Universidade Federal da Paraíba
Centro de Ciências e Tecnologia
Coordenação de Pós-Graduação em Informática

Katyusco de Farias Santos

Um Sistema de Operações para Suporte à Gerência de Centrais Digitais Heterogêneas Baseado em SNMP

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba - Campus II como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Orientadores:
J. Antão B. Moura – Ph. D
Marcelo Sampaio de Alencar - Ph. D

Área de Concentração: Ciência da Computação
Linhas de Pesquisa: Sistemas de Software
Gerência de Sistemas de Telecomunicações

Campina Grande, Paraíba, Brasil

S237s Santos, Katyusco de Farias
Um sistema de operacoes para suporte a gerencia de centrais digitais heterogeneas baseado em SNMP / Katyusco de Farias Santos. - Campina Grande, 1999.
117 f.

Dissertaca (Mestrado em Informatica) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Engenharia de Software 2. Gerencia SNMP 3. Gerencia de Telecomunicacoes 4. Dissertacao - Informatica I. Moura, Jose Antao Beltrao II. Alencar, Marcelo Sampaio de III. Universidade Federal da Paraiba - Campina Grande (PB)

CDU 004.41(043)

UM SISTEMA DE OPERAÇÕES PARA CENTRAIS DIGITAIS HETEROGÊNEAS BASEADO EM SNMP

KATYUSCO DE FARIAS SANTOS

DISSERTAÇÃO APROVADA EM 17.08.1999

PROF. JOSÉ ANTÃO BELTRÃO MOUTA, Ph.D
Orientador

PROF. MARCELO SAMPAIO DE ALENCAR, Ph.D
Orientador

PROF. MARCELO ALVES DE BARROS, Dr.
Examinador

PROF. CARLOS BECKER WESTPHALL, Dr.
Examinador

PROF. NELSON CAUAS ASFORA, M.Sc
Examinador

CAMPINA GRANDE – PB

RESUMO

A evolução dos atuais sistemas de telecomunicações tem criado um ambiente composto por redes e equipamentos heterogêneos, com diversidade de serviços e com um número cada vez maior de usuários. Gerenciar esta rede complexa é de fundamental importância para garantir a qualidade de seus serviços e a lucratividade do negócio de telecomunicações.

Uma das principais metas do negócio de telecomunicações é maximizar a produtividade da planta e otimizar o uso dos seus recursos disponíveis. As características multi-fornecedor da maioria das plantas de telecomunicações torna muito difícil a adoção do modelo de gerência TMN. Soluções intermediárias têm sido pesquisadas para prover determinadas funções de gerenciamento em equipamentos de diferentes fabricantes.

O trabalho desenvolvido nesta dissertação apresenta uma solução para realização de gerência de desempenho em redes de telecomunicações, de baixo custo, para uma planta telefônica heterogênea. É um sistema de *software* que monitora de cada uma das centrais que fazem parte do escopo de gerência, *e.g.* NEAX 61BR fabricada pela NEC, Trópico-RA e Trópico fabricadas pelo CPqD, o tráfego e a taxa de utilização de seus órgãos, a fim de determinar a eficiência global da rede. O sistema foi construído e implantado na Telemar-Paraíba, antiga Telpa, como um sistema de operações integrante do processo de desenvolvimento da filosofia TMN na empresa.

ABSTRACT

The evolution of the current telecommunications systems has created an environment composed of heterogeneous networks and equipments, with a plethora of services and an increasing number of users. Management of such a complex network is fundamental to deliver the quality of service and profit the telecommunications business demands.

One of the main goals of the telecommunications business is to maximize the plant productivity and optimize the use of available resources. The multi-vendor characteristic of most telecommunication plants makes it difficult to adopt a TMN approach as a primary paradigm. Intermediate solutions have been searched as options to provide certain functions of management to equipment from different vendors.

The project which is the subject of this dissertation presents a cost effective solution to implement the performance management of a heterogeneous telephonic network. It is a software system that monitors the traffic and utilization rate of a previously defined set of central offices (switches), e.g, NEAX 61BR, produced by NEC, Trópico-RA and Trópico-R, produced by CPqD, in order to determine the global efficiency of the network. The system has been implemented and deployed in Telemar-Paraíba, former Telpa, as an operations system which is part of the TMN development effort of the company.

AGRADECIMENTOS

Mais um passo foi dado, mais um sonho foi realizado, mais uma luta foi ganha e minha vida continua... Tenho muito a agradecer.

Agradeço ao ser supremo que rege o universo, Deus. E ao ser transcendental que criou o meu universo e é meu referencial de vida, minha mãe Mercês.

A J. Antão B. Moura, que me ajudou a identificar e a resolver os problemas de forma sensata e pragmática. Agradeço a Marcelo S. de Alencar, pela sua paciência imensurável e pelo seu voto de confiança, obrigado pela amizade que criamos e pelos conhecimentos transmitidos.

Aos meus familiares por me ajudarem nos momentos mais íntimos e mais difíceis da minha vida. Especialmente a Katyene, Júnior, Dando, Tony, Ada, Jujú, Beta, vovó Lenira, dona Nicinha, tio Edmilson e tia Marluce. Essa conquista também é de vocês.

A Light-Infocon pela oportunidade concedida. A todos os companheiros de trabalho, pela agradável troca de experiências. Em especial a Marcos Sebastian e a Adriano Sérgio pela a amizade e apoio.

Agradeço a minha namorada, Elaine, por "entender" a minha ausência e por sempre me incentivar. Aos meus amigos que sempre me perdoaram por faltar as farras e que sempre me apoiaram. Reservo aqui um agradecimento especial aos meus amigos, confidentes e irmãos: Johoson, Raul, Ary e Tenório.

Não posso deixar de agradecer ao prof. Jacques Sauvé, pela sua cobrança. A todos que fazem a COPIN, em especial a Ana Lúcia e a Vera pela paciência, incentivo e amizade.

SUMÁRIO

LISTA DE FIGURAS

GLOSSÁRIO

API – *Application Programming Interface* – Conjunto de funções e/ou classes de objetos disponibilizados para uso do programador.

ASN.1 – *Abstract Syntax Notation One* – Notação padronizada utilizada para representar objetos numa MIB.

CCITT - *International Telegraph and Telephone Consultative Committee.*

CHM – Comandos Homem-Máquina - Linguagem que possibilita a manipulação e a configuração dos blocos de *software* de uma central telefônica.

CMM - *Capability Maturity Model* – Modelo de aplicação de metodologia para desenvolvimento de *software.*

CMIP- *Common Management Information Protocol.*

CMIS - *Common Management Information Service.*

CMU - *Carnegie Mellon University.*

CPA – Central de Programa Armazenado.

GIRS - Gerência Integrada de Redes e Serviços.

GCR – Gerência Centralizada de Rede.

HTR – *Hard to Reach* – Direções de tráfego de difícil acesso.

HP - *Hewlett Packard* – Desenvolvedora de produtos de *hardware* e de *software.*

IANA - *InterNet Assigned Numbers Authority* - Entidade responsável pela padronização e organização lógica MIB.

IP – *Internet Protocol.*

ISO - *International Organization for Standardization.*

ITU - *International Telecommunications Union.*

ITU-T - *Telecommunications Standardization Sector.*

MIB – *Management Information Base.*

NE - *Network Element* (Elemento de Rede) - Pode variar desde um pequeno equipamento de uma rede de telecomunicações, como um modem, até uma central telefônica.

OAM&P - Conjunto de ações, que engloba as funções de operação, administração, manutenção e provisionamento.

OID - *Object Identifier.*

OMT - *Object Modeling Technique* - Metodologia para desenvolvimento de *software* orientada à objetos.

OS – *Operation System* (Sistema de Operação): É um sistema voltado à operação da redes de telecomunicações que desepenha um conjunto de funções definidas.

OSI - *Open System Interconnection.*

OSF – *Open Software Foundation.*

O&M - Operação e Manutenção

PDU – *Protocol Data Unit* – Unidade de pacote de dados que trafegam pela rede.

Qx – Pilha de protocolos, proposta pelo modelo TMN, que serve de interface entre os agentes e os sistemas de operações.

SMI - *Structure of Management Information.*

SNMP – *Simple Management Network Protocol.*

TCP – *Transmission Control Protocol.*

TMEST - Tempo Máximo Estimado Sem Transmissão de um caracter qualquer da central para o conversor.

TMLC - Tempo Máximo para Leitura de um Caracter.

TMN - *Telecommunications Management Network.*

1 INTRODUÇÃO

Nas atuais redes de telecomunicações observam-se, tipicamente, as seguintes características:

- Existência de vários equipamentos diferentes, tanto em sua aplicação (comutação, transmissão), quanto em sua composição (fabricantes diferentes);
- Diferentes tecnologias para a mesma aplicação (e.g. equipamentos analógicos *versus* digitais) que, geralmente, não oferecem ao usuário um sistema totalmente transparente;
- Diversidade de serviços, tais como: comunicação móvel privativa, sistemas de comunicação de dados e multimídia, telefonia móvel celular, serviços de telemensagem;
- Quantidade cada vez maior de usuários das redes.

Estas características proporcionaram o surgimento de uma rede complexa e de difícil gerenciamento, para a qual a necessidade de um sistema de gerência que possibilite seu funcionamento eficiente é de fundamental importância de forma a garantir a competitividade das operadoras e assegurar o atendimento do crescimento da demanda por serviços existentes e por novos. Esse sistema deve ter os seguintes objetivos gerenciais:

- Minimizar o tempo de reação a problemas e eventos da rede;
- Melhorar o serviço de assistência e interação com os clientes;
- Prover mecanismos de isolação para minimizar riscos de segurança e para localizar e conter falhas de rede;
- Ter controle dos diversos elementos de rede geograficamente dispersos, a partir de Centros de Operação;
- Minimizar a carga causada pelo tráfego de informações de gerenciamento.

Buscando resolver a problemática de gerenciamento de redes de telecomunicações surge o conceito Gerência Integrada de Redes e Serviços (GIRS). A Gerência Integrada de Redes e Serviços se constituí no conjunto de ações, que engloba as funções de operação, administração, manutenção e provisionamento (OAM&P) para todos os elementos da rede,

para a rede em si e para os serviços de telecomunicações, visando maximizar a produtividade da planta e dos recursos disponíveis.

Na atualidade do mundo das telecomunicações, em que prevalece a característica multi-fornecedor, para se proporcionar uma estrutura uniforme de Gerência de Redes de Telecomunicações é necessário fazer uso de padronização. As especificações das entidades de padronização, principalmente aquelas produzidas pelo ITU-T, definem um modelo denominado de *Telecommunications Management Network* (TMN), que propõe funções genéricas de gerência em diversos tipos de equipamentos de telecomunicações, funções essas que utilizam modelos genéricos de informação e interagem entre si através do uso de interfaces padronizadas [Sala94].

A adoção do modelo TMN por grande parte das atuais plantas de telecomunicações existentes, onde a heterogeneidade de equipamentos prevalece, é um processo complexo que requer um investimento alto por parte das operadoras detentoras desse tipo de planta [Pero93]. Esse investimento deve abranger diversos segmentos:

- Treinamento e contratação de mão-de-obra especializada;
- Mudança organizacional e administrativa da operadora;
- Compra de novos equipamentos que atendam aos requisitos do padrão TMN;
- Aquisição do ferramental de gerência para monitorar todo funcionamento da rede.

1.1 Motivação

Algumas soluções têm sido pesquisadas, desenvolvidas e adotadas pelas operadoras de telecomunicações para proporcionar um certo grau de gerência às suas plantas.

Existem alguns sistemas de gerência da rede de telecomunicações suportados por PCs, *workstations* ou minicomputadores baseados em arquiteturas proprietárias, que atendem a um determinado elemento de rede de um determinado fabricante, visando objetivos específicos. No entanto, o caráter proprietário de tais sistemas gera uma série de problemas:

- Interfaces não padronizadas de acesso aos elementos de rede;

- Diferentes conjuntos de comandos homem-máquina (CHM);
- Diferentes formas de operação e manutenção (O&M) para cada equipamento;
- Multiplicidade de sistemas de supervisão;
- Bases de informação duplicadas;
- Não interoperabilidade entre arquiteturas proprietárias distintas.

Além disso, os sistemas não possibilitam uma integração de processos na operação de redes e prestação de serviços de maneira que a partir de uma reclamação de um usuário qualquer, não é possível, automaticamente, realizar testes sobre a planta para verificar a autenticidade da reclamação. Essa falta de integração entre processos ocasiona uma série de dificuldades no gerenciamento da rede como um todo [Alen98]. Pode-se destacar as seguintes dificuldades:

- Impossibilidade de obter uma visão geral da rede e dos serviços, dificultando assim a identificação das causas das falhas;
- Incapacidade de automatizar as atividades operacionais.

Em função das dificuldades e problemas acima, podem ser elencados os principais fatores que motivaram o desenvolvimento deste trabalho. São eles:

- Custo elevado e a alta complexidade para adoção e implantação do modelo TMN;
- As dificuldades apresentadas pelos sistemas de gerência que não seguem o padrão TMN.

1.2 Objetivos

O principal objetivo deste trabalho é modelar e implementar uma solução de gerência, de baixo custo, para uma rede de telecomunicações heterogênea, na qual os recursos disponíveis não possibilitem adotar o padrão TMN. A solução deve também evitar ou minimizar os problemas elencados nos sistemas proprietários de gerência.

A solução desenvolvida neste trabalho se constitui de um sistema de *software* que fornece subsídios para realização de uma gerência de desempenho em uma rede de

telefonia heterogênea constituída de centrais do tipo: NEAX 61BR fabricada pela NEC, Trópico-RA e Trópico-R fabricadas pelo CPqD.

Além de permitir o acesso a todas as centrais, a solução monitora em cada uma delas o tráfego e a taxa de utilização de seus órgãos (componentes de uma central), a fim de oferecer uma visão da eficiência da rede. Utilizam-se mediadores específicos que preenchem um modelo de informação único com as informações referentes ao tráfego e taxa de utilização dos órgãos. Sobre o modelo de informação realizam-se consultas através de primitivas SNMP, cujo resultado é analisado a fim de se determinar a eficiência global da rede.

1.3 Organização do Restante da Dissertação

Além deste capítulo introdutório, esta dissertação é composta de mais quatro capítulos e dois apêndices, são eles:

O **Capítulo 2** aborda conceitos e resumos de áreas afins enfocando a Gerência Integrada de Redes e Serviços de Telecomunicações (GIRS) e a Gerência de Desempenho em Redes de Telecomunicações. O Capítulo ainda direciona em um contexto conceitual, o problema que o trabalho destina-se a resolver: Gerência de Desempenho sobre Redes de Telefonia Heterogêneas.

No **Capítulo 3**, primeiramente é apresentada a solução desenvolvida de forma genérica a fim de fornecer uma visão do ambiente como um todo. Depois define-se sua arquitetura funcional na qual estão descritos detalhadamente a funcionalidade de seus componentes funcionais, além de apresentar arquitetura física da solução.

Já no **Capítulo 4**, são descritos, em alto nível, detalhes de implementação para os principais componentes de *software* dos módulos que fazem parte da solução. Para cada componente funcional descrito no Capítulo 3 é apresentada sua hierarquia de classes com seus respectivos atributos e comportamentos.

O **Capítulo 5** apresenta as conclusões e os resultados obtidos com o trabalho desenvolvido, além de apresentar propostas para continuação da pesquisa e para o desenvolvimento de novos trabalhos.

Os **Apêndices A** e **B** tratam respectivamente, de conceitos relacionados a *Telecommunications Management Network* (TMN) e ao *Simple Network Management Protocol* (SNMP). No decorrer deste trabalho o conhecimento de conceitos referentes a esses apêndices são necessários.

2 CONCEITOS NECESSÁRIOS E DESCRIÇÃO DO PROBLEMA

2.1 Introdução

Com objetivo de localizar e fornecer subsídios ao entendimento da dissertação, este capítulo apresenta conceitos e resumos de áreas afins. A principal ênfase é dada à Gerência Integrada de Redes e Serviços de Telecomunicações (GIRS) e à Gerência de Desempenho em Redes de Telecomunicações que são os principais conceitos nos quais se baseia o trabalho desenvolvido. São apresentadas também as definições sobre *Telecommunications Management Network* (TMN) e *Simple Network Management Protocol* (SNMP), sobre os quais os **Apêndices A** e **B**, respectivamente, podem ser referenciados em conjunto para maiores detalhamentos conceituais.

Em seguida, o capítulo identifica e situa no contexto conceitual o problema que o trabalho objetiva-se resolver, que é a realização de uma gerência de desempenho sobre redes de telecomunicações heterogêneas. Na sequência, apresenta o esboço da solução desenvolvida para realização de tal gerência.

2.2 Gerência Integrada de Redes e Serviços de Telecomunicações

A contínua evolução dos sistemas de telecomunicações deu origem a um ambiente de rede bastante heterogêneo, seja pelas diferenças tecnológicas, seja pela variedade de fornecedores de equipamentos e produtos que são, usualmente, gerenciados por sistemas proprietários e incompatíveis entre si. A complexidade e falta de integração do ambiente atual eleva os custos operacionais, diminui o grau de segurança e compromete a agilidade do gerenciamento de redes e serviços. A sofisticação e qualidade atualmente exigidas dos serviços de telecomunicações não são compatíveis com esse ambiente heterogêneo. Fica evidente que as soluções proprietárias de gerência, originalmente adotadas, por não terem a capacidade de atuar de maneira integrada sobre redes e serviços, constituem um entrave para que as empresas fornecedoras desses serviços atinjam o máximo de qualidade e produtividade de sua planta.

Buscando resolver toda essa problemática de gerenciamento de redes de telecomunicações surge o conceito Gerência Integrada de Redes e Serviços (GIRS)

[Rebe93]. Apesar do centro das atividades a respeito de GIRS ser o ITU-T (*International Telecommunications Standardization Sector*, antigo CCITT), diversas entidades como a *Telemanagement Forum*, OSF (*Open Software Foundation*), ISO (*International Organization for Standardization*), tem colaborado para o desenvolvimento de documentos no sentido de definir padrões para o conceito de GIRS.

A Gerência Integrada de Redes e Serviços (GIRS) se constituí no conjunto de ações, que englobe as funções de operação, administração, manutenção e provisionamento (OAM&P) para todos os elementos da rede, para a rede em si e para os serviços de telecomunicações, visando maximizar a produtividade da planta e dos recursos disponíveis [SDT501-100-104], [SDT501-100-105].

Dessa forma, pode-se afirmar que o principal objetivo da GIRS é proporcionar o funcionamento ininterrupto e otimizado da rede, com vistas à eficiência do negócio de telecomunicações. Contudo, pode-se listar alguns outros objetivos básicos e não menos importantes:

- Automação de tarefas, tornando a gerência mais eficiente e mais simples;
- Fornecimento de dados em tempo real, agilizando a manutenção da planta de telecomunicações;
- Além da deteção de falhas, determinação de suas causas, possibilitando a resolução do problema em sua raiz;
- Flexibilidade para suportar as diferentes arquiteturas e estruturas organizacionais, além das evoluções tecnológicas e expansões;
- Ambiente aberto, permitindo a interconectividade e interoperabilidade entre sistemas, e também um ambiente multi-fornecedor;
- Terminal de operação universal de acesso aos elementos de rede, com apresentação padrão;
- Facilidade de reconfiguração em tempo real;
- Alta confiabilidade, tanto nos dados e informações de gerência obtidos da planta, como nas ações realizadas sobre a planta;

- Segurança, implicando em ter um acesso restrito ao tipo de gerência;
- Eliminação da multiplicidade de sistemas de supervisão.

Como conceito, GIRS não deve ser confundido com a simples compra de um conjunto de sistemas de operações ou de equipamentos com interfaces padronizadas. Sua efetiva introdução é um trabalho abrangente de reengenharia dos processos que são utilizados na operação do negócio de telecomunicações e que deve ser iniciado mesmo antes de uma completa reformulação dos sistemas atuais existentes.

O objetivo das especificações das entidades de padronização, principalmente o ITU-T é de proporcionar uma estrutura uniforme de Gerência de Redes de Telecomunicações, por meio da introdução de modelos genéricos, possibilitando o desempenho de funções genéricas de gerência em diversos tipos de equipamentos de telecomunicações que suportam modelos genéricos de informação e interfaces padronizadas. O modelo mais difundido e estudado é o TMN, *Telecommunications Management Network*, cujas características são abordadas no Apêndice A.

2.2.1 Estrutura Funcional da GIRS

Em virtude de sua complexidade, divide-se a Gerência Integrada de Redes e Serviços quanto a sua funcionalidade em camadas de abstração ou níveis de gerência, conforme a Figura 2.1

A Gerência do Negócio é responsável pela consolidação de dados (demanda, faturamento, tempo de atendimento, qualidade de serviço, planejamento), dando subsídios aos sistemas corporativos e de informação gerencial.

A Gerência de Serviço se encarrega do suporte aos centros de atendimento ao usuário. Relaciona-se com os sistemas das outras camadas para ativação e criação de novos serviços, recuperação, testes, avaliação fim a fim de qualidade.

A Gerência de Rede permite uma visão sistêmica (global) de todos os elementos que fazem parte de uma rede telecomunicações. É justamente, nessa camada que se insere o Serviço de Gerência de Desempenho que busca atender à Supervisão de Código e a Supervisão de Órgãos.

A Gerência dos Elementos de Rede é responsável pela gerência de um conjunto de elementos que fazem parte da rede. Essa gerência pode abranger uma área geográfica ou um tipo de tecnologia.

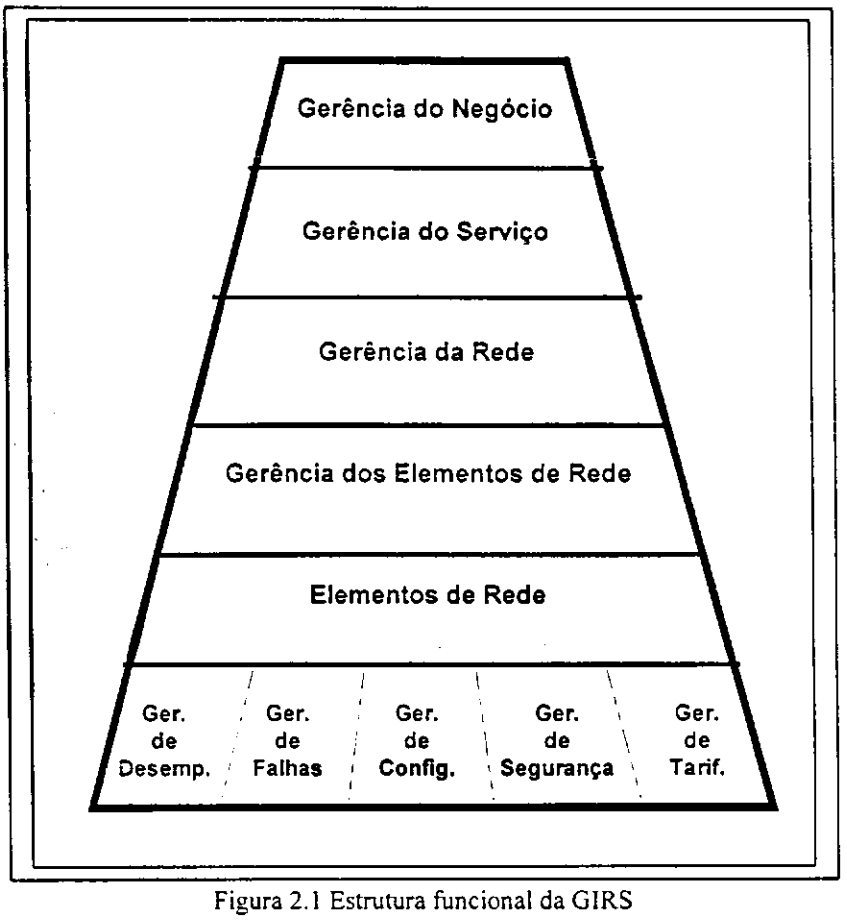

Figura 2.1 Estrutura funcional da GIRS

Por fim, a camada Elementos de Rede é vista como um conjunto de equipamentos a serem gerenciados. Dentro do escopo da solução desenvolvida neste trabalho, sempre que se fizer referência a um Elemento de Rede ou Elemento considera-se uma central telefônica digital.

2.2.2 *Atividades Funcionais de Gerência*

A cada Nível de Gerência apresentado na Figura 2.1, estão associadas cinco atividades funcionais responsáveis por desempenhar, de acordo com as necessidades de administração e de gerenciamento de redes, as funções de planejamento, instalação, operação, manutenção e provisionamento de redes e serviços de telecomunicações:

- A Gerência de Desempenho provê funções que consistem na monitoração de desempenho e qualidade dos equipamentos e dos serviços de telecomunicações com a finalidade de avaliar e corrigir o comportamento e eficácia da rede, dos elementos de rede ou equipamentos e para auxiliar o planejamento e análise dos mesmos;

- A Gerência de Falhas é um conjunto de funções que permite a detecção, isolamento e correção de condições anormais de operação da rede de telecomunicações. Essa função de gerência está relacionada a falhas que afetam o serviço prestado, mesmo que tenham sido conseqüência das outras funções de gerência;

- Já a Gerência de Configuração fornece funções para criar e modificar o modelo de gerenciamento de recursos físicos e lógicos da rede de telecomunicações;

- A Gerência de Segurança inclui as funções que dizem respeito ao controle de acesso de operadores e usuários às informações e ações de gerenciamento da rede, além da função de rastreamento para auditoria no caso de violação.

- Por fim, a Gerência de Tarifação também chamada de Gerência de Contabilização, inclui um conjunto de funções que permitem medir a utilização dos serviços oferecidos e associar custos a tal utilização. Prevê facilidades para a coleta de dados de tarifação para a utilização dos mesmos.

2.3 TMN - *Telecommunications Management Network*

A TMN é um modelo genérico de arquitetura baseada no modelo **OSI** (***Open System Interconnection***) de gerência, que provê suporte à implantação de uma Gerência Integrada de Redes e Serviços.

O objetivo básico da TMN é fornecer uma estrutura organizada para interconectar vários tipos de Sistemas de Suporte à Operação (**OS** - ***Operations System***), equipamentos e serviços de telecomunicações para a troca de informação de gerenciamento utilizando interfaces padronizadas que incluem a definição de protocolos e mensagens [BRISA93].

Atualmente os trabalhos realizados sobre TMN estão sobre a responsabilidade da entidade de padronização **ITU-T** (***ITU-International Telecommunication Standardization***

Sector). No Apêndice A é apresentado um resumo dos aspectos conceituais da padronização TMN.

2.4 Gerência de Desempenho em Redes de Telecomunicações

A Gerência de Desempenho busca monitorar o desempenho e qualidade dos equipamentos e dos serviços de telecomunicações avaliando e corrigindo o comportamento e eficácia da rede, dos elementos de rede ou equipamentos e auxiliando o planejamento e análise dos mesmos.

No escopo do trabalho, a Gerência de Desempenho é estudada apenas no nível de Gerência de Rede de maneira a fornecer subsídios para monitoração do desempenho da rede telefônica como um todo.

A Gerência de Desempenho em Redes Telefônicas tem como suporte a Teoria de Tráfego Telefônico. O objetivo da Teoria de Tráfego Telefônico é planejar sistemas telefônicos para que as chamadas realizadas pelos assinantes tenham alta probabilidade de sucesso, mesmo nos períodos de tráfego mais intenso – chamadas de horas de maior movimento (HMM) [Alen98]. Seguem abaixo alguns conceitos básicos e termos mais usados na Teoria de Tráfego Telefônico:

Assinante: Cada um dos terminais telefônicos ligados à central;

Órgão ou circuito: Um equipamento da central telefônica que realiza alguma função específica durante o estabelecimento de uma comutação, por exemplo, juntor;

Tempo de ocupação: O intervalo de tempo em que um assinante ocupa um órgão da central ou centrais;

Volume de tráfego: Somatório dos tempos de ocupação dos órgãos;

Período de observação: Intervalo de tempo em que se observa o comportamento das ocupações de um sistema telefônico;

Intensidade de ocupação: Número de ocupações num sistema telefônico, durante o período de observação;

Tempo médio de ocupação: Definido como a média aritmética dos tempos de ocupação das chamadas, num dado sistema, para um determinado período de observação;

Intensidade de tráfego: Relação entre o volume de tráfego e o período de observação. É medido em *Erlangs* (Erl);

Erlang: Unidade de tráfego que representa o número médio de órgãos ocupados durante um intervalo de tempo. Um tronco ocupado continuamente por um período de 60 minutos corresponde a um tráfego de 1 Erl.

É através da medição do tráfego telefônico que se pode identificar quais são os ofensores da rede telefônica, para que se possa tomar as providências corretivas e preventivas. Além disso o estudo de tráfego possibilita que as companhias de telefonia dimensionem e ampliem suas centrais e os meios de transmissão que as interligam de forma a oferecer um serviço telefônico de qualidade aos seus assinantes, ao mais baixo custo possível para elas.

A análise do tráfego se baseia pela coleta de dados de dois tipos de indicadores: Indicadores do Grupo de Prestação de Serviços de Telecomunicações e Indicadores do Grupo de Completamento de Chamadas.

