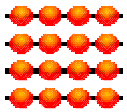
UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO



Dissertação de Mestrado

ÁTILA, Uma Arquitetura Inteligente e Distribuída para o Gerenciamento Pró-ativo de Redes ATM

por

Marcelo Vidal Vasconcelos

Orientador: Prof. Dr. Mauro Oliveira

Co-Orientador: Prof. Msc. Cidcley Teixeira de Souza

Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Mestrado em Ciências da Computação

Marcelo Vidal Vasconcelos

ÁTILA, Uma Arquitetura Inteligente e Distribuída para o Gerenciamento Pró-ativo de Redes ATM

Este trabalho será apresentado à Pós-Graduação em Ciências da Computação do Centro de Ciências da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Mestre em ciências da Computação.

Orientador: Prof. Dr. Mauro Oliveira
Co-Orientador: Prof. Msc. Cidcley Teixeira de Souza

Resumo

Este trabalho questiona o atual modelo de gerência ATM e propõe o ÁTILA, uma nova arquitetura para atender os requisitos de redes baseadas nesta tecnologia.

Três visões são utilizadas na proposição da nova arquitetura. A primeira é uma visão conceitual, independente de implementação. As Arquiteturas Funcional e Física, as quais são melhor explicitadas com o Fluxo Dinâmico de Informações representam esta visão. A segunda visão refere-se às tecnologias selecionadas para a implementação do modelo conceitual das arquiteturas.

Finalmente, a terceira visão mostra o funcionamento do ÁTILA através de ferramentas escolhidas, dentre as tecnologias selecionadas (na segunda visão), para a sua prototipagem.

Sumário

Capítulo 01 – Introdução 01

1.1 Um Pequeno Histórico 01

1.2 Necessidade de Gerenciamento 02

1.3 Identificação do Problema 03

1.4 Organização da Dissertação 04

PARTE I – Estado Atual da Gerência ATM 07

Introdução 07

Capítulo 02 - Gerenciamento de Redes 08

2.1 Introdução 08

2.2 SNMP (Simple Network Management Protocol) 09

2.3 CMIS/CMIP (Common Management Information Service/Protocol) 11

2.4 Áreas Funcionais 12

Capítulo 03 – ATM (Asynchronous Transfer Mode) 16

3.1 Introdução 16

3.2 Modelo da Arquitetura dos Protocolos RDSI-FL 17

3.3 Interfaces ATM 18

3.4 Conexões ATM 19

3.5 Organização dos Protocolos 20

3.6 LAN Emulation (LANE) 26

3.7 MPOA (Multi-Protocol Over ATM) 28

3.8 Endereçamento, Sinalização e Roteamento 29

3.9 Categoria de Serviço e Contrato de Tráfego 30

Capítulo 04 – Gerenciamento de Redes ATM 34

4.1 Introdução 34

4.2 Áreas Funcionais de Gerência ATM 36

4.3 Abordagens Atuais para Gerência ATM 39

4.3.1 ILMI (Integrated Local Management Interface) 40
4.3.2 Células OAM 41
4.3.3 Modelo de Referência de Gerenciamento de Redes ATM 42
4.4 Gerenciamento de Tráfego 44

PARTE II – Descrição do ÁTILA 46

Introdução 46

Capítulo 05 – Arquitetura 47

5.1 Introdução 47
5.2 Arquitetura Funcional 47
5.2.1 Diagrama da Arquitetura Funcional 47
5.2.2 Blocos Funcionais 48
5.2.2.1 BAU (Bloco Acesso ao Usuário) 48
5.2.2.2 BIG (Bloco Inteligência do Gerente) 49
5.2.2.3 BIA (Bloco Inteligência do Agente) 49
5.2.2.4 BER (Bloco Elemento de Rede) 50
5.2.3 Interfaces 50
5.2.3.1 Interface “G1” 50
5.2.3.2 Interface “G2” 51
5.2.3.3 Interface “G3” 51
5.3 Arquitetura Física 51
5.3.1 Diagrama da Arquitetura Física 51
5.3.2 Elementos Físicos 52
5.3.2.1 EAU (Elemento Acesso ao Usuário) 52
5.3.2.2 EIG (Elemento Inteligência do Gerente) 53
5.3.2.3 EIA (Elemento Inteligência do Agente) 53
5.3.2.4 ER (Elemento de Rede) 54
5.3.2.5 Plataforma de Distribuição 54

Capítulo 06 - Fluxo Dinâmico de Informações 55

6.1 Introdução 55
6.2 Diagrama 55

6.3 Exemplificação 56

PARTE III – Implementação do ÁTILA 65

Introdução 65

Capítulo 07 – Tecnologias Selecionadas 67

7.1 Introdução 67

7.2 EAU (Elemento Acesso ao Usuário) 68

7.3 EIG (Elemento Inteligência do Gerente) 68

7.4 EIA (Elemento Inteligência do Agente) 73

7.5 ER (Elemento de Rede) 74

7.6 Plataforma de Distribuição 74

Capítulo 08 – Prototipagem 75

8.1 Introdução 75

8.2 Ferramentas Utilizadas 75

8.2.1 Linguagem JAVA – WWW 75

8.2.2 ODE (Object Database Environment) – Banco de Dados Ativo 76

8.2.3 JATLite – Agentes Inteligentes 77

8.2.4 Advent API SNMP/JAVA – SNMP 78

8.2.5 ÁBACO – CORBA 80

8.3 Descrição do Protótipo 82

8.3.1 Elemento Acesso ao Usuário 82

8.3.2 Elemento Inteligência do Gerente 90

8.3.3 Elemento Inteligência do Agente 96

8.3.4 Elemento de Rede 97

8.3.5 Plataforma de Distribuição 98

8.3.6 Funcionamento Geral do Protótipo 102

Capítulo 09 - Estudo de Caso 104

9.1 Introdução 104

9.2 Modelagem da Base de Informações do Gerente 104

9.3 Estudo de Caso: Contabilização 107
9.3.1 Visão Geral 107
9.3.2 Gerenciamento de Contabilização e o Protótipo 108
9.3.3 Especificação do Agente *ag-contabil* 112
9.4 Estudo de Caso: Falha 115
9.4.1 Visão Geral 115
9.4.2 Gerenciamento de Falha e o Protótipo 116
9.4.3 Especificação do Agente *ag-falha* 120

Capítulo 10 – Conclusão / Trabalhos Futuros 122

Referências Bibliográficas 127

ANEXO A – Modelagem de MIBs ATM 134
1. Introdução 134
2. MIB AToM – IETF 134
3. MIB ILMI 4.0 – ATM-Forum 142
4. Objetos Suplementares para Gerenciamento ATM – IETF 151

ANEXO B – Banco de Dados Ativo 161
1. Introdução 161
2. Evento-Condição-Ação 162
3. Composição de Eventos 164
4. Modelos de Execução 165
5. Modos de Acoplamento 165
6. Exemplos de Ferramentas que Implementam um Bancos de Dados Ativo 167

ANEXO C – Agentes Inteligentes 169
1. Introdução 169
2. Comportamentos Sociais Possíveis 171

ANEXO D – Plataforma de Distribuição – CORBA 173

1. Introdução 173

2. Object Request Broker (ORB) 177

3. Interface Definition Language (IDL) 181

4. Requisição e Tratamento de Pedidos 182

5. Estrutura do Adaptador de Objetos 184

Lista de Figuras

Figura 2.1 : Paradigma Gerente X Agente 08
Figura 2.2 : Visão Geral do Ambiente de Gerenciamento SNMP 10
Figura 3.1 : Modelo da Arquitetura dos Protocolos RDSI-FL 18
Figura 3.2 : Estrutura das Interfaces de um Ambiente ATM 19
Figura 3.3 : Estrutura Geral de uma Célula ATM 22
Figura 3.4 : Pilha de Protocolos LANE 27
Figura 4.1 : Visão Geral do Ambiente de Gerência ATM 34
Figura 4.2 : Ambiente CNM (Customer Network Management) 36
Figura 4.3 : Ambiente ILMI (Integrated Local Management Interface) 40
Figura 4.4 : Modelo de Referência de Gerenciamento de Redes ATM 43
Figura 5.1 : Arquitetura Funcional do ÁTILA 48
Figura 5.2 : Arquitetura Física do ÁTILA 52
Figura 6.1 : Diagrama do Fluxo Dinâmico de Informações do ÁTILA 56
Figura 6.2 : Exemplo 01 58
Figura 6.3 : Exemplo 02 59
Figura 6.4 : Exemplo 03 60
Figura 6.5 : Exemplo 04 62
Figura 6.6 : Exemplo 05 62
Figura 7.1 : Ambiente de Implementação 67
Figura 8.1 : Estrutura do Componente do Ambiente ÁBACO 81
Figura 8.2 : Interface Gráfica do Protótipo 82
Figura 8.3 : Interface de Definição de um Objeto “Obj_SubMp” 85
Figura 8.4 : Interface de Definição de um Objeto “Obj_Snmp” 87
Figura 8.5 : Interface de Definição das Interfaces de um Objeto “Obj_Snmp” 87
Figura 8.6 : Interface de Instanciação dos Agentes Inteligentes 89
Figura 8.7 : Hierarquia do Agente *ag-atila* 96
Figura 8.8 : Especificação de um Componente em CDL 98
Figura 8.9 : Composição de Componentes no ÁBACO 99
Figura 8.10 : Formato de uma Especificação CCL 100
Figura 8.11 : A arquitetura física do ÁBACO 101
Figura 9.1 : Visão Geral do Estudo de Caso 104

Figura 9.2 : Modelo de Objetos da MIB-ILMI (OMT) 105
Figura 9.3 : Diagrama de Seqüência do Protótipo em Contabilização 111
Figura 9.4 : Diagrama de Seqüência do Protótipo em Falha 118
Figura A.1 : Estrutura da MIB AToM 135
Figura A.2 : Modelo Entidade/Relacionamento : MIB AToM 140
Figura A.3 : Modelo Objeto (OMT) : MIB AToM 141
Figura A.4 : Estrutura da MIB ILMI 142
Figura A.5 : Modelo Entidade/Relacionamento : MIB ILMI 149
Figura A.6 : Modelo Objeto (OMT) : MIB ILMI 150
Figura A.7 : Estrutura da MIB Suplementar 151
Figura A.8 : Modelo Entidade/Relacionamento : MIB Suplementar 160
Figura B.1 : Estrutura de um Sistema Gerenciador de Banco de Dados Ativo 162
Figura C.1 : De Sistemas Baseados em Conhecimento a Agentes 170
Figura D.1 : Estrutura de um ORB 178
Figura D.2 : Pedido de Cliente via ORB 182

Lista de Tabelas

Tabela 3.1 : Estrutura e Funções das Camadas 21
Tabela 3.2 : Synchronous Digital Hierarchy (SDH) 24
Tabela 3.3 : Plesiochronous Digital Hierarchy (PDH) 25
Tabela 3.4 : Classes de Serviços na Camada AAL 26
Tabela 3.5 : Categorias de Serviço da Camada ATM 31
Tabela 4.1 : Tipos de Células OAM 42
Tabela 6.1 : Fluxos e Restrições de Interação dos Blocos Funcionais 57
Tabela 8.1 : Hierarquia de “menus” do Banco de Dados Ativo 89
Tabela B.1 : Modos de Acoplamento 166

Capítulo 01 - Introdução

1.1 Um Pequeno Histórico

Em passado recente, as organizações tinham o seu modelo de informação baseado em arquiteturas de computadores de grande porte ("*mainframes*"), centralizadas e proprietárias, em acordo às restrições tecnológicas existentes na época.

O acesso e a geração de informações nestes sistemas se dava através do uso de terminais, sem qualquer inteligência local. A administração dos recursos computacionais era simples, o que acarretava o fácil diagnóstico e correção de problemas. Por outro lado, sendo as arquiteturas de rede proprietárias, havia um tratamento homogêneo para os problemas de cada arquitetura, restringindo o leque de soluções na administração dos sistemas e na gerência da rede de terminais.

Os sistemas proprietários foram, naturalmente, projetados para um conjunto particular de equipamentos, com protocolos e serviços específicos, o que resultava na dificuldade, senão na impossibilidade destes de interoperarem. Em geral, ferramentas de gerenciamento de um determinado fabricante de redes de computadores não permitem o gerenciamento global em um ambiente heterogêneo.

Em meados da década de 70, começa uma profunda transformação na forma de se obter, processar e disponibilizar as informações existentes no mundo digital. Começam a aparecer os microcomputadores *stand-alone* e, em seguida, as chamadas estações de trabalho, já projetadas para trabalhar em redes, distribuindo física e logicamente processos e arquivos. Os anos 80 são marcados pelo modelo cliente/servidor em rede, inaugurando, assim, uma nova era na computação em grande escala, até então monopolizada pelos *mainframes*.

No ambiente das operadoras de telecomunicações, até então dominado pela comutação de circuito, surge o conceito de RDSI-FE (Redes Digitais de Serviços Integrados Faixa Estreita), fazendo uso de uma salada de tecnologias de transporte de informação. Suportada pela tecnologia ATM (Asynchronous Transfer Mode), não tardou a aparecer a RDSI-FL (Faixa Larga), como a nova solução, integrando em uma única tecnologia (comutação de células), o transporte do tráfego multimídia.

Embora tenha sido projetada no contexto das RDSI-FL, a tecnologia ATM fez seu "début" também no ambiente da comunicação de dados, cujos usuários passaram a ter novas exigências de tráfego, tanto em velocidade, quanto em QoS (*Quality of Service*). Assim, ATM começa a figurar dentre as chamadas tecnologias de redes de comunicação de alta velocidade. Tem-se, portanto, início a possibilidade de se agregar à rede de dados, diversas aplicações multimídia (voz, texto, vídeo, etc.) com QoS garantida, e ainda disponibilizar serviços em tempo-real, impraticáveis em tecnologias passadas. Estes serviços

possuem, naturalmente, perfis de tráfego totalmente diferentes dos tradicionais.

1.2 Necessidade de Gerenciamento

O imediato crescimento de qualquer sistema acarreta, de forma não linear, um significativo aumento em seu mecanismo de gerenciamento, isto é, seu monitoramento e controle. Em não sendo diferente, as redes de computadores têm experimentado um aumento substancial de seus problemas e a conseqüente necessidade de gerenciamento cada vez mais eficiente de seus recursos físicos e lógicos. Além da heterogeneidade de equipamentos e de soluções, resultantes deste crescimento, equipamentos inteligentes para processamento e armazenamento de informações estão agora distribuídos através da rede.

Assim, ambientes homogêneos, onde os sistemas operacionais e os protocolos utilizados são os mesmos em toda a rede, cedem lugar a múltiplos tipos de ambientes operacionais e múltiplos protocolos de rede, com características específicas e particularidades na maneira de tratar e transportar as informações. Uma Intranet corporativa, por exemplo, pode comportar computadores de grande porte, servidores e estações de trabalho UNIX, computadores pessoais (PCs) executando Windows, OS/2, e talvez, dispositivos como comutadores telefônicos, etc. As redes e protocolos conectando estes sistemas também podem ser variados : ethernet, FDDI, TCP/IP, ATM.

Outro aspecto a considerar, é que este conjunto heterogêneo precisa ser gerenciado de forma integrada, para que a rede corporativa possua qualidade de serviço, disponibilidade de acesso, tempo de resposta e custos compatíveis com a capacidade e com as necessidades da organização. A complexidade na atividade de gerenciamento será tão maior quanto mais diversificados forem os equipamentos de redes fornecidos por diferentes fabricantes. Essa complexidade implicará no uso de ferramentas automatizadas associadas ao esforço humano. À medida que os sistemas baseados em redes se tornam maiores, mais complexos e heterogêneos, os custos inerentes à atividade de gerenciamento de redes se destacam.

Uma rede não gerenciada pode apresentar congestionamento anormal de tráfego, má utilização de recursos, problemas com segurança, etc., com conseqüências desastrosas para a empresa. A fim de contribuir para minimizar custos é que modelos e ferramentas padronizados de gerenciamento de redes vêm sendo pesquisados/propostos, permitindo o uso de uma gama variável de produtos, num ambiente multi-vendedor.

A tendência atual na utilização de aplicações com recursos multimídia, aliada ao desempenho crescente das estações de trabalho, vêm despertando um grande interesse, principalmente no que concerne à possibilidade da oferta de novos serviços e aplicações, tais como interconexão pública de redes locais em alta velocidade, transferência e edição remota de imagens, transmissões de vídeo digital bidirecional, videoconferência de alta resolução, entre outras. A tecnologia ATM surge então com a promessa de tornar viável essas necessidades.

A gerência de redes ATM passa a assumir uma importância fundamental na próxima geração, tanto de redes de computadores quanto de redes de telecomunicações.

1.3 Identificação do Problema

Comutadores ATM estão cada vez mais presentes em ambientes locais e metropolitanos, embora tenham sido idealizados para "backbones" de longa distância (RDSIs). A tecnologia ATM já foi criada com uma abordagem de gerenciamento. Esta arquitetura possui um plano de gerência que se volta para os problemas e soluções existentes neste ambiente. Este plano se divide em níveis, onde temos uma visão segmentada do ambiente. Atualmente, o ITU-T (International Telecommunication Union) [ITU-T], ATM Forum [ATM-Forum] e o IETF (Internet Engineering Task Force) [IETF] vêm desenvolvendo esforços no sentido de criar padrões que permitam que equipamentos de diferentes fabricantes possam interoperar.

Ambientes baseados em ATM possuem características específicas exigindo, provavelmente, uma nova abordagem na arquitetura de gerenciamento, o que envolve o acréscimo de novas funcionalidades a cada uma das áreas de gerenciamento especificadas pela ISO (falha, desempenho, contabilização, segurança e configuração).

Em função das necessidades impostas pelas aplicações existentes, redes ATM, são hoje de importância estratégica para o sucesso das organizações. Maximizar a utilização dos recursos e manter a QoS (*Quality of Service*) em níveis aceitáveis se tornam imprescindíveis para o ambiente ATM, através do qual os sistemas de informação são interconectados. A maioria das soluções existentes de gerenciamento de redes é inadequada para o processamento eficiente de eventos e informações de gerência disponibilizados por dispositivos ATM.