Através de uma análise sobre indicadores de completamento de chamadas pode-se: verificar a qualidade das chamadas saintes e entrantes da central analisada, identificar a taxa de chamadas completadas e perdidas pela central, além de ser possível detectar as áreas de difícil acesso (**HTR** - ***Hard To Reach***) e se evitar sua propagação na rede. A essa análise dá-se o nome de Supervisão de Código. Os Indicadores de Completamento de Chamadas são descritos a seguir:

OK: Indica o total de chamadas completadas sem erro;

NR: Indica o total de chamadas que o assinante de destino não respondeu;

LO: Indica o total de chamadas que encontraram o assinante de destino na condição de ocupado;

CO0: Indica o número de chamadas não completadas devido ao esgotamento de temporização de sinalização no receptor;

CO1: Indica o número de chamadas não completadas por congestionamento ou defeito na central de origem;

CO2: Indica o número de chamadas não completadas por congestionamento ou defeito na central de destino;

CO3: Indica o número de chamadas não completadas devido à falha na troca de sinalização;

CO: Somatório dos valores de CO0, CO1, CO2 e CO3;

OU: Indica chamadas não completadas devido a fatores diferentes dos acima citados;

PAB: Indica a taxa de perda do assinante B, diz respeito às chamadas encaminhadas e não completadas, seja porque o telefone está ocupado (LO) ou não responde (NR).

Já sobre os indicadores de prestação de serviços pode-se: verificar o tráfego em cada uma das rotas da central, averiguando se a quantidade de juntores satisfaz o tráfego na rota observada, avaliar a perda numa rota e se determinar a quantidade de juntores necessários para suprir o tráfego. A essa análise dá-se o nome de Supervisão de Órgãos. Os dados obtidos dos Indicadores de Prestação de Serviços apresentam:

Tráfego nas rotas: Tráfego que circula nas rotas de saída e entrada da central;

Tráfego nos órgãos comuns: Os órgãos comuns são responsáveis pela troca de sinalização da central, assim é tráfego que passa pelos juntores, enviadores e receptores;

Tráfego na HMM: É o trafego medido na hora de maior movimento;

Tráfego trânsito: Tráfego de passagem na central, refere-se às chamadas que são escoadas por uma central sem que sejam originadas ou terminadas nesta;

Tráfego interno: Tráfego entre os assinantes de uma mesma central.

Baseado no conceito de GIRS a realização de uma Supervisão de Órgãos e de Código sobre uma dada central caracteriza uma atividade de Gerência de Desempenho no nível de Gerência de Elemento de Rede.

Contudo, desde que se possa e se consiga realizar a coleta das informações acerca dos indicadores de serviços de telecomunicações e de completamento de chamadas para todas as centrais de uma planta de telefonia, a análise de tráfego proporcionará uma visão sistêmica da rede telefônica, indicando seus ofensores. Dessa forma, a Gerência de Desempenho passa a ser realizada no nível de Gerência de Rede.

2.5 Gerência de Desempenho em Redes de Telefonia Heterogêneas

A grande dificuldade em se obter as informações, acerca dos indicadores de serviços de telecomunicações e de completamento de chamadas é a característica multi-forncedor de cada umas das centrais que formam a maioria das atuais redes de telecomunicações.

Apesar de no modelo TMN estarem previstas interfaces padronizadas q_3/q_x para centrais de comutação que dentre várias funções, uma delas seria prover mecanismos que disponibilizassem os indicadores para as camadas superiores da TMN, alguns aspectos precisam ser mencionados:

- A realidade é que nem os próprios fabricantes possuem de fato tais interfaces;
- Os custos para a implantação de interfaces q_3/q_x são altos, já que seria necessário pessoal qualificado, compra de equipamentos, treinamentos e aquisição de tecnologia.

Em relação às centrais não existe tipo padronização entre os fabricantes. Na verdade cada tipo de central possui seus mecanismos proprietários para disponibilizar tais indicadores, impossibilitando e inviabilizando a coleta automática dos seus respectivos indicadores para realização de uma gerência de desempenho, no nível de rede.

O principal desafio do trabalho é organizar, no sentido de modelar e implementar, uma solução em que se possa obter os indicadores de serviços de telecomunicações e de completamento de chamadas das principais centrais de uma planta para, a partir desses, determinar e averiguar a eficiência da rede telefônica como um todo. Assim, baseado nos conceitos apresentados anteriormente pode-se afirmar que o trabalho procura resolver o problema de Gerência de Desempenho em Redes de Telefonia Heterogêneas.

A solução precisa estar adequada à concepção de Sistemas de Operações de GIRS de maneira a fornecer mecanismos através do qual se possa prover uma interligação com os demais Sistemas de Operações construídos. Ou ainda, que tenha características TMN para que facilmente possa se adequar aos padrões internacionais.

Nesse trabalho, a solução levou em consideração um conjunto de requisitos definidos a partir das necessidades intrínsecas da companhia operadora de telefonia para o qual o sistema foi desenvolvido, a Telpa (Empresa de Telecomunicações da Paraíba), que atualmente integra o complexo Telemar. Pode-se listar os seguintes requisitos:

- Uso da própria rede de telecomunicações para gerenciar a planta telefônica;
- O escopo de gerência deve abranger as principais centrais existentes na rede: NEAX - tecnologia NEC, Trópico-R e Trópico-RA - tecnologia nacional CPqD;
- Possuir um modelo único de informação;
- O projeto e a implantação da solução devem acontecer de maneira modular;
- Fazer uso intensivo de *hardware* e *software* comerciais que se baseiem nos padrões de sistemas abertos;
- Os custos da solução devem ser inferiores às soluções proprietárias existentes como também inferior ao desenvolvimento de soluções baseadas no padrão TMN;
- A solução deve proporcionar um melhoramento na qualidade do serviço de telefonia da planta de telecomunicações da operadora.

2.6 SNMP - *Simple Network Management Protocol*

Mais que um protocolo de comunicação, o SNMP em sua especificação define-se como uma arquitetura que oferece suporte ao gerenciamento de redes **IP** (*Internet Protocol*) e que se constitui:

- Das entidades gerenciadas denominadas agentes e dos gerenciadores chamados gerentes;
- Do protocolo SNMP para troca de informações entre um ou mais sistemas de gerência e agentes;

- Do modelo de informação de gerência, denominado MIB (*Management Information Base*), que contém um número de variáveis ou objetos de propósito geral;
- Uma ferramenta para formatação e armazenamento das informações de gerência.

No Apêndice B apresenta-se um resumo conceitual de cada um dos componentes arquitetura de gerência SNMP e a interação entre eles.

2.7 Gerência de Desempenho em Redes de Telefonia Heterogêneas usando SNMP

A solução desenvolvida neste trabalho apresenta a modelagem e implementação de conversores que funcionam como mediadores entre as centrais e um Sistema de Operações. A função dos conversores é obter de cada uma das centrais pertencentes à planta, as informações acerca dos indicadores necessários à realização de gerência de rede e armazená-las numa base de informações de gerência – MIB. A MIB possui um modelo de informação único para todas as centrais e pode ser manipulada através de entidades que se comuniquem através do protocolo de gerência SNMP.

Dessa forma, para se obter as informações acerca dos indicadores de cada uma das centrais, basta consultar a base de gerência, proporcionando assim que se realize uma gerência sobre o serviço de telefonia de uma rede de telecomunicações heterogênea que possibilite uma visão global da planta telefônica em função dos seus indicadores.

2.8 Conclusão

O Capítulo faz um resumo analítico dos conceitos necessários ao entendimento e a localização do trabalho desenvolvido direcionando para a identificação do problema e sua respectiva solução.

Apresenta o conceito de Gerência Integrada de Redes e Serviços (GIRS) para realizar funções de OAM&P sobre a rede com o objetivo de que esta atinja o máximo de qualidade e produtividade. Enfatiza uma arquitetura genérica, criada pelo ITU-T, denominada *Telecommunications Management Network* – TMN.

Aponta os principais conceitos da Teoria de Tráfego Telefônico e como, a partir deles, pode-se realizar uma Gerência de Desempenho sobre serviço de telefonia de uma rede de telecomunicações. Apresenta também o modelo de gerência SNMP com seus componentes e conceitos.

Segue esboçando o problema a ser solucionado em que se almeja obter de todas as centrais da planta, os indicadores de serviços e de completamento de chamadas para a partir desses gerenciar a eficiência completa da rede.

Em seguida apresenta a solução desenvolvida para realização de Gerência de Desempenho em Redes de Telefonia Heterogêneas utilizando a arquitetura SNMP. O próximo capítulo apresentará a descrição completa da solução proposta.

3 SOLUÇÃO - O AMBIENTE DE GERÊNCIA

3.1 Introdução

Este capítulo apresenta a solução desenvolvida, para realização de Gerência de Desempenho em Redes de Telefonia Heterogêneas.

A solução se constitui de em um ambiente de gerência que numa primeira instância é apresentado de forma genérica a fim de fornecer uma visão do ambiente como um todo. Em seguida define-se sua arquitetura funcional, onde estão descritos seus componentes funcionais, e sua arquitetura física na qual se localiza fisicamente cada um dos seu componentes funcionais, dentro do ambiente.

O desenvolvimento maior do capítulo encontra-se na descrição detalhada de cada um dos componentes funcionais que constituem um ambiente de gerência.

3.2 Visão Geral do Ambiente de Gerência

O ambiente de gerência de centrais CPAs desenvolvido, chamado de **Gerência Centralizada de Rede (GCR)**, tem como maior objetivo coletar de maneira automática dados de centrais digitais relativos à Teoria de Tráfego Telefônico para que quando processados, possam fornecer subsídios e informações suficientes para a realização de uma gerência de desempenho de redes de telefonia.

Inserido no contexto de gerência de telecomunicações, pode-se afirmar que o ambiente se enquadra no nível de gerência de rede, realizando uma monitoração de diversas CPAs, proporcionando assim uma visão da situação global da planta de centrais.

Pela própria natureza das atuais redes de telecomunicações, a diversidade dos equipamentos é um fator preponderante, em que é possível encontrar diversos tipos de equipamentos de muitos fornecedores. Como conseqüencia, a obtenção dos dados de tráfego muda de equipamento para equipamento, assim como de fornecedor para fornecedor, já que as centrais telefônicas que fazem parte das plantas não possuem especificações de interfaces padronizadas que possibilitem a obtenção de tais dados de maneira única. Em virtude deste fato, ambiente o GCR se utiliza de conversores, ou

mediadores [Sala94], para a obtenção dos dados de tráfego de cada uma das centrais a ser monitorada pelo ambiente.

Por ser um ambiente de gerência é inerente seu paradigma distribuído, em que alguns módulos do sistema podem executar suas tarefas em plataformas distintas e separadas fisicamente, sendo contudo, conectadas por algum meio de comunicação. No caso do GCR, a comunicação entre esses módulos é suportada por um ambiente de rede TCP/IP, utilizando um protocolo de gerência padrão já bem conhecido, o SNMPv1 [Stal93].

Os elementos de rede (Nes) que fazem parte do escopo de gerência do ambiente são centrais digitais, dentre os quais pode-se listar as seguintes:

- Centrais do tipo NEAX - tecnologia NEC;
- Centrais do tipo Trópico-R - tecnologia nacional CPqD;
- Centrais do tipo Trópico-RA - tecnologia nacional CPqD.

O ambiente pode ser considerado um OS (*Operating System*) [Sala94], que atua na camada de gerência de rede da GIRS. O sistema desenvolvido procura realizar, de maneira automatizada, dois tipos de supervisão:

Supervisão de Código – Neste tipo de supervisão busca-se obter dados dos contadores de desempenho (OK, LO, CO, NR, OU, PAB) das diversas centrais monitoradas. Para que por intermédio desses contadores se consiga detectar as áreas de difícil acesso (**HTR - *Hard To Reach***) e se evitar sua propagação na rede;

Supervisão de Órgãos – Caracteriza-se pela obtenção dos dados referentes à taxa de utilização de órgãos (juntores, receptores, enviadores) das centrais. De forma que, a partir desses, seja possível identificar quais órgãos degradam o desempenho da rede e quais as necessidades de expansão da planta.

As principais funcionalidades oferecidas pelo ambiente são:

- Interação entre os elementos de rede (centrais digitais NEAX, Trópico-R e Trópico-RA) e o ambiente, possibilitando o envio de comandos às centrais e a coleta dos diversos relatórios emitidos por cada uma delas para a extração dos dados de gerência;

- Processamento dos dados de gerência, possibilitando a apresentação de vários tipos de relatórios cujo conteúdo são informações sobre supervisão de código e supervisão de órgãos;
- Emulação dos terminais de qualquer CPA, que esteja sob seu domínio, de um único ponto centralizado;
- Uma interface gráfica (padrão MOTIF), através da qual é realizada toda operação sobre o ambiente.

O escopo de trabalho da dissertação dá suporte para realização das três primeiras funcionalidades citadas, que serão melhor detalhadas no decorrer do trabalho. Mesmo não fazendo parte do escopo do trabalho, a funcionalidade gráfica é citada para fornecer uma visão completa da arquitetura do ambiente de gerência.

3.2.1 Arquitetura Funcional

Como observado na Figura 3.1, funcionalmente os principais módulos do ambiente são os seguintes:

Emulação de Terminais – Permite que a partir de um ponto central do ambiente de gerência seja possível ter acesso aos terminais de quaisquer das centrais gerenciadas.

Conversores (coletores de dados) - Os conversores são responsáveis por enviar comandos homem-máquina específicos, às CPAs, por ler os diversos relatórios emitidos pelas mesmas em função dos comandos enviados e de extrair desses as informações para a realização do gerenciamento. Para cada tipo de CPA existe um tipo de conversor.

Agentes – São entidades de gerência que atendem às solicitações do gerente, de maneira que se o gerente necessita de alguma informação de gerência, é o agente que deve fornecer. Para cada conversor de uma dada CPA existe um agente específico.

MIB (***Management Information Base***) – É o modelo de informação do ambiente, cujo conteúdo é formado pelos dados coletados pelos conversores.

Gerente - É a entidade que solicita periodicamente, informações aos diversos agentes por ele monitorados. Assim, através de um mecanismo de coleta periódica, chamado *polling* [Sala94] o gerente obtém informações de todos os agentes das centrais, com a finalidade de identificar o atual *status* da planta.

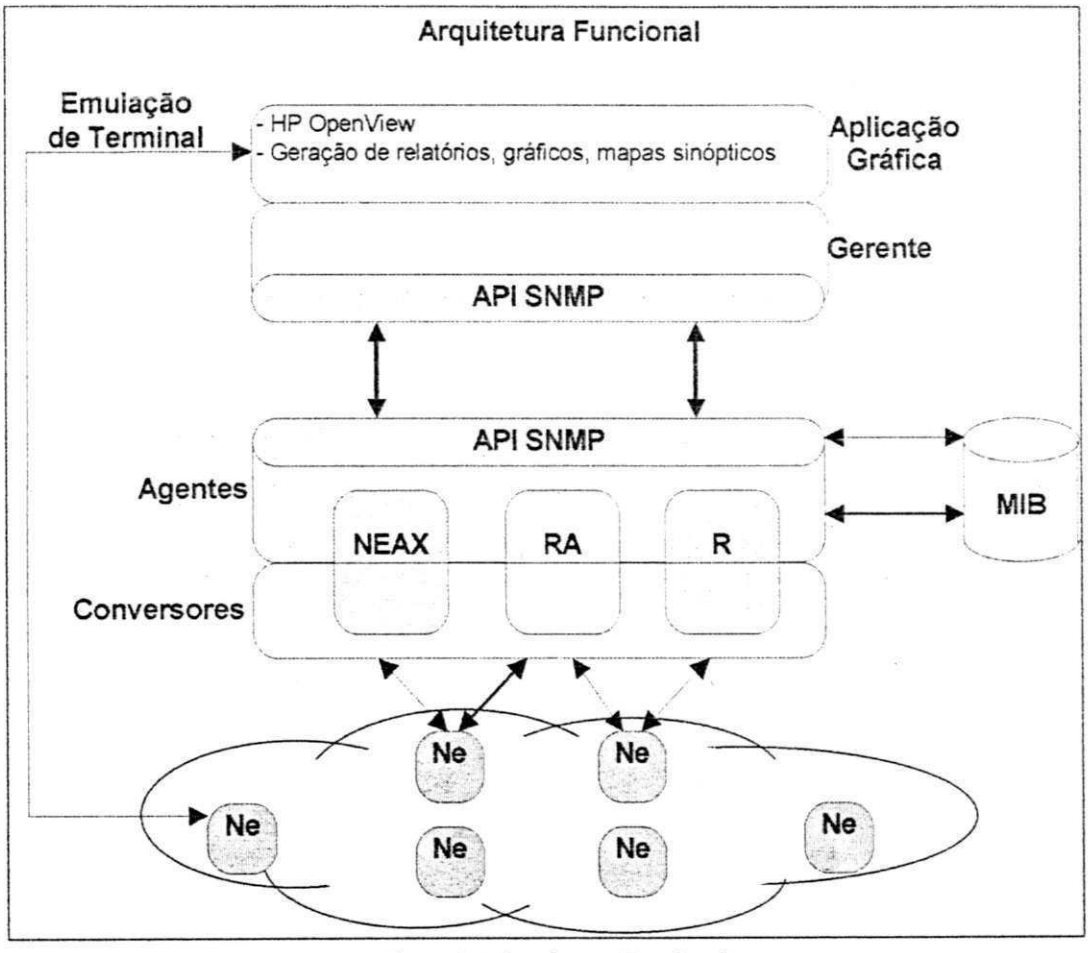

Figura 3.1 Arquitetura Funcional

Aplicação Gráfica - É baseada no padrão XWindow/MOTIF, construído utilizando o *framework* (ambiente gráfico para desenvolvimento de aplicações) de gerência da *Hewlett Packard, HP OpenView*. Essa camada permite que os usuários acessem todas as funcionalidades do ambiente e verifiquem o atual *status* da rede por meio de gráficos, mapas sinópticos, relatórios, e alarmes. Ainda nessa camada de aplicação encontram-se mecanismos de verificação de acesso dos diversos usuários do ambiente e mecanismos para geração de *logs* de acesso e de ações também de usuários.

O ambiente baseia-se no modelo de gerência *gerente-agente* e a comunicação entre essas entidades é realizada através do protocolo SNMPv1 da família TCP/IP, conferindo ao sistema características de um sistema aberto. O ambiente é de fácil evolução no tocante ao seu potencial de gerenciar outras centrais digitais além das NEAX, Trópico-RA e Trópico-R, visto que para a inclusão de um novo tipo de CPA no escopo de gerência, seria apenas necessário construir um novo conversor específico e seu respectivo agente [Sant99a].

Os módulos emulação de terminais, conversores, MIB, agentes e gerente delimitam o escopo do trabalho desenvolvido e, em conjunto, formam o Sistema de Operações para realização de Gerência de Desempenho em Redes de Telefonia Heterogêneas.

3.2.2 *Arquitetura Física*

Por ser um ambiente de gerência, seu carácter distribuído é inerente e pode ser observado na Figura 3.2 na qual é apresentada a arquitetura física do GCR. Fisicamente pode-se identificar as seguintes entidades:

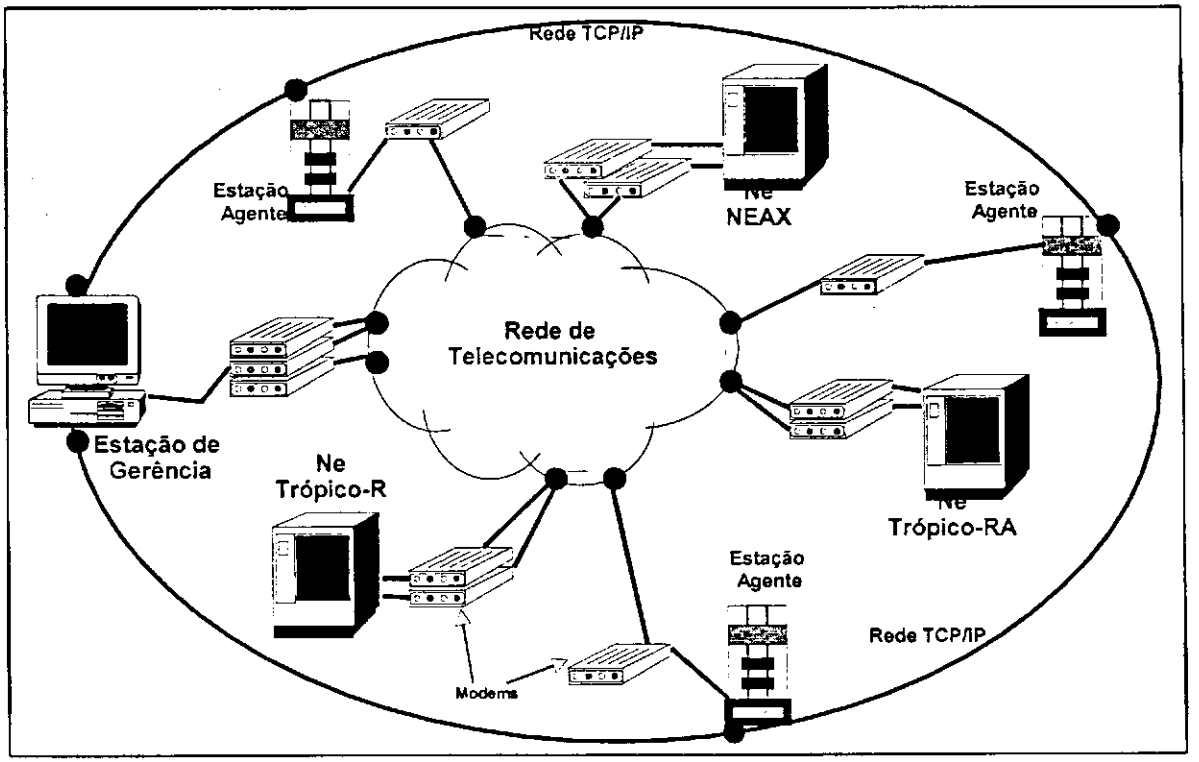

Figura 3.2 Arquitetura Física do GCR.

Estação de Gerência – É ponto central de gerência. Constituí-se de uma plataforma RISC da *Hewlett Packard* e Sistema Operacional Unix HP-UX, padrão POSIX. Nessa plataforma são executados os módulos funcionais Emulação de Terminais, Gerente e Aplicação Gráfica. À Estação de Gerência estão associados *modems* cuja finalidade é servir como ponto de conexão para emulação dos terminais de acesso de todas as centrais a serem gerenciadas pelo ambiente;

Elementos de Rede (Ne) – São as centrais digitais (NEAX, Trópico-R, Trópico-RA) a serem gerenciadas pelo ambiente. A cada elemento tem-se associado dois *modems*: um para emulação de um terminal de acesso pela Estação de Gerência e outro para realizar a comunicação entre a Estação Agente e a própria central;

Estações Agentes – Associada a cada elemento de rede tem-se uma plataforma PC (*Personal Computer*) CISC 486 ou superior e Sistema Operacional SCO Unix. Nas Estações Agentes são executados os módulos funcionais Conversor e Agente, específicos para cada tipo de central, e a MIB contendo as informações de gerência para a central associada à Estação Agente;

Rede de Telecomunicações – É a própria rede de telecomunicações que, no GCR, serve para comunicação entre as Estações Agentes e sua respectiva central e entre a Estação de Gerência e todas as centrais da planta a serem gerenciadas;

Modems – São os pontos de conexão entre as entidades Estação de Gerência e todas as centrais a serem gerenciadas e as estações agentes e suas respectivas centrais;

Rede TCP/IP – É através dessa rede que os módulos funcionais Gerente (da Estação de Gerência) e Agentes (das suas respectivas Estações Agentes) interagem entre si para a troca de informações de gerência constituídas de pacotes SNMP.

No ambiente de gerência GCR, é possível realizar a gerência sobre as centrais de uma planta mesmo que essas não possuam uma Estação Agente associada. Nesse casso, os módulos funcionais que seriam executados na Estação Agente passam a ser executados na Estação Gerente juntamente com os demais módulos que originalmente são executados nessa estação. Apesar de ser uma situação desaconselhável, já que sobrecarrega a Estação Gerente, essa solução pode resolver temporariamente a falta de gerência sobre um determinado elemento da planta. Isso só é possível pelo fato de que a comunicação, através

do protocolo SNMP, entre o agente de uma central e o gerente é transparente em relação à localização dos mesmos. Assim como também é transparente a comunicação entre os agentes e a central a ser gerenciada, que utiliza a própria rede de telecomunicações e *modems*.

Nos tópicos seguintes, apresenta-se o funcionamento de maneira detalhada de cada um dos módulos que constituem o sistema.

3.3 Emulação de Terminais das CPAs

Em toda CPA é executado um conjunto de blocos de *software* responsável pelo funcionamento geral da central. A manipulação e a configuração desses blocos de *software* é feita através de comandos específicos que podem ser considerados a linguagem de uma central telefônica. Essa linguagem é denominada comandos homem-máquina (**CHM**).

Assim como cada central possui seu próprio conjunto de blocos de *software*, que varia de fabricante para fabricante e até mesmo entre modelos diferentes de um mesmo fabricante, ela também possui comandos CHM totalmente diferenciados. Esses comandos se diferenciam em relação a sua sintaxe, semântica e resultado.

Contudo, se faz necessária a existência de algum tipo de interface para o envio desses comandos CHM para os blocos de *software* das centrais. Um mecanismo utilizado para a realização dessa tarefa é o terminal de vídeo de acesso. Terminais, são disponibilizados pelas próprias CPAs, como portas de entradas para o envio dos comandos que controlam, manipulam e configuram todo seu funcionamento. Outra função desses terminais é a de apresentar tanto os relatórios resultantes dos comandos CHM enviados, quanto aqueles emitidos automaticamente para indicar o estado de funcionamento da central.

A não padronização dos terminais de vídeo, com o uso de uma interface única, impossibilita o acesso a todas as centrais de uma planta de maneira uniforme, considerando que cada uma dessas centrais possui terminais proprietários. Desta forma, para haver um ponto central de acesso a todas as centrais da planta, é necessário dispor de terminais referentes aos diversos tipos de centrais presentes na rede monitorada.

Essa diversidade de tipos de terminais numa mesma planta de centrais, do ponto de vista de gerência de telecomunicações, é desaconselhável e apenas dificulta o gerenciamento da rede. Essa dificuldade pode ser comprovada pelos seguintes fatos: para cada tipo de central é necessário um treinamento específico para operar seu respectivo tipo de terminal (seu conjunto CHM) e entender as informações relatadas pelas centrais, como conseqüencia é necessário ter um ou mais operadores para cada tipo CPA. Um outro problema é que cada tipo de central precisa de um *hardware* específico (terminal de vídeo) para que se possa ter acesso aos seus terminais.

Para haver uma gerência de telecomunicações eficiente, um dos principais requisitos é dispor de um mecanismo de unificação de terminais para troca de mensagens entre os operadores e todos os elementos de rede e que, além disso, seja possível o acesso a qualquer um desses elementos de um ponto central da planta, por exemplo, uma estação de trabalho.

Numa primeira etapa, no ambiente proposto existe a preocupação em apenas resolver o problema da necessidade de haver um único ponto de acesso a todas as centrais que fazem parte da planta monitorada. A intenção é fornecer um mecanismo através do qual não sejam necessários diversos tipos de *hardware* para os diferentes tipos de terminais de acesso, mas apenas um *hardware* usando uma camada de *software* para emular os diversos tipos de terminais vídeo [Sant99a], [Sant99b].

A grande maioria das centrais digitais disponibilizam terminais padrão da categoria RS-232. É o caso das centrais NEAX, Trópico-R e Trópico-RA. Esses terminais são dispositivos constituídos de um teclado e um *display* que se comunicam com a unidade central de processamento usando uma interface serial, com conectores de 25 pinos [Bach86].

Para criar um único ponto de acesso às centrais da planta, de seus respectivos terminais de vídeo, o ambiente disponibiliza numa porta serial de cada central a ser gerenciada, um *modem* padrão *Hayes* associado a uma linha telefônica. Essa é uma linha convencional, com um número de assinante, cujo ponto de origem é a própria CPA, já que pertence à central, e com potencial para se conectar a qualquer outro ponto da planta telefônica. Por outro lado, na estação de gerência, que é o ponto único para o acesso às

centrais, também disponibilizam-se *modems* padrão *Hayes* associados a linhas telefônicas convencionais.

Desse modo, para se criar uma conexão entre a estação de gerência e uma central da planta é suficiente realizar uma chamada telefônica com origem na estação gerente e destino uma das centrais que se deseja ter acesso ao seu terminal de vídeo. Isso pode ser observado na Figura 3.3.

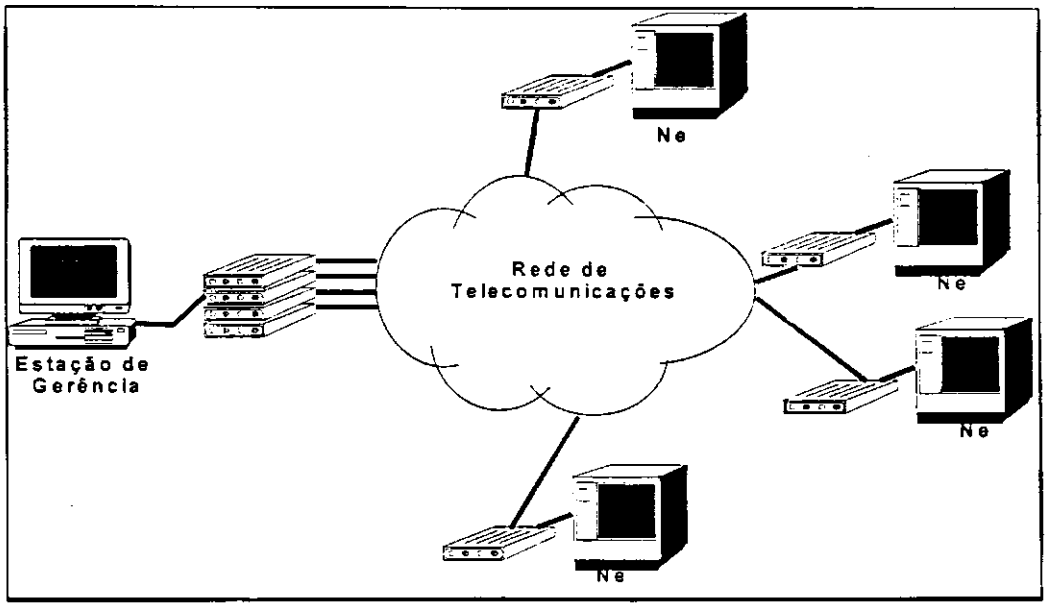

Figura 3.3 Acesso às centrais através de *modems*.

Além dos pontos de conexão entre as centrais e a estação de gerência (os *modems*) e o meio pelo o qual se faz esta conexão (a própria rede de telecomunicações) é necessário uma camada de *software* com potencial para emular os diversos tipos de terminais de vídeo na estação gerente.

A camada de *software* responsável pela emulação é constituída de alguns aplicativos do próprio sistema operacional (Unix HP-UX A.09.03) da plataforma de gerência, de aplicativos construídos para manipulação, configuração dos *modems*, e de alguns arquivos de configuração. Os aplicativos utilizados do sistema operacional foram: o ***cu*** e o ***Xterm*** [Bach86] do Unix.

O ***cu*** é um *software* cuja função é chamar outros sistemas e tentar conectá-los ao seu sistema hospedeiro. Os sistemas chamados podem ser outros sistemas Unix ou não, assim

como também pode ser um determinado terminal. O *cu* gerencia toda interação entre os sistemas conectados, incluindo possíveis transferências de arquivos ASCII [Pack].

Já o *Xterm* é um emulador de terminais de vídeo para sistemas *X Window* (padrão de interface gráfica para sistemas Unix). Ele provê terminais compatíveis para sistemas que não podem usar diretamente o padrão *X Window*.

3.4 Conversores

A realização de uma gerência de rede eficiente depende muito da origem e da confiabilidade dos dados de gerência que são coletados dos objetos a serem monitorados. De maneira que esses dados, depois de processados, possam fornecer informações gerenciais verídicas a respeito do estado da planta de telefonia.

No ambiente GCR, os dados de gerência coletados são referentes às centrais digitais a serem monitoradas. Esses dados estão armazenados em memórias, quer sejam voláteis ou não, das próprias centrais que para serem consultados ou manipulados é necessário ativar um bloco de *software*, através de comandos CHM específicos da central, que possibilitem essas operações.

Pelo caráter multifornecedor da planta de telecomunicações e a inexistência de implementação de padrões para os atuais elementos de rede que compõem essa planta, tanto o conteúdo quanto a forma de armazenamento dos dados gerenciais diferem entre os diversos tipos de centrais existentes. Ou seja, cada fornecedor implementa seu próprio bloco de *software* responsável pelo armazenamento, acesso e configuração dos dados gerenciais de suas centrais digitais.

Dessa forma, para se conseguir o conteúdo dos dados gerenciais das diversas centrais da planta é explícita a necessidade de se conhecer os comandos CHM de cada central, para se ativar o respectivo bloco de *software* responsável pela manipulação dos dados gerenciais. No GCR, essa é uma das principais funções do módulo **conversores**. Para cada tipo de central a ser gerenciada pelo o ambiente existe um conversor específico que conhece o conjunto de comandos CHM específico e necessário para manipulação e coleta, dos dados gerenciais daquele tipo de central [Sant99a], [Sant99b].

É por meio dos conversores que se viabiliza a comunicação e interação entre os elementos de rede, no caso as centrais, e o Sistema de Operação (OS) do ambiente de gerência. Pode-se afirmar que os conversores funcionam como mediadores entre as CPAs e o ambiente [Sala94].

Aos conversores estão associadas outras funções que auxiliam os mesmos a manipularem os dados de gerência e de funcionarem como mediadores, são elas: filtragem dos relatórios emitidos pelas centrais, para extração dos dados de gerência; envio periódico de comandos CHM de maneira automática; mecanismos para recuperação de erros decorrentes de ruídos quando no envio dos relatórios para os conversores; armazenamento dos dados coletados. Assim, pode-se definir os conversores como uma camada de software que desempenha todas essas funções acima citadas [Sant99a]. [Sant99b].

A Figura 3.4 esboça a arquitetura funcional de um conversor genérico, apresentando os seus sub-módulos.

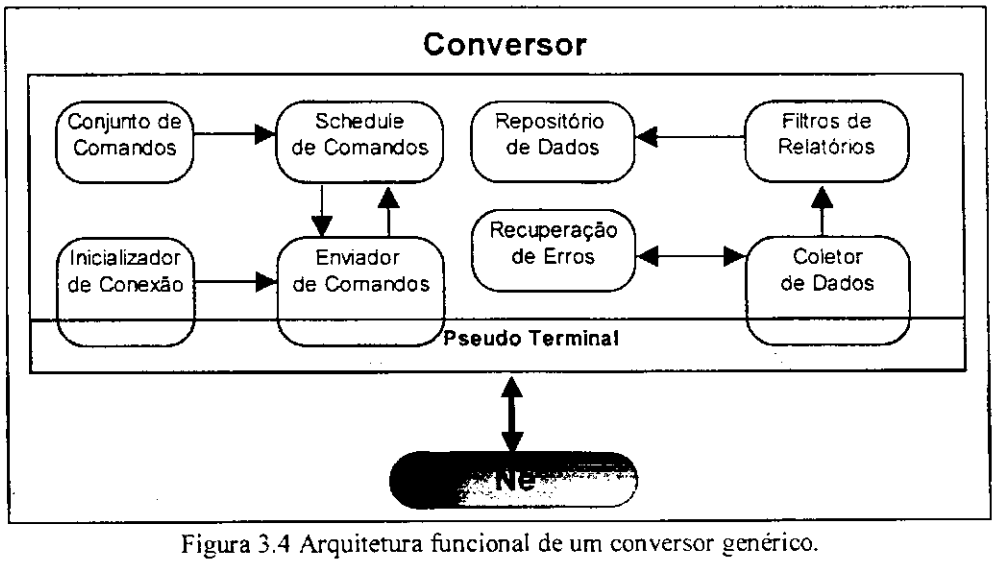

Figura 3.4 Arquitetura funcional de um conversor genérico.

3.4.1 *Conexão entre o Elemento de Rede e o Ambiente*

Para que a camada de *software* conversores realize seu papel de mediador na interação entre os elementos de rede gerenciados e o ambiente, é preciso algum mecanismo que possibilite essa comunicação.

Existem diversas formas para interconectar um elemento de rede a um Sistema de Operação:

• Pelo modelo TMN, através de interfaces q_3/q_x que sejam disponibilizadas pelos fabricantes daquela determinada central. As interfaces q_3/q_x servem de adaptadores cuja função é traduzir uma interface não-TMN em uma interface TMN [Sala94];

• Através do conhecimento do protocolo proprietário de comunicação da CPA, conectando-se a uma porta serial da própria central;

• Implementando-se uma pilha de um protocolo de comunicação para a plataforma operacional da central a ser gerenciada. Por exemplo uma pilha TCP/IP;

• Através de acesso via linha discada convencional.

No ambiente GCR, a comunicação é disponibilizada através de linhas convencionais discadas, visto que as outras formas de interconexão são inviáveis para a realidade do ambiente, pelas seguintes razões:

- Interfaces q_3/q_x para os modelos de centrais a serem monitoradas não estão disponíveis, seu desenvolvimento é bastante complexo e dispendioso, além de que o enfoque do ambiente é uma solução não TMN;
- Implementar uma pilha TCP/IP para as centrais não seria justificável, já que seria preciso conhecer a fundo toda plataforma operacional de cada uma das centrais;
- Fazer uso dos protocolos proprietários, do ponto de vista burocrático é muito difícil, visto que os fornecedores mantêm seus protocolos de comunicação sigilosamente guardados como forma de defender e garantir a rentabilidade de seu negócio.

Semelhante ao mecanismo de emulação de terminais, exposto anteriormente, é associado a cada central um *modem* dedicado que faz uso de uma linha convencional da própria CPA, como ponto de conexão entre os conversores e o elemento de rede. Do outro lado, para cada conversor é disponibilizado um outro *modem*, associado também a uma linha convencional, que serve como ponto de origem para conexão.

A comunicação ocorre quando o conversor de uma dada central entra em execução. Assim como na emulação de terminais, o conversor utiliza o aplicativo *cu* existente no sistema operacional Unix, juntamente com alguns programas e *scripts* específicos para realização do acesso discado. O conversor disca para o *modem* de sua respectiva central, e

após a negociação entre os *modems* (velocidade de transmissão, número de *bits*, paridade), a central disponibiliza para o conversor um terminal de acesso.

Do lado conversor, não existe qualquer emulador de terminais ou *display* para apresentar as informações enviadas pela central, contudo essas informações são tratadas pelo conversor, de maneira que a central não tem conhecimento da inexistência de um *display*. O conversor, na verdade, filtra os caracteres enviados pela central até que ele identifique alguma marca ou caractere que indique que a central está pronta para receber comandos CHM conhecidos. O conversor passa então, a funcionar como um simulador de ações de um operador da central.

3.4.2 Abertura de Sessão num Elemento de Rede

Efetivada a conexão entre o conversor e o elemento de rede, ainda não se deve considerar que a CPA já se encontra num estado pronto para receber comandos CHM, pois alguns tipos de CPAs necessitam que antes seja criada uma sessão de acesso.

As centrais digitais possuem blocos de *software* que implementam mecanismos de segurança tanto para cadastrar as entidades (pessoa ou grupo) que devem ter acesso ao elemento, como também controlar quais entidades devem ou não ter acesso a determinados recursos. A criação de uma sessão de acesso é o procedimento através do qual se verifica se uma determinada entidade tem acesso ao elemento de rede. Seria o equivalente a efetivação de um *login* em um ambiente computacional. As ações necessárias para criação de uma sessão diferem de central para central, assim cada tipo de conversor deve saber quais ações devem simular para criar uma sessão de acesso no seu respectivo tipo de central.

Além de ter inteligência para simular as ações necessárias para a abertura de uma sessão, os conversores precisam ter ainda informações sobre uma entidade que tenha acesso à central. Essa entidade deve possuir privilégios (autorização) para utilizar os recursos do elemento de rede responsáveis pela manipulação dos seus dados de gerência.

Para tornar mais claro o que significa a criação de sessão de acesso, toma-se como exemplo uma central do tipo Trópico-RA. Neste tipo de central para se abrir uma sessão é necessário informar primeiramente um *login*, que identifica um usuário para a central, e em seguida uma senha de autenticação para aquele usuário. Assim, o conversor para centrais

Trópico-RA tem inteligência para conceder essas informações, nessa ordem, simulando as ações de um operador que deseja interagir com este tipo de elemento de rede.

3.4.3 *Envio de Comandos Homem-Máquina*

Depois de criada a sessão de acesso, o elemento de rede se encontra num estado em que a interação entre os conversores e o elemento pode ser realmente estabelecida, por meio da linguagem reconhecida pela central, o seu conjunto de comandos CHM.

Para o ambiente de gerência, a interação entre as centrais e os conversores tem como objetivo obter o máximo de dados de gerência de cada uma das centrais, levando em consideração a regularidade dos intervalos entre os períodos de coleta e a confiabilidade dos dados. Em função disso, os comandos CHM a serem manipulados e enviados pelos conversores às centrais possuem características específicas voltadas para a ativação dos blocos de *software* responsáveis pelos dados de gerência de cada uma das centrais.

A diversidade entre os elementos de rede a serem monitorados, traz como conseqüência a existência de diferentes linguagens para cada um desses elementos. Desta maneira, o conversor de cada tipo de central precisa, necessariamente, saber qual ou quais comandos CHM, com seus devidos parâmetros, devem ser transmitidos às centrais para que essas possam responder e emitir aos seus respectivos conversores os dados questionados. Assim, os conversores funcionam como operadores das centrais, questionando-as de maneira automatizada e periódica, sobre o seu atual *status* com relação a seus indicadores de desempenho e sua taxa de utilização de órgãos. Depois de recebidos os dados, os conversores os filtram obtendo apenas os dados de gerência relevantes para sua monitoração e os armazenam em um estrutura de dados bem definida, que será explicada mais a frente.

Dependendo do tipo da central, cada uma delas tem um período mínimo para atualizar os seus dados internos, quer sejam indicadores de desempenho quer sejam indicadores de taxa de utilização de órgãos. Pois, esses indicadores são manipulados pelos blocos de *software* da central que possuem seus próprios mecanismos de atualização. Esse período mínimo de alteração dos dados internos também depende do tipo de comando

CHM que está sendo executado na central, visto que cada comando pode ativar blocos de *software* diferentes, cujos intervalos de atualização de dados também são variados.

São essas discrepâncias entre os comandos a serem dados para uma central que determinam quais os intervalos de tempo entre o envio de um tipo de comando e o seu próximo reenvio. Isto significa que o envio do mesmo comando, sem levar em consideração o tempo necessário para que a central atualize os seus contadores, implicará que os dados que serão coletados não trarão informação adicional, pois são idênticos aos armazenados na última coleta.

Apesar de existirem práticas e normas da Telebrás [SDT501-100-105], [SDT501-100-107] que especificam a frequência ideal para coleta de informações de gerência, na prática e no ambiente de gerência proposto isso não é utilizado. Cada equipamento possui seu período próprio de atualização, imposto por sua arquitetura proprietária de *software*. No GCR os conversores têm sua frequência de envio de comados para realização de coletas periódicas, em função da frequência de atualização que as centrais utilizam sobre seus indicadores.

Cada conversor possui então, seu próprio conjunto de comandos CHM, seu próprio *schedule* de comandos e um enviador de comandos. A Figura 3.5 mostra a sistemática de envio de comandos ao elemento de rede.

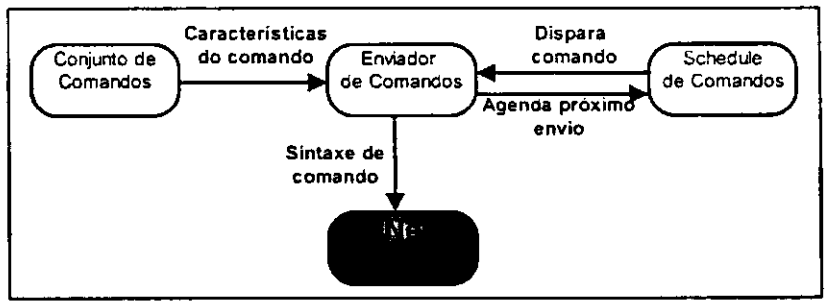

Figura 3.5 Envio de comandos ao elemento de rede.

O conversor para cada comando a ser utilizado, sabe qual a sua sintaxe, quais os seus parâmetros necessários, sua dependência a algum outro comando e qual a sua frequência de envio. Estas informações são guardadas pelos conversores nos seus conjuntos de comandos CHM a serem utilizados.

O *schedule* de comandos é o mecanismo que garante a periodicidade da coleta dos dados de gerência para todos os comandos utilizados pelo conversor. É o *schedule* que, após o envio de um determinado comando, agenda-o para ser reenviado pelo conversor em função da frequência de coleta daquele comando.

O enviador de comandos é responsável por realizar o envio propriamente dito do comando. É nesse módulo que se organiza toda a sintaxe do comando e o envia através da conexão, via linha discada convencional, criada entre o conversor e o elemento de rede.

3.4.4 Coleta dos Relatórios

Após o envio de um determinado comando à central por seu conversor, é necessário esperar algum *feed-back* da própria central referente ao comando enviado. É função dos conversores coletar essa informação de *feed-back* dando-lhe os seus devidos tratamentos.

Assim como é realizado o envio dos comandos, o *feed-back* de resposta das centrais aos conversores é feito através da conexão criada a partir das linhas convencionais. Os conversores têm como canal de *input* para essa conexão e são alimentados com as informações geradas pelos *feed-backs* de suas centrais.

Caso o comando tenha sido enviado com sucesso, levando em consideração sua sintaxe, seus parâmetros e sua freqüência de envio, o seu *feed-back* será um relatório emitido pelo bloco de *software* ativado, contendo as informações de gerência interrogadas, num formato proprietário da central. Além dos dados de gerência, esses relatórios possuem outras informações de controle da própria central, tais como: identificadores numéricos, descrição do relatório, delimitadores de blocos de informações. Ou seja, os dados de gerência desejados estão misturados a outras informações que não são relevantes para a realização da gerência da rede.

Em virtude dessa mistura de dados é que advém a necessidade dos filtros de dados. Para cada comando de cada tipo de central existe um relatório específico com suas próprias particularidades: palavras-chaves, identificadores de tipo de relatório, descrição do relatório, caracteres de controle, marcas de blocos de dados. Assim, para cada relatório emitido pela central como resposta ao envio de um dado comando, os conversores precisam

ter um filtro específico para extrair desse relatório apenas os dados relevantes para realização da gerência da planta de telefonia.

Os relatórios são emitidos pelas centrais aos conversores caractere a caractere de maneira que, para que se possa entender as informações existentes nesses relatórios, é necessário se realizar uma análise sequêncial desse conjunto de caracteres. Para tal tarefa os conversores utilizam um analisador léxico (*scanning*), um analisador sintático (*parser*) e uma camada de *software* adicional para extração dos dados específicos denominada tratador de relatório [Sant99b], como mostrado na Figura 3.6.

Para o GCR, um analisador léxico ou *scanning* pode ser definido como um bloco de *software* cuja função é ler uma tira contínua de caracteres sem significado aparente, da esquerda para direita, agrupando-os em *tokens* que também são seqüências de caracteres, porém em conjuntos definidos que possuem um significado.

O *scanning* dos conversores agrupa os caracteres, em três tipos de *tokens*:

Palavras - constituídas de caracteres pertencentes a faixa de **'a'** a **'z'** e de **'A'** a **'Z'**. Ex: Rota, DTR, IFALCE.

Dígitos - constituídos de caracteres pertencentes a faixa de **'0'** a **'9'**. Ex: 635.

Símbolos - constituídos de caracteres que não se encaixam nem na faixa de palavras nem na de dígitos. Ex: '>', '=', '.'(ponto), '*', '#'.

Já um analisador sintático ou *parser* é também um bloco de *software* que tendo um conjunto de *tokens* como entrada, tenta agrupá-los em frases que tenham algum significado.

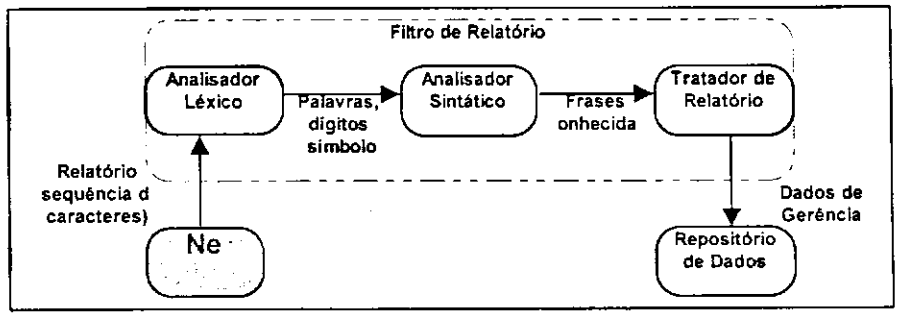

Figura 3.6 Mecanismo de leitura de relatórios

No GCR, para cada comando existe uma analisador sintático específico, que tenta agrupar os *tokens* originados do analisador léxico em frases com significados bem

conhecidos. Essas frases, na realidade, são em conjunto o próprio relatório da central. Contudo, nesse ponto os conversores já conhecem o significado desse relatório, já que conhecem o significado de cada uma de suas frases.

Nessas frases além dos dados de gerência ainda se encontram misturadas outras informações não relevantes. Para extrair dessas frases apenas os dados importantes para gerência, os filtros fazem uso de uma camada de *software* específica para cada comando. É nesta fase que o tratador de relatório entra em operação.

Internamente a camada de tratamento de relatório possui inteligência para identificar quais são os *tokens* das frases analisadas que podem ser considerados dados relevantes para gerência. Depois de identificados, esta camada transforma tais *tokens* em dados de gerência propriamente ditos. Veja o exemplo com as frases da Figura 3.7.

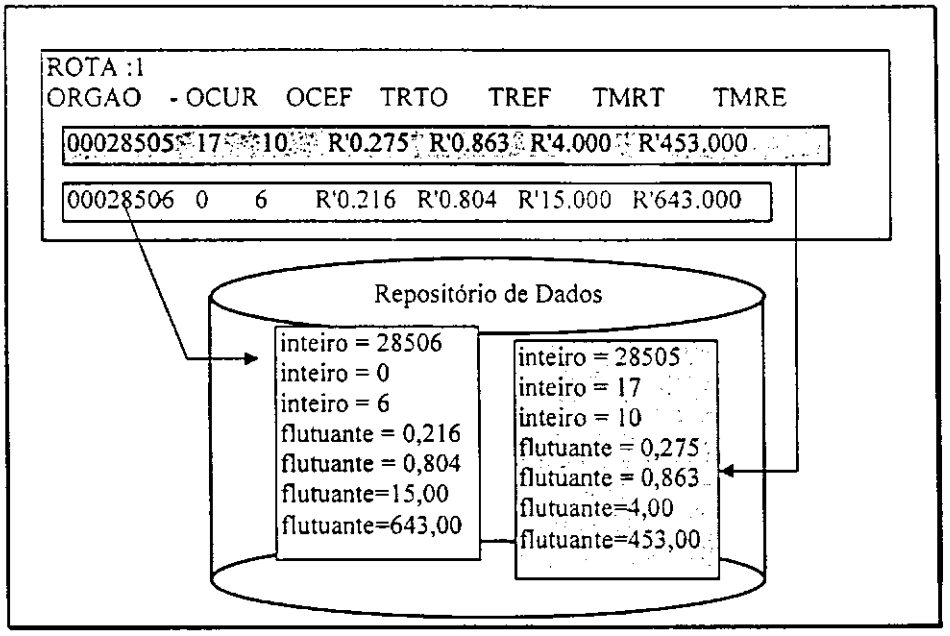

Figura 3.7 Exemplo de relatório e extração de dados de gerência.

Uma outra função importante da camada de tratamento de relatório é a de recuperação de erros. Por serem emitidos pelas centrais através de uma linha discada convencional, não é muito difícil ocorrer erros na transmissão dos caracteres que compõem um relatório. Tais erros podem ser gerados, por exemplo, por ruídos na linha serial ou por perda de sincronização entre os *modems*.

O mecanismo de recuperação baseia-se em um ou mais pontos de referência dentro de um determinado relatório. Esses pontos são *tokens* ou frases cuja semântica indica alguma marca especial do relatório, são eles:

- marca de início ou de fim do relatório;
- marca de início ou fim de blocos de dados;
- símbolos especiais ou *tokens* pré-definidos - Estes são definidos como marcas, na fase de projeto da camada de tratamento para um determinado tipo de relatório.

Internamente, a camada de tratamento de relatório de cada comando possui conhecimento de suas marcas especiais, e baseado nesse conhecimento é que essa camada consegue, na ocasião de um erro ou de um estado anormal da transmissão, se recuperar e voltar a um estado considerado normal.

O mecanismo de recuperação ocorre da seguinte forma: se o *token* transmitido pela central ou a frase formada pela seqüência de *tokens* enviados não é o esperado pelo conversor, o mecanismo de recuperação ignora o *token* ou a frase lida e os seguidos *tokens* e frases até que uma marca especial, definida para aquele tipo de comando, seja enviada pela central e consequentemente seja identificada pela a camada de tratamento de relatório. Quando identificada alguma marca, o conversor passa a desempenhar suas funções de maneira normal.

Essa camada traz como benefício uma maior confiabilidade na veracidade dos dados. Contudo, apesar de diminuir a perda dos dados de gerência, esse mecanismo não impede que se houver um erro no meio da construção de uma frase que contenha dados de gerência, esses sejam perdidos visto, que serão ignorados.

A perda se dá em função da quantidade das marcas especiais para um dado tipo de relatório, da quantidade de dados existentes no relatório e do formato do relatório em si. Por exemplo: para um relatório que tenha um bloco de dados com apenas três frases com dados de gerência e com marcas especiais entre cada frase, acontecendo um erro de transmissão no meio de uma frase, só acarretará o prejuízo a 33% dos dados coletados.

Contudo, se o mesmo relatório contiver apenas marcas para o início e o fim do relatório, as perdas podem chegar a 100% do relatório.

3.4.5 *Falta de Comunicação entre o Elemento de Rede e os Conversores*

Apesar de serem equipamentos projetados para oferecerem um alto grau de confiabilidade e de disponibilidade [Alen98], mesmo assim as centrais estão sujeitas a faltas, quer seja de um equipamento, de um conjunto de equipamentos, de um bloco de *software*, ou até mesmo da central como um todo. Uma falta qualquer pode fazer com que a central pare de receber comandos CHM do conversor e de enviar relatórios como resposta. Então, é necessário ter algum mecanismo para se detectar se existe realmente comunicação entre os conversores e suas respectivas centrais, já que não faz sentido manter um conversor em execução impossibilitado de obter dados de gerência do elemento de rede.

O mecanismo é baseado em um tempo máximo estimado no qual não tenha sido realizada qualquer transmissão de um caractere qualquer da central para o conversor, é o Tempo Máximo Estimado Sem Transmissão (TMEST) [Sant99b].

A estimativa desse tempo máximo é fundamentada em testes e medições realizados com cada uma das centrais a serem gerenciadas, levando em consideração os seguintes aspectos: o tempo máximo que a central mantém sua sessão de acesso ativa sem que haja interação entre ela e o seu respectivo conversor, o tempo necessário para a central transmitir o seu maior relatório para o conversor, a maior diferença de tempo entre os intervalos de envio dos comandos CHM utilizados por seu conversor.

Todo conversor tem um acumulador geral de tempo de leitura que é o responsável por monitorar há quanto tempo a central não transmite qualquer tipo de caractere ao conversor. A cada caractere a ser lido pelo analisador léxico, um tempo máximo para a leitura daquele caractere é definido, denominado Tempo Máximo para Leitura de um Caractere (TMLC). Se durante esse tempo nenhum caractere for lido pelo conversor, o TMLC é adicionado ao acumulador geral de leitura e se verifica se o tempo acumulado é maior que o TMEST. Caso esse tempo máximo tenha sido alcançado, o conversor assume que a comunicação entra ele sua central foi quebrada e essa informação é então repassada

para as outras camadas de mais alto nível do ambiente, até chegar ao conhecimento do operador do sistema.

Para todo comando CHM de um determinado conversor é definido um TMLC específico. Isto decorre do fato de não existir uma taxa de transmissão constante de caracteres da central para o seu conversor, pois a construção e o envio de um determinado relatório por parte de uma central dispara a execução de seu respectivo bloco de *software*, cujo tempo de processamento de toda tarefa varia entre os diversos tipos de comandos CHM. O TMLC é estimado tomando-se como base o maior *delay* (atraso) de intervalo de tempo entre a transmissão de dois caracteres consecutivos de um relatório.

Assim, estipulando-se um TMLC para cada tipo de comando enviado à central, pode-se garantir uma maior segurança na identificação de falta de comunicação entre a central e o seu conversor.

A maioria da centrais digitais desativam suas sessões de acesso caso essas permaneçam um determinado período de tempo sem serem utilizadas, ou seja, elas possuem um temporizador para suas sessões ativas não utilizadas. Esse tempo é variado de central para central e como cada comando tem sua própria freqüência de envio para garantir a periodicidade dos dados coletados, pode acontecer casos em que o intervalo de tempo para o envio do próximo comando CHM seja maior que o tempo suportado pela central, sem que essa desative a sessão vigente. Para evitar que sua sessão de acesso seja desativada, todo conversor faz uso de um comando CHM nulo, cuja freqüência de envio é menor que o período suportado pelo temporizador, além de que seu envio à central não faz com que esta emita qualquer tipo de relatório e que apenas realize um *refresh* na temporização da sessão ativa.

3.4.6 Estrutura Intermediária de Armazenamento

Após o processo de envio de comando CHM à central, da leitura do seu devido relatório e da extração dos dados de gerência por parte do conversor, é necessário dar algum destino a esses dados coletados. É ainda trabalho do conversor armazenar esses dados em algum lugar e de alguma forma.