ÁTILA, uma Arquitetura Inteligente e Distribuída para o Gerenciamento Pró-ativo de Redes ATM, é a proposta deste trabalho. Além das arquiteturas funcional e física do ÁTILA, este trabalho também apresenta um protótipo cuja implementação faz uso de uma plataforma distribuída, de Banco de Dados Ativo e de Agentes Inteligentes. Uma contribuição relevante do ÁTILA consiste na proposição de uma metodologia original, baseada em um fluxo dinâmico de informações, para a realização da gerência pró-ativa em redes ATM, o que proporciona vantagens funcionais sobre os modelos atuais de gerência ATM.

1.4 Organização da Dissertação

Este trabalho encontra-se organizado em três partes. A parte I (capítulos 2, 3 e 4) discorre sobre o estado atual da gerência ATM. A parte II (capítulos 5 e 6) apresenta, com detalhes, as arquiteturas física e funcional do ÁTILA, bem como o seu fluxo dinâmico de informações. A parte III (capítulos 7, 8 e 9) descreve a implementação de um protótipo da arquitetura proposta neste trabalho.

No capítulo 02 é conceituado gerência de rede, descrevendo os processos que compõe um ambiente de gerência de rede de computadores. Neste capítulo, conceitos de gerente, agente e MIB - Management Information Base, comuns às arquiteturas de gerência de rede são descritos. Além disso, são apresentados os ambientes de gerência baseados em sistemas abertos, problemas de heterogeneidade existente entre os processos e as dificuldades na sua integração. Esta descrição leva em conta os padrões definidos pelo IAB - Internet Activities Board para o Ambiente de Protocolos TCP/IP - *Transmission Control Protocol/Internet Protocol* e pela ISO - *International Organization for Standardization* através do Modelo de Referência OSI. Para finalizar, são detalhadas as áreas funcionais de gerência, sob as quais as informações de cada elemento de rede são organizadas, conforme proposto no RM-OSI.

O capítulo 03 trata da tecnologia ATM. São descritas as características funcionais e estruturais desta tecnologia, que foi projetada para redes de telecomunicações (RDSI-FL), mas que se tornou também interessante para redes de computadores. Detalhes sobre a pilha de protocolos, os serviços, as interfaces que compõe o ambiente são apresentadas. Uma visão geral sobre as tecnologias LANE - *Lan Emulation* e MPOA- *Multi-Protocol Over ATM* é apresentada com o objetivo de prover uma idéia geral da utilização de ATM como suporte a outras tecnologias existentes (IP, IPX, ...). Uma descrição sucinta sobre endereçamento, sinalização e roteamento também é apresentada. Na ultima seção, são descritas as categorias de serviços e contratos de tráfego utilizados no ambiente.

O capítulo 04 finaliza a parte I apresentando uma coletânea dos ambientes disponíveis de gerência de redes ATM. Detalhes sobre os padrões de gerência ATM, especificados pelos organismos de padronização são descritos. É dada ênfase nas especificações do IETF, que tratam de MIBs relacionadas ao ambiente ATM; do ATM Forum relacionado à ILMI (*Integrated Local Management Interface*) e do Modelo de Referência de Gerência de Redes ATM; e do ITU-T em relação às células OAM. É apresentada também uma visão detalhada sobre cada área funcional de gerência, voltada ao ambiente ATM. Na última seção é descrito o gerenciamento de tráfego, e em especial diversas funções a serem empregadas no monitoramento e controle de tráfego no ambiente ATM.

O capítulo 05 aborda as arquiteturas funcional e física do ÁTILA. Estas arquiteturas detalham, em diferentes níveis e visões, o ambiente proposto, o qual se propõe a dar suporte a gerência de redes ATM.

O capítulo 06 finaliza a parte II apresentando o Fluxo Dinâmico de Informações relativo à arquitetura proposta. Posteriormente, são apresentadas exemplificações de fluxos de informação dos procedimentos a serem realizados com o objetivo de se gerenciar uma rede ATM.

O capítulo 07 descreve as tecnologias selecionadas, bem como apresenta as justificativas do emprego delas na implementação.

O capítulo 08 apresenta o protótipo do ÁTILA. Nele são mostrados o conjunto de ferramentas que implementam as tecnologias abordadas no capítulo 07 e é descrito a estrutura e o funcionamento do protótipo através da modelagem/especificação dos elementos que o compõe. O capítulo 09 finaliza a parte III apresentando um estudo de caso, onde é realizado nas áreas de Contabilização e Falha, uma análise comparativa entre as soluções de gerência do ÁTILA e dos modelos atuais de gerência ATM.

As conclusões são apresentadas, mostrando a viabilidade de se utilizar esta arquitetura no auxílio à gerência ATM.

Para finalizar são apresentados quatro anexos. O anexo A apresenta um estudo detalhado sobre algumas MIBs baseadas na tecnologia ATM. No anexo B são descritos em detalhe a estrutura e o funcionamento de um banco de dados ativo. O anexo C mostra um resumo sobre a tecnologia de agentes inteligentes. O último anexo apresenta uma descrição detalhada da plataforma de distribuição CORBA, utilizada na implementação do ÁTILA.

PARTE I

“Estado Atual da Gerência ATM”

Introdução

ATM se destaca das demais tecnologias utilizadas em redes de computadores por integrar comportamentos diversos, necessários à transmissão de diferentes mídias. Projetada para suportar os serviços RDSI-FL, ATM incorpora em sua arquitetura nativa um Plano de Gerência, além dos Planos de Controle e de Usuário. O Plano de Gerência oferece dois conjuntos de funções: o gerenciamento dos planos e o gerenciamento das camadas. O gerenciamento dos planos trata das funções associadas ao sistema como um todo, permitindo a coordenação entre os demais planos. Gerenciamento de camada trata dos recursos e parâmetros residindo em cada protocolo das camadas da arquitetura.

Existem, atualmente, três abordagens no tratamento do plano de gerenciamento da arquitetura RDSI-FL, com vistas a gerência de um sistema baseado em ATM. São elas: *Integrated Local Management Interface* (ILMI) do ATM Forum, Células OAM do ITU-T e o Modelo de Referência de Gerenciamento de Redes ATM, especificado pelo ATM Forum (MR-ATM Forum).

A Parte I deste trabalho, apresenta os conceitos gerais de Gerência de Redes, um pequeno tutorial sobre redes ATM, e os fundamentos de Gerência ATM onde as três abordagens de gerenciamento ILMI, Células OAM e MR-ATM Forum são relacionadas com o plano de gerenciamento da arquitetura RDSI-FL.

Capítulo 02 - Gerenciamento de Redes

2.1 Introdução

As arquiteturas de gerência de rede em sistemas abertos utilizam-se do paradigma gerente x agente, uma espécie de modelo cliente/servidor no gerenciamento. Estes trocam informações entre si a respeito dos objetos gerenciados, através dos protocolos específicos de cada arquitetura. Uma estrutura de dados denominada MIB é o repositório destes objetos gerenciados. A seguir (Figura 2.1) é mostrado o relacionamento entre gerente, agente e MIB, bem como o conceito destes componentes básicos de gerência.

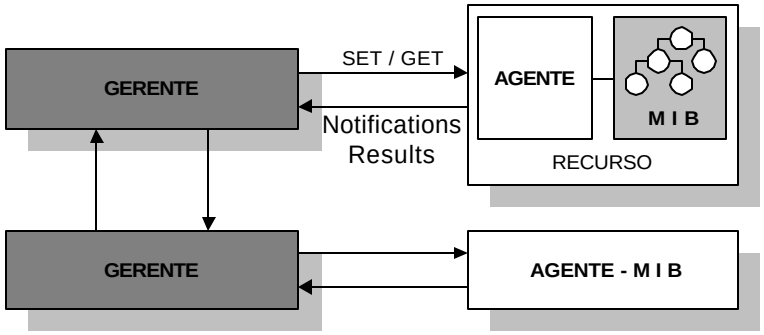


Figura 2.1 : Paradigma Gerente X Agente

Gerente : Processo de aplicação com a finalidade de coletar informações dos agentes distribuídos pela rede. O gerente trata estas informações no seu conjunto para análise e apresentação visando a solução de problemas ou tomadas de decisão. Ao gerente cabe também o envio de operações a serem realizadas pelos agentes sobre os elementos gerenciados.

Agente : Processo de aplicação que é implementado em cada elemento de rede passível de gerência. Tem como finalidade prover uma interface com o elemento a ser gerenciado para coletar as informações relativas ao seu funcionamento no mundo real ou proceder alguma alteração no elemento gerenciado.

MIB : Presente em cada elemento de rede a ser gerenciado, é uma estrutura conceitual e se constitui na visão que o gerente possui dos objetos gerenciáveis do elemento de rede. A MIB modela a estrutura e o estado do recurso gerenciado. Tem como finalidade descrever os objetos gerenciáveis em uma rede.

Dois padrões abertos se destacam no gerenciamento de redes : SNMP (*Simple Network Management Protocol*) e CMIS/CMIP (*Common Management Information*

Service/Protocol), descritos nas próximas seções.

2.2 SNMP (Simple Network Management Protocol)

Padrão de fato no mercado, SNMP [Rose95] foi idealizado para trabalhar no ambiente TCP/IP. Ele é definido como um sistema de gerência simples, implementado como um processo de aplicação. É um sistema centrado no gerente, com o agente possuindo uma função passiva comparado ao CMIS/CMIP. O envio de notificações do agente para o gerente em caso de ocorrência de eventos excepcionais, é efetuado de forma simples, praticamente informando a ocorrência. A figura 2.2 [Stallings93], abaixo, apresenta uma visão geral do ambiente SNMP, juntamente com as primitivas disponibilizadas por tal protocolo.

Com o objetivo de manter SNMP simples, este não foi modelado para trabalhar com conceitos sofisticados. De fato, os objetos em SNMP não são realmente objetos, do ponto de vista da tecnologia orientada a objeto; em vez disso, objetos SNMP são simplesmente variáveis com características básicas, tais como tipo de dados e se a variável é somente de leitura ou de leitura e gravação.

As principais especificações que definem as funções e base de dados relacionados a SNMP são descritas abaixo :

- Structure and Identification of Management Information for TCP/IP - Based Internets - (RFC1155) : descreve como objetos gerenciados contidos na MIB são definidos;
- Management Information Base for Network Management of TCP/IP - Based Internets : MIB II (RFC1213) : descreve os objetos gerenciados contidos na MIB II;
- Simple Network Management Protocol (RFC1157) : define o protocolo usado para troca de informações de gerência entre Gerente e Agente, com o objetivo de se gerenciar os objetos da MIB e consequentemente o recurso gerenciado.

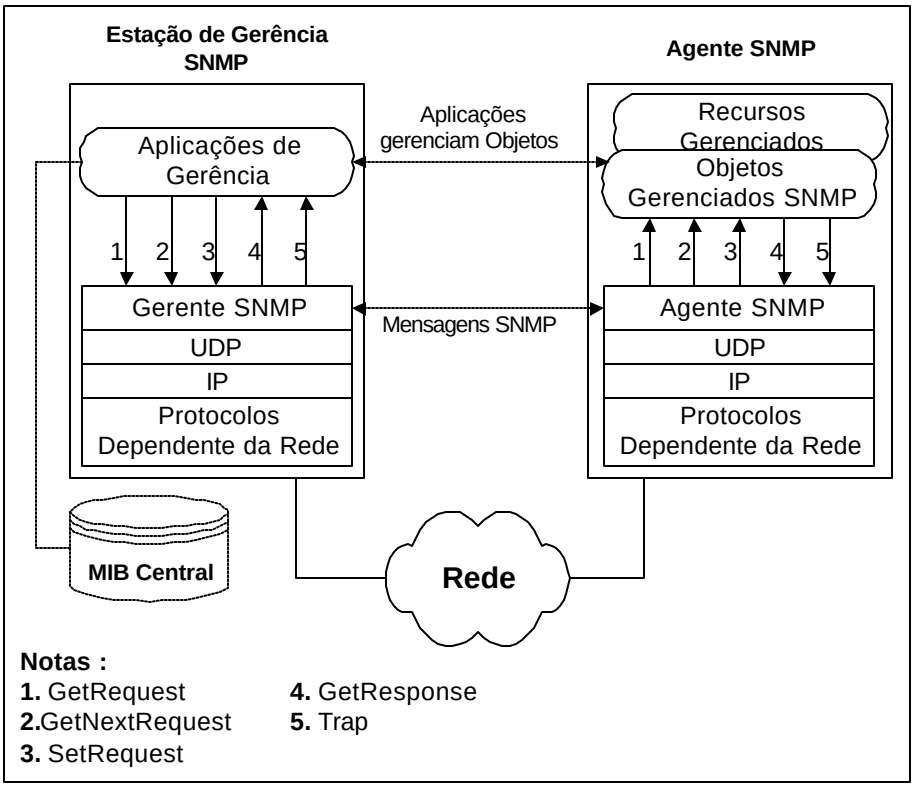


Figura 2.2 : Visão Geral do Ambiente de Gerenciamento SNMP

Atualmente, todos os maiores vendedores de equipamentos de rede (computadores, hubs, comutadores, roteadores) suportam o SNMP. Trabalhos estão em progresso no sentido de se utilizar SNMP sobre OSI (CMOT) e outros protocolos não SNMP. Além disso, melhoramentos no SNMP tem sido objetos de estudo em diversas direções.

Entretanto, quando SNMP é aplicado a redes maiores e mais sofisticadas, suas deficiências começam a aparecer. Estas deficiências são principalmente na área de segurança. Algumas destas deficiências são em virtude das seguintes limitações: rigidez da estrutura da MIB (MIB Fixa), limitações nas notificações assíncronas, passividade do processo agente, relação entre processo agente e objetos gerenciados (MIBs), etc.

Com o objetivo de se sobrepor a estas e outras limitações impostas pela arquitetura SNMP, foram desenvolvidas várias iniciativas. Talvez a mais importante seja a capacidade de monitoramento remoto (RMON - RFC1271) para SNMP. A especificação de monitoramento remoto define adições à MIB SNMP básica, tão bem quanto funções que exploram a MIB RMON.

RMON possibilita o gerente da rede monitorar sub-redes como um todo em vez de dispositivos da rede individualmente. RMON representa uma forma de se tentar aumentar as funcionalidades de SNMP adicionando semântica as MIBs.

Outra especificação que surgiu com objetivo de suprir algumas das limitações do SNMP, foi apresentada em 1992, denominada SMP. SMP foi aceito como a base para a definição de uma segunda geração SNMP, conhecida como SNMPv2 (RFCs 1441, 1442, 1450, etc.), provendo melhoramentos, tanto funcionais como em termos de segurança. Em particular, SMP adiciona ao SNMP novas PDUs.

2.3 CMIS/CMIP (Common Management Information Service/Protocol)

Desenvolvido em conjunto entre ISO e ITU-T, o conjunto de padrões para gerência de redes no ambiente OSI, é referenciado como Gerência de Sistemas OSI. Este inclui padrões de gerência de serviço, protocolo, banco de dados e conceitos associados.

O Modelo de Gerência OSI, utiliza o CMIS/CMIP [Stallings93] e uma arquitetura de gerência baseada no princípio do balanceamento da carga de trabalho entre o gerente e o agente, com a comunicação podendo ser estabelecida por iniciativa de ambos os lados. Diferentemente do SNMP, o CMIS/CMIP tem a sua concepção orientada a objeto. O gerente solicita operações a serem executadas pelos agentes, informações dos agentes e ativa ou desativa instâncias de classes de objetos. O agente por sua vez, emite notificações para o gerente assincronamente, sem que este tenha que efetuar qualquer solicitação.

Os padrões desenvolvidos para a gerência OSI se dividem em cinco categorias gerais :

- OSI Management Framework and Overview : que inclui ISO 7498-4, que provê uma introdução geral aos conceitos de gerência; e ISO 10040, que é uma visão geral dos documentos restantes.
- CMIS/CMIP : define o "*Common Management Information Service*" (CMIS), que provê os serviços de gerência OSI para aplicações de gerência, e o "*Common Management Information Protocol*" (CMIP), que disponibiliza a capacidade de troca de informação para suportar o CMIS.
- System Management Functions : define as funções específicas que são realizadas pela gerência de sistemas OSI.
- Management Information Model : define a base de informação de gerência (MIB), que contém a representação de todos os objetos dentro do ambiente OSI sujeitos à gerência.
- Layer Management : define informações, serviços e funções de gerência relacionados com camadas específicas OSI.

Na gerência de redes OSI, objetos gerenciados são vistos como entidades sofisticadas com : atributos, procedimentos associados, capacidade de notificação e outras características complexas associadas com a tecnologia orientada a objeto.

2.4 Áreas Funcionais

Os padrões de gerência OSI (*Open Systems Interconnection*), definiram cinco áreas funcionais pertinentes ao gerenciamento de redes de comunicações. A descrição destas áreas funcionais é independente de como elas são implementadas. Estas áreas aplicam-se a todos os três domínios, chamados de redes privadas, públicas e híbridas. As áreas funcionais são discutidas posteriormente.

2.4.1 Gerência de Falhas

Para manter a operação de uma rede complexa, cuidados devem ser tomados no sistema como um todo, e individualmente em cada componente essencial. Quando ocorre uma falha, é importante tão rapidamente quanto possível :

- Determinar exatamente a localização da falha;
- Isolar o resto da rede da falha, assim ela pode continuar a funcionar sem interferência;
- Reconfigurar ou modificar a rede de tal maneira a minimizar o impacto da operação mesmo sem o(s) equipamento(s) com falha;
- Reparar ou trocar os equipamentos com falha para retornar a rede ao seu estado inicial.

As informações obtidas através das gerências de configuração e de desempenho devem ser utilizadas de forma pró-ativa para evitar falhas previsíveis.

2.4.2 Gerência de Contabilização

Em muitas redes corporativas, ocorrem divisões individuais, centros de custo, divisões com base em projetos, com o objetivo de se cobrar pelo uso dos serviços da rede. Estes são procedimentos de contabilização interna. Além disso, mesmo se nenhuma cobrança é realizada internamente, o gerente da rede precisa estar apto a investigar a utilização dos recursos da rede pelos usuários ou classe de usuários. O registro das informações de utilização dos recursos da rede tem como objetivo : distribuição de custos, tarifação, planejamento de capacidade, planejamento de crescimento da rede, verificação de má utilização dos recursos dentre outros.

2.4.3 Gerência de Configuração

Redes de comunicação modernas são compostas de uma grande variedade e quantidade de componentes individuais e subsistemas lógicos que podem ser configurados com o objetivo de suportar muitas aplicações diferentes. Além disso podem estar dispersos em diferentes localidades.

O gerenciamento de configuração é responsável pela inicialização e bom funcionamento de toda a rede. Este também é responsável por manter, adicionar, e atualizar o relacionamento entre os equipamentos e seus estados, durante a operação da rede. O sistema de gerência deve disponibilizar maneiras de coleta automática de informações na rede, de forma a manter as informações sobre estado e configuração da rede sempre atualizados, permitindo também meios para se fazer análise sobre estas informações.

2.4.4 Gerência de Desempenho

Em virtude da grande variedade de componentes, é de importância crítica para a utilização efetiva de certas aplicações, que a comunicação sobre a rede esteja dentro de certos limites de desempenho. Os componentes da rede devem ser monitorados com o objetivo de se comparar e avaliar o seu comportamento (níveis de desempenho). O sistema deve possuir limiares de comportamento para que possa ser avaliado se o estado atual de um determinado componente ou sub-rede é crítico ou não. Além disso, deve-se possuir mecanismos para realizar ajustes e aumentar o desempenho da rede.

Resumindo, gerência de desempenho deve monitorar os recursos disponibilizando informações para se determinar os níveis de operação da rede, e a partir da análise, o gerente da rede pode ficar mais informado sobre situações de indicação ou presença de degradação de desempenho.

2.4.5 Gerência de Segurança

Gerência de segurança se preocupa com o controle de acesso às informações, definindo quais informações devem ser mantidas ou distribuídas. Além disso, deve se preocupar com acesso à determinadas aplicações e também às informações de gerência disponibilizadas pelo sistema. É importante também se definir os pontos de acesso, e os procedimentos de segurança, informando inclusive sobre tentativas de ataque com o objetivo de executar ações preventivas. "*Logs*" são uma ferramenta importante de segurança, portanto esta área está envolvida com a coleta, armazenamento e avaliação dos registros de auditoria e *"logs"* de segurança. Dentre as soluções hoje vigentes pode-se citar algoritmos de criptografia e *"firewalls"*.

Gerência de segurança envolve a proteção da transferência de informações autorizadas do usuário de um local para outro. Os cinco principais serviços para uma gerência de segurança efetiva são :

- Autenticação : Verificação da fonte de informação, isto é, a informação veio do fonte esperado.
- Controle de Acesso : A habilidade de se restringir e controlar o acesso a uma fonte ou recurso da rede. Esta função tenta assegurar que somente usuários autorizados tenham acesso a arquivos correspondentes, partições de rede, bancos de dados, etc.
- Integridade : A verificação que os dados recebidos são realmente os dados que foram enviados.
- Confidencialidade : A propriedade dos dados, como o receptor necessita saber e autorizar o acesso à informação.
- Não-repudiação : A verificação que ou o receptor ou o remetente recebeu ou enviou o dado.

Capítulo 03 - ATM (Asynchronous Transfer Mode)

3.1 Introdução

Até pouco tempo, cada tipo de aplicação necessitava de um tipo específico de rede de comunicação para ser implementada. Por exemplo, o serviço de voz utiliza rede comutada de circuito, enquanto o serviço de dados utiliza a rede comutada de pacotes. Atualmente deseja-se que uma rede simultaneamente suporte todos estes serviços, o que é chamado de Rede de Integração de Serviços (RDSI).

Com o advento das redes de comunicação de alta velocidade, há a possibilidade de se integrar estes serviços, e ainda disponibilizar outros serviços de tempo-real que eram impraticáveis em tecnologias passadas. Estes serviços, possuem freqüentemente perfis de tráfego totalmente diferentes dos tradicionais, o que levou ao estudo de novas tecnologias de rede como ATM [Alles95] [Hott96] [Soares95].

ATM é uma tecnologia de comunicação que utiliza um protocolo comutado por circuito de alta velocidade, que oferece a capacidade de transmissão de dados à altas velocidades com "*delay*" mínimo e qualidade de serviço (*QoS*) garantida. ATM é uma tecnologia orientada à conexão que transfere pacotes de tamanho fixo de 53 bytes, denominados células. É assíncrono de maneira que a ocorrência de células contendo informação de um usuário não seja necessariamente periódica.

Por ser uma tecnologia de transmissão mais flexível, ATM oferece a promessa de integração de diversos serviços (ex. voz, vídeo e dados). ATM também possibilita a utilização de transmissões ponto-a-ponto e multiponto, enquanto provê largura de banda de maneira escalonável; e oferece a promessa de integrar redes de longa distância (WAN) e redes locais (LAN).

Recursos ATM, tais como, largura de banda e *"buffers"* são compartilhados entre os usuários, e são alocados somente quando eles tem algo a transmitir. Logo a rede usa multiplexação estatística para aumentar o "*throughput*" efetivo.

As recentes modificações na estrutura da rede telefônica em todo o mundo, primeiro com a crescente utilização de computadores e segundo com a introdução de fibras de alta capacidade, aumentam a possibilidade de ATM desempenhar um papel importante na área de tecnologia de transmissão num futuro próximo [Partridge93].

Vários órgãos tem definido padrões para redes ATM : ITU-T, ATM Forum, Belcore e ANSI. Este processo assegura uma alta probabilidade de interoperabilidade entre os produtos dos fornecedores.

3.2 Modelo da Arquitetura dos Protocolos RDSI-FL

O modelo de arquitetura dos protocolos RDSI-FL, do qual surgiu a tecnologia ATM, consiste de três planos e quatro camadas (Figura 3.1). Este modelo difere do modelo de referência OSI, pois ele se utiliza de três e não duas dimensões.

A função de cada plano é:

- Plano do Usuário: Transferência de informações do usuário, com controle de fluxo e recuperação de erros.
- Plano de Controle : Funções de controle de chamada e de estabelecimento de conexão, tal como a sinalização.
- Plano de Gerenciamento: Controla os dispositivos ATM, tais como, comutadores ou hubs. Este plano oferece dois tipos de funções : gerenciamento dos planos e gerenciamento das camadas. O gerenciamento dos planos trata das funções associadas ao sistema como um todo, proporcionando coordenação entre os planos. Gerenciamento de camada trata dos recursos e parâmetros residindo em cada protocolo das camadas, como por exemplo, fluxo OAM.

As camadas incluem : camada física, ATM e camada de adaptação ATM (AAL) que serão descritas nas próximas seções.

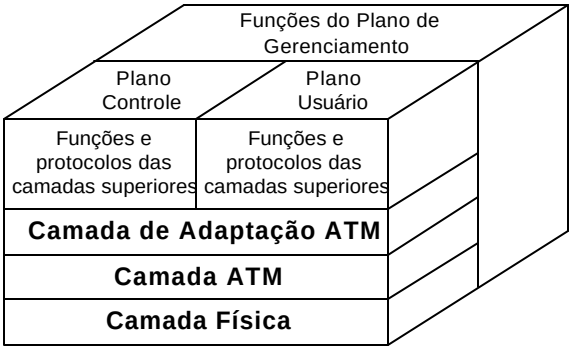


Figura 3.1 : Modelo da Arquitetura dos Protocolos RDSI -FL

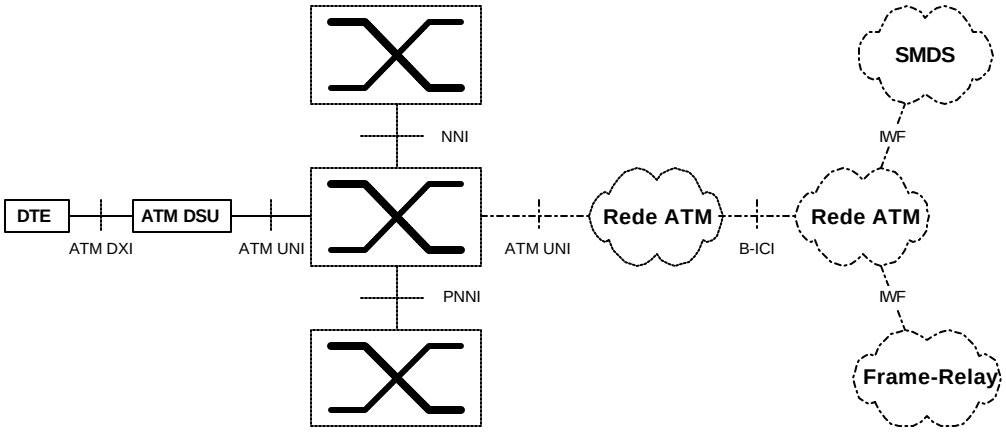
3.3 Interfaces ATM

ATM possui várias interfaces (Figura 3.2) : a UNI, ATM DXI, Network-Node Interface (NNI), Private Network-Node Interface (PNNI) e Broadband Inter-Carrier Interface (B-ICI) [Miller97].

Padronizada pelo documento do ATM Forum a UNI [UNI4.0] conecta uma rede ATM a um equipamento, o qual pode ser um comutador ou *host*. Existem dois tipos de UNI : pública e privada. A UNI pública conecta um comutador ATM privado a uma rede de provedor de serviço ATM pública. A UNI privada conecta usuários ATM ao comutador ATM.

Algumas aplicações dividem as funções do protocolo ATM entre o DTE, tal como um roteador, e a interface de hardware para a UNI, tal como um ATM DSU. A interface ATM DXI define a operação do protocolo entre estes dois dispositivos.

A interface NNI descreve a interconexão de rede dentro de uma rede de um único provedor (carrier) ou entre duas redes de provedores. A PNNI especifica um protocolo através do qual comutadores ATM, que são partes de uma rede ATM privada, podem se comunicar. A especificação da PNNI define duas configurações possíveis : private network-node interface, operando entre dois comutadores; ou a private network-network interface, operando entre dois grupos de comutadores, ou redes ATM. Quando uma NNI interconecta redes públicas ATM, esta é chamada B-ICI.



Notas :
ATM DXI : ATM Data Exchange Interface
ATM UNI : ATM User-Network Interface
B-ICI : Broadband Inter-Carrier Interface
IWF : Internetworking Function
NNI : Network-Node Interface
PNNI : Private Network-Node Interface

Figura 3.2 : Estrutura das Interfaces de um Ambiente ATM

Caso uma rede ATM se conecte com outra rede pública ou privada, tal como Frame-Relay ou SMDS, é necessário uma conversão entre elas. Processos IWF, definidos pela especificação da interface B-ICI, realizam esta conversão.

3.4 Conexões ATM

Como dito anteriormente, a camada ATM utiliza-se de um serviço orientado à conexão, onde a transferência de informações é baseada em circuitos virtuais. Independente de ser enviada através de uma UNI ou NNI, uma célula ATM possui o identificador do seu canal (circuito) virtual. Este identificador possui duas partes, onde ambas são usadas na camada ATM : VPI (Identificador de Caminho Virtual) e VCI (Identificador de Canal Virtual).

Um caminho virtual (VP) é formado por um conjunto de enlaces de canais virtuais (VCL), todos tendo o mesmo ponto de término. O VPI ou é associado ou removido para originar ou terminar um enlace de caminho virtual (VPL). Estes enlaces são concatenados formando uma conexão de caminho virtual (VPC). Cada enlace de canal

virtual (VCL) dentro de um VPC mantém a seqüência de transmissão das células, mas não assegura a sua integridade.

O ITU-T define um canal virtual, como sendo de comunicação unidirecional para o transporte de células. Um VCI ou é associado ou removido, respectivamente, para originar ou terminar um enlace de canal virtual (VCL). Enlaces de canais virtuais são concatenados para formar uma conexão de canal virtual (VCC), ou seja, um caminho fim-a-fim para células na camada ATM.

A operação básica de um comutador ATM é muito simples: receber uma célula através de um enlace com um valor de VPI/VCI conhecido; verificar o valor da conexão numa tabela local para determinar a porta de saída da conexão e o seu novo valor do VPI/VCI neste enlace; e então retransmitir a célula no enlace de saída com os identificadores de conexão apropriados.

A subdivisão do identificador de um canal virtual tem como função principal agrupar conexões similares entre pontos comuns, obtendo as seguintes vantagens:

- Simplificação no estabelecimento de conexões, pois a parte mais complexa desse processo, que é a determinação da rota entre os pontos envolvidos, só é feita na primeira conexão entre estes pontos.
- Simplificação da administração da rede no que se refere a controle de performance, tipo de serviço fornecido e gerenciamento de conexões.

3.5 Organização dos Protocolos

A Tabela 3.1 mostra uma visão ampla da pilha de protocolos ATM, juntamente com as funções de cada camada ou sub-camada. Abaixo são descritas em detalhes estas funções.

Camada de Adaptação ATM (AAL)	CS	Convergência
	SAR	Segmentação e Remontagem
Camada ATM		Controle de Fluxo Genérico Geração/extração de cabeçalho de célula Tradução de VPI/VCI Multiplexação/demultiplexação de células
Camada Física	TC	Geração/recuperação de transmissão Delineação de células Geração/verificação de sequencia do HEC Adaptação de transmissão de "frame" Adaptação de taxas de tranmissão
	PM	Temporização de Bit Meio físico

Tabela 3.1 : Estrutura e Funções das Camadas

3.5.1 Camada ATM

A camada ATM fornece um serviço do tipo orientado à conexão para as camadas superiores, isto é, a transferência de informações entre as aplicações em uma rede ATM é baseada em circuitos virtuais. Ela trabalha com pacotes de tamanho fixo e também define a estrutura destes pacotes especificando o significado de cada campo do cabeçalho.

Esta camada também se preocupa com o estabelecimento e término dos circuitos virtuais. Esses circuitos virtuais podem ser tanto circuitos comutados (SVC - *Switched Virtual Circuit*), alocados conforme demanda; ou circuitos permanentes (PVC - *Permanent Virtual Circuit*), em geral alocados por uma entidade administrativa de forma estática.

Os pacotes em que são agrupadas as informações para transmissão na tecnologia ATM são conhecidos como células ATM e possuem 53 bytes (Figura 3.3).

O significado dos campos das células é o seguinte:

- GFC (*Generic Flow Control*) (4 bits) : Usado apenas nas células ATM da UNI tem

como função auxiliar o controle de fluxo através desta interface.

- VPI (*Virtual Path Identifier*) (8/12 bits, UNI/NNI) : Identificam um caminho virtual em um meio físico de transmissão, para a comutação de células ATM nos comutadores.
- VCI (*Virtual Channel Identifier*) (16 bits) : Identificam um canal virtual em um meio físico de transmissão, para a comutação de células ATM nos comutadores.
- PTI (*Payload Type Identifier*) (3 bits) : Indica o tipo de informação do campo 'Information Field' da célula ATM (dados de usuário ou de controle, por exemplo).
- CLP (*Cell Loss Priority*) (1 bit) : Associa uma prioridade à célula, no caso de necessidade de descarte de células ATM em situações de congestionamento; células com CLP = 1 podem ser descartadas nessas situações.
- HEC (*Header Error Control*) (8 bits) : Valor calculado para validar o cabeçalho da célula, a partir dos 32 bits restantes desse cabeçalho; é o único campo não gerado na camada ATM, e sim na sub-camada TC da camada física.
- Information Field (48 bytes) : Também conhecido como *"information payload"*, constitui os dados das camadas superiores a serem transportados via células ATM.

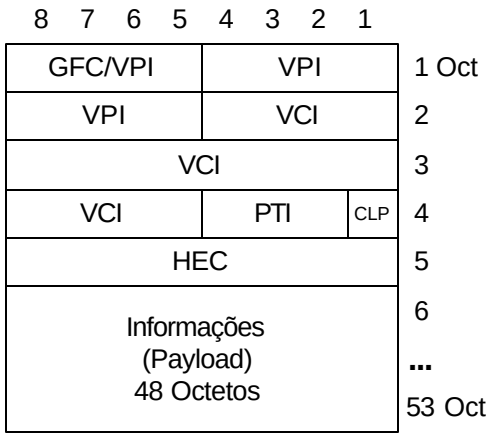


Figura 3.3 : Estrutura Geral de uma Célula ATM

A camada ATM funciona independentemente da camada física e realiza quatro funções principais : multiplexação/demultiplexação de células ATM, tradução de VPI/VCI, geração/extração de cabeçalho e controle de fluxo. Estas funções são detalhadas abaixo.

- Multiplexação/demultiplexação de células ATM de conexões lógicas distintas de uma mesma interface física. Na direção de transmissão a camada ATM multiplexa

células de caminhos virtuais (VPs) e canais virtuais (VCs) individuais em um fluxo de células composto. Na direção do receptor, a demultiplexação direciona células do fluxo de células composto para o VP ou VC apropriado.

- Tradução VPI/VCI, ou seja comutação de células de acordo com os identificadores da conexão (VPI/VCI). Os campos VCI e VPI nas células de entrada podem requerer mapeamento para novos valores.
- Geração/extração dos cabeçalhos das células. A camada ATM gera o cabeçalho ATM e o adiciona ao "payload" para transmissão ou extrai o "payload" de uma célula recebida e o repassa para as camadas mais altas.
- Controle de fluxo genérico : a camada ATM pode gerar células que contenham informações de Controle de Fluxo Genérico (GFC).

3.5.2 Camada Física

A camada física possui duas sub-camadas : Physical Medium (PM) e Transmission Convergence (TC).

A sub-camada PM provê transmissão a nível de bit. Entre suas funções incluem-se funções dependentes do meio de transmissão, seja ele uma interface ótica ou elétrica, tais como, cabo a ser utilizado, recuperação de bits, temporização e método de acesso ao meio, etc.