Para cada tipo de relatório é extraído um tipo de dado necessário para realização da gerência de rede. Assim, associada a cada comando manipulado por um conversor, tem-se uma estrutura de dados sobre a qual é feito o armazenamento para aquele tipo de dado extraído do relatório. Muitas vezes, um mesmo relatório traz consigo não só um único bloco de dados, mas sim um conjunto de blocos de dados de um mesmo tipo. Nesse caso, faz-se uso de uma lista encadeada cujos nodos (de um mesmo tipo) guardam o conteúdo dos blocos de dados extraídos de um mesmo relatório. Em função deste fato, o GCR utiliza no conversor uma estrutura de dados que é um conjunto de listas encadeadas para armazenar os conjuntos de dados extraídos de cada um dos relatórios.

Essa estrutura de dados é uma estrutura de armazenamento intermediária, de maneira que esses dados nunca são consultados, são apenas atualizados em função da periodicidade do envio dos comandos CHM e usados como fonte alimentadora para a estrutura de dados principal, que é a base de informações de gerência - MIB.

3.5 A MIB

A MIB é um repositório de informações onde estão armazenados os objetos de gerência que são abstrações, para o mundo computacional, dos elementos de rede gerenciados. Esses objetos em conjunto, formam o modelo de informação do ambiente GCR. Em função dos tipos de supervisão a ser realizada sob os elementos e dos dados de gerência coletados, foi construído um modelo de informação genérico para a realização de gerência de desempenho.

Os principais objetos armazenados na MIB do GCR são apresentados a seguir. É usada a notação ASN1 (*Abstract Syntax Notation*) [Stal93] para representar os objetos.

3.5.1 Objetos Utilizados na Supervisão de Código

O objeto **DadosCodigoEntry** contém as informações referentes aos valores dos indicadores de completamento de chamadas. Através desse objeto pode-se identificar qual a taxa de aproveitamento das chamadas, assim como taxa de Perda no Assinante B (PAB) para uma determinada direção de tráfego de um dado elemento.

```
DadosCodigoEntry ::= SEQUENCE {
        linhaTabelaIndex        INTEGER, --índice da instância do objeto
```

```
        dtr             OCTET STRING, --DTR. Um par (Origem - Destino)
        ok              INTEGER, --Chamadas completadas com sucesso
        lo              INTEGER, --Chamadas incompletas com linha ocupada
        co0             INTEGER, --Incompletas por congestionamento na rede
        co1             INTEGER, --Incompletas por congestionamento na rede
        co2             INTEGER, --Incompletas por congestionamento na rede
        co3             INTEGER, --Incompletas por congestionamento na rede
        nr              INTEGER, --Chamadas não atendidas pelo destino
        dsc             INTEGER, --Chamadas desconectadas na origem
        ou              INTEGER, --Incompletas por outras causas especiais
        horaModif       TimeTicks --TimeStamp de controle
}
```

O objeto **DadosChamadaEntry** contém as informações referentes aos valores de uma chamada efetuada por um elemento. A partir desse objeto pode-se também extrair a taxa de aproveitamento das chamadas, assim como taxa de PAB para uma determinada direção de tráfego.

```
DadosChamadaEntry ::= SEQUENCE {
        linhaTabelaIndex INTEGER, --Índice da instância do objeto
        num-A           OCTET STRING, --Número do assinante de origem
        num-B           OCTET STRING, --Número do assinante de destino
        fds             INTEGER, -- Fim de seleção, i.e. status da chamada
        horaChamada     OCTET STRING, --Tempo de duração da chamada
        horaModif       TimeTicks --TimeStamp de controle
}
```

3.5.2 Objetos Utilizados na Supervisão de Órgãos

Objeto **DadosTrafJuntEntry** armazena informações referentes à taxa de utilização de um determinado juntor pertencente a uma determinada rota, dentre as quais: o tráfego efetivo, tráfego total, ocupação efetiva e ocupação total. Com esse objeto pode-se identificar juntores defeituosos (e.g. *killers* - são juntores que sempre são ativados, porém nunca completam a comutação de uma chamada; ou juntores que estão presos a uma comutação não mais existente) e rotas com dimensionamentos incorretos.

```
DadosTrafJuntEntry ::= SEQUENCE {
        linhaTabelaIndex INTEGER, --Índice da instância do objeto
        idJuntor        INTEGER, --Identificador do juntor
        idRota          OCTET STRING, --Rota a qual pertence o juntor
        trafEfet        OCTET STRING, --Tráfego efetivo
        trafTot         OCTET STRING, --Tráfego total
        ocpEfet         INTEGER, -- Número de ocupações efetivas
        ocpTot          INTEGER, -- Número de ocupações total
        horaModif       TimeTicks --TimeStamp de controle
}
```

Objeto **DadosTrafEnvEntry** armazena informações referentes a taxa de utilização de um outro órgão pertencente a uma central, os enviadores. De posse dessas informações pode-se detectar a existência de enviadores defeituosos.

```
DadosTrafEnvEntry ::= SEQUENCE {
        linhaTabelaIndex INTEGER, --Índice da instância do objeto
```

3.6 Agentes

No conceito de gerência de redes de computadores pode-se afirmar que os agentes são entidades passivas responsáveis por atender às solicitações da entidade ativa, o gerente, acerca de objetos existentes numa base de gerência. Eles eventualmente podem enviar interrupções abruptas ao gerente com o objetivo de avisar a respeito de eventos inesperados ocorridos na rede monitorada.

No caso do GCR, os agentes são responsáveis por fornecer ao gerente as informações de gerência coletadas das centrais e armazenadas pelos conversores na MIB definida para o GCR.

Assim como se tem um tipo de conversor para cada tipo de central monitorada, existe também um agente específico para cada tipo de conversor. Os agentes oferecem ao gerente pontos de acesso para que esses obtenham os valores dos diversos elementos da MIB.

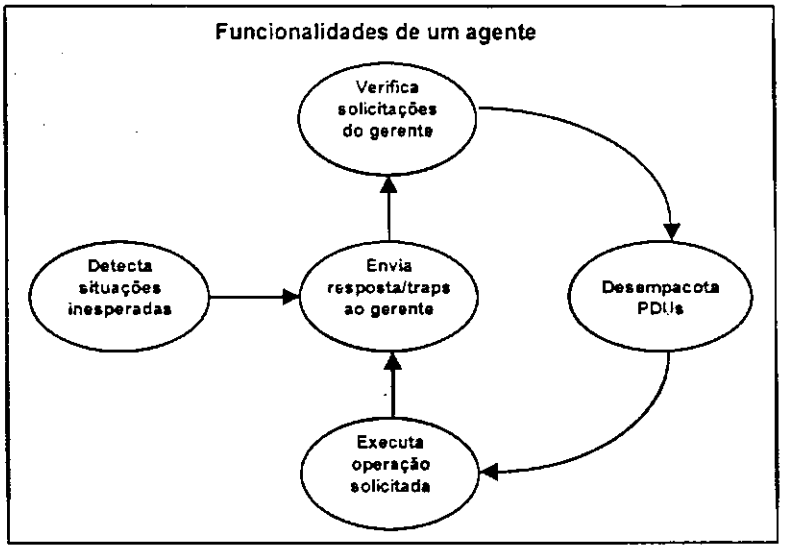

Figura 3.10 Funcionalidades desempenhadas por um agente do ambiente GCR.

De maneira geral, a funcionalidade dos agentes implementados para o ambiente GCR é apresentada na Figura 3.10

Quando em execução, um agente sempre permanece num estado de espera cujo funcionamento é semelhante a de um *daemon* [Tane92]. Para se manter nesse estado de espera o agente fica associado a uma porta do sistema operacional sobre o qual está em execução, até que algum pacote SNMP chegue a essa porta. Assim como os vários serviços

da família do protocolo TCP/IP, como **ftp** e **telnet** por exemplo, que estão associados a portas bem definidas, por padronização do protocolo SNMP a porta através da qual os agentes devem atender as solicitações de qualquer gerente é a 161 [Stal93].

No estado de espera o agente permanece verificando se existe alguma primitiva SNMP de solicitação originada pelo gerente na porta em que ele se encontra associado. Permanece em espera, exceto no caso de algum evento inesperado ocorrer na rede gerenciada e ser necessário que o agente envie ao gerente uma interrupção abrupta, os chamados *traps*. Ao receber um **PDU** (***Packet Data Unit***) SNMP, o agente precisa tratar esse pacote no sentido de identificar qual o tipo de operação está sendo solicitada pelo gerente, e sobre quais objetos da MIB esta operação deverá ser realizada. Essas informações estão contidas dentro do pacote PDU recebido pelo agente e é função desse desempacotar essa PDU para obter as informações.

Um ponto importante a ser levantado é o da validação das solicitações. Dentre as várias informações existentes numa PDU, a informação chamada de ***community name*** é utilizada pelo protocolo SNMPv1 para autenticar as solicitações realizadas pelo gerente [Stal93]. O conteúdo do *community name* é uma seqüência de caracteres alfanuméricos definida localmente em cada agente, de maneira que o gerente que realizar alguma solicitação, só será atendido caso o *community name* de suas PDUs seja válido. Assim, um gerente só poderá ter acesso aos dados da MIB por meio de um agente caso ele tenha conhecimento do *community name* do agente solicitado. No caso do GCR, apenas o gerente que tenha conhecimento do *community name* usado, tem permissão para consultar e modificar os valores dos diversos objetos que fazem parte da MIB do GCR.

Após a validação da solicitação e do desempacotamento da PDU o agente executa a operação solicitada sobre os objetos interrogados. Os objetos podem ser qualquer um dos existentes na MIB e a operação qualquer uma das seguintes primitivas do SNMP: *get*, *getnext*, ou *set*.

A semântica das operações é facilmente identificada pelo agente, visto que cada uma delas é representada internamente por um código único. Contudo, a localização dos dados na MIB é um pouco mais complicada, já que a organização lógica e a forma de

endereçamento dos objetos é hierárquica, enquanto que sua representação interna é um conjunto de listas encadeadas.

Como cada objeto na MIB é conceitualmente uma tabela, uma função de tradução foi utilizada para mapeá-los para um objeto que é representado internamente como uma lista encadeada. O comportamento dessa função é descrito a seguir:

- Qualquer coluna de uma tabela é mapeada para seu respectivo atributo de um nodo da lista que representa internamente o objeto;
- A *n*-ésima linha de uma tabela é mapeada para sua respectiva n-ésima posição na lista encadeada do objeto.

A execução da operação contida em uma PDU, por um agente, pode ser observada na Figura 3.11.

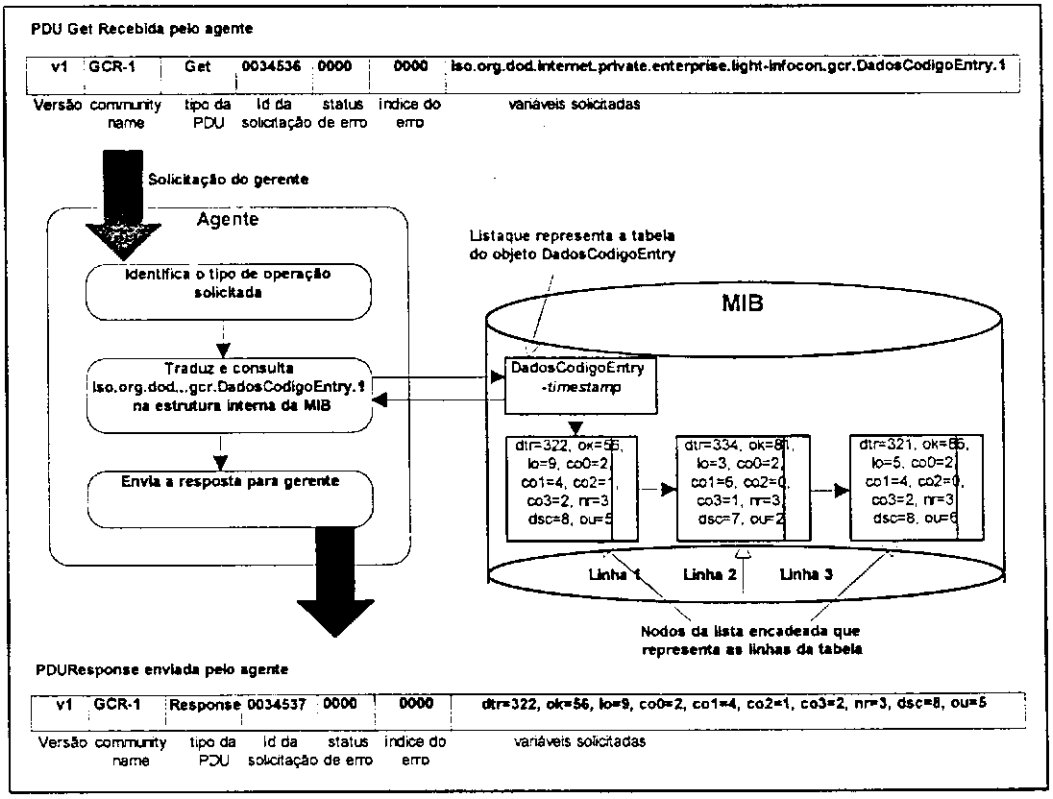

Figura 3.11 Execução de operação solicitada pelo gerente através de PDU Get.

De posse do resultado da operação executada sob os respectivos objetos da MIB, cabe ao agente enviá-lo de volta à entidade solicitante, o gerente. O resultado tanto pode ser

decorrente da operação ter sido executada com sucesso como também resultante de algum erro produzido pela operação.

Para enviar ao gerente quaisquer das respostas, o agente primeiro empacota numa PDU SNMP todas as informações necessárias para a resposta e a emite de volta através da mesma porta de conexão por onde o agente recebeu a solicitação a qual está associado.

Uma outra função importante dos agentes é de enviar mensagens abruptas, chamadas de *traps* ou interrupções, ao gerente. Esse tipo de mensagem tem a função de avisar ao gerente sobre eventos repentinos que acontecem na rede gerenciada e que precisam ser sanados rapidamente. No caso do GCR, existem *traps* para que os agentes avisem ao gerente que não existe mais comunicação entre a central e os conversores impossibilitando o funcionamento do ambiente, visto que como não mais existe comunicação, não há coleta periódica dos dados de gerência.

3.7 Gerente

O gerente é a entidade ativa numa plataforma de gerência distribuída, sendo responsável por realizar a coleta periódica dos dados contidos na MIB dos agentes ativos. Coletados, esses dados podem ser disponibilizados para as camadas de *softwares* superiores, para eventuais processamentos, ou mapeados para serem apresentados da melhor maneira possível aos usuários do sistema.

No GCR, o gerente tem conhecimento e se comunica com todos os agentes específicos e ativos associados às centrais a serem gerenciadas. Para coletar os dados e repassá-los à camada de aplicação, periodicamente o gerente solicita a cada um dos agentes informações a respeito de suas respectivas MIBs. Esse mecanismo é denominado de *polling*.

Assim como os agentes, o gerente utiliza as primitivas do protocolo SNMP para desempenhar as solicitações de coleta. A Figura 3.11 apresenta uma PDU enviada por um gerente a um agente.

Ainda para solicitar aos agentes os dados presentes na MIB, o gerente precisa saber de algumas informações: como endereçar e quais os objetos da MIB de cada um dos agentes dos quais solicita informações; quais são os agentes que estão disponíveis (ativos)

para solicitações; que objetos devem ser solicitados de um dado agente, com qual periodicidade deve ser realizado o *polling* sobre cada um dos agentes. A Figura 3.12 elenca os componentes do módulo gerente do GCR e suas interações.

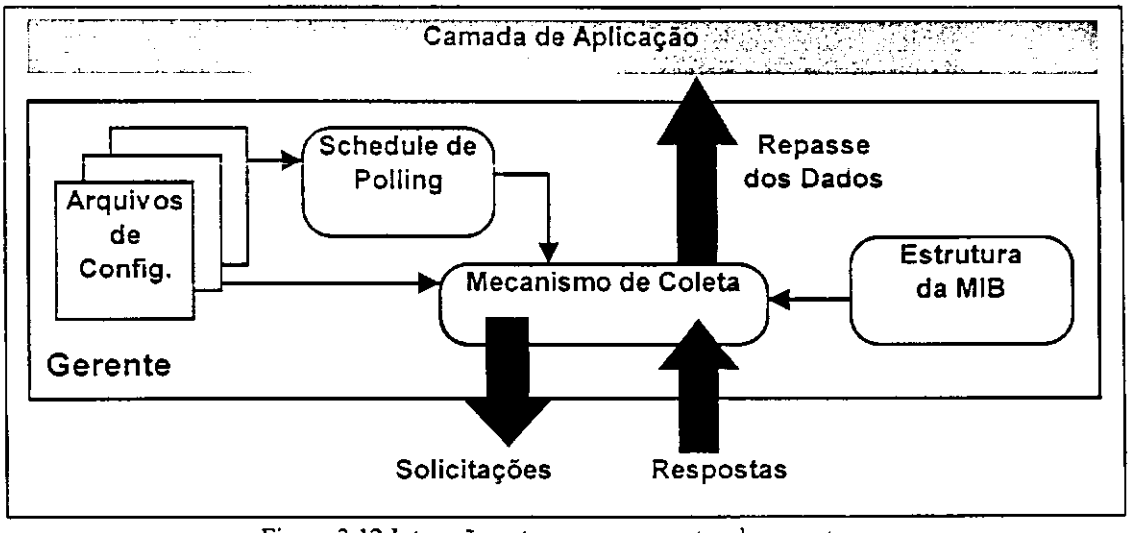

Figura 3.12 Interação entre os componentes do gerente.

O *Schedule de Polling* é o componente responsável por disparar periodicamente a solicitação de coleta para cada um dos agentes ativos. Após toda coleta dos dados contidos na MIB de um determinado agente ter sido realizada, é o próprio *schedule de polling* que reagenda a próxima solicitação de coleta. A freqüência com que o gerente deve realizar as solicitações de coleta, ou melhor, o *polling*, varia entre os diversos tipos de agentes

O intervalo de frequência é estimado em função da quantidade de objetos a serem questionados de cada tipo de agente e do tempo que é levado para a realização da coleta dos dados dos objetos. São nos arquivos de configuração de *polling* que se encontram as informações acerca de qual intervalo deve ser adotado para se realizar as solicitações de coleta consecutivas para um dado tipo de agente.

Existe um outro tipo de arquivo que é o de configuração de objetos que servem para especificar quais dos objetos contidos na MIB de um determinado tipo de agente devem ter seus valores solicitados pelo gerente a cada *polling*. Para cada tipo de agente existe um tipo de arquivo de configuração de objetos.

Esses arquivos de configuração oferecem uma flexibilidade ao ambiente GCR. O arquivo de configuração de *polling* possibilita dar prioridades em relação a quais tipos de

agentes devem ou não participar da coleta periódica realizada pelo gerente. Já o arquivo de configuração de objetos possibilita amenizar o tráfego de informações de gerência entre agentes e gerente, especificando apenas quais objetos devem ter seus valores coletados, ao invés de todos objetos que pertencem a MIB do agente questionado. Esse arquivo é muito útil para casos em que é necessário apenas obter informações acerca de um dado objeto específico.

Para ser capaz de coletar os valores de qualquer objeto de qualquer agente é necessário que o gerente tenha conhecimento sobre esses objetos. É no componente estrutura da MIB que se encontram todas as informações importantes à coleta, sobre todos os objetos que compõe a base de gerência do GCR, desde suas estruturas e sua organização lógica, até a localização de cada um dos elementos e dos atributos que o compõe.

O mecanismo de coleta é a camada de *software* utilizada para obter os valores dos objetos da MIB de qualquer agente monitorado. Tendo o *schedule de polling* disparado uma solicitação para um determinado tipo de agente, automaticamente ele dispara também o mecanismo de coleta para aquele tipo de agente. Esse mecanismo verifica a partir de informações contidas no arquivo de configuração de objetos, quais dos elementos da MIB daquele agente devem ter seus valores coletados. Em seguida, sequencialmente para cada um desses elementos o mecanismo busca no componente estrutura da MIB, a localização do elemento e de seus atributos, empacota-os em uma PDU de solicitação e a envia para o agente interrogado[Sant99a], [Sant99b].

Após o envio de uma solicitação qualquer, o gerente permanece num estado de espera até que uma resposta àquela solicitação chegue. De posse da resposta o gerente desempacota a PDU de resposta e repassa os valores dos dados solicitados à camada de aplicação que manipulará os dados de forma apropriada.

Os *traps* enviados abruptamente por um agente são desempacotados e suas informações recebidas também são repassadas à camada de aplicação que dará o seu devido tratamento. Não é função do gerente realizar qualquer ação em função das informações contidas nos *traps*.

3.8 Conclusão

Este Capítulo apresentou uma visão completa do ambiente de gerência desenvolvido neste trabalho, além de mostrar sua arquitetura física e funcional. Em seguida, para cada um dos componentes que fazem parte da arquitetura funcional do ambiente e que estão no escopo do trabalho, apresenta-se seu detalhamento descrevendo de forma suscinta as tarefas desempenhadas por cada componente, dependências entre as tarefas e interações entre os componentes.

É através dessa descrição funcional detalhada do ambiente que se verifica e conclui como essa solução desenvolvida realiza Gerência de Desempenho em Redes de Telefonia Heterogêneas.

4 IMPLEMENTAÇÃO

4.1 Introdução

Este capítulo apresenta em nível mais elevado, detalhes de implementação para os componentes de *software* dos módulos que fazem parte do ambiente GCR, cuja funcionalidade foi discutida no capítulo anterior: emulação de terminal, conversores, agentes e gerente.

O sistema foi modelado seguindo o paradigma de orientação à objetos, em que a modelagem oferece um melhor entendimento das necessidades, um projeto mais claro e um sistema fácil de evoluir e de manter. Para cada um dos módulos, é esboçado o conjunto de classes de objetos que o compõe, incluindo seus atributos e principais comportamentos.

4.2 Características Gerais da Implementação

A construção do ambiente baseou-se na metodologia de Engenharia de Software utilizada pela Light-Infocon Tecnologia S/A para o desenvolvimento de seus produtos de *software*. Essa metodologia atende ao modelo CMM (*Capability Maturity Model*) no nível 2/3 [Light97].

Para esboçar os componentes de *software*, agora denominados de objetos, dos módulos do GCR, faz-se uso da técnica de modelagem de objetos **OMT** (*Object Modeling Technique*) [OMT91]. OMT é uma metodologia independente de linguagem de programação e constituída de três modelos:

- Modelo Objeto – descreve a estrutura estática de um sistema por intermédio de objetos e relacionamentos correspondentes às entidades do mundo real;
- Modelo Dinâmico – descreve a estrutura de controle do sistema em termos de diagramas de evento/estado;
- Modelo Funcional – apresenta as transformações dos valores dos dados do sistema por meio de diagramas de funções.

Efetivamente, apenas o modelo objeto foi utilizado na modelagem do GCR, com o objetivo de apresentar a estrutura de classes para os quais os seus objetos são instâncias. A Figura 4.1 apresenta um resumo da notação gráfica do modelo objeto OMT.

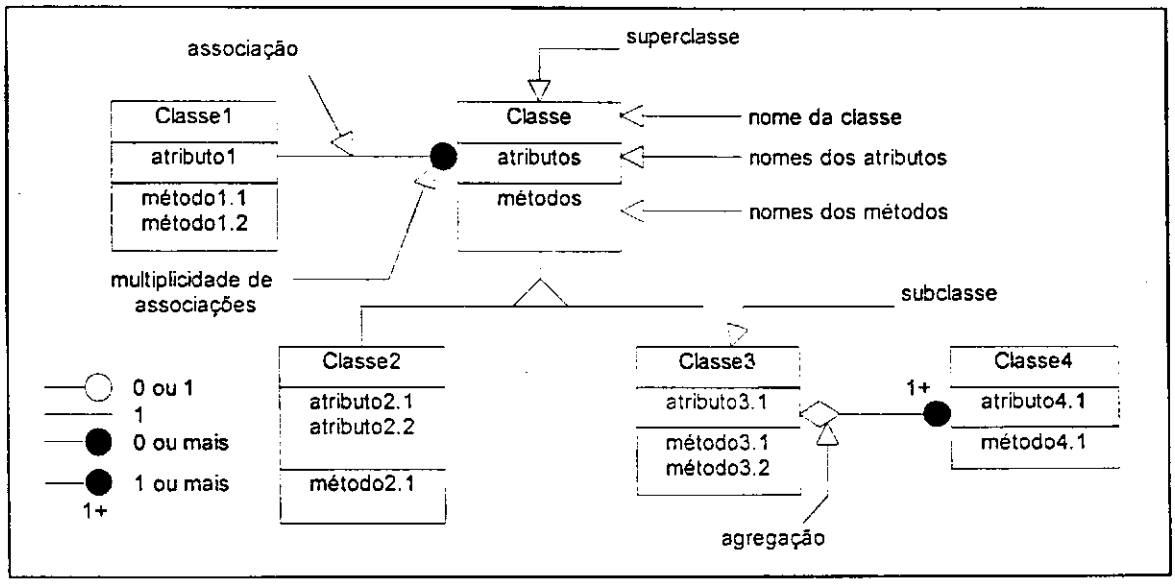

Figura 4.1 Resumo da notação OMT [Jato98].

Para cada um dos módulos (emulação de terminal, conversores, agentes e gerente) é apresentada sua estrutura hierárquica de classes. Alguns aspectos relativos à descrição das classes e determinados detalhes de implementação são também apresentados por serem mais relevantes e por possibilitarem sua utilização como referência em trabalhos futuros.

A implementação do ambiente foi realizado em fases visando atender os requisitos de modularidade citado no Capítulo 2, para a implantação de um Sistema de Operações para gerência de desempenho em redes de telefonia

A linguagem utilizada para o desenvolvimento dos módulos de *software* do ambiente foi C++ (orientada à objetos) padrão ANSI, e compilador gcc da GNU. Foram utilizados também módulos de *software* e bibliotecas para dinamizar e automatizar o desenvolvimento:

Templates de Listas Encadeadas - Encapsulamento de objetos cuja estrutura de dados são listas encadeadas. Fornecem API e operadores para manipulação (acesso, inclusão, alteração) de nodos em uma lista encadeada;

Pacotes SNMP – São bibliotecas destinadas ao desenvolvimento de aplicações de gerência baseadas em SNMP. Para o desenvolvimento dos agentes foi utilizado o SNMP CMU da *Carnegie Mellon University*, e para o gerente foi usado o SNMP da *Hewlett Packard*.

Nos tópicos seguintes são apresentados os objetos, incluindo seus atributos e comportamentos, de cada um dos módulos de *software* que compõem a solução de gerência desenvolvida neste trabalho.

4.3 Emulação de Terminal

O módulo Emulação de Terminal permite que a partir de um ponto central do ambiente de gerência seja possível ter acesso aos terminais de vídeo de quaisquer das centrais gerenciadas.

Na construção desse módulo não foi utilizado nenhum desenvolvimento em código orientado à objetos C++. O módulo de emulação foi construído por meio da integração dos seguintes componentes:

Xterm – É um aplicativo do sistema operacional Unix cuja função é emular terminais de vídeo padrão como: vt100, vt220, ansi.

cu – É também um aplicativo do sistema operacional Unix, cuja função é chamar outros sistemas (Unix ou não), para tentar conectá-los ao sistema requisitante e gerenciar toda interação entre os sistemas conectados.

Arquivos de Configuração das Centrais – Para cada tipo de central a ter seu terminal de acesso emulado existe um conjunto de informações que deve ser considerado: velocidade de transmissão de *bits* durante a emulação, número de bits e tipo de paridade, número de *stop bits*, o dispositivo no qual o terminal de acesso emulado será associado e o número para o qual o módulo Emulação de Terminal deve discar, para assim conseguir obter um terminal de acesso.

Arquivo de *script* – É um *script* de execução cuja função é obter do arquivo de configuração de um determinado tipo de central o seu respectivo conjunto de informações,

e passá-las para o **Xterm** e **cu** preenchendo os parâmetros corretamente de cada um desses aplicativos.

A Figura 4.2 abaixo descreve um exemplo de integração dos aplicativos **Xterm** e **cu** para emular um terminal de acesso de uma CPA Trópico-RA.

Para o **Xterm** são utilizados os parâmetros:

–title: Permite especificar um título para a emulação;

–e: Possibilita a chamada de um outro executável para que quando for criado o terminal de emulação o executável seja disparado. No GCR, o executável é o comando **cu**.

Para o **cu** são utilizados os parâmetros:

-s: Especifica a velocidade de transmissão com a qual o dispositivo de comunicação deve trabalhar;

-l: Permite especificar um dado dispositivo de comunicação;

-n: Caso o dispositivo seja um *modem*, esse parâmetro indica o número a ser discado pelo mesmo.

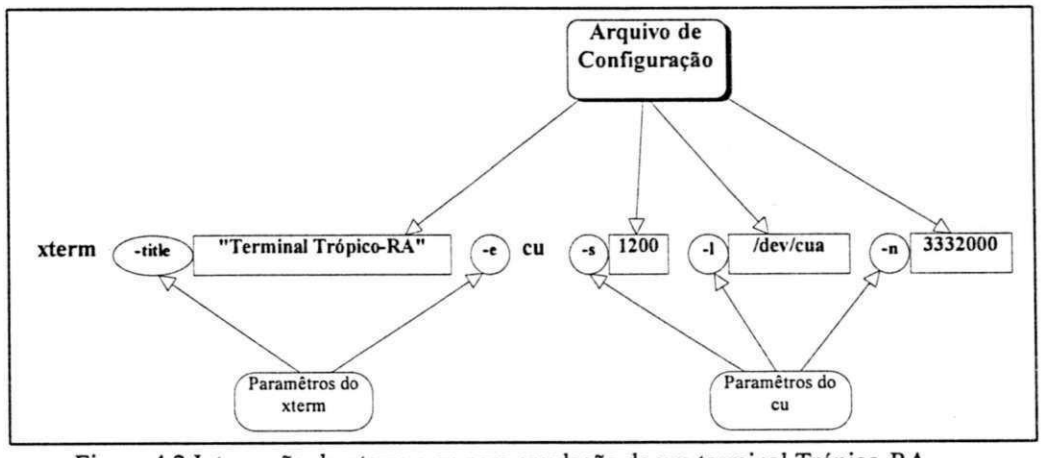

Figura 4.2 Integração do xterm e cu para emulação de um terminal Trópico-RA.

4.4 Conversores

Como relatado no Capítulo 3, os conversores são módulos responsáveis pela comunicação entre o GCR e as CPAs a serem gerenciadas. Basicamente eles enviam

comandos CHM às centrais, coletam os relatórios emitidos pelas mesmas e extraem dados de gerência desses relatórios.

As principais classes que compõem um conversor genérico são esboçadas a seguir na Figura 4.3.

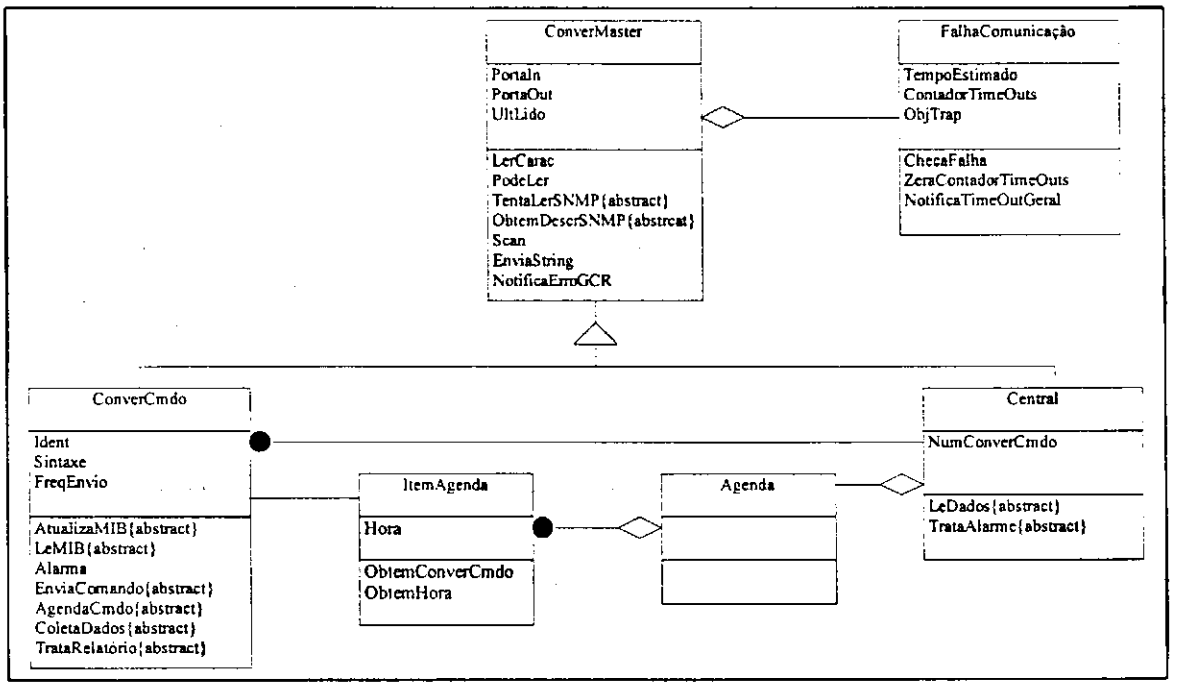

Figura 4.3 Modelo de Objetos (parcial) do módulo conversores.

A classe **ConverMaster** é uma superclasse estática, que serve como base para a modelagem das subclasses derivadas **Central** e **ConverCmdo.** Estas correspondem, respectivamente, às abstrações de uma central propriamente dita e de um conversor para um comando CHM.

ConverMaster é agregado a um objeto **FalhaComunicação** e possui uma porta de entrada (PortIn) e uma de saída (PortOut), e um *buffer* (UltLido) para o último caractere lido.

Um objeto **FalhaComunicação** possui um valor de tempo máximo para se avaliar se houve ou não falha de comunicação (TempoEstimado), de um contador de tempo (ContadorTimeOuts) e é agregado a um objeto da classe **Trap** (ObjTrap) não inclusa na Figura 4.3 para garantir a legibilidade.

comandos CHM às centrais, coletam os relatórios emitidos pelas mesmas e extraem dados de gerência desses relatórios.

As principais classes que compõem um conversor genérico são esboçadas a seguir na Figura 4.3.

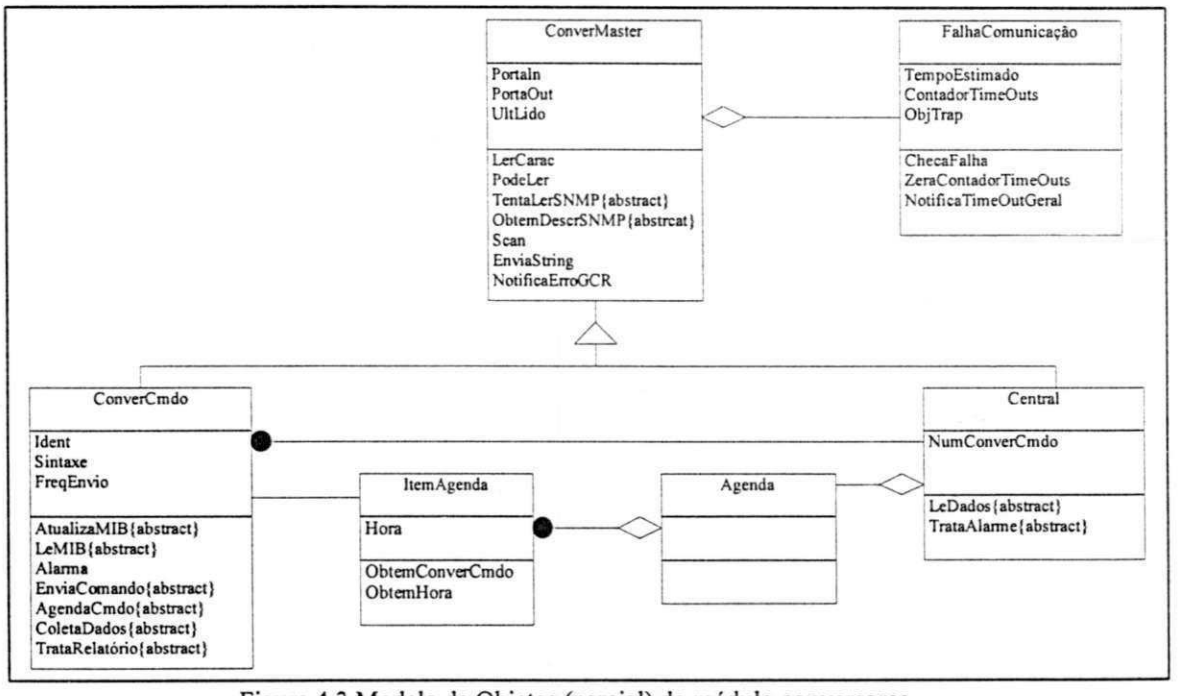

Figura 4.3 Modelo de Objetos (parcial) do módulo conversores.

A classe **ConverMaster** é uma superclasse estática, que serve como base para a modelagem das subclasses derivadas **Central** e **ConverCmdo.** Estas correspondem, respectivamente, às abstrações de uma central propriamente dita e de um conversor para um comando CHM.

ConverMaster é agregado a um objeto **FalhaComunicação** e possui uma porta de entrada (PortIn) e uma de saída (PortOut), e um *buffer* (UltLido) para o último caractere lido.

Um objeto **FalhaComunicação** possui um valor de tempo máximo para se avaliar se houve ou não falha de comunicação (TempoEstimado), de um contador de tempo (ContadorTimeOuts) e é agregado a um objeto da classe **Trap** (ObjTrap) não inclusa na Figura 4.3 para garantir a legibilidade.

A classe **Central** se relaciona com vários objetos da classe **ConverCmdo** e é agregado a um objeto da classe **Agenda**, que por sua vez é um agregado de vários objetos da classe **ItemAgenda**. Um **ItemAgenda** relaciona-se com um objeto **ConverCmdo,** e possui a hora (Hora) que este **ConverCmdo** deve ser tratado.

A classe **ConverCmdo** possui como atributos um identificador do comando (Ident), a sintaxe completa do comando (Sintaxe) e a freqüência de envio do comando (FreqEnvio).

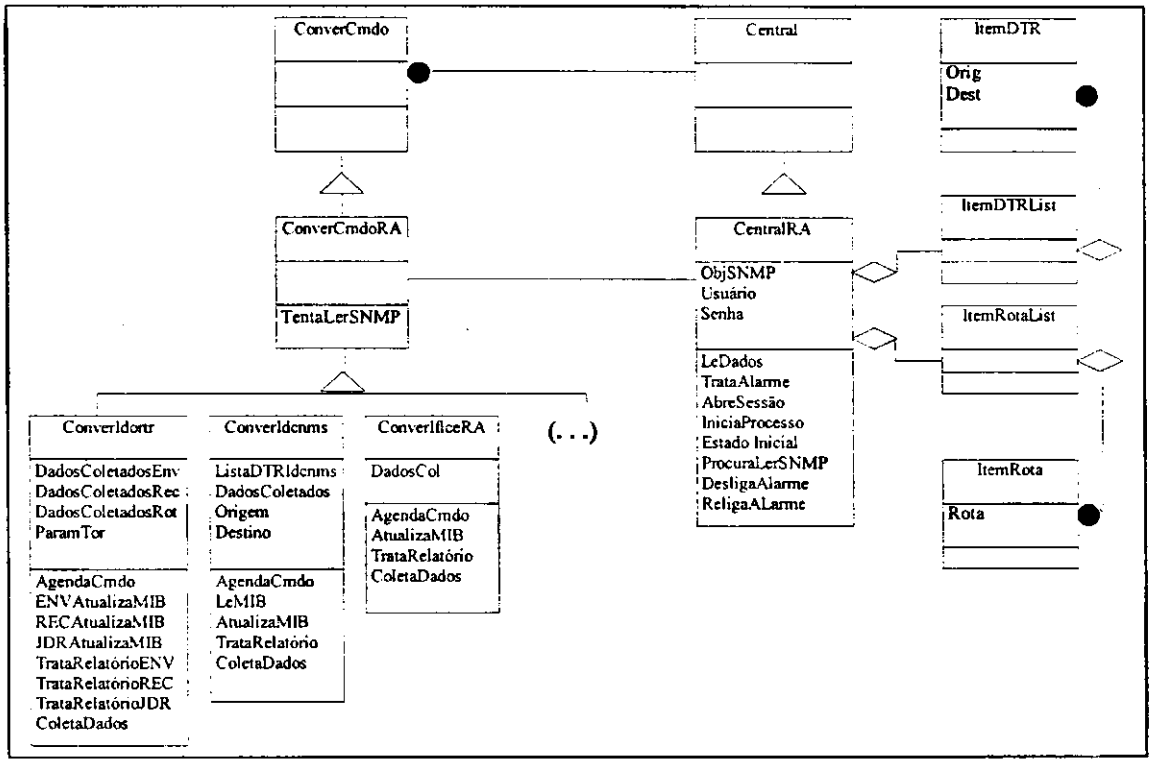

Figura 4.4 Modelo de Objetos do conversor para uma CPA Trópico-RA.

O modelo de objetos da Figura 4.3 apresenta as classes presentes em qualquer conversor genérico presente no ambiente GCR. Contudo, cada CPA monitorada tem características intrínsecas e diferentes umas das outras como dito no capítulo anterior. Apresenta-se na Figura 4.4, o modelo de objetos para uma CPA específica, no caso uma Trópico-RA, para que se possa esboçar, as funcionalidades de um conversor específico através dos objetos que o compõe.

Como apresentado na Figura 4.4 a classe **CentralRA** cuja abstração representa uma CPA Trópico-RA, é uma subclasse derivada da classe **Central** e a classe **ConverCmdoRA** é uma subclasse derivada da classe **ConverCmdo** cuja abstração representa um conversor

para um comando de uma CPA Trópico-RA. Já as classes **ConverIfalceRA**, **ConverIdcnms**, **ConverIdortr** são derivações da classe **ConverCmdoRA** e representam conversores para comandos específicos de uma Trópico-RA.

Um objeto **CentralRA** é constituído da agregação com um objeto da classe **SNMPRA** (ObjSNMP) não incluso na Figura 4.4 por questões de legibilidade, um da classe **ItemDTRList** e outro da classe **IdgrtrRotaList**. Um **CentralRA** possui ainda como atributos um identificador de *login* na Central (Usuário) e a autenticação para o *login* (Senha).

Os objetos **ItemDTRList** e **IdgrtrRotaList**, são respectivamente, constituídos de vários objetos da classe **ItemDTR** e da classe **IdgrtrRota** que representam as direções de tráfego e as rotas monitoradas pelo objeto **CentralRA**. O **ItemDTR** possui atributos que determinam a origem (Orig) e o destino (Dest) de uma direção de tráfego, enquanto que **IdgrtrRota** tem como atributo o identificador da rota (Rota).

A seguir é apresentado como estão implementadas as funcionalidades de um conversor de uma central Trópico-RA, em função dos comportamentos (métodos) modelados em sua classe.

4.4.1 Criação de um Conversor

A criação de um conversor se dá com a inicialização de um objeto cuja classe seja uma subclasse derivada da classe **Central**. É utilizado como exemplo a classe **CentralRA**.

A criação é disparada através de uma chamada ao construtor da classe passando seus devidos parâmetros:

```
ConversorRA = new CentralRA(Usuário, Senha, PortaIn, PortaOut);
```

Os parâmetros listados acima informam, respectivamente, ao conversor: o nome da entidade (Usuário) e sua senha (Senha) que será usada para criar uma sessão na central; os dispositivos pelos quais serão enviados comandos à central (PortaOut) e por meio dos quais serão recebidos os relatórios emitidos pela central (PortaIn).

Neste método é criada uma tabela de conversores de comandos específica para aquele tipo de central, no caso uma Trópico-RA. Essa tabela constitui o módulo Conjunto de Comandos apresentado no Capítulo 3.

```
CentralRA::CentralRA(Usuário, Senha, PortaIn, PortaOut)
{
...
        TabConverCmdo[1] = new ConverIfalceRA(PortaIn, PortaOut, TimeOutGeral);
        TabConverCmdo[2] = new ConverIdcnms(PortaIn, PortaOut, TimeOutGeral);
        ...
        TabConverCmdo[n] = new ConverIdortr(PortaIn, PortaOut, TimeOutGeral);
...
}
```

4.4.2 *Abertura de uma Sessão*

Para que possam desempenhar suas funções, os conversores promovem a abertura de uma sessão no elemento de rede para poder enviar comandos CHM e ter acesso aos seus recursos.

O método da classe **CentralRA** responsável por esta função é o **AbreSessão**. Seu comportamento baseia-se num autômato finito e específico para cada tipo de central. O algoritmo em alto nível para uma Trópico-RA é definido a seguir.

```
CentralRA::AbreSessao()
{
        Enquanto conversor não recebe a sequência "USUARIO="
                Envia sequência de controle
        Fim
        Enquanto sequência igual a "USUARIO="
                Envia Nome do Usuário
                Se conversor achou a sequência "SENHA="
                        Envia sequência Senha do Usuário
                        Sessão criada
                        Sair do método
                Senão
                        Procura a sequência "USUARIO="
                        Volta para o início do laço
                Fim
        Fim
}
```

4.4.3 *Inicialização do Conversor*

A inicialização é realizada pelo método **IniciaProcesso** da classe **CentralRA**. É nesse método que o conversor começa a exercer realmente suas funcionalidades, já que o *Schedule* de Comandos (apresentado no capítulo anterior) inicia os agendamentos de todos os comandos existentes na tabela de comandos, para que seja possível, periodicamente, questionar a central sobre seus contadores internos. Os agendamentos dos comandos são realizados em função dos seus atributos de tempo de freqüência de envio (FreqEnvio).

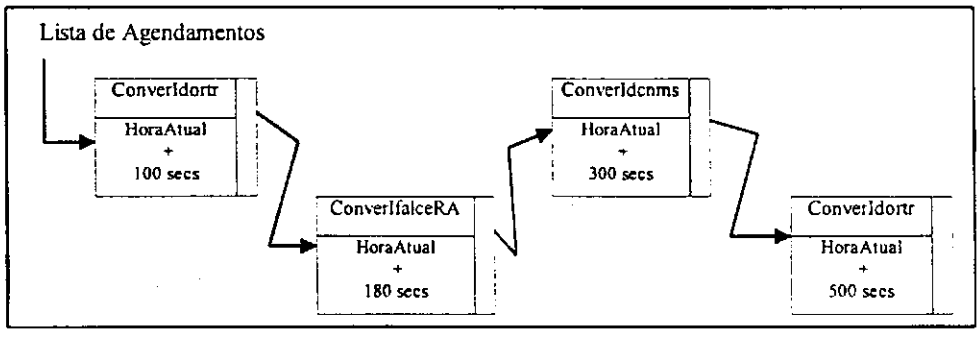

Figura 4.5 Lista de agendamentos de comandos.

Internamente o *Schedule* de Comandos cria uma lista de comandos ordenada por seus tempos de freqüência de envio e programa um *timer* para o menor tempo de freqüência, ou seja para o primeiro agendamento da lista ordenada. Na programação do *timer* é acrescido o valor da hora atual. Um exemplo de agendamento de comandos é esboçado na Figura 4.5

De acordo com a Figura 4.5, o *timer* programado é o do primeiro comando ConverIdortr e só expirará depois de 100 segundos contados a partir da hora atual.

4.4.4 *Envio de Comandos Homem-Máquina às Centrais*

Os comandos CHM são enviados às centrais periodicamente em função dos agendamentos dos mesmos, realizados pelo *Schedule* de Comandos. Basicamente, o envio se dá quando o *timer* programado pelo o *Schedule* se expira.

Assim que o *timer* expira, é função do método **TrataAlarme** da **CentralRA** obter quais os comandos presentes na lista de agendamentos do *Schedule* cujo período é menor ou igual a hora de expiração do *timer*. Os comandos que atendem essa condição são

retirados da lista de agendamentos e estão habilitados a enviarem sua sintaxe de comando à central, através da PortaOut.

Ainda é comportamento do método **TrataAlarme** re-agendar os comandos que foram enviados à central e programar um novo *timer*, utilizando a freqüência de envio do primeiro comando da lista de agendamentos. O *timer* é renovado justamente para garantir a periodicidade do envio de comandos.

4.4.5 Coleta dos Relatórios

Todas as vezes que se expira o *timer* do *Schedule* e são retirados comandos da lista de agendamentos, o conversor passa para fase de coleta de dados. Este comportamento é implementado pelos métodos ColetaDados dos objetos derivados da classe **ConverCmdoRA** e que foram retirados da lista do *Schedule* de Comandos. O comportamento do método ColetadoDados, de maneira genérica, é descrito a seguir.

```
ConverCmdoRA::ColetaDados
{
        Preenche sintaxe do comado;
        Envia sintaxe à central;
        Trata o relatório emitido;
}
```

O preenchimento da sintaxe do comando consiste em anexar numa seqüência de caracteres todos os parâmetros inerentes de cada comando, tais como: hora atual, data, identificadores de configuração e palavras chaves.

O envio da sintaxe é realizada pelo método EnviaString. Toda subclasse derivada da classe **ConverCmdoRA** possui este método e, desde que seja um comando válido para central e sua sintaxe esteja correta, o comando será enviado pela dispositivo de saída, PortaOut, que significa escrever na PortaOut a seqüência de caracteres.

Após o envio da sintaxe do comando o método ColetaDados se prepara para tratar o relatório emitido pela central em resposta ao comando recebido do conversor. O tratamento do relatório se constitui basicamente, em extrair do conjunto de caracteres que formam um relatório, os dados relevantes para realização da gerência da planta telefônica. Essa função é desempenhada pelo método **TrataRelatório** próprio de cada subclasse derivada da classe **ConverCmdoRA.**

A função do analisador sintático (*parser*), citado no Capítulo 3, que busca agrupar os *tokens* em frases significativas, é implementado no método **TrataRelatório** e são justamente dessas frases significativas que este método extrai os dados relevantes para gerência.

Todo caractere emitido por uma central é recebido, do lado conversor, pelo analisador léxico (*scanning*), referenciado no Capítulo 3, cuja função é receber toda seqüência de caracteres sem significados e agrupá-los seguindo regras de formação para construção de palavras, dígitos e símbolos. Assim, todas as vezes que o analisador sintático necessita de um ou mais *tokens* para produzir uma frase ele requisita os *tokens* ao analisador léxico. O comportamento do método Scan é apresentado a seguir.

```
ConverMaster::Scan(char *Token, long TimeOut)
{
        Caractere  = LeCarac(TimeOut);
        Enquanto (Caractere  é igual a espaços em branco) {
                Caractere  = LeCarac(TimeOut);
        }
        Se (Caractere é letra)  {
                Concatena Caractere ao Token ;
                Caractere = LeCarac(TimeOut);
                while (Caractere é letra ) {
                        Concatena Caractere ao Token ;
                        Caractere = LeCarac(TimeOut);
                }
                Se ( Caractere  igual a VAZIO )
                        Retorna (FALSE )
                 Senão
                        Retorna(TRUE );
        }
        Se (Caractere é dígito ) {
                Concatena Caractere ao Token ;
                Caractere = LeCarac(TimeOut);
                Enquanto (Caractere é dígito ) {
                        Concatena Caractere ao Token ;
                        Caractere  = LeCarac(TimeOut);
                }
                Se ( Caractere  igual a VAZIO )
                        Retorna (FALSE )
                 Senão
                        Retorna(TRUE );
        }
        Concatena Caractere ao Token ;
        Se ( Caractere  igual a VAZIO )
                Retorna (FALSE )
         Senão
                Retorna(TRUE );
}
```

O analisador léxico é implementado pelo método **Scan** e se utiliza de mais dois outros métodos: o **LeCarac** e o **PodeLer**, que pertence também a superclasse **ConverMaster**.

LeCarac e PodeLer são, literalmente, os métodos responsáveis pelo recebimento de qualquer caractere alfanumérico ou de controle enviado pela central ao ambiente GCR.

O método LeCarac tem como comportamento obter um caractere do dispositivo de entrada, o PortaIn. Outra tarefa desempenhada é a de verificar se houve quebra de conexão e consequentemente falha de comunicação entre a central e o conversor. Se houver falha de comunicação uma notificação é gerada.

```
ConverMaster::LeCarac(TimeOut)
{
        Loop {
                Se ( !PodeLer(TimeOut) ) {
                        Se ( FalhaComunicação->ChecaFalha(TimeOut) )
                                FalhaComunicação ->NotificaTimeOutGeral();
                        Retorna (VAZIO);
                }
                Caracter = Read (PortaIn);
        } Enquanto ( Caracter igual a EOF OU Caracter igual a BEEP );
        FalhaComunicação ->ZeraContadorTimeOuts();
        Retorna (Caracter);
}
```

Já o método PodeLer busca identificar se existe algo realmente para ser lido da PortaIn ou se existe alguma requisição SNMP feita pelo gerente. Se existe algo para ser lido, o LerCarac poderá obter o caractere, caso contrário, o LerCarac verifica a possibilidade de ter ocorrido uma falha de comunicação.

O método PodeLer recebe como parâmetro um valor de tempo que representa o tempo de escuta máximo (*TimeOut)* que deve ser utilizado para se verificar se há algo para ser lido dos dispositivos de entrada: PortaIn ou *socket* SNMP. É bom lembra que o valor de tempo é configurado para cada tipo de relatório a ser recebido pelo conversor em resposta ao seu respectivo comando enviado à central. A seguir, é apresentado o comportamento do método PodeLer.

```
ConverMaster::PodeLer(long MaxSec)
{
        Enquanto ( TRUE ) {
                TimeOut = MaxSec;
                HoraAntes = ObtemHoras();
                Resultado = Escutando PortaIn  e socket SNMP (TimeOut);
                HoraDepois = ObtemHoras ();
                Caso (Resultado) {
                        Se igual -1:// A escuta interrompido por algum sinal; tentar novamente
                        Se igual 0:  // Nada para ser lido, nem socket nem PortaIn
                                Retorna (FALSE);
                        Senão:  // Algo para ser lido, ou socket SNMP ou PortaIn
                                TentaLerSnmp( socket SNMP );
                                Se ( Tem Caracter  na PortaIn ) {
                                        Retorna (TRUE);
                                }
                }
                // Novo timeout, descontado o tempo da escuta do socket SNMP e da PortaIn
                TimeOut = TimeOut - (HoraDepois - HoraAntes);
                Se (Estourou TimeOut)
                        Retorna (FALSE);
        }
}
```

Caso o resultado da escuta (Resultado) seja maior que zero é porque existe algo para ser lido dos dispositivos. Então, verifica-se primeiramente se existe alguma requisição SNMP para ser atendida, para depois verificar se o dispositivo PortaIn tem algo a ser lido. Se o resultado da escuta (Resultado) for 0 (zero) é porque nada existe para ser lido.

Durante a escuta pode ocorrer algum tipo de interrupção inesperada que provoque algum erro na escuta nos dispositivos de entrada, esse é o caso em que o resultado da escuta é –1 (menos um). Em função disso, é calculado um novo *TimeOut* descontando-se o tempo gasto no processo de escuta que foi interrompido. Se o novo *TimeOut* for igual ou menor a zero é porque nada existe nos dispositivos para ser lido.

De posse das frases significativas que compõem um relatório, cabe ao método TrataRelatório retirar os *tokens* que representam dados importantes à aplicação de gerência. O algoritmo apresenta a idéia de comportamento de um TrataRelatório genérico.

De acordo com o algoritmo apresentado a seguir, é realizada uma análise das frases obtidas. Para cada conversor de comando existe uma análise própria que representa a comparação entre as frases obtidas e as frases que o conversor de comando espera e pode receber. Na comparação se verifica se todos os *tokens* que compõem a frase são conhecidos e se estão num seqüência organizacional também conhecida pelo conversor.

```
ConverCmdo::TrataRelatório( )
{
        Enquanto (Não é fim de relatório) {
                Token = Scan(TimeOut) // Obtendo token da PortaOut
                Frase = Frase + Token // Formando frases significativas
                AnalisaFrase (Frase)
                Se (Frase está correta) {
                        Extrai dados relevantes à gerência
                        Limpa Frase
                        Guarda dados em Estrutura Intermediária
                }
                Senão  {  // Frase incorreta. Erros devido à ruídos
                        Despreza dados e Frase inteira
                        Limpa Frase
                }
        }
        AtualizaMIB( ) // Copia os dados da Estrutura Intermediária para MIB
}
```

É ainda no método TrataRelatório que são realizadas as recuperações de erros, decorrentes de ruídos da linha discada, quando na transmissão dos caracteres que compõem um relatório. Um erro é caracterizado e identificado quando numa análise de frase, esta possui *tokens* desconhecidos para o conversor de comando ou quando os *tokens* são conhecidos, porém sua ordem dentro da frase é também desconhecida. Quando se detecta um erro, o TrataRelatório faz um *bypass* da frase, desprezando todos os possíveis dados de gerência, até que encontre outros *tokens* numa seqüência organizacional conhecida pelo conversor.

Caso o resultado da análise indique que a frase é correta, então pode-se realizar a extração dos dados de gerência. A extração se dá em função da seqüência organizacional da frase obtidas, de maneira que o TrataRelatório de cada conversor de comando conhece os *tokens*, pelas suas posições dentro da frase, que representam dados de gerência. Além disso, o TrataRelatório deve conhecer para que tipo de dado deve ser transformado um determinado *token* que representa um dado de gerência.

De cada frase é extraído um conjunto de dados de gerência que é guardado temporariamente numa estrutura de dados intermediária, citada no Capítulo 3. Então, após a leitura de todo o relatório e da extração de todos os dados de gerência, o TrataRelatório dispara uma atualização do objeto da MIB que representa os dados coletados, com os dados contidos na estrutura intermediária.

4.4.6 Detecção de Falta de Comunicação entre Conversor e CPA

Para cada conversor existe um único objeto **FalhaComunicação** que é responsável pela detecção da impossibilidade da coleta de dados pelo conversor devido à falta de comunicação deste com a central.