A sub-camada TC, desempenha cinco funções : geração/recuperação de *frames* de transmissão (se existentes), empacotamento de células ATM em *frames* (se for utilizado), delimitação das células, geração/recuperação do HEC (*Header Error Control*) das células ATM, inserção e retirada de células ATM de preenchimento.

A Geração de *frames* cria e recupera os *frames* de dados enviados pela sub-camada PM. Células transmitidas pela camada ATM devem ser adaptadas para o formato utilizado pela sub-camada PM. Na direção receptora, a função de empacotamento extrai as células dos *frames*. A função de delineação de célula identifica os limites das células, para a camada ATM poder decodificá-las. A seqüência do HEC é calculada e adicionada no cabeçalho ATM para *frames* transmitidos. Para *frames* recebidos os cabeçalhos das células são checados para erros. Se erros são encontrados, eles são

recuperados quando possível, ou no outro caso as células são descartadas. Finalmente, a retirada ou inserção de células de preenchimento adaptam as taxas de transmissão das células ATM válidas para a capacidade do sistema de transmissão.

A camada física especifica as duas formas de transmissão da arquitetura RDSI-FL:

- As células podem ser empacotadas em envelopes (*frames*) a serem transportados pela técnica TDM; neste caso, a estrutura desses *frames* é denominada como SDH (*Synchronous Digital Hierarchy*), definida pelo ITU-T no documento G-709; nos EUA essa estrutura é conhecida como SONET (*Synchronous Optical Network*).
- As células podem ser transmitidas através do meio físico sem nenhum envelope adicional, sendo a sincronização feita através do campo HEC das mesmas.

O ATM Forum define várias tecnologias de transmissão, tanto para UNIs públicas quanto privadas. Abaixo as tabelas 3.2 e 3.3 citam as mais comuns :

SONET / SDH (Synchronous Digital Hierarchy) Rates		
Velocidade (Mbps)	**SONET**	**SDH**
51.840	STS-1	-
155.520	STS-3	STM-1
466.560	STS-9	STM-3
622.080	STS-12	STM-4
933.120	STS-18	STM-6
1244.160	STS-24	STM-8
2488.370	STS-48	STM-16

Tabela 3.2 : Synchronous Digital Hierarchy (SDH)

3.5.3 Camada de Adaptação ATM (AAL)

A camada AAL possui duas sub-camadas : Segmentation e Reassembly Sublayer (SAR) e a Convergence Sublayer (CS).

A sub-camada SAR segmenta/reagrupa as informações da camada superior em células

ATM. Ela segmenta as informações de tamanho variáveis para serem transmitidos nos *payload* de tamanho fixo ATM, e reagrupa os *payloads* recebidos em informações de camadas superiores.

Plesiochronous Digital Hierarchy (PDH) Rates		
Velocidade (Mbps)	**América do Norte**	**Europa**
0.064	DS0	-
1.544	DS1	-
2.048	-	E1
3.152	DS1C	-
6.312	DS2	-
8.448	-	E2
34.368	-	E3
44.736	DS3	-
139.264	-	E4
274.176	DS4	-

Tabela 3.3 : Plesiochronous Digital Hierarchy (PDH)

A sub-camada CS realiza funções requisitadas pelo tipo de AAL sendo utilizada, e é portanto dependente do serviço. Em alguns casos as suas funções podem ser subdivididas em Common Part Convergence Sublayer (CPCS), e Service Specific Convergence Sublayer (SSCS).

A camada AAL tem como função principal a de permitir o uso da tecnologia ATM por outros protocolos não ATM, ou seja, permitir o uso da tecnologia ATM por vários tipos de aplicações.

Esta camada fornece alguns tipos de serviços com características orientadas a cada tipo de aplicação; esses serviços são denominados como QoS (*Quality of Service*) e são agrupados em classes de serviços. Além disso, existem nessa camada alguns protocolos conhecidos como AAL Types. Segue abaixo relação entre as classes de serviços e os protocolos mais adequados para a classe.

Características	Classe A	Classe B	Classe C	Classe D
Tempo Real	Sim	Sim	Não	Não
Taxa de Transferência	Constante	Variável	Variável	Variável
Tipo de Serviço	Orientado à conexão	Orientado à conexão	Orientado à conexão	Não orientado à conexão
Protocolo AAL	AAL 1	AAL 2	AAL 3/4 e 5	AAL 3/4
Aplicações	Vídeo e Voz.	Voz e Vídeo comprimidas	Frame-Relay e X.25	SMDS e tráfego LAN

Tabela 3.4 : Classes de Serviços na Camada AAL

3.6 LAN Emulation (LANE)

Padronizada pelo ATM Forum, LAN Emulation [LANE1.0] [Greaves94] permite as aplicações de usuários hoje existentes acessarem uma rede ATM. Este acesso deverá ser transparente para a aplicação, ou seja, se comportar como se ela estivesse utilizando protocolos tradicionais, tais como TCP/IP ou IPX, e rodando sobre LANs como Ethernet e Token Ring.

As restrições encontradas são devido as diferenças entre os modelos dos protocolos. ATM é orientado à conexão, enquanto IP e IPX não o são. Algumas funções, tais como, estabelecimento de uma conexão e tradução de endereços LAN para ATM, devem ser ocultadas das camadas superiores, fazendo com que as aplicações pensem que estão rodando sobre redes tradicionais.

Neste caso, foram definidas duas interfaces diferentes : a interface entre o usuário LANE e a rede (LUNI) e a interface entre duas redes LANE (LENNI). Além disso, são especificados dois cenários : no primeiro a rede ATM é utilizada para interconectar duas Ethernets, uma Ethernet e um dispositivo ATM ou dois dispositivos ATM; no segundo caso temos no lugar de Ethernets, Token Rings. Para estes sistemas interagirem, é necessário uma pilha de protocolos LANE (Figura 3.4).

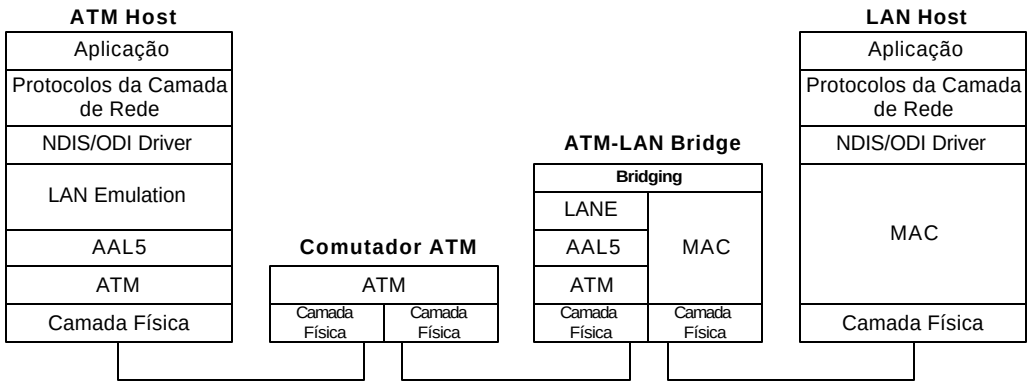


Figura 3.4 : Pilha de Protocolos LANE

Computadores em LANs e suas aplicações operam sobre protocolos tradicionais, tais como, TCP/IP e IPX, e drivers como NDIS e ODI provêem uma interface entre as camadas mais altas e a camada MAC. O equipamento que converte ATM para LAN é um elo da rede rodando duas pilhas de protocolo; uma que se comunica com a LAN e outra que se comunica com a rede ATM. Note que o equipamento que realiza a conversão está funcionando como uma ponte, operando independente dos protocolos de rede e os acima deste.

Resumindo, LANE realiza o mapeamento das funções das camadas MAC da Ethernet e Token Ring em conexões virtuais ATM, enquanto protegem a aplicação do estabelecimento da conexão e funções de controle que os comutadores ATM necessitam.

A arquitetura LANE é baseada no paradigma cliente/servidor, com o Cliente de Emulação de LAN (LEC) derivando informações que este processa a partir de um ou vários servidores : o servidor de configuração (LECS), o servidor de emulação de LAN (LES) e o servidor de "*broadcast*" ou "*unknown*" (BUS).

3.7 MPOA (Multi-Protocol Over ATM)

Também padronizado pelo ATM Forum, MPOA [MPOA1.0] é um modelo de serviço para *internetworking* fim-a-fim através de uma rede ATM. Estende o conceito de *internetworking* aos protocolos típicos de nível 3 (Internet IP ou Novell IPX, por exemplo), permitindo conexões de várias sub-redes, sob o ponto de vista desses protocolos, através da tecnologia ATM.

MPOA pode ser encarada como uma extensão da LANE. A principal diferença entre LANE e MPOA é nas camadas de operação dos protocolos. LANE opera no nível 2 (*bridging*), enquanto MPOA opera nos níveis 2 e 3 (*bridging e routing*). MPOA utiliza LANE para realizar as funções da camada 2. Assim temos que o escopo de LANE é uma simples sub-rede de camada 3, enquanto MPOA permite os dispositivos estabelecerem comunicação direta através de conexões ATM, até se os dispositivos estiverem em sub-redes diferentes.

Sua arquitetura inclui vários componentes : *hosts* ATM, dispositivos de elo (*edge devices*) e servidores de roteamento (*route servers*). Além disso dispositivos podem ser agrupados logicamente em um Grupo de Resumo de Endereços Internet (IASG). Este é definido como um conjunto de endereços da camada de rede, resumidos em roteamento da camada de rede, que é similar para uma sub-rede e seu conjunto de endereços. Em adição, um IASG depende do protocolo, assim um dispositivo que opera com dois protocolos de rede, deve ser membro de no mínimo dois IASGs.

Um dispositivo de elo provê conectividade entre redes legadas e ATM. O servidor de roteamento é um dispositivo físico e/ou lógico que provê informações de roteamento para outros dispositivos dentro da rede. Esta informação inclui endereços das camadas 2, 3 e ATM. Se dois dispositivos que estão ligados ao mesmo dispositivo de elo precisarem se comunicar, o dispositivo de elo enviará o pacote usando LANE. Se o pacote precisar sair do seu IASG, o dispositivo de elo obterá o endereço ATM a partir do seu cache ou através de uma consulta ao servidor de roteamento. O dispositivo de elo e o servidor de roteamento se comunicarão para descobrir e atualizar suas informações a respeito de endereços e rotas disponíveis.

3.8 Endereçamento, Sinalização e Roteamento

O fato de ATM ser orientado a conexão implica na necessidade de protocolos de sinalização [Signalling] [Stiller] e estruturas de endereçamento, tão bem quanto protocolos para rotear requisições de conexão através de uma rede ATM.

Todo equipamento ATM precisa de um endereço ATM em adição a qualquer endereçamento de camadas superiores. O espaço de endereçamento ATM deve ser logicamente disjunto do espaço de endereçamento de qualquer protocolo que execute sobre a camada ATM. Portanto, todos os protocolos operando sobre uma sub-rede ATM deverão possuir alguma forma de protocolo de resolução de endereço ATM, para mapear endereços de camadas mais altas (ex. IP) para o seu endereço ATM correspondente.

Sinalização em ATM utiliza o método "*one-pass*" de estabelecimento de conexão, que se comporta da seguinte maneira: uma requisição de conexão a partir do equipamento fonte é propagado através da rede, estabelecendo a conexão por onde passa, até chegar ao equipamento final. O roteamento da requisição de conexão, e portanto qualquer fluxo de dados subsequente é governado pelos protocolos de roteamento. Tais protocolos roteiam a requisição de conexão baseado no endereço destino, e nos parâmetros de QoS e tráfego requisitados pelo equipamento fonte.

O protocolo de sinalização ATM opera em cima do SSCOP (*Service Specific Convergence Protocol*), o qual garante a entrega, através da utilização de janelas e retransmissões.

Como falado anteriormente endereçamento é de fundamental importância numa rede ATM. O ITU-T propôs uma estrutura de endereçamento hierárquico para redes públicas ATM baseado no padrão E.164, usado na numeração internacional de sistemas telefônicos. O ATM Forum estendeu o endereçamento ATM para incluir redes privadas, e avaliou dois modelos diferentes de endereçamento. O primeiro é o formato E.164 e o outro é um endereço de 20 bytes modelado a partir do formato de endereços de um OSI NSAP (*Network Service Access Point*) [Juha92].

3.9 Categoria de Serviço e Contrato de Tráfego

ATM é uma tecnologia que suporta uma variedade de serviços e aplicações. Os mecanismos de controle do tráfego numa rede ATM se baseiam na qualidade de serviço definida para cada aplicação.

Estas categorias de serviço dizem respeito a características de tráfego e requisitos de qualidade de serviço (*QoS*) para o comportamento da rede. Funções como roteamento, CAC (Controle de Admissão de Conexão), alocação de recursos são, em geral, estruturados diferentemente para cada categoria de serviço. Categorias de serviço são distinguidas como sendo de tempo-real ou não. Para tráfego em tempo-real , existem duas categorias, CBR (*Constant Bit Rate*) e rt-VBR (*Real-Time Variable Bit Rate*). Para tráfego que não leva em conta o tempo, ou seja, não possui tráfego em tempo-real, temos três categorias ABR (*Available Bit Rate*), UBR (*Unspecified Bit Rate*) e nrt-VBR (*Non-Real Time Variable Bit Rate*).

Cada categoria de serviço possui um conjunto de parâmetros que identificam o tráfego a ser entregue pelo usuário à rede e a qualidade de serviço (*QoS*) requisitada à rede. Abaixo é descrito em mais detalhes cada categoria.

Real-Time Variable Bit Rate (rt-VBR): Utilizado por aplicações de tempo-real que exigem atrasos e variações de atraso bem limitadas (voz, vídeo), este serviço é baseado nos seguintes parâmetros de tráfego : PCR, SCR e MBS e maxCTD. É esperado da fonte uma taxa de transmissão variável no tempo, podendo a fonte ser chamada equivalentemente de "*bursty*" (explosiva). Células que são atrasadas além do valor especificado para máximo atraso de transferência de células (maxCTD) são assumidas como sendo de valor significativamente reduzido para a aplicação. O serviço VBR pode suportar multiplexação estatística de fontes de tempo-real.

Atributos	Categorias de Serviço da Camada ATM				
	CBR	**rt-VBR**	**nrt-VBR**	**UBR**	**ABR**
Parâmetros de Tráfego					
PCR e CDVT [4,5]	Especificado			Especificado [2]	Especificado [3]
SCR, MBS e CDVT [4,5]	n/a	Especificado		n/a	
MCR	n/a				Especificado
Parâmetros de QoS					
CDV ponto-a-ponto	Especificado		Não Especificado		
MaxCTD	Especificado		Não Especificado		
CLR	Especificado			Não Especifica	[1]
Outros Atributos					
Feedback	Não Especificado				Especificado

Categorias de Serviço

CBR	Constant Bit Rate
rt-VBR	Real-Time Variable Bit Rate
nrt-VBR	No-Real-Time Variable Bit Rate
UBR	Unspecified Bit Rate
ABR	Available Bit Rate

Parâmetros de Tráfego

MCR	Minimum Cell Rate
MBS	Maximum Burst Size
CDVT	Cell Delay Variation Tolerance
PCR	Peak Cell Rate
SCR	Sustainable Cell Rate

Parâmetros de QoS

CDV	Cell Delay Variation
CLR	Cell Loss Ratio
CTD	Cell Transfer Delay

Notas :

(1) CLR é baixo para fontes que ajustam o fluxo de células em resposta à informações de controle. O valor quantitativo para CLR é especificado dependendo da rede.

(2) Pode não estar sujeito aos procedimentos de CAC (Controle de Admissão de Conexão) e UPC (Usage Parameter Control).

(3) Representa a taxa máxima a qual um fonte ABR pode enviar informações. A taxa real está sujeita à informações de controle.

(4) Estes parâmetros são especificados implícita ou explicitamente para PVCs ou SVCs.

(5) CDVT refere-se à tolerância de variação de "delay" da célula. CDVT não é sinalizado. Em geral, CDVT não precisa possuir um valor único para uma conexão. Valores diferentes podem aplicar-se em cada interface ao longo do caminho de uma conexão.

Tabela 3.5 : Categorias de Serviço da Camada ATM

Constant Bit Rate (CBR): Este serviço é usado por aplicações que exigem uma quantidade estática de largura de banda contínua durante todo o tempo de existência da conexão. Ele é caracterizado pelo parâmetro PCR. Utiliza também o parâmetro *máximo CTD* (maxCTD). O compromisso básico assumido pela rede com o usuário que reserva recursos através da capacidade CBR é que uma vez estabelecida a conexão, a QoS da camada ATM é assegurada para todas as células que estão de acordo com os testes de conformidade relevantes. A fonte pode emitir células na PCR a qualquer instante e por qualquer duração de tempo e os compromissos de QoS ainda são mantidos. É voltada para aplicações de tempo-real que requerem variações de atraso de transferência (peak-to-peak CDV) bastante limitadas (voz, vídeo) mas não é restrita a estas aplicações. A fonte pode emitir células na PCR, ou abaixo (e pode até mesmo permanecer em silêncio), por períodos de tempo. Células que são atrasadas além do valor especificado para máximo atraso de transferência de células (maxCTD) são assumidas como sendo de valor significativamente reduzido para a aplicação. A categoria de serviço CBR pode ser usada tanto para VPCs como para VCCs.

Non-Real-Time VBR (nrt-VBR): Usado por aplicações que não sejam de tempo-real e que têm tráfego em rajadas, ou explosivo, este serviço é caracterizado pelos seguintes parâmetros : PCR, SCR e MBS. Para aquelas células que são transferidas dentro do contrato de tráfego, a aplicação espera uma baixa taxa de perda de bits (CLR). Pode suportar multiplexação estatística de conexões. Não há limite de atraso relacionado com essa categoria de serviço.