Todos as vezes que se verifica se existe algum caractere para ser lido do dispositivo PortaIn, e o seu *TimeOut* de escuta se expira, é enviado uma mensagem ao objeto **FalhaComunicação** para que este verifique se houve ou não uma falha de comunicação entre a central e seu conversor. Este comportamento é exercido pelo método **ChecaFalha** do objeto **FalhaComunicação** que é descrito a seguir.

```
FalhaComunicacao::ChecaFalha(Time)
{
        ContadorTimeOuts = ContadorTimeOuts + Time;
        Se ( ContadorTimeOuts Maior ou igual a TempoEstimado ) {
                ContadorTimeOuts = 0 ;
                Retorna (Ocorreu Falha);
        }
        Retorna (Não há Falha);
}
```

O objeto **FalhaComunicação** tem um atributo chamado TempoEstimado cujo valor reflete o tempo máximo em que, mesmo na ausência de caracteres enviados aos conversores por uma central, considera-se ainda existir comunicação real entre a central e o conversor. Outro atributo do objeto é o ContadorTimeOuts cujo o valor é incrementado todas as vezes que o mecanismo de escuta do dispositivo PortaIn nada identifica para ser lido. Quando o ContadorTimeOuts extrapola o valor do atributo TempoEstimado pode-se afirmar que ocorreu falha de comunicação entre o conversor e sua central. Assim, o objeto **FalhaComunicação** realiza uma notificação às camadas de *software* superiores. Essa notificação se constituí no disparo de um *Trap* SNMP.

Contudo, se existe algo para ser realmente lido da PortaIn, o objeto **FalhaComunicação** recebe uma mensagem para parar o incremento do atributo ContadorTimeOuts e reinicializar o mesmo.

4.4.7 *Estrutura Intermediária de Armazenamento*

A estrutura intermediária de armazenamento contém todos os dados de gerência extraídos dos relatórios emitidos pelas centrais. Todo objeto derivado da classe **ConverCmdo** tem seu respectivo depósito de armazenamento. O conjunto de todos os depósitos de armazenamento de cada objeto derivado da classe **ConverCmdo** constituí a estrutura de armazenamento intermediária. No modelo de objetos do conversor da Trópico-RA apresentado na Figura 4.4, os objetos **ConverIdortr**, **ConverIdcnms** e **ConverIfalceRA** possuem apresentados seus depósitos. Para o **ConverIdortr** tem-se os DadosColetadosEnv, DadosColetadosRec, DadosColetadosRot; o **ConverIdcnms** possui ListaDTRIdcnms e DadosColetados e **ConverIfalceRA** tem DadosColetados.

Os depósitos são implementados como listas encadeadas de nodos, em que cada nodo contém os dados de gerência extraídos dos relatórios. O preenchimento dessas listas é feito à medida que o conversor trata o relatório emitido pela central e extrai do mesmo os dados de gerência. A cada bloco de dados de gerência obtido o conversor, mais especificamente o Tratador de Relatório, utiliza os comportamentos das *Templates* de Listas para inserir os dados na lista e manter tais dados armazenados de forma encadeada.

4.5 MIB

Como apresentado no Capítulo 3 a MIB é a base de informação de gerência constituída de objetos (que não seguem o paradigma da orientação a objetos) que abstraem para o mundo computacional os elementos de rede gerenciados.

No caso do GCR, esses objetos que formam a MIB são implementados como um conjunto de listas encadeadas. A implementação de cada lista está encapsulada em classes (de orientação à objetos) cuja hierarquia é apresentada na Figura 4.6.

A classe **MIB** é superclasse estática da hierarquia e funciona como base para modelagem das demais classes. Possui como atributos um marcador temporal (*TimeStamp*) e o Identificador do Objeto dentro da MIB (OID). Cada uma das classes derivadas da classe **MIB**, encapsula uma lista encadeada que representa um objeto da base de informação de gerência.

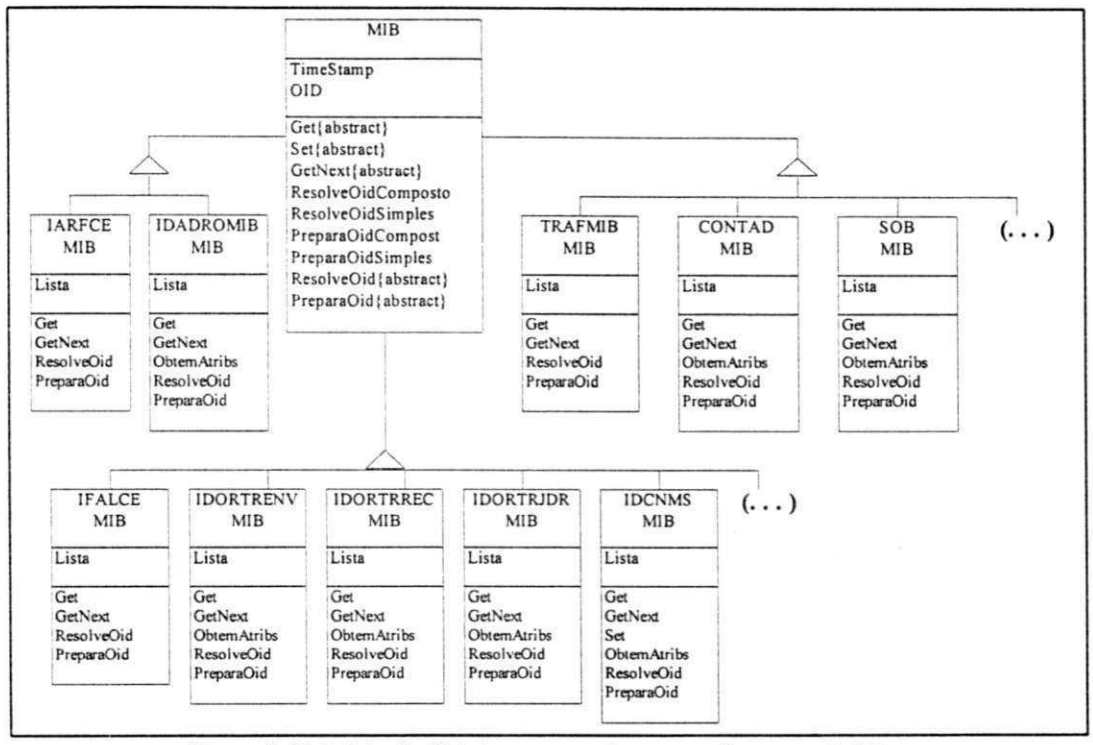

Figura 4.6 Modelo de Objetos para as classes que formam a MIB.

Cada elemento da lista possui um conjunto de atributos. O modelo de objetos da Figura 4.7 exemplifica o conjunto de classes que compõem uma lista e consequentemente um objeto da MIB. Neste caso, são esboçadas as classes que conceitualmente compõem o objeto **DadosTrafRotaEntry** (apresentado no Capítulo 3) da MIB do GCR.

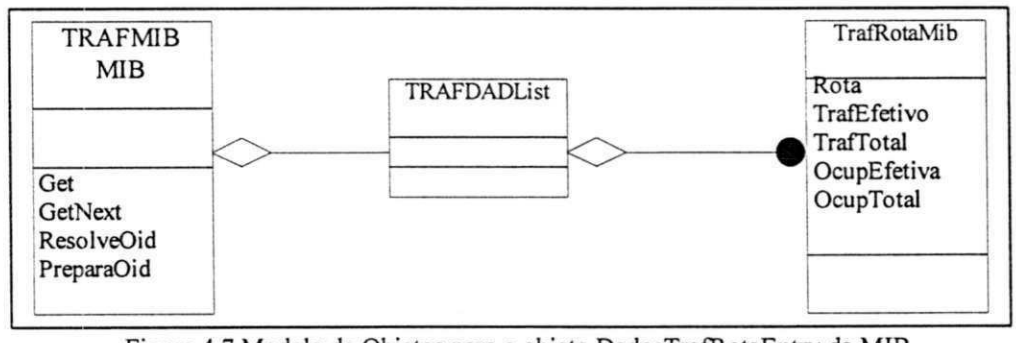

Figura 4.7 Modelo de Objetos para o objeto DadosTrafRotaEntry da MIB.

O objeto **TRAFMIB** é agregado a um objeto **TRAFDADList** (representando a lista) que é composto de vários objetos **TrafRotaMib** (representando os vários elementos da lista). O objeto **TrafRotaMib** possui os seguintes atributos: a rota de identificação do tráfego (Rota), o tráfego efetivo, o tráfego total, a ocupação efetiva e a ocupação total da rota.

4.5.1 Obtenção dos dados da MIB

Conceitualmente os objetos da MIB são representados por tabelas em que suas colunas representam os atributos do objeto. Uma linha da tabela, composta dos valores de suas colunas, representa uma instância do objeto representado.

No GCR, este conceito é mapeado para o conjunto de listas que compõe sua MIB. Uma tabela é representada por uma lista encadeada, as colunas da tabela são representados pelos atributos dos itens da lista e uma linha da tabela é representado por cada item da lista.

Cada um dos objetos derivados da superclasse **MIB** possui seus próprios métodos Get, GetNext e Set que são as operações realizadas sobre os objetos da MIB. É função desses métodos, além de executar sua semântica de operação sobre os objetos, mapear o conceito de tabelas de MIB para a estrutura de listas da MIB do GCR.

O comportamento do método Get para o objeto **TRAFMIB** é apresentado a seguir para exemplificar o mecanismo de obtenção dos dados sobre os objetos da MIB.

```
TRAFMIB::Get( variable_list Var )
{
        TrafRotaMib Dados;
        int IdLinha, IdColuna;

        IdColuna = Var->name[MAXNAMELEN-2];
        IdLinha = Var->name[MAXNAMELEN-1] - 1;
        // Obtem os atributos do n-ésimo item da lista que representa IdLinha
        Se ( Dados = ObtemAtribs(IdLinha) ) {
                Retorna ("Objeto não encontrado");
        }
        // tenta obter o valor dos atributos que representam a coluna IdColuna
        Caso ( IdColuna ) {
                Se IdColuna igual a IDROTA_TRAF:
                        Var->val = Dados.Rota;
                Se IdColuna igual a TRAFEFET_TRAF:
                        Var->val = Dados.TrafEfetivo;
                Se IdColuna igual a TRAFTOT_TRAF:
                        Var->val = Dados.TrafTotal;
                Se IdColuna igual a OCUPEFET_TRAF:
                        Var->val = Dados.OcupEfetiva;
                Se IdColuna igual a OCUPTOT_TRAF:
                        Var->val = Dados.OcupTotal;
                default:
                        Retorna ("Objeto não encontrado");
        }
}
```

O método Get recebe uma estrutura do tipo *variable_list* que contém o identificador do objeto para o qual se deseja obter os dados. Dele se obtém a linha e a coluna da tabela nas quais se localizam os valores desejados. No caso do GCR, o método ObtemAtribs obtém da lista encadeada o item que representa a linha da tabela desejada. Em seguida, dos atributos dos itens, obtém-se aquele que representa a coluna, conseguindo então o valor do objeto questionado.

4.6 Agentes

Os agentes são as entidades passivas que atendem as solicitações do gerente acerca das informações de gerência coletadas e armazenadas na MIB. Como mostrado no Capítulo 3, para cada tipo de conversor existe seu respectivo agente. Na Figura 4.8 são apresentadas as principais classe de um agente genérico.

Um agente é constituído basicamente das classes apresentadas na Figura 4.8. A classe principal que representa um agente é denominada de **Snmp**. É agregada de vários objetos da classe **MIB**, e mantém ligação com um objeto da classe **Central** que é justamente o conversor ao qual o agente é associado. A classe **Snmp** é responsável por atender às solicitações do gerente. Outra classe importante é a **Trap**. É constituída de um objeto da classe **Snmp** e é responsável por enviar interrupções abruptas ao gerente.

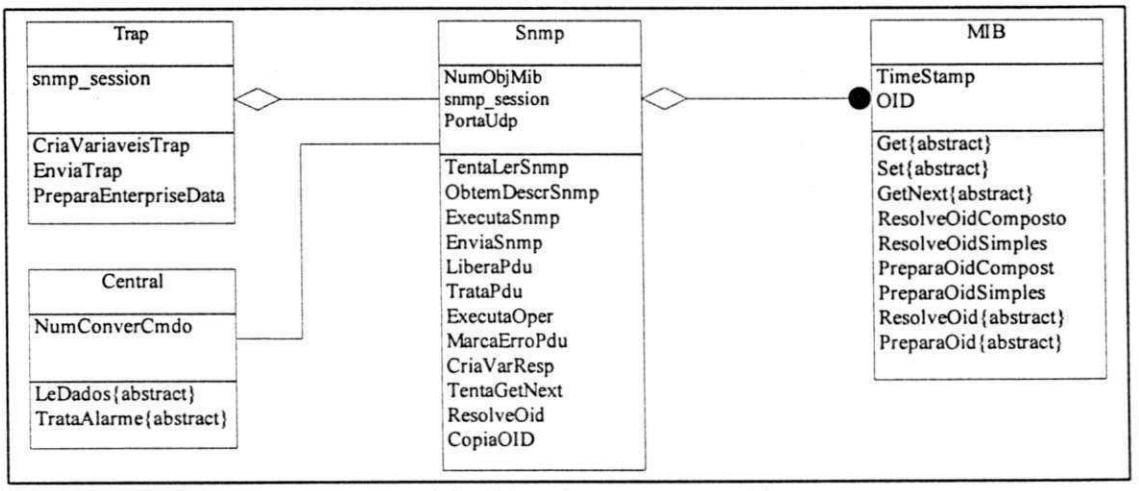

Figura 4.8 Modelo Objeto de um agente genérico.

A classe **Snmp** possui os seguintes atributos:

NumObjMib - Quantidade de objetos pertencentes a MIB que é referenciada pela classe;

snmp_session - Uma estrutura de dados utilizada para envio, transmissão, empacotamento e desempacotamento de pacotes SNMP. Essa estrutura será detalhada mais a frente;

PortaUdp – Representa o dispositivo por meio do qual serão recebidos e enviados os pacotes SNMP.

Já a classe **Trap** possuí como atributo apenas uma estrutura do tipo snmp_session, cuja função é a mesma da snmp_session da classe **Snmp**.

Um objeto da MIB têm basicamente dois atributos:

TimeStamp – É a marcação temporal utilizada para determinar se os valores do objeto estão atualizados ou não, em relação aos dados armazenados na estrutura intermediária dos conversores;

Object Identifier (OID) – Representa o identificador único do objeto dentro da organização hierárquica de árvore da MIB.

Cada tipo de central possui um conversor para coletar os dados de gerência e armazená-los em objetos da MIB do GCR. Em alguns casos, os dados coletados não são suficientes para que todos os objetos da MIB de uma dada central sejam preenchidos. Assim, para cada conversor existe um agente que conhece quais são os objetos da sua MIB que são preenchidos e qual a sua localização. A Figura 4.9 esboça o modelo de objetos para um agente de um conversor de uma central Trópico-RA.

A classe **SnmpRA** é derivada da classe **Snmp** e representa o agente para um conversor de uma central Trópico-RA propriamente dito e para simplificar nomeia-se esse tipo de agente como um agente Trópico-RA.

Da classe MIB são derivadas as classes que compõem os objetos da MIB manipulados por um agente Trópico-RA. O símbolo "(. . .)" na Figura 4.9 indica a existência de outros objetos que são derivados da classe MIB e que por questão de legibilidade não estão apresentados na Figura 4.9.

Como feito anteriormente para os conversores, é apresentado a seguir o comportamento de um agente Trópico-RA em função do métodos modelados em sua classe. É importante relatar que o comportamento de um agente encapsula as funcionalidades disponibilizadas pelo pacote de desenvolvimento de aplicação de gerência baseadas no protocolo SNMP da *Carnegie Mellon University* (**CMU**), chamado de **SNMP CMU**.

É através da estrutura snmp_session, que é um atributo da classes Snmp e Trap, que se pode utilizar as funcionalidades do pacote de desenvolvimento SNMP CMU.

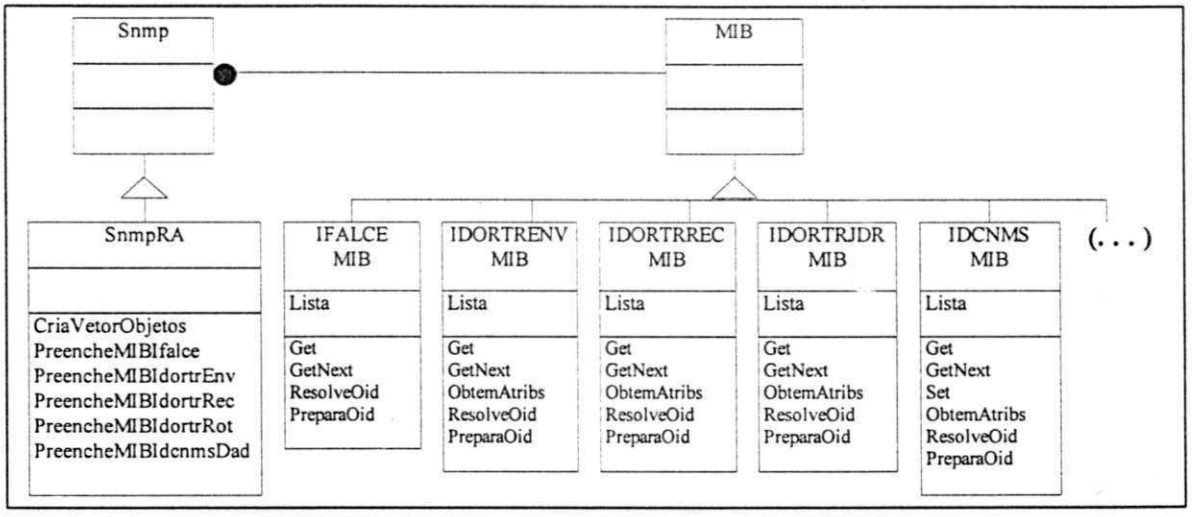

Figura 4.9 Modelo Objeto para o agente de um conversor da Trópico-RA.

4.6.1 Criação de um Agente

A criação de um agente se constitui do preenchimento de informações da estrutura snmp_session necessários para inicialização do protocolo SNMP, que proporcionará a comunicação entre o agente e o gerente; e da criação da organização hierárquica lógica da MIB para localização dos objetos a serem manipulados pelo agente. Além da criação de toda a estrutura de dados que armazena as informações dos objetos que fazem parte da MIB. O construtor da classe SnmpRA é o método que realiza todas essas funções citadas e seu comportamento é apresentado a seguir.

```
SnmpRA::SnmpRA(PortaLocal, PortaRemota, IPRemoto, CommunityName;)
{
        snmp_session  SessIni;
        SessIni.community  = CommunityName;
        SessIni.timeout = SNMP_DEFAULT_TIMEOUT;
        SessIni.retries = SNMP_DEFAULT_RETRIES;
        SessIni.peername =  IPRemoto;
        SessIni.remote_port = PortaRemota;
        SessIni.local_port = PortaLocal;
        SessIni.authenticator = TrataAuth;
        SessIni.callback = TrataSNMP;
        SessIni.version = SNMP_VERSION_1;
        SessSnmp = snmp_open( &SessIni );
        Se ( Não conseguiu abrir SessSnmp ) {
                ("Erro fatal na abertura da sessao SNMP (snmp_open):");
        }

        CriaVetorObjetos( );
}
```

Os parâmetros passados para o construtor da classe SnmpRA são utilizados para preencher algumas das informações da estrutura snmp_session. A estrutura possuí os seguintes atributos:

community – Contém uma *string* que representa o *community name* a ser aceito, pelo agente, nas autenticações das solicitações SNMP;

timeout – Valor de tempo máximo para verificação de existência de solicitação;

retries – Número de tentativas para ler ou enviar um pacote SNMP antes de ocorrer *timeout*;

peername - Nome ou número IP do *host* remoto para o qual será criada a sessão SNMP;

remote_port – Porta UDP do *host* remoto utilizada pela sessão. Para o GCR é utilizada a porta padrão para agentes SNMP, a porta 161;

local_port – Porta UDP do *host* local utilizada pela sessão. Também é utilizada a porta padrão 161;

authenticator – Endereço da função que realiza a autenticação do pacotes SNMP. Essa função é definida pelo desenvolvedor que está fazendo uso do SNMPCMU. No caso do GCR é a função TrataAuth;

callback - Endereço da função que realmente trata as solicitações do gerente. É também definida pelo desenvolvedor e no caso do GCR é a TrataSNMP;

version – Indica qual a versão do protocolo que está sendo usada. No GCR utiliza-se a versão SNMP v1 (versão 1);

Após o preenchimento das informações da estrutura snmp_session, é feita uma chamada a uma função snmp_open da API do SNMPCMU, cujo objetivo é a criação de fato de uma sessão SNMP que é a conexão de um agente ao gerente através do protocolo SNMP. É através desta sessão SNMP que são realizados os recebimentos de solicitações do gerente pelo agente, e que também são enviadas as repostas dos agentes para o gerente.

Para criar os objetos que fazem parte da MIB de um agente e para organizá-los logicamente, o construtor solicita a funcionalidade do método CriaVetorObjetos.

```
SnmpRA::CriaVetorObjetos()
{
        NumObjMib = 5;

        // Criando os objetos da MIB
        Objetos = new (MIB *[NumObjMib]);
        Objetos[IFALCE] = new IFALCEMIB;
        Objetos[IDOENV] = new IDORTRENVMIB;
        Objetos[IDOREC] = new IDORTRRECMIB;
        Objetos[IDOJUN] = new IDORTRJDRMIB;
        Objetos[IDCNMS] = new IDCNMSMIB;

        // Localizando hierarquicamente os objetos na MIB
        Objetos[IFALCE]->OID = ISO.ORG.DOD.INTERNET.PRIVATE.ENTERPRISES.
        LIGHTINFOCON.GCR.FALHAS.FALHASTABLE.FALHASENTRY;

        Objetos[IDOENV]->OID = ISO.ORG.DOD.INTERNET.PRIVATE.ENTERPRISES.
        LIGHTINFOCON.GCR. ORGAOS. EXTENVTABLE. EXTENVENTRY;

        Objetos[IDOREC]->OID = ISO.ORG.DOD.INTERNET.PRIVATE.ENTERPRISES.
        LIGHTINFOCON.GCR. ORGAOS. EXTRECTABLE. EXTRECENTRY;

        Objetos[IDOJUN]->OID = ISO.ORG.DOD.INTERNET.PRIVATE.ENTERPRISES.
        LIGHTINFOCON.GCR. ORGAOS. EXTJUNTABLE. EXTJUNENTRY;

        Objetos[IDCNMS]->OID = ISO.ORG.DOD.INTERNET.PRIVATE.ENTERPRISES.
        LIGHTINFOCON.GCR. CODIGO. CODIGOTABLE. CODIGOENTRY;
}
```

Como apresentado, esse método inicia cada um dos objetos que compõe a MIB do agente corrente e atribui a cada um dos objetos sua posição, dentro da organização hierárquica, em relação à padronização da MIB.

4.6.2 Verificação de Solicitações

A tarefa de verificar se existe solicitações requisitadas pelo gerente para serem atendidas pelo agente é realizada pelo método TenteLerSNMP da superclasse **Snmp** e consiste em identificar se há algum pacote SNMP na porta local de entrada estabelecida para comunicação entre um agente e o gerente.

A periodicidade da verificação é feita todas as vezes que se verifica se existe algo para ser lido pelo conversor como apresentado anteriormente. O método PodeLer da classe **ConverMaster** é o responsável por esta tarefa, de maneira que todas as vezes que o conversor tenta ler algo de sua PortaIn, o agente verifica se existe alguma solicitação SNMP.

O método TentaLerSNMP se constitui apenas em chamar uma função da API SNMPCMU que é o snmp_read. A função snmp_read verifica se existe algum pacote SNMP na porta local (*local_port*) para ser lido.

Caso exista, a função lê o pacote e obtém dele informações referentes ao *community name*, a fim de repassar tais informações para a função que autentica e valida o pacote como um todo. Essa função no caso do GCR é a TrataAuth cujo endereço foi passado para API SNMP através do atributo *authenticator* da estrutura snmp_session e definido pelo programador. A função TrataAuth apenas verifica se o *community name* do pacote lido é igual ao *community* da MIB do GCR. Se sim, o pacote é valido e a solicitação tem permissão para acessar as informações contidas na MIB do GCR, caso contrário a solicitação é negada.

```
TrataAuth( community_name )
{
        Se ( comunity_name diferente do Community_GCR )
                Retorna (Acesso Negado);

        Retorna (Acesso OK);
}
```

Se o pacote é válido e o solicitante tem acesso à MIB, a função snmp_read repassa o pacote como um todo, para a função *callback* construída pelo desenvolvedor e cujo endereço foi passado também para API SNMP através da estrutura snmp_session. É a função chamada TrataSNMP.

4.6.3 Execução de uma Solicitação

Executar uma solicitação é tarefa do agente e significa atender às requisições, no formato de pacotes SNMP, do gerente sobre as informações contidas na MIB.

Duas outras estruturas da API SNMPCMU são de fundamental importância para o execução das operações solicitadas: snmp_pdu e *variable_list*. É o conhecimento dos seus atributos que proporcionará o correto recebimento e tratamento das solicitações, como também um correto envio das respostas ao gerente.

A estrutura snmp_pdu é o pacote SNMP propriamente dito onde estão contidas informações essenciais para transmissão do pacote de uma origem a um destino, pela rede TCP/IP, carregando as informações trocadas entre agentes e gerente. Abaixo descreve-se os principais atributos dessa estrutura:

version – Versão do protocolo que está sendo utilizado;

address – Endereço IP do *host* remoto para o qual a pdu está sendo enviada;

community – *Community name* para autenticação do pacote;

command – Tipo de operação (*get*, *set*, *getnext*) que deve ser realizada sobre os objetos da MIB;

reqid – Identificador de requisição. Para cada requisição existe uma requisição única. A resposta a uma requisição possui o mesmo reqid;

errstat – Indica o *status* de erro de uma requisição;

errindex – Indica qual objeto cuja a requisição não foi realizada com sucesso através de um índice de erro;

variables – É uma estrutura do tipo *variable_list* cujo conteúdo representa os objetos sobre os quais a operação (*command*) será realizada.

Já na estrutura *variable_list*, encontram-se as informações referentes aos objetos da MIB a serem manipulados pela solicitação. Uma snmp_pdu contém uma *variable_list*. Os atributos dessa estrutura são mostrados a seguir:

next_variable – Apontador responsável pela criação da lista de variáveis;

name – Identificador do objeto na MIB, e.g.: ISO.ORG.DOD.INTERNET.PRIVATE.LIGHTINFOCON.GCR.FALHAS.;

name_length – Tamanho do identificador;

type – Tipo da variável;

val – Valor da variável;

val_len – Tamanho da variável em *bytes*.

Apesar do pacote SNMP depois de lido, ser entregue a *callback* TrataSNMP, esta função apenas repassa o pacote para o método ExecutaSnmp, descrito a seguir.

```
Snmp::ExecutaSnmp( struct snmp_pdu *Pdu )
{
        struct snmp_pdu *PduResp; // Pdu de resposta
        struct variable_list *Result; //Resultado

        PduResp->errstat = SNMP_ERR_NOERROR; // Status de erro OK
        PduResp->errindex = SNMP_DEFAULT_ERRINDEX; // Não existe índice de erro
        PduResp->command = GET_RSP_MSG; // Mensagem de resposta a uma requsição
        Operação = Pdu->command; // Tipo da operação a ser realizada sobre os objetos

        Loop: Para todo objeto existente em Pdu->variables faça {
                Se ( (Result = ExecutaOper (Pdu->variables, Operação )) é diferente de NOERROR ) {
                        MarcaErroPdu ( PduResp );
                        Sai do loop;
                }
                Pdu->variables = Pdu->variables->next;
                Guarda Result em PduResp->variables;
        }

        // preenche os últimos campos da Pdu a ser respondida
        PduResp->version = Pdu->version;
        PduResp->address = Pdu->address;
        PduResp->community = Pdu->community;
        PduResp->reqid = Pdu->reqid;

        EnviaSnmp( PduResp );

}
```

É no método ExecutaSnmp que será realizada a operação, sobre os objetos da MIB, solicitada pelo gerente. O ExecutaSnmp pertence a superclasse **Snmp**, contudo ele tanto faz uso de outros métodos da classe **Snmp** como também das classes de agentes derivados.

Esse método é responsável por preencher parte das informações da PDU a ser enviada como resposta, extraindo essas informações da PDU recebida. É ele também quem obtém o tipo de operação a ser executada e quais os objetos que devem manipulados.

Para cada um dos objetos solicitados, é invocado o método ExecutaOper passando como parâmetros o objeto e a operação a ser realizada sobre ele. O resultado da operação é armazenado na lista de variáveis da PDU de resposta para posterior envio. Caso aconteça algum erro na execução do método ExecutaOper uma marcação de erro é feita na PDU de resposta, de maneira que o solicitante (gerente) tenha conhecimento do erro ocorrido.

Depois de se executar a operação sobre todos os objetos requisitados, são preenchidas as informações restantes da PDU de resposta e enviada para a entidade solicitante.

O método ExecutaOper pertence a superclasse **Snmp** e sua função é realizar a operação requisitada sobre um objeto da MIB. A seguir apresenta-se o método.

```
Snmp::ExecutaOper(Variables, Operação)
{
        Se (Operação igual a GETNEXT_REQ_MSG ) { // tratamento em separado
                Retorna ( TentaGetNext( Variables ) );
        }
        Loop: Para ( todos objetos da MIB do agente corrente) faça {
                Caso ( Operação ) {
                        Se Operação igual GET_REQ_MSG:
                                CodErro = Objetos->Get ( Variables );
                                Se ( CodErro igual a SNMP_ERR_NOERROR )
                                        Retorna ( CodErro );
                                Sai do Loop;
                        Se Operação igual SET_REQ_MSG:
                                CodErro = Objeto->Set ( Variables );
                                Se ( CodErro igual a SNMP_ERR_NOERROR )
                                        Retorna ( CodErro );
                                Sai do Loop;
                        Senão:
                                Retorna ( SNMP_ERR_RESOURCEUNAVAILABLE );
                }

        }
        Retorna ( SNMP_ERR_NOSUCHNAME );
}
```

A idéia do método é varrer todo o conjunto de objetos pertencentes à MIB do agente corrente, até identificar o objeto solicitado e dependendo do tipo de operação requisitada, enviar uma mensagem para o objeto de maneira que este realize tal operação devolvendo o resultado.

Essa idéia é válida quando a operação solicitada é um *Get* ou um *Set*, porém se a solicitação for um *GetNext* o método TentaGetNext é invocado para, como o próprio nome diz, realizar a busca do próximo objeto da seqüência hierárquica da MIB.

4.6.4 Envio de uma PDU de Resposta

Enviar uma PDU de resposta significa mandar para a entidade solicitante gerente, uma PDU contendo o resultado da operação requisitada.

Como pode ser observado no quadro que apresenta o comportamento do método ExecutaSnmp da classe **Snmp**, algumas informações de controle (*errstat*, *errindex*, *version*, *address*, *community*) da PDU de requisição são utilizadas para preencher as informações de controle da PDU de resposta. Contudo, a informação de controle *command* da PDU de resposta é preenchida com o valor GET_RSP_MSG indicando que a PDU é uma resposta a uma solicitação.

O resultado de uma operação solicitada pelo gerente sobre objetos, é um conjunto de informações de gerência obtidas da MIB do agente questionado e que devem ser transmitidas para o gerente. Esse conjunto de informações é obtido no momento de execução da operação, através do método ExecutaOper da classe **Snmp**, quando as informações são aglutinadas aos objetos questionados através do parâmetro *Variables*.

Tendo todas as informações de controle preenchidas e as informações de gerência aglutinadas aos objetos, pode-se então enviar a PDU de resposta. O método ExecutaSnmp invoca o método EnviaSnmp pertencente também a classe **Snmp**, para realizar esta tarefa. Esse método realiza apenas uma chamada a mais uma função da API SNMPCMU que é a função snmp_send como pode ser observado no quadro, que apresenta o comportamento do método EnviaString.

A função snmp_send envia uma PDU através de uma sessão SNMP criada anteriormente.

```
Snmp::EnviaSnmp( struct snmp_pdu *Pdu )
{
        Se ( snmp_send( SessaoSnmp, Pdu ) diferente de SNMP_NOERROR) {
                ("Erro fatal na sessao SNMP (snmp_send):");
        }
}
```

4.6.5 Traps

Os *traps* (interrupções) são utilizados pelo agente para enviar, de maneira abrupta, uma notificação ao gerente acerca de eventos inesperados. No caso do GCR, todas as vezes que um conversor identifica que houve quebra de comunicação entre ele e sua respectiva CPA, um *trap* é criado pelo seu agente com o objetivo de avisar tal evento ao gerente.

Para que um agente possa enviar um *trap* SNMP para o gerente, é necessário que este crie e abra uma sessão SNMP através da qual o *trap* será enviado, e prepare uma PDU SNMP que conterá as informações referentes ao *trap* propriamente dito. O objeto responsável por essas funções é o **Trap** (apresentado no Modelo de Objetos da Figura 4.8).

O comportamento do construtor do objeto **Trap** é apresentado a seguir.

```
Trap::Trap(PortaLocal, PortaRemota, IPRemoto, CommunityName,)
{
        snmp_session SessTrap;
        SessIni.community  = CommunityName;
        SessIni.timeout = SNMP_DEFAULT_TIMEOUT;
        SessIni.retries = SNMP_DEFAULT_RETRIES;
        SessIni.peername =  IPRemoto;
        SessIni.remote_port = SNMP_TRAP_PORT;
        SessIni.local_port = 0; // A primeira porta que esteja disponível
        SessIni.authenticator = NULL;
        SessIni.callback = TrataSNMPTrap;
        SessIni.version = SNMP_VERSION_1;
        SessTrap = snmp_open( &SessIni );
        Se ( Não conseguiu abrir SessTrap ) {
                ("Erro fatal na abertura da sessao SNMP de Trap (snmp_open):");
        }

}
```

Assim como na criação de um agente, o construtor de um **Trap** preenche um conjunto de informações (atributos) necessários para criação de uma sessão SNMP. Apesar de ser o mesmo conjunto de informações, algumas delas são preenchidas com valores que são específicos para os *traps*, são eles:

remote_port – Porta UDP do *host* remoto utilizada para receber os *traps*. No caso do GCR é utilizada a porta padrão para recebimento de *traps*, 162;

local_port – Porta UDP do *host* local por onde será enviado o *trap*. Para GCR, é utilizado o valor 0 (zero), fazendo com que o próprio sistema operacional escolha a primeira porta disponível;

callback – É o endereço de uma função que nada deve fazer, já que o agente não trata nenhuma PDU de trap apenas as envia. O objetivo da existência desse atributo é apenas para manter a compatibilidade entre sessões SNMP.

Tendo sido preenchidos todos os atributos de maneira coerente, a criação de uma PDU de *trap* e o envio da mesma, é similar a criação e envio de uma PDU de resposta. Apenas o tipo de operação deve ser modificado, de maneira que o atributo *command* de uma PDU de *trap* seja a operação *trap*, e não um *get*, *set*, *getnext* ou *response*.

4.7 Gerente

Como apresentado no Capítulo 3 o gerente é a entidade ativa que coleta, periodicamente, os dados contidos na MIB de todos os agentes ativos associados às centrais a serem monitoradas. A Figura 4.9 apresenta o Modelo de Objetos parcial para o gerente do ambiente GCR.

A principal classe do Modelo de Objetos apresentado (Figura 4.9) é a classe **Gerente**, pois é a partir dela que se inicia toda funcionalidade do módulo gerente do ambiente GCR. É derivada da classe **Master** que é uma classe abstrata cuja existência é justificada pela necessidade de reaproveitamento de código.

A classe Gerente é formada por diversas outras classes:

BasePollingLista – Lista que contém o valor de *polling* para cada um dos tipos de centrais a serem monitoradas. Cada item dessa lista possuí dois atributos: o tipo de central (Tipo) e a freqüência de *polling* (ValPolling) para aquele tipo de central;

AlarmeLista – Lista de alarmes a serem tratados pela classe Gerente. Cada item dessa lista é um alarme que quando disparado indica que deve ser iniciada a coleta de dados de um determinado agente. Um item da classe **AlarmeLista** possuí os seguintes atributos: a

hora de realizar o *polling* (Hora), identificador da central (IdentCentral), porta UDP na qual o agente está em comunicação com o gerente (PortaUDP), tipo da central (TipoCentral) e o tempo de intervalo de *polling* (IntervaloPolling). É essa lista que proporciona o funcionamento do *Schedule* de *Polling* apresentado no Capítulo 3.

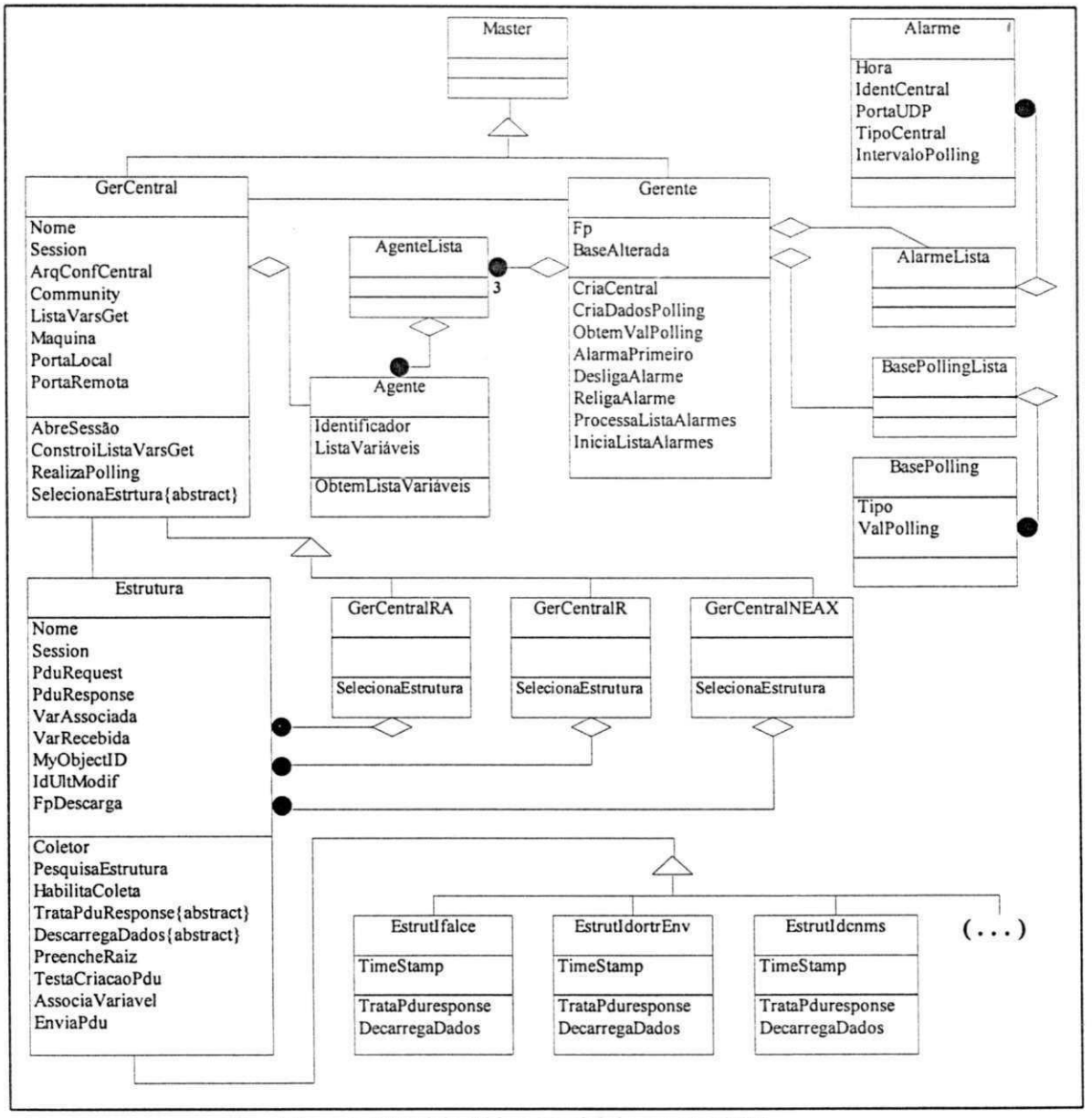

Figura 4.9 Modelo Objeto parcial do gerente GCR.

AgenteLista – Lista de agentes ativos (em funcionamento) de um determinado tipo. Cada item da classe **AgenteLista** é da classe **Agente** e possui dois atributos: o identificador do agente ativo (Identificador) e a lista de objetos da MIB que devem ser solicitados pelo

gerente ao seu respectivo agente ativo (ListaVariaveis). O **Gerente** possuí 3 (três) objetos da classe **AgenteLista**, um para cada tipo de agente questionado pelo gerente: agente RA, agente R e agente NEAX;

A classe **Gerente** possui ainda dois atributos: um apontador para um arquivo de configuração de *polling* (Fp), e um indicador de mudança na quantidade de agentes ativos.

Uma outra classe também importante do modelo de objetos do gerente é a classe **GerCentral**. É responsável por realizar a funcionalidade do gerente para cada tipo de agente associado as suas respectivas centrais monitoradas. A GerCentral possui os seguintes atributos: nome que identifica uma central e seu agente (Nome), uma sessão SNMP pelo qual serão enviados e recebidos os pacotes SNMP (Session), um apontador para um arquivo de configuração para um tipo de central gerenciada (ArqConfCentral), lista de objetos da MIB a serem solicitados para aquele tipo de central (ListaVarsGet), um *community name* para autenticação dos pacotes SNMP (*Community*), endereço IP no qual se encontra o agente interrogado (Maquina), a porta UDP local (PortaLocal) e remota (PortaRemota) utilizada na comunicação do agente com o gerente. É uma classe abstrata da qual se deriva as classes **GerCentralRA**, **GerCentralR**, **GerCentralNEAX**.

Além dos atributos herdados da classe **GerCentral**, cada uma de suas classes derivadas (**GerCentralRA**, **GerCentralR** e **GerCentralNEAX**) é constituída de um agregado de diversos objetos da classes derivadas de uma classe denominada **Estrutura**.

A classe **Estrutura** é a abstração da estrutura da MIB do GCR, no gerente. Um objeto dessa classe é responsável por realizar a coleta dos dados de gerência propriamente dita, de um determinado objeto da MIB, como também de repassar os dados coletados para uma camada de aplicação gráfica.

Dessa maneira, uma classe derivada de **GerCentral**, é agregada a objetos derivados da classe **Estrutura,** que representem objetos da MIB existentes para aquela determinada classe derivada de **GerCentral**.

Uma classe **Estrutura** possui os seguintes atributos: um nome que identifica a estrutura (Nome), uma sessão SNMP (Session) através da qual serão enviados os pacotes SNMP, uma estrutura de um pacote SNMP para requisições de dados da MIB (PduRequest), um outro pacote SNMP para tratar as repostas às requisições (PduResponse),

uma lista de variáveis a serem requisitadas (VarAssociada), um lista de variáveis recebidas em função das requisitadas (VarRecebida), o identificador do objeto da MIB representado pelo objeto **Estrutura** corrente (MyObjectID), o identificador do objeto da MIB que tem o valor a última hora de modificação dos valores representado pelo atributo MyObjectID (IdUltModif) e um apontador para um dispositivo ou bloco de *software* o qual serão repassados os dados coletados (FpDescarga).

Apesar de na construção dos agente ter-se utilizado o pacote para desenvolvimento de aplicações de gerência baseadas no protocolo SNMP da *Carnegie Mellon University*, o SNMPCMU, para construção da *interface* SNMP do gerente utilizou-se o pacote comercial do *framework HP OpenView* da *Hewlett Packard*.

4.7.1 Criação do Gerente

A criação do gerente se constitui da inicialização das classes que o compõem (**BasePollingLista**, **AlarmeLista** e **AgenteLista**) e da própria classe **Gerente**. Basicamente, são dois métodos da classe Gerente responsáveis por esta tarefa: o construtor da classe e o método IniciaListaAlarmes.

```
Gerente::Gerente()
{
        ...
        // Inicializa os atributos da classe
        Abrir(Fp) // Abrir arquivo de cofiguração de polling
        BaseAlterada = FALSE; // Base de agentes ativos sem alterações
        ...
        // Cria BasePollingLista
        CriaDadosPolling();
}
```

O construtor, além de realizar as alocações de memória necessárias para execução do gerente, inicializa os atributos da classe e invoca o método CriaDadosPolling que é responsável por ler do arquivo de configuração de *polling* (Fp) os intervalos de *polling* para cada tipo de central, e construir uma lista contendo essas informações. Pode-se observar o quadro que apresenta o construtor do Gerente.

O IniciaListaAlarmes cria as listas de agentes ativos para cada tipo de central do ambiente GCR e constrói a lista de alarmes em que cada item dessa lista representa quando devem ser realizados os *pollings* sobre cada agente ativo.

Similarmente ao *Schedule* de Comandos apresentado na seção 4.2.3 (Inicialização do conversor), a lista de alarmes é ordenada pelos tempos de freqüência de *polling*. É essa estrutura que garante o funcionamento do *Schedule* de *Polling* citado no Capítulo 3.

O método IniciaListaAlarmes ainda invoca um outro método denominado AlarmaPrimeiro cuja função é programar um *timer* para o menor tempo de freqüência, ou seja, para o primeiro agendamento da lista ordenada. Na programação do *timer* é acrescido o valor da hora atual. Esse *timer* é responsável por iniciar o funcionamento do *Schedule* de *Polling*.

```
Gerente::IniciaListaAlarmes()
{
    ...
    // Cria Alarmes para um tipo de central
    Loop: Para todos os agentes ativos faça {
                // Adicionando o Agente na sua respectiva Lista de Agentes
                Se ( agente é do tipo "RA" ) {
                        Adiciona agente na ListaAgentesRA
                }

                Se ( agente é do tipo "R" ) {
                        Adiciona agente na ListaAgentesR
                }

                Se ( agente é do tipo "NEAX" ) {
                        Adiciona agente na ListaAgentesR
                }

                // Adicionando um Alarme na Lista de Alarmes
                Cria Alarme para agente
                Adiciona Alarme na ListaAlarmes
    }
    AlarmaPrimeiro(); // Dispara o primeiro Alarme.
}
```

4.7.2 Polling dos Agentes

Quando o *timer* do alarme programado expira é iniciado um outro funcionamento do *Schedule* de *Polling* que é tratar o alarme expirado. Essa função é desempenhada pelo

método ProcessaListaAlarmes da classe Gerente cujo comportamento é apresentado no quadro a seguir.

```
Gerente::ProcessaListaAlarmes()
{
        Alarme = Primeiro item da ListaAlarmes;
        Enquanto ( Tem alarme na ListaAlarmes ) {
                HoraAtual = Hora do sistema;
                Se ( Alarme->Hora é maior que HoraAtual ) {
                        // É um alarme para depois
                        Sai do Loop;
                }
                // Cria um GerCentral em função do atributo TipoCentral da classe Alarme
                CriaCentral(Alarme);
                GerCentral->RealizaPolling();
                // Cria uma replica do Alarme vigente
                NovoAlarme = Duplica Alarme;
                Retira Alarme da ListaAlarmes
                Insere NovoAlarme na ListaAlarmes
                Alarme = Primeiro item da ListaAlarmes;
        }
        Se ( Existe Alarme) {
                // Programa novo timer
                ValorTimer = ( Alarme->Hora - HoraAtual );
                Timer (ValorTimer);
        }
}
```

O ProcessaListaAlarmes retira da ListaAlarmes os alarmes (elementos da lista) cujo valor do atributo Hora seja menor ou igual a hora atual do sistema. Para cada alarme retirado é criado um objeto derivado da classe GerCentral em função do atributo TipoCentral da classe Alarme, para a partir dele realizar o *polling* sobre o agente de um determinado tipo de central. Após a realização do *polling*, um novo Alarme é inserido na ListaAlarmes.

O método RealizaPolling varre todos os itens da **ListaVarsGet** que representam quais os objetos da MIB serão solicitados para um determinado tipo de central. Para todo item da **ListaVarsGet** é invocado o método SelecionaEstrutura específico de cada classe derivada da **GerCentral** com o objetivo de identificar qual o objeto derivado da classe **Estrutura** que deve ser ativado para realização da coleta propriamente dita. O quadro abaixo apresenta o RealizaPolling.

```
GerCentral::RealizaPolling()
{
        Enquanto ( Existe Variável na ListaVarsGet ) {
                Se ( SelecionaEstrutura( Variável ) é igual a COLETA_ERRO ) {
                        ("Erro na Realização do Polling");
                        Sai do laço;
                }
        }
}
```

A seguir é apresentado o método SelecionaEstrutura específico para classe GerCentralRA que em função da variável recebida, envia uma mensagem para seu respectivo objeto derivado da classe **Estrutura** para que este dispare o seu processo de coleta.

```
GerCentralRA::SelecionaEstrutura(VarName)
{
        Se ( VarName é igual "DadosFalhasEntry" ) {
                Retorna (EstrutIfalce->Coletor());
        }
        Se ( VarName é igual "DadosTrafExtEnvEntry" ) {
                Retorna (EstrutIdortrEnv->Coletor());
        }
        if ( strcmp(VarName,"DadosCodigoEntry") {
                Retorna (EstrutIdcnms->Coletor());
        }
        ...
        return(COLETA_ERRO);
}
```

4.7.3 Coleta dos dados da MIB

O método Coletor da classe **Estrutura** é utilizado por cada uma de suas classes derivadas afim de obter informações de gerência, dos objetos que representam, e que estão armazenadas na MIB. O comportamento do método se encontra no quadro a seguir.

Vale a ressalva de que todos os métodos chamados a partir do método Coletor da classe **Estrutura**, utilizam funções pertencentes a API do *framework HP OpenView* cuja funcionalidade é similar à API do pacote SNMPCMU usado na construção dos agentes, mudando apenas o nome das funções e estruturas de dados utilizadas. Contudo, as duas APIs mantém o mesmo objetivo que é o de prover suporte para o desenvolvimento de aplicações de gerência baseadas no protocolo SNMP. Como não é ênfase do trabalho

apresentar as funções da API do *framework HP OpenView*, limita-se aqui a apresentação dos métodos que utiliza tais funções.

Primeiramente o Coletor verifica se as informações de gerência contidos na MIB sofreram alguma mudança. Essa tarefa é desempenhada através do método HabilitaColeta que compara o *TimeStamp* da MIB com o atributo *TimeStamp* da classe **Estrutura**. Os valores sendo iguais revela que as informações de gerência não foram modificadas e que não há necessidade de se realizar coleta para o objeto. Esse mecanismo evita o desperdício de tráfego na rede de pacotes SNMP. A obtenção do valor do *TimeStamp* dos objetos da MIB é realizado através de uma solicitação SNMP do gerente acerca desse atributo e em conseqüencia, o agente envia uma resposta SNMP contendo o valor requisitado.

```
Estrutura::Coletor()
{
        // Verifica o valor do TimeStamp da MIB com o da estrutura corrente
        Se ( HabilitaColeta() igual a FALSE ) {
                ("TimeStamp não foi alterado");
                Retorna (COLETA_OK);
        }
        //Processo de Coleta
        Enquanto (TRUE ) {
                // Cria PDU utilizando a API do framework HP OpenView
                PduRequest = OVsnmpCreatePdu(GET_REQ_MSG);

                // Associa a PDU os atributos, a serem coletados, do objeto da MIB
                AssociaVariavel(VarAssociada, PduRequest, MyObjectID);

                // Envia a PDU de requisicao de dados
                RetEnvia = EnviaPdu();
                Se ( RetEnvia igual a FALSE ) {
                        Sai do loop;
                }
                Se ( RetEnvia igual a SNMP_ERR_NO_RESPONSE ) {
                        Retorna (COLETA_ERRO);
                }

                // Trata a PDU Respondida.
                TrataPduResponse();
                // Prepara os atributos para coleta de um novo conjunto de valores
                IncrementaVetorIds();
        };

        // Manipula todos os dados de um objeto
        DescarregaDados();
        Retorna (COLETA_OK);
}
```

Tendo sida habilitada a coleta, a tarefa agora se constitui em coletar todas as informações de gerência contidas no objeto.

Como o gerente, a princípio, não tem conhecimento da quantidade de informações de gerência, a coleta é feita até que não se tenha resposta (*timeout* de recebimento) para a variável solicitada ou ocorra algum erro de transmissão/recebimento de pacotes.

O processo de coleta acontece da seguinte forma:

- Cria-se primeiro um pacote PDU SNMP de solicitação denominada de PduRequest. Usa-se a função OvsnmpCreatePdu da API do *framework HP OpenView;*
- Associa-se a esta PDU variáveis que representam os atributos a serem coletados e que compõem o objeto na MIB. O método responsável por esta tarefa é o AssociaVariável;
- Envia-se à PDU de solicitação para o agente através de um mecanismo de envio síncrono, de maneira que o processo de coleta permanece bloqueado até que se obtenha uma resposta ou ocorra um *timeout* de recebimento. O método EnviaPdu desempenha esta função encapsulando a chamada de diversas funções da API HP OpenView;
- Obtém-se a PDU de resposta denominada de PduResponse e desta obtém os valores solicitados. Procedimento realizada pelo método TrataPduResponse;
- Prepara-se as variáveis para coleta de um novo conjunto de valores dos atributos do objeto questionado. O método IncrementaVetorIds é quem executa esta tarefa.

Após a coleta de todas as instâncias de valores dos atributos do objeto, cabe ao gerente manipular as informações obtidas da forma que lhe convir. Uma delas é repassá-la à camada de aplicação gráfica. Essa tarefa é desempenhada pelo gerente através do método DescarregaDados específico de cada uma das classes derivada da superclasse **Estrutura**.

É a camada de aplicação que dará um tratamento mais amigável e legível as informações, de maneira a facilitar o entendimento das mesmas pelos usuários (operadores) do ambiente GCR.

4.8 Conclusão

Neste capítulo foram descritos, em alto nível, detalhes de implementação para os componentes de *software* dos módulos que fazem parte da solução. Para cada componente funcional descrito no Capítulo 3 foi apresentada sua hierarquia de classes, por meio de Modelos de Objetos, com seus respectivos atributos e comportamentos.

5 CONCLUSÕES

Os atuais sistemas de telecomunicações têm evoluído continuamente o que leva a um ambiente composto por redes e equipamentos heterogêneos. Esta heterogeneidade implica na complexidade de se gerenciar este ambiente. As soluções proprietárias dos fabricantes tornam a gerência de redes de telecomunicações ineficiente, envolvendo problemas como, por exemplo: a presença de múltiplas interfaces para diferentes sistemas, sistemas não interoperáveis, insuficiência de informações coletadas, bases de dados específicas e isoladas com informações redundantes e inconsistentes.

O ambiente de competitividade vem obrigando muitas empresas a adotarem padrões que possibilitem uma gerência integrada de rede, proporcionando a interoperabilidade de sistemas de diferentes fabricantes. A gerência integrada de rede exige que os fabricantes forneçam seus produtos dentro de padrões estabelecidos por recomendações com interfaces padronizadas que possibilitam a interação com sistemas de gerenciamento de outras empresas.

Apesar de existirem normas e estudos referentes ao modelo TMN, que determinam e especificam o que e como deve ser a padronização, a prática e mundo real revelam que a existência desses padrões ainda não é uma realidade disponível de fato, e o seu desenvolvimento requer um investimento elevado de recursos, pessoas, tempo, ferramental e financeiro que não se justifica nas atuais plantas de telecomunicações existentes no mercado.

O trabalho desenvolvido nesta dissertação apresentou uma solução para realização de gerência de desempenho em redes de telefonia, de baixo custo, para uma planta heterogênea no qual os recursos disponíveis não possibilitam aderir aos padrões TMN, além de minimizar as características indesejáveis existentes nos sistemas proprietários de gerência. A solução ainda pode ser considerada um OS (Sistema de Operações) que funciona como parte integrante de uma (GIRS) Gerência Integrada de Redes e Serviços de Telecomunicações.

A solução é um sistema de *software* que, para realizar a gerência de desempenho de rede, monitora de cada uma das centrais que fazem parte do escopo de gerência (centrais do

tipo: NEAX 61BR fabricada pela NEC, Trópico-RA e Trópico fabricadas pelo CPqD), o tráfego e a taxa de utilização de seus órgãos. São utilizados mediadores (conversores) específicos que preenchem um modelo de informação único com as informações coletadas de cada central. Sobre o modelo de informação realizam-se consultas através de primitivas SNMP, cujo resultado é analisado para se determinar a eficiência global da rede. A solução permite ainda que a partir de um ponto único seja possível ter acesso a todas as centrais gerenciadas.

5.1 Comentários Sobre a Solução

No Capítulo 2 deste trabalho foram apresentados os requisitos exigidos para a construção do ambiente de gerência. Os parágrafos a seguir descrevem como o GCR atende a cada um deles, de maneira total ou parcial, citando algumas observações nos pontos mais importantes:

- Uso da própria rede de telecomunicações para gerenciar a planta telefônica– Não foi necessária a criação de nenhuma rede adicional de comunicação. Toda comunicação entre os elementos de rede gerenciados e o ambiente é realizada via linha telefônica convencional;

- O escopo de abrangência da gerência – O ambiente consegue interagir e obter dados de gerência das principais centrais da planta para a qual foi desenvolvido (Telemar Paraíba) : NEAX (tecnologia NEC), Trópico-R e Trópico-RA (tecnologia nacional CPqD);

- Possui um modelo único de informação – Esse requisito foi alcançado por meio do uso do conceito de MIB do protocolo SNMP. Apesar da existência de elementos de redes de diferentes fornecedores, o modelo de informação desenvolvido foi criado de maneira genérica para fornecer subsídios a realização de Gerência de Desempenho sobre a rede, e não para um elemento específico;

- Projeto e implantação modulares – O desenvolvimento e implantação se deu em função de cada componente da arquitetura funcional do GCR;

- Uso de *hardware* e *software* que se baseiem nos padrões de sistemas abertos – Foram utilizadas plataformas de software que se consolidaram mercadologicamente como padrões: modelo de gerência Gerente/Agente, modelo de infomação caracterizado pelo conceito de MIB, protocolo de gerência SNMP;

- Baixo custo – Contabilizando pessoal, tempo, ferramental e dinheiro, os gastos no desenvolvimento da solução foram inferiores as soluções proprietárias existentes, como também inferiores ao desenvolvimento de soluções baseadas no padrão TMN;

- Proporcionar melhora na qualidade do serviço de telefonia da planta – Problemas na rede que antes eram identificados em até quinze dias, com o ambiente, passaram a ser detectados em duas horas após a sua causa. É o caso de identificação de rotas sobrecarregadas e de juntores defeituosos. Além disso, o GCR possibilita identificar áreas de difícil acesso, as HTR que podem congestionar a rede como um todo e que reduzem a receita da operadora.