Unspecified Bit Rate (UBR): Este serviço é utilizado por aplicações que não são de tempo-real e que não exijam restrições de atraso e variação de atraso, como transferência de arquivos e correio eletrônico. Não especifica garantias de serviço relacionadas ao tráfego. Não há nenhum compromisso numérico com relação ao CTD ou ao CLR. Uma rede pode ou não aplicar o PCR às funções do CAC e do UPC. No caso das redes em que PCR não é exigido, o valor de PCR é apenas informativo. Quando PCR não é exigido, ainda assim é útil tê-lo negociado, porque isto pode permitir a fonte descobrir a limitação da menor largura de banda ao longo da via de conexão. O controle de congestionamento para UBR pode ser executado numa camada mais alta ou numa base fim-a-fim. É especificado pelo uso do Indicador de Melhor Esforço (BEI - Best Effort Indicator) no ATM User Cell Rate Information Element

(UCRIE)

Available Bit Rate (ABR): Neste tipo de serviço, as características das conexões ATM podem ser alteradas a qualquer momento após o seu estabelecimento. Um mecanismo de controle de fluxo é especificado e suporta vários tipos de realimentação para controlar a taxa da fonte em resposta às mudanças nas características de transferência da camada ATM. Esta realimentação é levada até a fonte através de células de controle específicas chamadas *Resource Management Cells*, ou ***RM-cells***. No estabelecimento de uma conexão ABR, o usuário deve especificar para a rede tanto uma taxa de transmissão máxima (PCR) quanto uma taxa mínima (MCR) de células, podendo este último ser especificado como zero. Não é voltada para aplicações de tempo-real, desta forma não especifica limites para o atraso (maxCTD) e para a variação do atraso (CDV). É esperado que um usuário que adapta seu tráfego de acordo com a realimentação obtenha baixo CLR e consiga uma divisão justa da largura de banda disponível de acordo com a política de alocação específica da rede.

Capítulo 04 – Gerenciamento de Redes ATM

4.1 Introdução

Gerenciamento de rede é um dos maiores desafios da tecnologia ATM. ATM que utiliza uma tecnologia para transmitir diversos tipos de tráfego e suporta diferentes aplicações tanto em LANs quanto em WANs, necessita de uma visão fim a fim de uma variedade de condições, desde transferência de células à altas velocidades, até milhares de conexões virtuais simultaneamente.

Vários organismos se detiveram com a questão da padronização da gerência de redes ATM. Abaixo são citados e detalhados alguns deles, como ilustrado na figura 4.1.

- Plano de Gerência (M-Plane). Padrão ITU-T que discute o modelo RDSI-FL [Miller97];
- Gerência de camadas e LMEs (Layer Management Entities). ISO e ITU-T [Minoli97];
- Customer Network Management (CNM) e SNMP [CNM94]. ATM Forum e IETF;
- Integrated Local Management Interface (ILMI) [ILMI4.0]. ATM Forum;
- Telecommunication Management Network (TMN) [M.3000]. ITU-T .

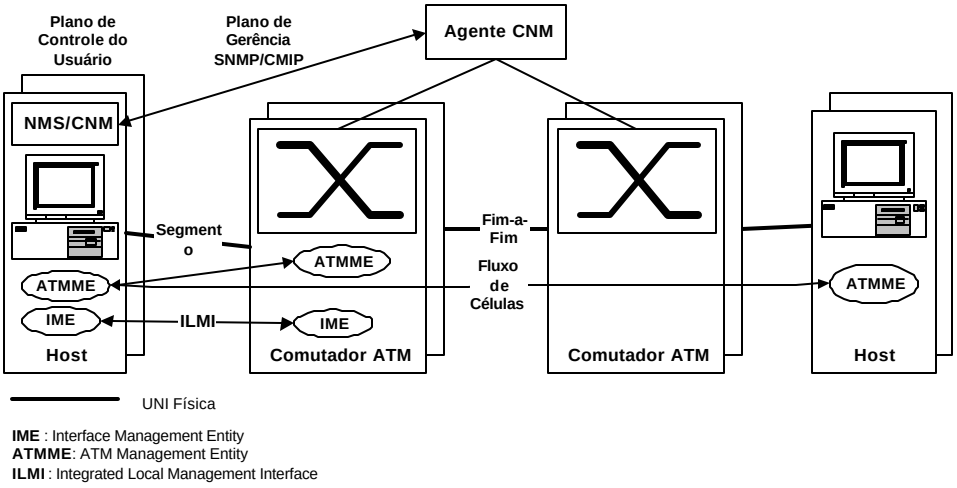


Figura 4.1 : Visão Geral do Ambiente de Gerência ATM

A arquitetura de protocolos RDSI-FL especifica dentro do seu contexto o plano de gerência ATM. A capacidade de gerência é subdividida em Gerência de Camada e Gerência de Sistema. A Gerência de Camada está relacionada ao nível lógico, enquanto Gerência de Sistema tanto atua a nível de serviço quanto realiza outras funções específicas. A Gerência de Camada usualmente trabalha em conjunto com um protocolo em uma determinada camada. A Gerência de Sistemas supervisiona totalmente as funções de comunicação desempenhadas, e coordena o comportamento dos subsistemas individuais. O software de gerência de sistema interage com o software de gerência de camada para coletar valores de atributos e para controlar e operar os aspectos de comunicação em questão.

Na proposta da ISO e ITU-T, em todas as camadas da arquitetura da rede devem ser coletadas informações de gerência específicas de um protocolo e aquelas relacionadas à camada como um todo. As funcionalidades de gerência de camadas incluem capacidades, tais como : *"loopback"*, monitoramento de desempenho, supervisão de alarmes, etc. Como dois tipos de conexão são definidas a nível de camada ATM, VPCs e VCCs, os usuários estão interessados em obter informações de gerência a respeito destes dois tipos de conexão. A gerência de camada é usualmente implementada através do fluxo de entrada, usando células especializadas.

A figura 4.1 também apresenta uma abordagem chamada ILMI, que trabalha com a gerência a nível de interface e será detalhada posteriormente..

Funções mais sofisticadas e relacionadas com serviço são implementadas em um nível mais alto. Estas funções trabalham com diversos aspectos, tais como: alterações de parâmetros de tráfego, obtenção de informações de contabilização, reconfiguração VCCs e VPCs, etc. Estas capacidades são suportadas através de uma estrutura CNM separada e por protocolos como SNMP ou CMIP.

A visão da gerência de redes segundo a abordagem CNM é mostrada na figura 4.2 [Minoli97]. Provedores de serviço (Concessionárias) gerenciam a sua infra-estrutura para prover serviços ATM aos usuários, através dos Sistemas de Operação (OS). Numa rede ATM privada , os usuários, através de um Sistema de Gerência de Redes (NMS –

Network Management System), precisam suportar muitas, mas não necessariamente todas, as funções de gerência que um provedor disponibiliza.

Outros usuários simplesmente obtém serviços ATM de um provedor. Neste caso, eles precisam somente de um mecanismo para gerenciar certos aspectos de seu serviço. Esta funcionalidade é conhecida como "*Customer Network Management*" (CNM). O serviço CNM suportado pelos provedores facilita o planejamento, operação, manutenção, administração, reconfiguração da porção da rede do cliente alugada ao provedor.

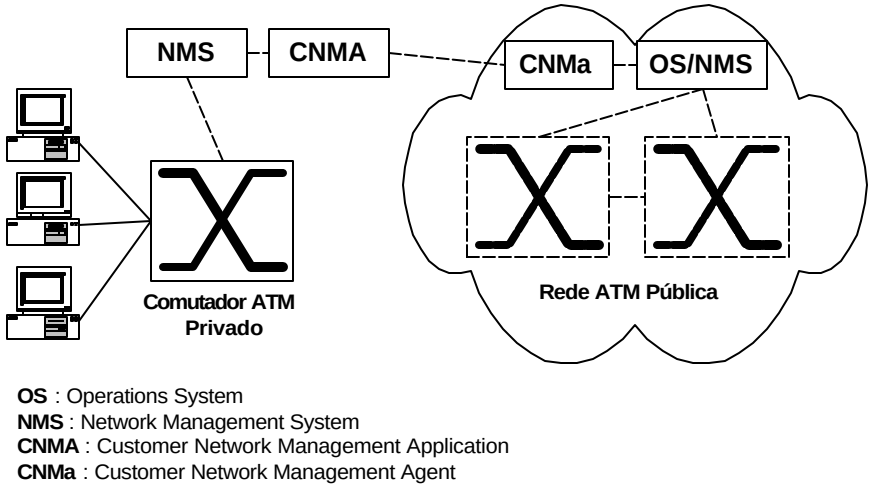


Figura 4.2 : Ambiente CNM (Customer Network Management)

4.2 Áreas Funcionais de Gerência ATM

Como dito, ATM é uma tecnologia que possui características próprias. Isto acarreta no acréscimo de novas funcionalidades em cada uma das áreas funcionais de gerenciamento. Algumas destas funcionalidades são descritas a seguir.

Configuração : As expectativas dos usuários com os níveis de serviço da rede e demanda por capacidade são determinadas quando da configuração da rede. É importante que estas expectativas sejam alcançadas de maneira que a relação custo-benefício seja valorizada. Assim, a gerência de configuração consiste em obter dados e usá-los para gerenciar todos os recursos por meio de policiamento. Alguns exemplos de

requisitos de configuração em uma rede ATM são :

- Criar e liberar conexões ATM (VPCs e VCCs);
- Monitorar o *status* da conexão;
- Determinar o número de conexões ativas numa interface;
- Determinar o número máximo de conexões suportadas numa interface;
- Determinar o número de conexões pré-configuradas numa interface;
- Configurar o número de bits suportadas para VPI/VCI ;
- Configurar e determinar o *status* do endereço da interface.

Segurança : As funções de segurança em redes ATM, são principalmente implementadas por concessionárias que utilizam este tipo de serviço, com o objetivo de garantir privacidade das informações das corporações e a integridade tanto destas informações quanto da rede pública. Algumas das funções desempenhadas pela área de segurança no ambiente ATM são:

- Realizar o tratamento das notificações recebidas;
- Obter informações de configuração da UNI;
- Iniciar testes;
- Reconfigurar PVCs;
- Requisitar a notificação de determinados eventos.

Falha : As funções de falha aplicadas a redes ATM, necessitam lidar com altas velocidades e terem a habilidade de trabalhar com vastos volumes de dados gerados a partir de diversas categorias de serviço, fazendo com que os problemas e soluções de gerência nestes ambientes tenham outra dimensão. Aplicado às redes ATM, algumas funções de gerência de falha incluem:

- Notificação da inabilidade de se estabelecer conexões ATM;
- Notificação de falha de uma conexão ATM;
- Notificação de múltiplas e simultâneas falhas em conexões;
- Notificação de falha nos pares adjacentes de uma UNI;
- Suporte a fluxo de células OAM de gerência de falha.

Desempenho : Os mecanismos de análise de desempenho de hardware, software e meio de transmissão da rede também necessitam lidar com altas velocidades. A investigação quantitativa dos recursos da rede, para verificar a garantia dos níveis de serviço, sofre contínuas alterações em virtude das características da tecnologia ATM. Além disso, ajustes com o objetivo de melhora no desempenho também se tornam mais complexos e devem levar em conta um período de tempo aceitável para a sua execução. Aplicado às redes ATM, os requisitos de gerência de desempenho incluem:

- Determinar se ou não uma conexão ATM reúne os requisitos de QoS;
- Determinar o número de células que violam o contrato de tráfego;
- Suportar fluxo de células OAM de desempenho;
- Suportar atualizações dos parâmetros, para receber e enviar operações nos níveis de VP e VC.

Contabilização : As funções de contabilização devem prover ao sistema uma maneira de identificar tanto as classes de serviço utilizadas, quanto suas medidas de utilização. Em função disto, os cálculos de limitação da quantidade de recursos alocados para os usuários e de recursos utilizados também se tornam mais complexos. Dentre as funções de contabilização aplicadas à redes ATM temos:

- Registro da QoS de uma conexão;
- Registro de largura de banda de uma conexão;
- Registro de duração da conexão;
- Registro do número de células transmitidas e recebidas com sucesso;
- Registro do número de células recebidas com erro;
- Registro do número de células recebidas com violação ao contrato de tráfego.

Alguns autores [Minoli97] julgam necessário, no caso do ATM, a definição de outras áreas funcionais. Dentre elas temos:

Planejamento de Recursos : Esta área tem como principais funções o planejamento da rede e a modelagem de recursos (ex. comutador ATM). Esta funções devem ser

desempenhadas por ferramentas inteligentes de modelagem a partir de dados reais da rede.

Provisionamento de Serviços : Esta área pretende garantir a qualidade de serviço desejada, através de funções de tempo-real como: configuração dos parâmetros das conexões virtuais através do gerenciamento de prioridade de tráfego e da viabilização de novas rotas. Isto tudo levando-se em conta a disponibilidade de banda e outros fatores que descrevem o estado corrente da rede.

4.3 Abordagens Atuais para Gerência ATM

As abordagens de gerência de redes ATM estão atualmente definidas em três áreas gerais: gerência de interface, incluindo UNI, DXI e LUNI; gerência de camada (ex. células OAM) e gerência global da rede.

A gerência de interface trabalha com a troca de informações a nível de interface. Ela tem como principal objetivo, a configuração e a geração de alarmes de interfaces ATM, e prover as funções para que dois dispositivos tenham uma interface padronizada de comunicação. A especificação ILMI realizada pelo ATM Forum, a ser descrita na seção 4.3.1, apresenta uma das abordagens para a gerência a nível de interface.

A gerência a nível de camada permite testes de continuidade e *"loopback"* em um segmento ou a nível de circuito virtual fim-a-fim (VC e VP). Células OAM, especificadas pelo ITU-T e apresentadas na seção 4.3.2, são o principal mecanismo disponível para o gerenciamento de camadas.

A gerência de rede global tanto fornece uma visão de redes públicas, privadas e híbridas como padroniza as funções de monitoramento e controle dos dispositivos e serviços ATM. O ATM Forum definiu um modelo de cinco interfaces dentro desta abordagem, que será discutido na seção 4.3.3.

4.3.1 ILMI (Integrated Local Management Interface)

Este modelo é incluído como parte da UNI (*User-Network Interface*) 4.0. O propósito deste padrão é auxiliar no gerenciamento de configuração e "status" de uma interface específica. Ele também incorpora um mecanismo para registro de endereços e serviços ATM através da UNI.

Cada UNI de um dispositivo possui uma IME (*Interface Management Entity*), associada a esta, para suportar as suas funções. As informações de gerenciamento são trocadas somente entre IMEs adjacentes.

ILMI define a sua MIB separadamente do seu mecanismo de acesso (Figura 4.3). Esta abordagem não requer que existam agentes nos dois lados da UNI. Por outro lado, foram definidas IMEs que acessam as informações da MIB, diretamente sobre um VCC predefinido (0/16), através de comandos SNMP transportados sobre AAL5. Neste caso, UDP/IP não são utilizados. Outro mecanismo de acesso a esta MIB se dá através de uma sistema de gerenciamento usando SNMP sobre UDP/IP/AAL5.

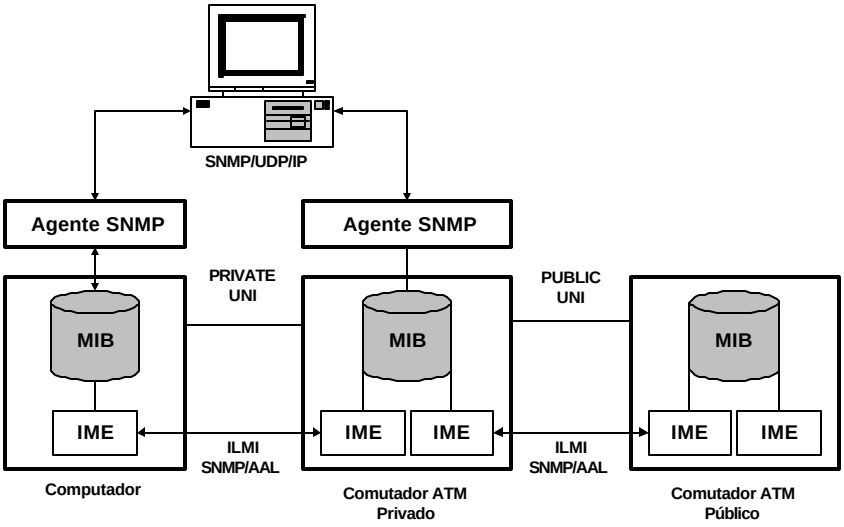


Figura 4.3 : Ambiente ILMI (Integrated Local Management Interface)

Apesar de ser bastante divulgada, ILMI não provê funções de gerenciamento nas áreas de falha, segurança e contabilização, possui limitações em relação ao gerenciamento de interfaces entre redes, e não distribui inteligência de gerenciamento sobre a rede [Alexander95].

4.3.2 Células OAM

Especificada juntamente com a UNI, as células OAM provêem funcionalidades para a gerência a nível de camada. Estas células são utilizadas na distribuição automática de informações de gerenciamento através de toda a rede e na execução de testes (falha e desempenho) dentro destas. Iniciadas a partir de dispositivos ATM, estas células tem a capacidade de conseguir informações sobre conexões fim-a-fim, reduzindo tanto a necessidade de se distribuir MIBs por toda a rede, como a quantidade de tráfego relacionado ao gerenciamento.

Para VPs (Virtual Paths), as funções OAM são suportadas por células ATM especiais, que são transmitidas sobre VCs com VCI específico (Fluxo F4). Para VCs (*Virtual Channels*), estas funções são desempenhadas por células que possuem um código no campo PTI (*Payload Type Identifier*) (Fluxo F5).

Pontos finais de uma conexão ou de segmentos de conexão, terminam e processam todas as células OAM. Estes podem tanto gerar, como inserir células OAM para processamento "*downstream*". Pontos intermediários podem monitorar ou inserir células OAM, mas não terminar o seu fluxo.

Várias funções são desempenhadas por estas células. Em função disto, elas foram divididas em três tipos:

- Células OAM para Gerenciamento de Falhas: estas células são transmitidas para indicar condições de falha, como descontinuidade em um circuito virtual. Podem ser utilizadas para realizar vários tipos de teste num circuito virtual.
- Células OAM para Ativação / Desativação: são responsáveis pela ativação/desativação de células OAM e de funções de processamento associadas com certas capacidades de gerenciamento em VPC/VCC.
- Células OAM para Gerenciamento de Desempenho: estas células são transmitidas regularmente entre pontos finais de um circuito virtual específico e são utilizadas para monitorar parâmetros de desempenho, como CLR (*Cell Loss Ratio*).

Na Tabela 4.1 abaixo são descritas as funções de cada célula, de acordo com a sua área funcional de gerenciamento.

Áreas de Gerência (Funções)	**Tipos de células OAM**
Falha	Alarm Surveillance: **AIS** (Alarm Indicator Signal) Alarm Surveillance: **RDI** (Remote Defect Indicator) – FERF Connectivity verification: **LB** (OAM Cell Loopback) Continuity Check (**CC**)
Desempenho	Forward Monitoring Backward Monitoring Monitoring / Reporting
Ativação e Desativação	Performance Monitoring Continuity Check (**CC**)

Tabela 4.1 : Tipos de Células OAM

4.3.3 Modelo de Referência de Gerenciamento de Redes ATM

Nesta seção é apresentado o modelo de gerenciamento de redes ATM sugerido pelo ATM Forum. Este modelo descreve os vários tipos de gerenciamento de redes necessários para suportar redes privadas, redes públicas e redes híbridas. Especifica também *"gateways"* entre sistemas baseados nos protocolos SNMP e CMIP, e sistemas com protocolo proprietário.

Para isto foram criadas cinco interfaces, nomeadas M1, M2, M3, M4 e M5 (Figura 4.4). Estas interfaces são essenciais para o monitoramento e controle fim-a-fim de uma rede ATM. Abaixo são descritas as funções de cada interface.

- M1 é necessária para realizar a gerência de um dispositivo ATM.
- M2 é utilizada para se realizar o gerenciamento de uma rede ATM privada.
- M3 permite que o usuário supervisione sua porção de uma rede pública ATM.
- M4 é necessária para se gerenciar uma rede pública ATM. Ela inclui as funções de

gerenciamento de elementos de rede e funções de gerenciamento de serviços [M4-96].

- M5 é utilizada para se gerenciar a interação entre dois sistemas de gerência de redes públicas ATM.

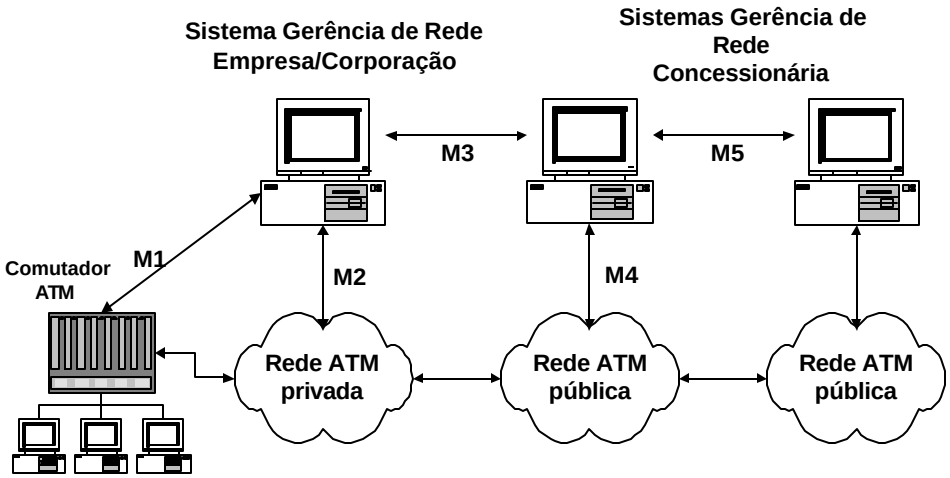


Figura 4.4 : Modelo de Referência de Gerencia mento de Redes ATM

As interfaces M1 e M2 possuem suas especificações baseadas no protocolo SNMP, em virtude deste ser amplamente empregado atualmente. Gerentes de uma rede privada estarão particularmente interessados nas interfaces M1 e M2. Estas interfaces utilizam MIBs padronizadas pelo IETF e ATM Forum, tais como: MIB II, MIBs padrões para conexões (DS-1, DS-3 e Sonet), MIB AToM e MIB ILMI.

O serviço M3 é disponibilizado pelo provedor de rede pública através do agente CNM (*Customer Network Management*) localizado no provedor de rede ATM. O agente CNM se baseia no protocolo SNMP e suporta as mesmas MIBs definidas para as interfaces M1 e M2. Os provedores de rede ATM planejam estender as informações CNM disponibilizadas para que os gerentes de redes privadas tenham controle em tempo-real sobre os serviços utilizados.

M4 [M4-94], que é a interface que delimita redes públicas e privadas, disponibiliza visões a nível de elementos de rede e de serviço para os sistemas de gerenciamento do provedor de rede pública. A interface M5 é a mais complexa de todas e especifica a interação entre dois sistemas de gerenciamento de provedores de rede pública.

4.4 Gerenciamento de Tráfego

O gerenciamento de tráfego e congestionamento em redes ATM é fundamentado na habilidade da rede de prover qualidade de serviço diferenciada a cada aplicação. O principal papel do gerenciamento de tráfego é proteger a rede e as estações de congestionamento, de maneira a garantir os níveis de desempenho desejado. Outro papel importante é promover a utilização eficiente dos recursos da rede.

A suposição que a multiplexação estatística pode ser utilizada para aumentar a utilização do enlace, ocorre em virtude dos usuários não usarem seus valores de taxa de pico simultaneamente. Mas já que as demandas de tráfego são estocásticas e não podem ser previstas, congestionamento é inevitável.

Mecanismos de controle de tráfego são utilizados pela rede para garantir que usuários disponham da qualidade de serviço (QoS) necessária. Estes mecanismos tentam evitar certas situações, tais como: taxa total de entrada maior que a capacidade do enlace de saída (congestionamento), comprimento da fila cresça rapidamente, (gerando "*buffer overflow*" e perdas de células), etc.

As seguintes funções formam uma abordagem para se gerenciar e controlar o tráfego e congestionamento numa rede ATM. Estas funções podem ser combinadas dependendo da categoria de serviço a ser suportada. Abaixo são descritas algumas delas.

- Controle de Admissão de Conexão (CAC): conjunto de ações que determinam a aceitação ou não de uma requisição de conexão. Estas ações são executadas pela rede durante a fase de estabelecimento de conexão.
- Usage Parameter Control (UPC): conjunto de ações que fazem o controle e o monitoramento do tráfego, baseados no tráfego negociado e efetivamente gerado. Seu principal objetivo é proteger os recursos da rede de usuários mal intencionados ou que mesmo sem intenção, afetam a qualidade de serviço de outras conexões. Esta proteção é realizada através da detecção de violações dos parâmetros negociados e da ação apropriada. Tais ações incluem descarte ou marcação de células.

- Descarte de *"Frames"*: ação a ser tomada em caso de congestionamento na rede. Neste caso são descartadas células a nível de *frame*.
- Gerenciamento dos Recursos da Rede (NRM).
- Controles de *"Feedback"*: conjunto de ações utilizadas para regular o tráfego submetido às conexões ATM, de acordo com o estado dos recursos da rede. Estas ações são tomadas pela rede e pelas estações.
- Controle de Prioridade de Perda de Células: Algumas categorias de serviço permitem que estações gerem células com o campo CLP (*Cell Loss Priority*) marcadas. A rede pode tratar estas células de forma transparente ou "significante". No segundo caso, a rede pode seletivamente descartar células marcadas de baixa prioridade, para proteger os objetivos de qualidade de serviço de células com maior prioridade.
- "*Traffic Shaping*": mecanismo utilizado para se alterar as características de tráfego de acordo com o desejado.
- Controle de Fluxo ABR: este protocolo pode ser utilizado para adaptativamente compartilhar a largura de banda disponível entre os usuários.

PARTE II

"Descrição do ÁTILA"

Introdução

Dadas as particularidades da comutação de célula na tecnologia ATM, surgem novos requisitos quanto as atividades de monitoramento e controle do ambiente gerenciado. É natural que um modelo de gerência bem formalizado, como o das redes de pacotes, fosse a primeira tentativa para o processo de gerência destas redes.

Portanto, o paradigma Gerente x Agente x MIB, usado tanto em redes OSI (CMIP/S) quanto em redes TCP/IP (SNMP), tem sido implementado como soluções em sistemas de gerência em produtos comerciais. No entanto, este modelo Cliente/Servidor não atende satisfatoriamente as exigências dos sistemas cada vez mais complexos baseados na tecnologia ATM.

A Parte II deste trabalho apresenta o ÁTILA, uma proposta no escopo de solução não convencional ao problema da gerência de redes ATM. São descritas diferentes visões da proposta, através de suas arquiteturas física e funcional. É feita uma análise de cada um dos elementos da arquitetura e são apresentadas diversas possibilidades para a implementação de um protótipo. A maior contribuição do ÁTILA consiste, provavelmente, na proposição de uma metodologia baseada em um fluxo dinâmico de informações que bem representa o comportamento diferenciado do ÁTILA ante problemas mal resolvidos (ou não resolvidos) pelas abordagens atuais de gerência ATM citadas no capítulo 04.

Capítulo 05 – Arquitetura

5.1 Introdução

Este trabalho avalia a eficiência do modelo de gerência que vem sendo utilizado em redes ATM. Duas abordagens podem ser adotadas na solução de problemas não completamente solucionados pelos atuais modelos :

- Melhoria do modelo tradicional, adicionando novas funcionalidades e pequenas alterações na sua estrutura, sem contudo alterar as características básicas do paradigma adotado.
- Redefinição total da estrutura de gerência, a partir de um questionamento da adequabilidade dos padrões de gerência atuais.

As abordagens de gerência ATM citadas no capítulo 04, referem-se a proposta relacionada a melhoria dos modelos existentes, descrevendo adaptações e extensões do paradigma Gerente/Agente. São elas, ILMI, Células OAM e MR-ATM Forum. A arquitetura proposta, ÁTILA, é um esforço na direção da segunda proposta, ou seja, a redefinição da estrutura de gerência.

5.2 Arquitetura Funcional

5.2.1 Diagrama da Arquitetura Funcional

A “Arquitetura Funcional” do ÁTILA (Figura 5.1) é representada por um conjunto de blocos funcionais, onde cada bloco realiza suas funções específicas relativas ao tratamento e processamento de informações de gerência. Além disso, é apresentado um conjunto de interfaces que definem as operações disponíveis aos blocos funcionais. Estas interfaces tem como objetivo prover a troca, acesso e manipulação de informações entre blocos adjacentes. Elas definem pontos conceituais que representam os limites entre os blocos funcionais, com o propósito de identificar o tipo de informação que é

trocada entre estes blocos. Estas interfaces são baseadas nos conceitos de orientação a objeto. Portanto, as mensagens trocadas manipulam objetos. Além dos objetos também estão definidas as operações válidas nestas interfaces.

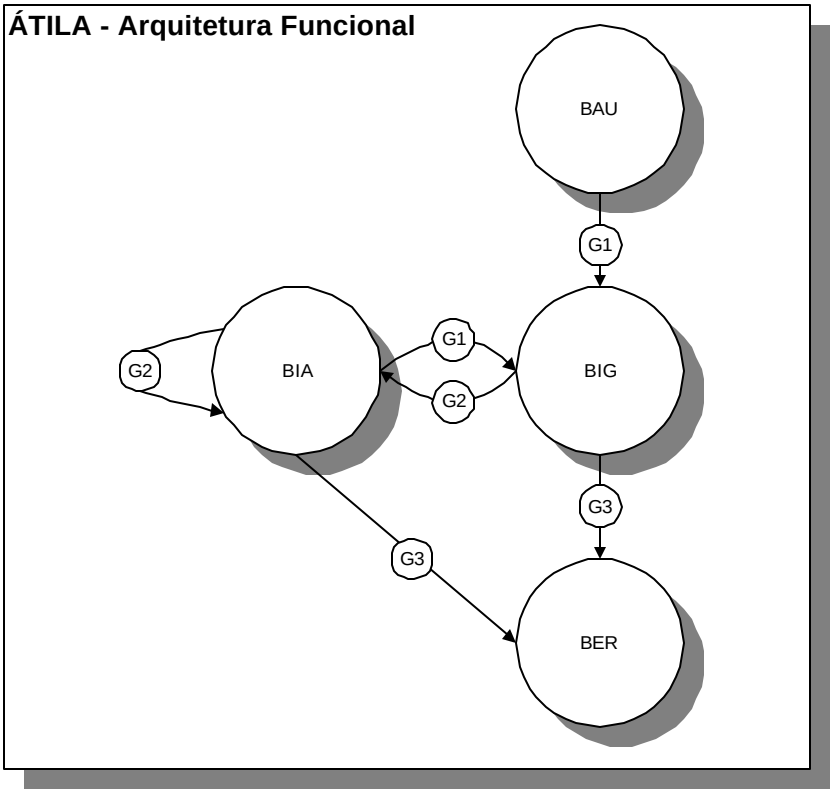


Figura 5.1 : Arquitetura Funcional do ÁTILA

5.2.2 Blocos Funcionais

5.2.2.1 BAU (Bloco Acesso ao Usuário)

Este bloco tem como principal objetivo disponibilizar uma interface de acesso e manipulação às informações de gerência da rede ao usuário, através de um conjunto de operações pelas quais este pode interagir com o ambiente a ser gerenciado.

Funções de tratamento e apresentação de informações, devem ser disponibilizadas com o objetivo de fornecer uma interface "amigável" de visualização da rede.

Com o objetivo de prover aos usuários a gerência de suas redes a partir de diferentes localidades, outra importante função é a facilidade de acesso ao sistema, obtida através de independência de plataforma.

O BAU possui interface com o bloco Inteligência do Gerente (BIG).

5.2.2.2 BIG (Bloco Inteligência do Gerente)

Este bloco processa e armazena informações globais relacionadas com o gerenciamento de redes para suportar, coordenar e controlar a realização das várias funções de gerenciamento.

As informações globais citadas acima, disponibilizam ao usuário uma visão ampla do ambiente gerenciado. Estas informações permitem a realização de avaliações e tomada de decisões, em função do processamento inteligente. Este é um dos fatores que caracteriza o comportamento pró-ativo da arquitetura proposta.

Este bloco possui interfaces com os blocos Acesso ao Usuário (BAU), Inteligência do Agente (BIA) e Elemento de Rede (BER), descritos a seguir.

5.2.2.3 BIA (Bloco Inteligência do Agente)

Este bloco realiza o monitoramento e o processamento local de informações relacionadas com o gerenciamento de um elemento de rede, com o objetivo de executar as funções de gerenciamento a este nível.

O BIA é baseado na distribuição das funções de gerência. A análise das informações e das mensagens a serem passadas ao BIG, através da interface, é realizada de forma distribuída caracterizando os vários níveis de processamento e de inteligência.

Problemas podem ser previstos, analisados e solucionados localmente ao elemento de rede, em função da distribuição da inteligência junto a estes elementos e da garantia de acesso a informações atualizadas em virtude da proximidade das fontes de informação.

Estas características fornecem subsídios para o comportamento pró-ativo da arquitetura também a nível de elemento de rede.

Este bloco possui interfaces com os blocos Inteligência do Gerente (BIG) e Elemento de Rede (BER).

5.2.2.4 BER (Bloco Elemento de Rede)

Este bloco representa as funções dos recursos de rede relacionadas ao gerenciamento. Ele disponibiliza as funções que permitem o monitoramento e o controle destes recursos, de forma a fornecer as informações necessárias ao gerenciamento e atuar nos elementos físicos da rede.

Este bloco possui interfaces com os blocos Inteligência do Gerente (BIG) e Inteligência do Agente (BIA).

5.2.3 Interfaces

5.2.3.1 Interface "G1"

A interface "G1" encontra-se entre os blocos funcionais "BAU" e "BIG" e entre o "BIA" e o "BIG".

Ela disponibiliza funções que possibilitam o acesso/manipulação das informações de gerência contidas no "BIG" aos blocos BAU e BIA. Estas funções, tem como objetivo, tanto fornecer ao usuário o suporte necessário ao desenvolvimento de aplicações de gerência, como possibilitar a troca de informações e mensagens entre os blocos "BIA" e "BIG".

Esta troca de informações e mensagens permite aos elementos do BIA interagirem com os elementos do BIG, com objetivo de obter informações gerais da rede, podendo assim, ter uma visão ampla da rede e desempenhar suas funções de maneira pró-ativa.

5.2.3.2 Interface "G2"

Esta interface encontra-se entre os blocos funcionais "BIG" e "BIA" e entre os elementos que compõe o "BIA".

Em virtude do caráter pró-ativo e distribuído da arquitetura, o monitoramento e controle junto ao elemento de rede, pode ser requisitado pelo BIG aos elementos do BIA. Estas requisições são realizadas através das operações disponibilizadas nesta interface. Isto visa a descentralização das atividades de gerência.

A interação entre os elementos do BIA, é realizada em função da necessidade de haver cooperação tanto de informações como de inteligência para análise, prevenção e solução de problemas localmente ao elemento de rede.

5.2.3.3 Interface "G3"

A interface "G3" encontra-se entre os blocos funcionais "BIG" e "BER", e entre o "BIA" e o "BER".

Ela permite o controle e monitoramento dos elementos de rede pelo "BIA" e "BIG". É através desta interface que se tem acesso aos elementos de rede propriamente dito, e às funcionalidades e informações de gerência que eles disponibilizam.

5.3 Arquitetura Física

5.3.1 Diagrama da Arquitetura Física

A "Arquitetura Física" (Figura 5.2) define os elementos físicos e as interfaces que os interligam. Estes elementos constituem os blocos que representam as implementações das funcionalidades definidas nos blocos contidos na "Arquitetura Funcional".

Esta arquitetura descreve a distribuição das funcionalidades, refletindo o caráter distribuído do ÁTILA. Ela agrupa os blocos funcionais em entidades físicas visando

atingir requisitos de flexibilidade. Um elemento físico pode implementar mais de um bloco da arquitetura funcional e as interfaces desta arquitetura eqüivalem as interfaces da arquitetura funcional.