Durante o desenvolvimento do ambiente de gerência GCR foram encontradas algumas dificuldades que pelo grau de complexidade valem ser ressaltadas:

A comunicação entre as centrais e o ambiente de gerência - Realizar esta comunicação foi um processo demorado em relação ao processo de construção dos demais módulos do ambiente do GCR. O módulo responsável pela comunicação é o conversor específico de cada central. Houve dois grandes problemas na implementação dos conversores: o primeiro foi como fazer para criar sessões de acesso em diferentes tipos de central e o outro era como conseguir fazer com que os conversores entendessem que a sequência de caracteres enviados pela central que formam os relatórios e como de cada relatório específico seria possível extrair apenas os dados importantes para realização de gerência de desempenho.

Definição do modelo de informação – Definir um modelo de informação que não atendesse às características específicas de um determinado tipo de central, mas que abrangesse de maneira genérica todas as características necessárias para realização de gerência de desempenho foi um tanto difícil. Na literatura pesquisada e utilizada, até o momento da escrita dessa dissertação, não existe uma definição formal de um modelo de

informação que forneça suporte a realização de gerência de desempenho em plantas de centrais digitais heterogêneas.

5.2 Contribuições

O trabalho desenvolvido trouxe contribuições importantes tanto no âmbito acadêmico quanto no mercadológico. Basicamente, pode-se listar as seguintes contribuições:

- Apresentação do panorama multi-fornecedor das atuais plantas de telecomunicações, enfatizando os principais problemas decorrentes dessa heterogeneidade, e as soluções existentes para resolver esses problemas;
- Desenvolvimento de um modelo conceitual para realização de gerência em redes de telefonia utilizando a arquitetura SNMP;
- Construção de um modelo de informação genérico que possibilite a realização de gerência de desempenho em redes de telefonia heterogêneas;
- Implementação e implantação de um sistema de operações, a solução desenvolvida, para gerenciar a taxa de utilização da rede, contribuindo para o processo de implantação de GIRS na empresa para a qual foi desenvolvido.

5.3 Sugestões para Continuação do Trabalho

No intuito de aprimorar a solução desenvolvida neste trabalho para Gerência de Desempenho em Redes de Telefonia Heterogêneas, são sugeridos alguns encaminhamentos futuros que devem contribuir para a evolução do ambiente GCR:

- Expansão e melhoramento do modelo de informação da MIB – Incluir novos objetos no atual modelo de informação do GCR, possibilitará estender a funcionalidade de gerência do mesmo. Seria possível adicionar ao modelo, objetos que pudessem armazenar informações sobre: falhas e alarmes existentes nas centrais, configuração de rotas e planos de encaminhamento de chamadas, além de informações a cerca de tarifação sobre os serviços oferecidos aos usuários.

- Aprimoramento do mecanismo de *polling* do Gerente – Em alguns casos, o *polling* sobre os agentes de determinadas centrais é bastante longo. Um estudo para se determinar *slots* (intervalos) de tempo máximo para que o Gerente permaneça realizando *polling* sobre um mesmo agente sem comprometer o *polling* sobre os demais agentes ativos seria de grande contribuição, além de garantir a escalabilidade do ambiente GCR;

- Desenvolvimento de conversores para CPAs de outros fabricantes – A construção de novos conversores para diferentes CPAs, possibilitará o aumento do atual escopo de gerência do GCR. Dessa forma, o ambiente poderá ser utilizado em diversas plantas telefônicas, além de que terá o potencial para gerenciar redes cujo grau de heterogeneidade seja acentuado. Podem ser construídos conversores para CPAs dos fabricantes BATIK, Zetax, Alcatel e Ericcson, por exemplo.

- Estender a gerência a outros níveis da estrutura de GIRS – A funcionalidade atual de gerência do GCR possibilita atender até o nível de Gerência de Rede, contudo pode-se estender essa funcionalidade de gerência para os níveis de Gerência de Serviço e de Negócio. Para tanto, é necessário estender o modelo de informação com objetos que retenham informações acerca de Gerência de Serviço e de Negócio, e que os conversores consigam obter essas informações das CPAs que fazem parte da rede ou através de algum processamento sobre um conjunto de dados.

- Possibilitar integração do GCR com as tecnologias da (Web) *World Wide Web* – A idéia seria permitir que a partir de um *browser* (Netscape, Explorer) se possa ter acesso às funcionalidades do ambiente de gerência e às informações de gerência coletados pelo gerente. Para tanto, seria necessário construir *applets* Java específicos que funcionassem como consoles de gerenciamento remoto nos *Web browsers*, de forma que o gerente pudesse disponibilizar as informações centralizadas por ele a partir dos dados solicitados aos agentes, através dos protocolos HTTP (*Hyper Text Transfer Protocol*) e TCP/IP;

- Adequação do GCR ao modelo TMN – Apesar de não seguir as padronizações do modelo de gerência TMN para redes de telefonia, é factível o

estudo e a implementação de um Bloco de Função Adaptadora Q (vide Apêndice A) para o ambiente de gerência GCR como um todo, cuja função seria integrar o GCR com arquiteturas de redes que seguem o modelo TMN. Obtendo–se essa integração o GCR seria considerado uma arquitetura de gerência TMN-*like* [M3010];

- Adequação de soluções proprietárias de gerência ao ambiente GCR – É comum encontrar sistemas de gerência proprietários nas atuais plantas de telefonia que realizam gerência apenas sobre os seus elementos de rede. Um trabalho interessante seria a construção de conversores específicos para tais sistemas cuja função seria integrá-los ao ambiente GCR. Esses conversores, basicamente, obteriam dos sistemas proprietários as informações necessárias para o preenchimento do modelo de informação do GCR.

Dessa forma, pode-se afirmar que a TMN é uma arquitetura organizada que viabiliza a aplicação de uma Gerência Integrada de Redes e Serviços atendendo aos seus objetivos e a sua estrutura funcional.

Arquitetura Geral do Modelo TMN

Na arquitetura geral do modelo TMN existem três aspectos arquiteturais que devem ser considerados: aspecto funcional, de informação e físico. A cada um desses aspectos é associada uma padronização para a formação do modelo arquitetural

Arquitetura Funcional

Esse modelo descreve a distribuição das atividades da TMN em blocos funcionais que possibilitem a implantação de TMNs de qualquer complexidade. É baseado em três conceitos fundamentais: bloco funcional, componente funcional e ponto de referência.

Os blocos funcionais representam agrupamentos de funções gerais da TMN, dando modularidade para implementação das diversas funcionalidades de uma arquitetura de gerência de telecomunicações. Cada bloco funcional é formado por um conjunto de componentes funcionais. Os blocos funcionais do modelo são:

OSF (***Operations Systems Function***) – Função Sistema de Operações: Representa o processamento de informação (associado a alguma aplicação de gerência) para monitorar, coordenar e /ou controlar funções de telecomunicações e/ou funções de gerência;

NEF (***Network Element Function***) – Função de Elemento de Rede: São as funções desempenhadas pelos recursos de telecomunicações, trocando informações com a TMN para permitir a gerência dos recursos. Os recursos em si não fazem parte da TMN;

QAF (***Q Adaptor Function***) – Função Adaptador Q: É usado para conectar à TMN e entidades não-TMN que possuam funções similares aos OSF e NEF;

MF (***Mediation Function***) – Função de Medição: Armazena, adapta, filtra e condensa as informações trocadas entre os blocos funcionais NEF (ou QAF) e OSF para garantir uma comunicação segura entre eles;

WSF (***Workstation Function***) – Função de Estação de Trabalho: Provê mecanismos para interpretação de informações da TMN pelo usuário e vice-versa.

Uma função de grande importância para a TMN, mas não considerada bloco funcional é a **DCF** (***Data Communication Function***), Função de Comunicação de Dados. A DCF transporta, sem modificar, as informações trocadas entre os blocos funcionais.

Os blocos funcionais são constituídos de agrupamentos de funções de caráter mais específico, denominados componentes funcionais. Esses componentes são os elementos básicos dos blocos funcionais da TMN:

MAF (***Management Application Function***) – Função de Aplicação de Gerência;

ICF (***Information Conversion Function***) – Função de Conversão de Modelo de Informação;

WSSF (***Workstation Support Function***) – Função de Suporte à Estação de Trabalho;

UISF (***User Interface Support Function***) – Função de Suporte à Interface do Usuário;

MCF (***Message Communication Function***) - Função de Comunicação de Mensagem;

DSF (***Directory System Function***) – Função de Sistema de Diretório;

DAF (***Directory Access Function***) – Função de Acesso à Dirctório;

SF (***Security Function***) – Função de Segurança.

Para delinear os limites entre os blocos de funções, tem-se o conceito de pontos de referência. É por intermédio dos pontos de referências que ocorrem as trocas de informações entre os blocos de funções.

Três classes de pontos de referência da TMN são apresentadas:

q – pontos entre OSF, QAF, MF e NEF;

f – pontos entre OSF ou MF e a WSF;

x – pontos entre OSFs de duas TMNs ou entre um OSF de uma TMN e um OSF-like (equivalente a OSF em funcionalidade) de uma outra rede de gerência.

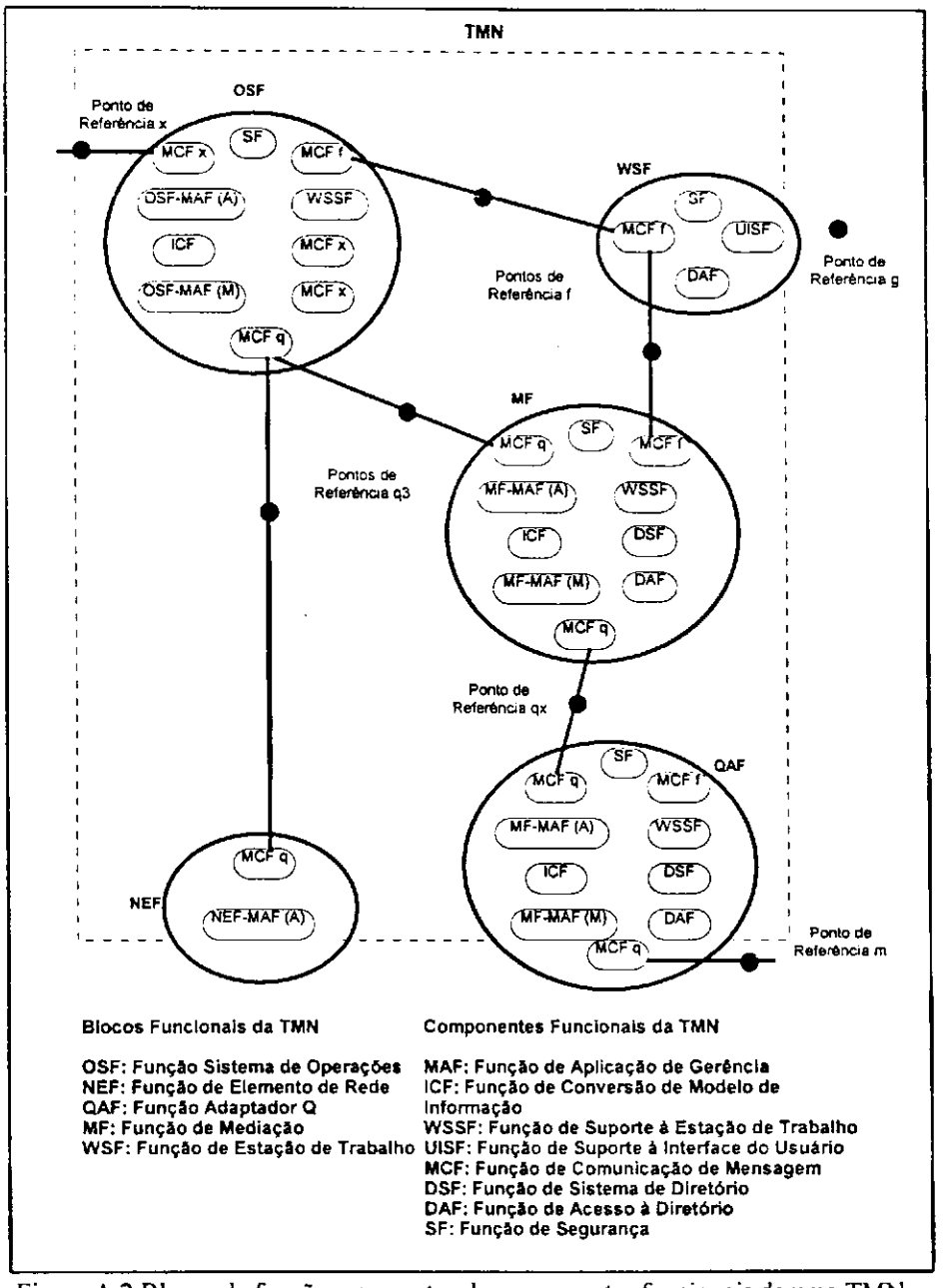

Figura A.2 Blocos de funções compostos de componentes funcionais de uma TMN.

Inseridos na classe q de pontos de referência tem-se os seguintes pontos:

$\mathbf{q_x}$ – pontos entre NEF e MF, QAF e MF e entre MF e MF;

$\mathbf{q_3}$ – pontos entre NEF e OSF, QAF e OSF, MF e OSF, e OSF e OSF.

Além destas, existem duas outras classes de pontos de referência que não pertencem aos trabalhos de especificação da TMN no ITU-TS, mas que são importantes para a compreensão de toda a funcionalidade envolvida no modelo. São elas:

g – pontos entre WSF e usuários;

m – pontos entre QAFs e entidades gerenciadas não-TMN.

A Figura A.2. apresenta um exemplo de uma arquitetura funcional de uma TMN, apresentando os blocos funcionais constituídos de seus respectivos componentes funcionais e interagindo através dos pontos de referência.

Arquitetura de Informação

O gerenciamento de um ambiente de telecomunicações é uma aplicação de processamento de informação. Sendo a rede de telecomunicações em si um ambiente distribuído, seu gerenciamento é intrinsecamente uma aplicação distribuída. Isto envolve a troca de informações de gerenciamento entre processos de gerenciamento para finalidade de monitoração e controle de vários recursos físicos e lógicos de rede como por exemplo, recursos de comutação e transmissão.

A Arquitetura de Informação da TMN descreve um modelo orientado a objeto para modelagem da informação de gerência trocada entre blocos funcionais da TMN, utilizando o modelo de Gerência de Sistemas OSI. Assim, todas as trocas de informações entre gerentes (entidades gerenciadoras) e agentes (entidades gerenciadas) obedecem a um conjunto consistente de operações de gerenciamento e notificações, efetuado sempre por meio do Serviço e Protocolo de Informações de Gerenciamento Comum (CMIS/CMIP – *Commom Management Information Service/ Commom Management Information Protocol*).

É por intermédio da padronização das estruturas lógicas que compreendem os sistemas gerenciador e gerenciado, bem como a comunicação entre eles que se consegue de fato, a compatibilidade dos sistemas de diferentes fabricantes (ambiente multi-fornecedor) dentro da filosofia TMN.

Arquitetura Física

O modelo genérico de arquitetura física define os elementos físicos e as interfaces que permitem interligá-los. Esses elementos físicos constituem os Blocos de Implementação TMN. Tais blocos, bem como as interfaces padrões da TMN, são apresentados na Figura A.3, que ilustra exemplos de arquitetura física. Os blocos de implementação representam implementações de funcionalidades (blocos funcionais) da TMN.

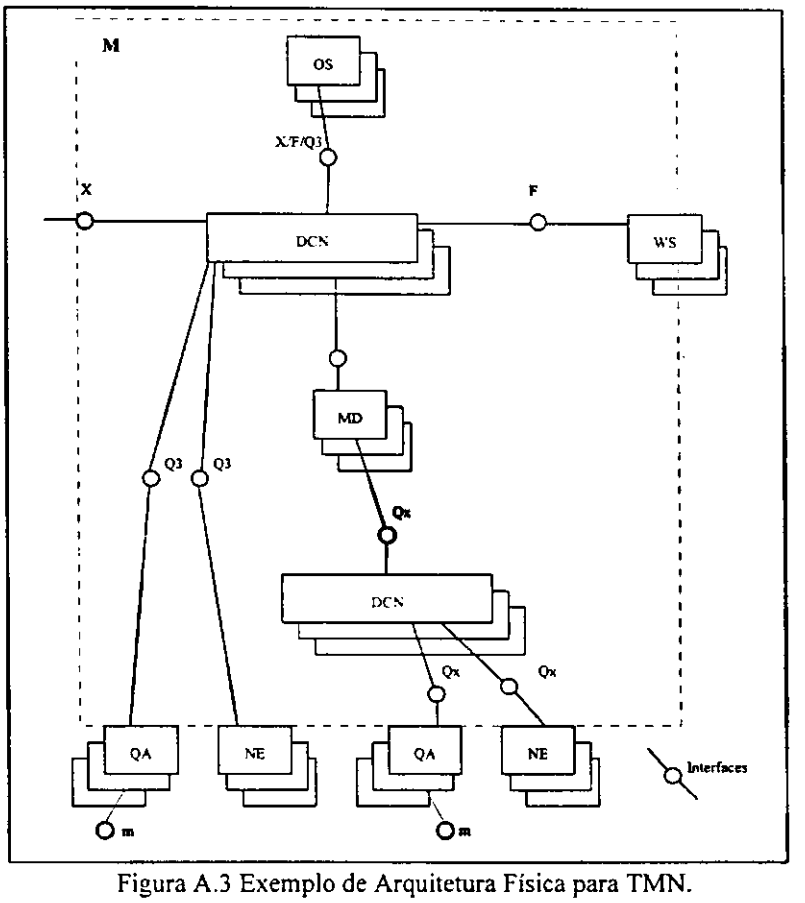

Figura A.3 Exemplo de Arquitetura Física para TMN.

A arquitetura física deve agrupar os blocos funcionais em entidades físicas visando atingir requisitos de flexibilidade, como por exemplo, adaptação a diferentes organizações empresariais e gerência de redes de complexidade variada. Um elemento físico da TMN pode implementar mais que um bloco funcional da arquitetura funcional e um ponto de referência passa a ser uma interface quando estiver entre dois elementos físicos da arquitetura física.

APÊNDICE B

SNMP - SIMPLE NETWORK MANAGEMENT PROTOCOL

O Gerenciamento das Redes de computadores atuais exige maneiras de simplificar os complexos problemas de comunicação e sincronização inerentes a esses tipos de aplicações. Uma das mais difundidas infra-estruturas que oferece suporte ao gerenciamento de redes **IP** (*Internet Protocol*) é o modelo que se baseia no protocolo **SNMP** (*Simple Network Management Protocol*). Este protocolo, criado em 1988, foi desenvolvido com o objetivo de que seus mecanismos pudessem ser facilmente implementados, com baixo *overhead* de trabalho, pelos diversos fornecedores de roteadores, servidores, *workstations* e outros recursos de redes. Em sua especificação a arquitetura SNMP define:

- Entidades gerenciadas denominadas agentes e os gerenciadores chamados gerentes;
- Um protocolo para troca de informações entre um ou mais sistemas de gerência e agentes;
- Um *framework* para formatação e armazenamento das informações de gerência;
- Um número de variáveis ou objetos de propósito geral com informações de gerência.

O modelo de gerência de rede utilizado pelo SNMP constitui-se dos seguintes elementos chaves: estação de gerência ou gerente; agentes; base de informação de gerência (MIB) e protocolo de gerência.

Gerente

Um gerente é um nodo que tem uma ativa participação no gerenciamento da rede. Ele solicita, ao agente, e interpreta dados sobre os dispositivos e tráfego na rede. O gerente também dispara ações no agente através de mudanças dos valores de variáveis gerenciadas pelo agente. Além disso, um gerente detém um sumário de todos os objetos gerenciados de cada uma das entidades gerenciadas.

REFERÊNCIAS BIBLIOGRÁFICAS

[Alen98] Alencar, Marcelo S. **Telefonia Digital**. Editora Érica Ltda. São Paulo, 1998.

[Alvi93] Alvin, Helvécio. **A Informática na Gerência Integrada de Redes e Serviços**. Revista Telebrás, dezembro de 1993.

[Bach86] Bach, Maurice J. ***The Design of the UNIX Operating System.*** Englewood Cliffs, Prentice-Hall, New Jersey,1986.

[BRISA93] BRISA (Sociedade Brasileira para Interconexão de Sistemas Abertos). **Gerenciamento de Redes - Uma Abordagem de Sistemas Abertos**. Makrom Books do Brasil Editora Ltda, 1993.

[Frei93] Freitas, José P.; Ishibashi, Walter W. **Princípios de Construção de Software para Gerência Integrada de Redes e Serviços**. Revista Telebrás, dezembro de 1993.

[Ghla98]Ghlamalah, Adel; Boutaba, Raouf. ***Implementing a Distributed Web-based Management System in Java***. SBT/IEEE International Telecommunications Symposium, 1998.

[Jato98] Jatobá, A. Alessandro. **Um Arcabouço Cooperativo para uma Sociedade de Agentes Tutores**. Dissertação de Mestrado. UFPB/CCT/COPIN, maio de 1998.

[Jr98] Jr., A. B. Dalmer. **Desenvolvimento e Adaptação de Aplicações Visando a Gerência com SNMP: Análise de Requisitos e Estudo de Caso para um Sistema de Impressão.** Dissertação de Mestrado. UFPB/CCT/COPIN, outubro de 1998.

[Light97] Documentação Técnica da Light Infocon Tecnologia S/A. **O Processo de Desenolvimento de *Software* na Light Infocon Tecnologia S/A**, 1997.

[Lope93] Lopes, Francisco J.S.; Jaime, Joaquim R.; Romariz, Renata E. **Planejamento de Sistemas de Operações**. Revista Telebrás, dezembro de 1993.

[M3010] Recomendação ITU-T M.3010. ***Principles for a Telecommunications Network.***

[M3020] Recomendação ITU-T M.3020. ***TMN Interface Specification Methodology.***

[M3100] Recomendação ITU-T M.3100. ***Generic Network Information Model Network.***

[M3200] Recomendação ITU-T M.3200. ***TMN Management Services – Overview.***

[M3400] Recomendação ITU-T M.3400. ***TMN Management Functions.***

[Mach93] Machado, Iara. **Segurança e Gerência de Segurança no Ambiente TMN**. Revista Telebrás, dezembro de 1993.

[Meir95] Nogueira, José Marcos S.; Meira, Dilmar M.; Vuong, Son T. ***On Telecommunications Network Management.*** *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing.* Victoria - Canadá, maio de 1995.

[Mend99] Mendes, Carlos L. dos S; Fernandes, Antônio O.; Coelho Jr., Claudionor N. **SupsWeb: Gerenciamento de Nobreaks Baseado na Web.** XVII Simpósio Brasileiro de Telecomunicações, Vitória - Espírito Santo, Setembro 1999.

[NEAX] **NEAX 61BR, Manual de Comandos.** Nec do Brasil S/A.

[Noga93] Nogay, Júlio; Pereira, Sérgio R. **Arquitetura para Sistemas de Operações.** Revista Telebrás, dezembro de 1993.

[Nogu97] Nogueira, José Marcos S. ***Integrating legacy systems: Networks elements and supervisory systems into new management systems or When you don't want to replace everything.*** *The Fifth IFIP/IEEE International Symposium on Integrated Network Management*, San Diego, CA, USA, 12-16 maio de 1997.

[Oliv99] Oliveira, Eduardo J.; Pinto Alexandre S.; Faina, Luís F.; Cardozo Eleri. **Desenvolvimento de Serviços de Telecomunicações Utilizando a Arquitetura TINA.** Simpósio Brasileiro de Telecomunicações, setembro de 1999.

[OMT91] Rumbaugh, James; Blaha Michael; Premerlani, William; Eddy, Frederick; Lorensen William. **Object-Oriented Modeling and Design.** Englewood Cliffs, Prentice-Hall, 1991.

[OMT96] Bakker, Gerbrand. **OMT Object Model.** http://www.comp.mq.edu.au/courses/comp331/documentation/OMT, Última modificação fevereiro de 1996.

[Pack], *Packard Hewllet.* ***Cu - Calls another (UNIX) system****; Man Pages.*

[Pero93] Perotoni, Ivanez A.; Simons, Humberto. **A Necessidade de Sistemas de Supervisão e Gerência de Rede nas Empresas Operadoras de Telecomunicações.** Revista Telebrás, dezembro de 1993.

[Rama93] Ramalho, Eduardo A. **Gerência Integrada de Redes e Serviços.** Revista Telebrás, dezembro de 1993.

[Rebe93a] Rebelles, Paulo R. L.; Freitas, José Pedro. **Introdução aos Modelos Genéricos de Arquitetura para a Gerência de Redes de Telecomunicações(TMN).** Revista Telebrás, dezembro de 1993.

[Rebe93b] Rebelles, Paulo R.L.; Freitas, José P. **Modelagem de Informação Aplicada a Gerência Integrada de Redes de Telecomunicações**. Revista Telebrás, dezembro de 1993.

[Rose91] Rose, M. T. ***The Simple Book:*** *An Introduction to management of TCP/IP-based Internets*. Englewood Cliffs, Prentice-Hall, 1991.

[Ruiz99] Ruiz, Linnyer Beatrys. **Minicurso: Rede de Gerência de Telecomunicações (TMN)**. XVII Simpósio Brasileiro de Telecomunicações, Vitória - Espírito Santo, Setembro 1999.

[Sala94] Salah Aidarous and Thomas Plevyak. ***Telecommunications Network Management into the 21st Century:*** *Techniques, Standards, Technologies, and Applications*.The Institute of Electrical and Electronics Engineers, Inc.,New York, USA, 1994.

[Sant95] Santos, Katyusco F. **Relatório de Estágio Supervisionado**. Universidade Federal da Paraíba - Campus II .Departamento de Ciência e Computação, 1995.

[Sant99a] Santos, Katyusco F.; Alencar, Marcelo S.; Moura, J. Antão B. ***An Operating System for Heterogeneous SPC Exchanges Based on SNMP***. 2ª Conferência Internacional de Telecomunicações (Conftele99), Sesimbra - Portugal, Abril 1999.

[Sant99b] Santos, Katyusco F.; Alencar, Marcelo S.; Moura, J. Antão B. **Um Sistema de Operações para CPAs Heterogêneas Baseado em SNMP**. XVII Simpósio Brasileiro de Telecomunicações (SBT99), Vitória - Espírito Santo, Setembro 1999.

[SDT501-100-104] Sistema de Documentação Telebrás 501-100-104. Série: Planta. **Prática: Conceitos de Gerência Integrada de Redes de Telecomunicações.**

[SDT501-100-105] Sistema de Documentação Telebrás 501-100-105. Série: Planta. **Prática: Princípios para Implantação de Gerência Integrada de Redes de Telecomunicações.**

[SDT501-100-107] Sistema de Documentação Telebrás 501-100-107. Série: Planta. **Prática: Gerência Integrada de Redes e Serviços. Ações na Planta Digital.**

[SDT501-100-110] Sistema de Documentação Telebrás 501-100-110. Série: Planta. **Prática: Requisitos Básicos de Arquitetura Computacional para a Gerência Integrada de Redes e Serviços.**

[SDT210-001-059] Sistema de Documentação Telebrás 210-001-059. Série: Engenharia. **Prática: Especificação de Perfis Funcionais de Protocolos da Interface Qx para Conexão de Equipamentos/Sistemas de Rede de Telecomunicações à Rede de Gerenciamento.**

[SDT210-001-060] Sistema de Documentação Telebrás 210-001-060. Série: Engenharia. **Prática: Especificação de Perfis Funcionais de Protocolos da Interface Q3 para Conexão de Equipamentos/Sistemas de Rede de Telecomunicações à Rede de Gerenciamento.**

[Shön98] Shönberger, Selena; Schweitzer, Alessandra; Ferreira, Alexandre P. ***Flexos, A Telecommunications Network System***. SBT/IEEE International Telecommunications Symposium, 1998.

[Stal93] Stallings William. ***SNMP, SNMPv2, and CMIP****: The Practical Guide to Network Management Standards*. Addison Wesley, 1993.

[Stev90] Stevens, W. R. ***Unix Network Programming*** – Prentice Hall,1990.

[Strou93] Stroupstrup, Bjarne; Margaret A. Llis. **C++: Manual de Referência Comentado**. Editora Campus. Rio de Janeiro, 1993.

[Taka93] Takanohashi, Marcos. **Padronização em Gerenciamento de Redes de Telecomunicações Estudo e Especificação de uma Interface Q3 da TMN**. Dissertação de mestrado. Escola Politécnica da Universidade de São Paulo. 1993.

[Tane92]Tanenbaum, Andrew S.. ***Modern Operating Systems.*** Englewood Cliffs, Prentice-Hall, New Jersey,1992.

[Telerj] TELERJ-Telecomunicações do Rio de Janeiro S/A. **Documento: Requisitos para Gerência de Rede.** 1994.

[Town95] Townsend, Robert L. ***SNMP Application Developer's Guide.*** VNR Communications Library 1995.

[TrR] **Trópico-R, Manual de Operação, Manutenção e Supervisão, Vol.3.** Elebra Telecom Ltda.

[TrRA] **Trópico-RA**, Documentação de Sistema, Vol.1. STC Telecomunicações Ltda.