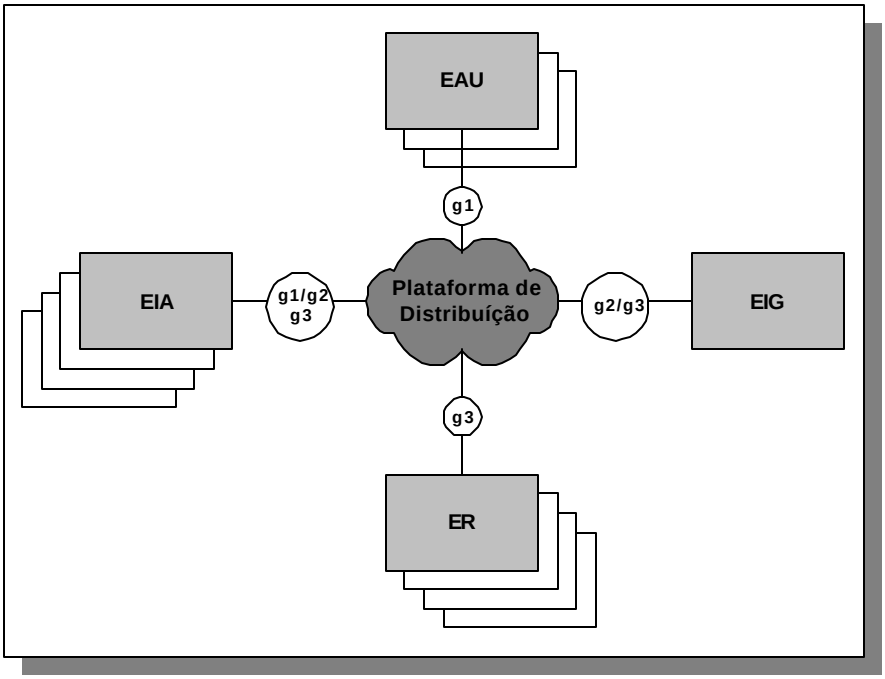


Figura 5.2 : Arquitetura Física do ÁTILA

5.3.2 Elemento Físicos

5.3.2.1 EAU (Elemento Acesso ao Usuário)

Correspondendo ao "BAU", este elemento físico realiza o tratamento das informações disponibilizadas pelo "BIG" e apresenta-as de forma "amigável" aos usuários de informações de gerência. Existem diversas tecnologias que podem ser utilizadas na implementação destas funções. Dentre elas temos:

- Interfaces proprietárias que são acessadas a partir de um cliente específico e se utilizam de linguagens de programação gráficas como: JAVA, Delphi, Visual Basic, PowerBuilder, etc.;
- Interfaces WWW (Applets JAVA, HTML, CGI, etc.).

5.3.2.2 EIG (Elemento Inteligência do Gerente)

Correspondendo ao "BIG", este elemento físico deve conter dentre outras funcionalidades, o armazenamento das informações de gerência, o tratamento inteligente destas informações e uma interface simples e funcional de acesso/manipulação deste elemento.

O EIG deve prover as alternativas de centralizar ou distribuir as funções globais, as quais incluem suporte às aplicações, funções de banco de dados, análise de informações, formatação de dados, relatórios, etc..

As tecnologias vigentes que podem implementar as referidas funções são :

- Sistemas Especialistas [Rocha96][Artola96];
- Redes Neurais [Vieira96];
- Banco de Dados Ativo [Hasan96], etc.

5.3.2.3 EIA (Elemento Inteligência do Agente)

Este elemento físico que corresponde ao bloco funcional "BIA", deve ter autonomia para realizar monitoramento e controle dos recursos da rede, a partir de uma visão global disponibilizada por outros elementos do "EIA", e pelo "EIG".

Em função da não trivialidade destas funções, cada vez mais utiliza-se mecanismos inteligentes para tal tarefa. Para implementar este elemento pode-se utilizar:

- Sistemas Especialistas [Rocha96][Artola96];
- Redes Neurais [Vieira96];
- Agentes Inteligentes [Gaiti93].

5.3.2.4 ER (Elemento de Rede)

O elemento ER corresponde ao bloco funcional "BER", e desempenha as funções de gerência nos elementos de rede. Dentre estas funções temos: armazenamento de informações locais de gerenciamento, controle destas informações e a interação com o recurso físico gerenciado.

Existem atualmente dois padrões que disponibilizam estas funções, que são :

- CMIS/CMIP (Common Management Information Service/Protocol) [Stallings93];
- SNMP (Simple Network Management Protocol) [Rose95].

5.3.2.5 Plataforma de Distribuição

Esse elemento físico representa o elo de ligação entre os demais elementos da arquitetura. Através dessa plataforma, ocorre troca de informações, usuários acessam/manipulam informações sobre o sistema, etc.

Várias tecnologias podem ser utilizadas para implementar esse elemento, dentre elas destacam-se as arquiteturas :

- ANSA [APM93] [Marshak91];
- CORBA [Siegel96].

Capítulo 06 – Fluxo Dinâmico de Informações

6.1 Introdução

A principal contribuição do ÁTILA, evidenciado em sua arquitetura funcional, é a utilização de inteligência e distribuição no gerenciamento pró-ativo de redes ATM. Essa característica é atingida pela interoperação de seus blocos funcionais.

Esta distribuição permite uma flexibilidade não existente nos atuais modelos de gerência ATM vistos no capítulo 04. Assim, os blocos BIG e BIA engendram um processo de cooperação que agrega ao ÁTILA uma gama de possibilidades de interação no processo da atividade de gerência.

Esta gama de interações, ou fluxo dinâmico de informações, favorece, como será visto a seguir, a pró-atividade do ÁTILA.

6.2 Diagrama

A figura 6.1 apresenta o diagrama de fluxos de informação do ÁTILA. A associação destes fluxos formam um conjunto de possibilidades de interação entre os elementos gerentes e agentes do processo de gerência através dos blocos funcionais inteligentes (BIG e BIA), o que constitui uma importante característica do ÁTILA.

Na tabela 6.1 são descritos os fluxos de informação apresentados na figura 6.1 e as restrições das interações entre os blocos da arquitetura funcional. Estas restrições são intrínsecas a concepção da arquitetura e dizem respeito às operações disponíveis entre os blocos funcionais, representadas na arquitetura funcional pelas interfaces entre esses blocos.

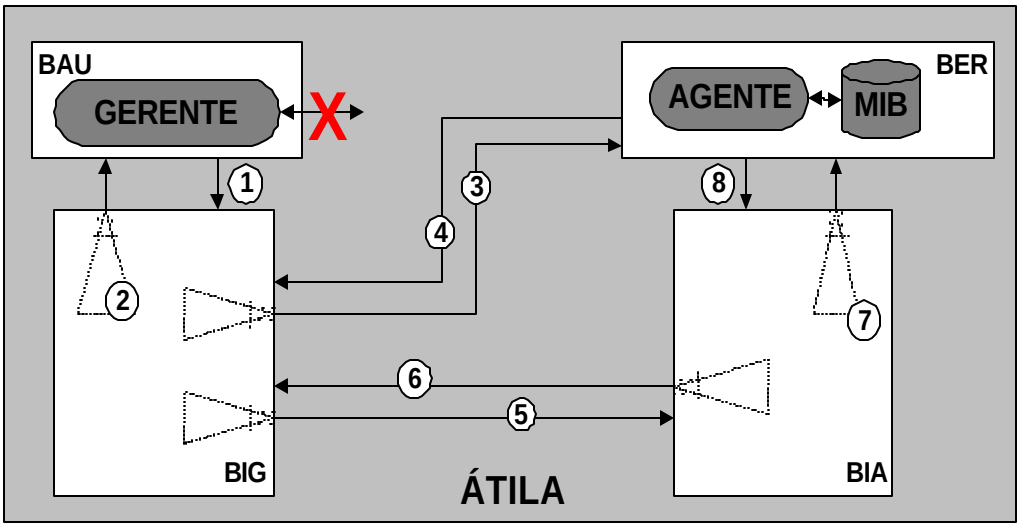


Figura 6.1 : Diagrama do Fluxo Dinâmico de Informações do ÁTILA.

6.3 Exemplificação

Após apresentar os fluxos simples de informação entre os blocos funcionais da arquitetura e suas restrições de interoperação, são descritos exemplos reais de interação entre estes blocos. Estes exemplos mostram os tipos de informação e as operações que são trocadas entre as partes, além do comportamento dos blocos funcionais no tratamento de problemas de gerenciamento.

Estes exemplos estão estruturados através de diagramas e de uma seqüência de passos, nomeados com letras do alfabeto. Caso haja a possibilidade de mais de um fluxo ser utilizado em um determinado instante, estes são seguidos por números. Além disso, os passos foram organizados de forma identada para facilitar o entendimento por parte do leitor. No caso dos diagramas, os fluxos são identificados pelo identificador do passo seguido do número do fluxo correspondente a figura 6.1.

De	Para	Restrições das Interações	Fluxos
BAU (Bloco Acesso ao Usuário)	BIG	Sem restrições	1
	BIA	Deve sempre passar pelo BIG	$1 \rightarrow 5$
	BER	Sem passar pelo BIA	$1 \rightarrow 3$
		Passando pelo BIA	$1 \rightarrow 5 \rightarrow 7$
BIG (Bloco Inteligência do Gerente)	BAU	Sem restrições	2
	BIA	Sem restrições	5
	BER	Sem passar pelo BIA	3
		Passando pelo BIA	$5 \rightarrow 7$
BIA (Bloco Inteligência do Agente)	BAU	Deve sempre passar pelo BIG	$6 \rightarrow 2$
	BIG	Sem restrições	6
	BER	Sem restrições	7
BER (Bloco Elemento de Rede)	BAU	Passando somente pelo BIG	$4 \rightarrow 2$
		Passando pelo BIG e BIA	$8 \rightarrow 6 \rightarrow 2$
	BIG	Sem passar pelo BIA	4
		Passando pelo BIA	$8 \rightarrow 6$
	BIA	Sem restrições	8

Tabela 6.1 : Fluxos e Restrições de Interação dos Blocos Funcionais

Exemplo 01:

Fluxos: **1 ® 3 ® 4 ® 2**

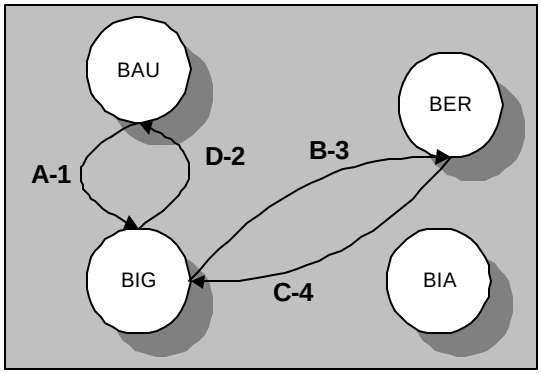


Figura 6.2 : Exemplo 01

A. O usuário do sistema, a partir do BAU requisita informações da rede (monitoramento) ao BIG (1);

B. O BIG (Inteligência do Gerente) avalia o pedido e requisita uma operação na MIB através do BER (3);

C. O resultado da requisição é retornado ao BIG pelo BER (4);

D. O BIG novamente analisa e trata as informações e as repassa ao usuário através do BAU (2).

Exemplo 02:

Fluxos: **3 ® 4 ® 3**

3 ® 4 ® 2

A. O BIG requisita informações da MIB através do BER. (3) (O BIG realiza uma análise do estado/comportamento da rede a partir de suas informações da rede como um todo. Estas avaliações podem resultar na necessidade de acesso/manipulação de informações da MIB, ou seja monitoramento e controle nos elementos de rede.);

B. As informações requisitas são retornadas ao BIG pelo BER (4);

C1. O BIG através do BER age nos elementos de rede (3) (De posse de novas informações o BIG realiza uma nova avaliação da rede e age junto aos elementos de rede.);

C2. Ou o BIG repassa as informações ao BAU (2) (Após análise nota-se que é preciso a interferência do operador para solucionar o problema.).

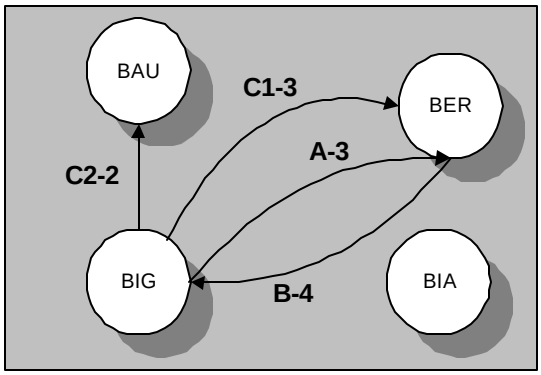


Figura 6.3 : Exemplo 02

Exemplo 03:

Fluxos: **1 ® 2**
1 ® 3 ® 4 ® 2

A. O BAU realiza uma consulta no BIG (1) (Esta consulta tem como objetivo a obtenção de informações de configuração e/ou estatística, especificas de um elemento de rede ou gerais da rede.);

B1. O BIG ou retorna as informações requisitadas ao BAU (2) (Se o BIG possuir as informações requisitadas ele retorna estas ao BAU. Voltando ao passo A.);

B2. Ou o BIG interagi com BER para acessar informações (MIB) de um ou mais elementos de rede (3) (Isto ocorre caso o BIG não contenha as informações requisitadas pelo BAU, ou caso as informações não estejam atualizadas.);

B2.1. As informações são retornadas pelo BER (4) ao BIG;

B2.2. O BIG retorna as informações requisitadas ao BAU (2) (Antes de retornar as informações ao BAU, pode ser necessário que o BIG realize um tratamento/filtragem destas informações.).

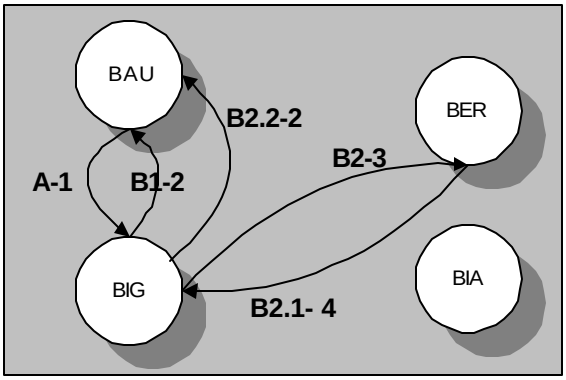


Figura 6.4 : Exemplo 03

Exemplo 04:

Fluxos: **1 ® 5 ® 7 ® 8 ® 7**

1 ® 5 ® 7 ® 8 ® 6 ® 5

1 ® 5 ® 7 ® 8 ® 6 ® 3

1 ® 5 ® 7 ® 8 ® 6 ® 2

A. O BAU solicita ao BIG informações de um ou mais elementos de rede (1) (A partir da necessidade de se ter estudos estatísticos detalhados realizados pelo sistema de gerência em determinados elementos de rede o bloco BAU pode solicitar ao bloco BIG tais informações.);

B. O bloco BIG solicita ao BIA que os seus elementos o auxiliem a realizar as funções solicitadas a ele pelo BAU (5) (O bloco BIG com o objetivo de distribuir as funções a ele requisitadas, ativa elementos do BIA para realizar monitoramento e controle junto aos elementos de rede.);

C. O BIA realiza o monitoramento e controle dos elementos de rede através do bloco BER (7) (Os elementos do BIA necessitam de informações de um ou mais elementos de rede para realizar as funções a eles delegadas. Estas informações são obtidas através do monitoramento contínuo das MIBs por intermédio do BER.);

D. O BER acessa as MIBs dos elementos de rede e retorna as informações obtidas para o BIA (8);

E1. O BIA ou age novamente na MIB através do bloco BER (7) (Os elementos do BIA de posse das informações atualizadas de um ou mais elementos de rede podem realizar uma análise/tratamento destas informações e verificar que necessitam agir novamente nos elementos de rede.) (Voltando ao passo C.);

E2. Ou o BIA retorna as informações ao BIG (6) (O BIA pode não ser capaz de resolver ou prever problemas já que sua visão da rede é limitada. Em virtude disto, seus elementos devem passar as informações obtidas/tratadas ao BIG, já que este tem uma visão mais ampla da rede podendo tomar decisões mais abrangentes.);

E2.1.1. O BIG pode interagir novamente com o BIA (5) (O BIG não possuindo as informações necessárias para realizar suas funções ou tendo a necessidade de executar funções mais complexas, interage com o BIA com o objetivo de distribuir estas funções.) (Voltando ao passo B.);

E2.1.2. Ou o BIG pode interagir com os elementos de rede através do BER (3) (Se as funções a serem executadas pelo BIG forem simples, estas podem ser executadas diretamente no elemento de rede através do BER.);

E2.1.3. Ou o BIG pode passar as suas informações para o BAU (2) (Caso o BIG não tenha condições de resolver/prever problemas, ele pode interagir também com o operador do sistema de gerência através do BAU.);

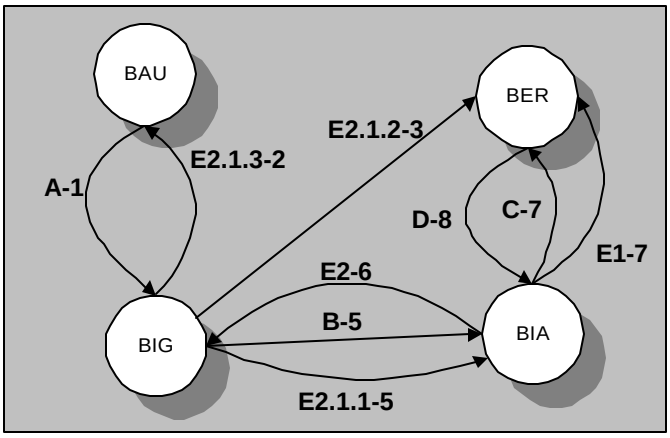


Figura 6.5 : Exemplo 04

Exemplo 05:

Fluxos: **5 ® 7 ® 8 ® 7**

5 ® 7 ® 8 ® 6 ® 5 ® 7

5 ® 7 ® 8 ® 6 ® 3 ® 2

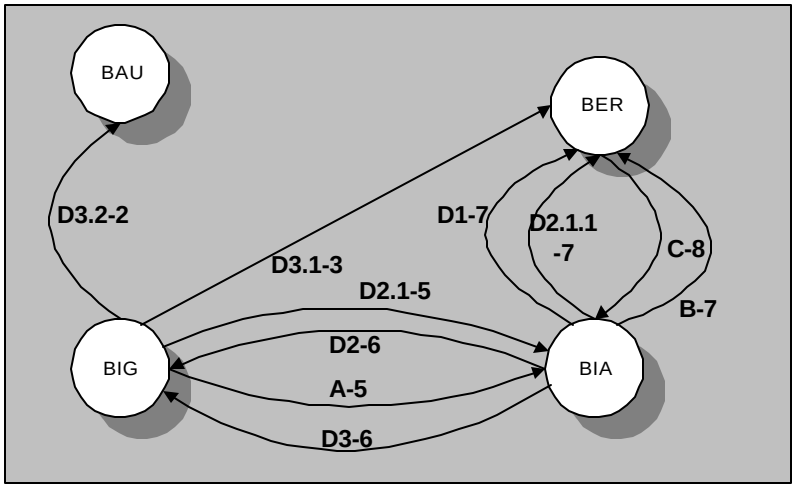


Figura 6.6 : Exemplo 05

A. O BIG ativa elementos do BIA (5) (O BIG toma a decisão de monitorar e/ou controlar a rede de maneira distribuída através de um ou mais elementos do BIA.);

B. Os elementos do BIA efetuam continuamente monitoramento nos elementos de rede através do BER (7) (Em função das requisições feitas ao BIA, seus elementos

monitoram um ou mais elementos de rede a fim de obter dados da rede para executar análises.);

C. As informações são retornadas ao BIA pelo BER (8);

D1. Os elementos do BIA ou voltam a interagir com os elementos de rede através do BER (7) (Os dados coletados no passo anterior são utilizados pelo BIA com o objetivo de realizar análises da rede e possivelmente prever, localizar ou até mesmo solucionar problemas sem a necessidade de trocar informações com o BIG ou BAU (elementos centralizadores). Se as informações e os conhecimentos são suficientes, o BIA age novamente nos elementos de rede (através do BER) para realizar ações de controle na rede.).

D2. Ou o BIA requisita informações ao BIG (6) (Se não existem informações suficientes para realizar uma análise local, e se prever/solucionar algum problema, os elementos do bloco BIA requisitam informações ao bloco BIG com o objetivo de obter informações globais da rede e poder avaliar e prever/solucionar problemas a nível de elemento de rede.);

D2.1. O BIG retorna as informações ao BIA (5);

D2.2. O BIA age nos elementos de rede por intermédio do BER (7) (Voltando ao passo D1.).

D3. Ou o BIA passa as informações ao BIG (6) (Caso não possua o conhecimento necessário, o BIA passa as informações obtidas junto aos elementos ao BIG. Este, possuindo uma visão ampla da rede pode analisar o problema com maior precisão já que também possui conhecimento (inteligência).);

D3.1.1. O BIG ou pode interagir diretamente nos elementos de rede através do BER (3) (Neste caso, se as informações a serem obtidas e/ou as funções a serem desempenhadas pelo BIG forem simples, ele as realiza interagindo diretamente com os elementos de rede através do BER.).

D3.1.2. Ou o BIG interage com o BAU (2) (Neste caso o BIG e BIA, através de suas interações (cooperações), não conseguiram prever/solucionar um problema que ocorreu na rede. Em virtude disto, o BIG repassa o "*status*" do problema ao operador via BAU para que este tome alguma decisão.).

PARTE III

“Implementação do ÁTILA”

Introdução

Em sendo o ÁTILA, a priori, uma arquitetura conceitual (não dependente de implementação), surgem várias possibilidades para sua prototipação.

Como mostrado nas arquiteturas física e funcional, na Parte II, a concepção do ÁTILA é alicerçada em cinco elementos funcionais: interface com o usuário, inteligência do gerente, inteligência do agente, elemento de rede e a distribuição.

A Parte III deste trabalho justifica, dentre as opções apresentadas na Parte II, as tecnologias e respectivas ferramentas escolhidas para a implementação de cada elemento da arquitetura física. Ou seja, as tecnologias e ferramentas referentes aos cinco elementos citados acima são respectivamente : a interface WWW implementada em JAVA, o banco de dados ativo ODE (*Object Database Environment*), o ambiente de suporte a agentes inteligentes JATLite, a API SNMP/JAVA Advent e o ambiente de distribuição ÁBACO, baseado no CORBA.

Em seguida é descrito formalmente cada um dos componentes do protótipo implementado. São descritas as interfaces IDL (*Interface Definition Language*) de acesso e manipulação do ODE, é especificado o agente genérico Átila, são descritas as interfaces IDL de acesso às primitivas do Advent e, finalmente, é apresentado o funcionamento do ambiente ÁBACO.

Os Estudos de Caso, apresentados no capítulo 09, permitem não só uma visão pragmática da utilização do protótipo, como também um mapeamento numa aplicação entre as três visões do ÁTILA: Funcional, Tecnológica e de Ferramentas. Estas visões são apresentadas nos capítulos 05, 07 e 08, respectivamente.

Capítulo 07 – Tecnologias Selecionadas

7.1 Introdução

A figura 7.1 apresenta uma visão geral do ambiente de implementação do ÁTILA. Nesta figura se destacam, o WWW, o Banco de Dados Ativo, os Agentes Inteligentes, o CORBA e o SNMP, que são as tecnologias escolhidas para a implementação de cada elemento da arquitetura física.

Em seguida é justificada a utilização de cada uma destas tecnologias para a implementação do protótipo.

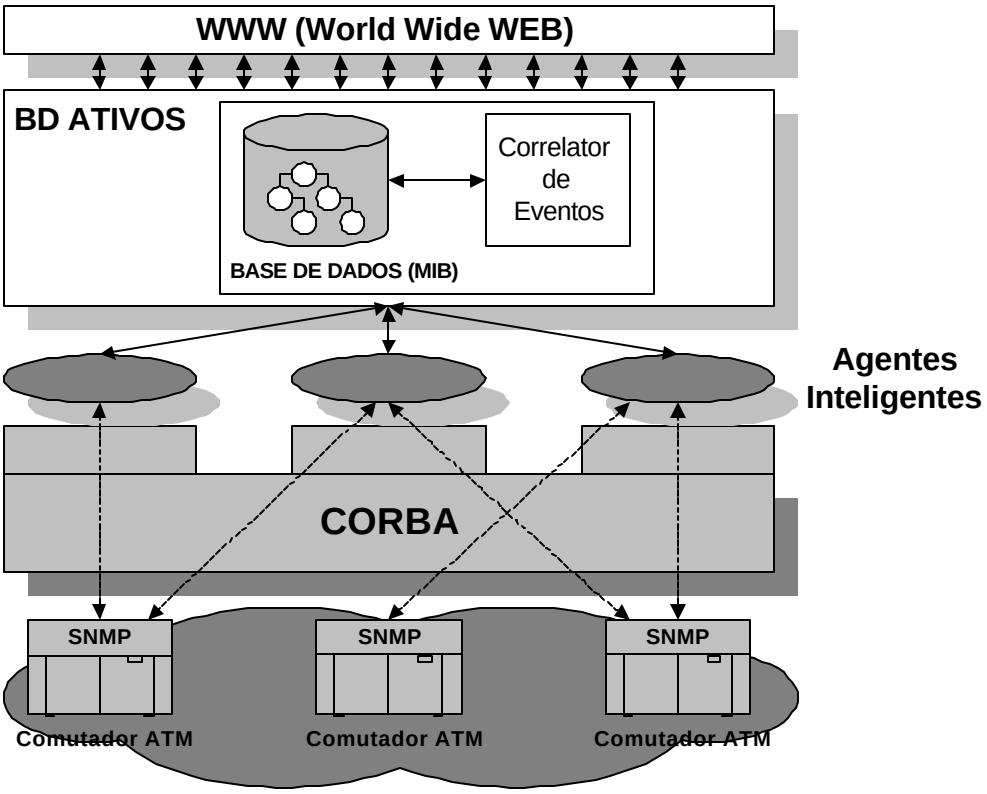


Figura 7.1 : Ambiente de Implementação

7.2 EAU – (Elemento Acesso ao Usuário)

As possíveis formas de implementação deste elemento, citadas na arquitetura física, devem levar em conta não só a simplicidade/expressividade da apresentação das informações de gerenciamento, como também a disponibilidade/independência deste elemento. Em função destas características, este elemento será implementado utilizando-se WWW (*World Wide Web*).

Outras características que nos levaram a utilização de WWW no protótipo foram :

- Protocolos abertos : Tanto o SNMP como o HTTP são protocolos da família TCP/IP. Eles são padronizados pelo IETF, e são largamente aceitos e empregados.
- Acesso remoto : Servidores WEB garantem acesso a partir de qualquer *host*, assumindo que um "*browser*" WEB esteja disponível, ao contrário de uma interface implementada em uma linguagem de programação gráfica, que depende da plataforma, ou seja, que requer um cliente específico instalado.

7.3 EIG (Elemento Inteligência do Gerente)

Dentre as opções citadas na arquitetura física sobre as tecnologias que implementam as funções do elemento físico IG, optou-se por Banco de Dados Ativo (BDA). Uma das principais motivações foi a experiência obtida com o uso desta tecnologia em trabalhos anteriores [Vasconcelos97] e a certificação de que esta tecnologia apresenta bem fundamentadas as funcionalidades necessárias para a implementação do protótipo.

- **Banco de Dados Ativo e Gerenciamento de Redes**

O propósito de um sistema de gerência de redes é monitorar e controlar o funcionamento da rede. Para tanto eles trabalham com grandes volumes de dados e eventos relevantes para realizar estas funções. A grande quantidade de informações (dados e eventos) a serem tratadas, tornou inviável a reação a qualquer simples evento acontecido na rede por parte do sistema. Em função disto, tornou-se obrigatório funcionalidades que permitam a reação inteligente para eventos, ou seja, reagir a eventos somente quando certa condição for satisfeita. Em outras palavras, eventos devem ser filtrados baseados em certos critérios [Hasan96].

O armazenamento destas informações, como é sabido, é realizado por um banco de dados. Os Banco de

dados utilizados atualmente na gerência de redes não possuem as funcionalidades e a integração desejada com o sistema de gerência, e consequentemente não são adequados à gerência pró-ativa. Em função disto, o ambiente proposto em [Vasconcelos97], apresenta o Sistema de Banco de Dados Ativo como uma "solução" para os problemas de gerência em um ambiente ATM.

O tratamento e filtragem de eventos a ser realizado pelas regras do BDA pode ser feito através de correlação de eventos baseado em vários relacionamentos, tais como: relacionamento temporais ou causais [Hasan95]. Quando o evento correlacionado (composto) ocorre, o sistema deve ser capaz de iniciar certas ações automaticamente. O BDA provê facilidades para especificação declarativa destas correlações, o que não acontece na maioria dos gerenciadores de eventos e sistemas de correlação de eventos. Além disso, estes gerenciadores e sistemas disponibilizam apenas correlação de eventos baseados em causalidade, não provendo funcionalidades temporais.

Alguns dos exemplos de funções que podem ser desempenhadas pelo BDA em auxílio à gerência de redes são:

- Detectar automaticamente gargalos na rede e sugerir a reconfiguração da rede;
- Avaliar/analisar inteligentemente numerosas mensagens de falha e tomar as ações corretivas automaticamente ou prover recomendações via console. (Ex. O sistema deverá permitir configurar os alertas que deverão aparecer na console e filtros baseados em atributos, tais como, severidade dos alarmes, localização, etc.);
- Utilizar dados do mundo real para planejamento eficiente de necessidade de recursos e justificativa de custos, sugerindo alterações na topologia da rede para otimizar o desempenho e iniciar novos serviços;
- Auxiliar a análise de tendências.

A correlação de eventos é uma tarefa complexa. De maneira a correlacionar um grande número de eventos, vários fatores devem ser levados em conta, tais como, tipo do evento, estrutura ou topologia da rede, relacionamentos causais e temporais entre eventos, etc. [Hasan96]. Ex. Uma simples falha ou problema (manifestada como um alarme ou notificação) pode causar a geração de vários outros alarmes, os quais são reportados à estação de gerenciamento (sem análise), então o operador será sobrecarregado e pode não estar habilitado a detectar a causa real do problema. Se os vários relacionamentos entre eventos (causais e temporais) e as informações de configuração (estrutura) da rede são conhecidas, então os alarmes podem ser

rapidamente isolados (ignorados).

A funcionalidade de herança de regras também pode ser utilizada para fornecer uma visão abstrata e hierárquica da rede, possibilitando o tratamento (solução) de problemas em vários níveis, dependendo do grau de importância, dificuldade ou criticidade do mesmo. Outra importante característica é a reusabilidade destas "regras" em diferentes aplicações.

Tipos de Informação

Para um sistema de gerência é freqüentemente necessário acessar e manipular tanto dados atualizados quanto dados históricos, com o objetivo de tomar decisões de controle. Para isto o banco de dados ativo possui dados que são classificados em três tipos [Schuhknecht95]: dados estatísticos (estado do sistema), dados estruturais (*layout* lógico e físico do sistema) e dados de controle (*baseline*).

Dados estatísticos que são recebidos pelo BDA enviado pela rede (Ex. tamanho do *buffer*, taxas de retransmissão, *status* do link.) representam, em qualquer momento, o seu estado atual. Elementos de rede podem reportar dados periodicamente ou não. Com atualizações periódicas, dados chegam em freqüências regulares. Alternativamente, eles também podem fornecer dados somente quando eventos extraordinários, tais como falha ou sobrecarga, ocorrem ou após requisição explícita do processo gerenciador da rede.

Já que o volume de informações disponibilizada é enorme, questões de armazenamento precisam ser planejadas. Claramente, não é possível armazenar toda informação, e cada informação possui requisitos de persistência diferentes. A partir desta perspectiva estes dados se dividem em duas categorias [Datta96]:

- Persistentes : dados de utilização em longos períodos, que precisam ser mantidos no BDA permanentemente. Ex. Alarmes de rede e violações de segurança. Já que este tipo de dado é de importância crítica e requer permanência, estes necessitam de mecanismos de recuperação provido pelos BD convencionais.
- Perecível: dados de utilidade limitada, que retém sua utilidade somente até a atualização seguinte. Ex. taxas de retransmissão e outras estatísticas de desempenho dinâmicas. Não existe necessidade de medidas de recuperação, já que a informação estará desatualizada no momento da recuperação. Também registrar atualizações para

estes dados não é tão crucial como registrar atualizações dos dados persistentes, dado a sua rápida modificação. Porém, manter um histórico destes dados é necessário tanto para análise *off-line* de desempenho e falha, quanto para planejamento estratégico.

Dados estruturais são em sua maioria armazenados no momento de inicialização do sistema, e mantém seus valores até quando o sistema não está em operação. Os valores destes dados mudam muito menos freqüentemente que dados estatísticos, isto é, eles são mais estáticos e duráveis. Ex. topologia da rede, configuração de um comutador, chaves de encriptação. Estes dados precisam ser recuperáveis tanto em virtude de sua importância, como por razões de eficiência e de segurança. O volume deste tipo de dado depende do nível de detalhes no qual o sistema deseja representar.

Os dados contidos no *baseline* modelam a harmonia (sintonia) da rede. Estes dados incluem informações como : fluxo máximo num tronco, número máximo de células perdidas num circuito virtual, sem que haja degradação do serviço, etc. Atualizações dos dados que compõe o *baseline* tipicamente resultam em alterações no desempenho do sistema. A modelagem (especificação) dos valores destes dados são manipuladas ou pelo operador humano ou por processos (regras do BDA) que são disparados automaticamente em virtude de atualizações em dados reativos.

- **Banco de Dados Ativo no ÁTILA**

Dentre as funções que podem ser desempenhadas pelo BDA no ÁTILA, destacam-se:

- A ativação/desativação de células OAM, que auxiliam no gerenciamento de desempenho e falha numa rede ATM, pode ser realizada a partir da execução de regras no BD Ativo, dando ao sistema de gerência a capacidade de realizar testes e monitoramento de desempenho e falha na rede sem a interferência do gerente.
- Monitoramento da rede de forma mais dinâmica através de *poolings* assíncronos. A partir das regras do BDA, podemos especificar critérios, baseados no comportamento da rede, que definam de forma dinâmica, a caracterização de limiares e a execução de *poolings*. Ou seja, a partir das regras e das informações da rede, pode-se saber quais são as áreas de maior criticidade na rede, com o objetivo de definir *poolings* mais constantes para estas.
- Através das regras do BDA pode ser especificado um *baseline* dinâmico, onde os valores são periodicamente analisados/tratados com o objetivo de fornecer ao sistema uma visão mais real da rede, propiciando assim a gerência pró-ativa.

O BDA no ÁTILA segue uma determinada seqüência de passos com o objetivo de identificar, analisar e possivelmente solucionar ou apresentar soluções a problemas que surgiram ou surgirão na rede, agindo assim de maneira pró-ativa. A seguir, é apresentado um exemplo do comportamento do BDA em relação a gerência de *FALHA*.

- Correlacionar múltiplos alarmes : Embora hajam comutadores ATM que implementam formas de reduzir o número de alarmes redundantes, podem existir alarmes redundantes gerados entre comutadores.
- Analisar alarmes e notificações : Após eliminar os alarmes redundantes, o BDA analisa os alarmes que restaram para determinar a sistemática ou causa fundamental para o problema detectado.
- Cadastrar alarmes : O BDA cadastra alarmes reportados por comutadores individuais, para análise futura.
- Iniciar e executar testes : Dentre as capacidades desempenhadas o BDA seleciona, executa e termina rotinas de diagnósticos tanto em comutadores individuais como num

conjunto de comutadores.

- Analisar os resultados dos testes e reportar as informações : O BDA analisa todos os resultados dos testes dos vários comutadores ATM, para seccionalizar o problema e determinar a causa da falha.

Algumas funções que podem ser desempenhadas automaticamente pela nova arquitetura são mostradas a seguir:

- Inserir, remover, interconectar e desconectar circuitos;
- Requisição e recepção de status dos VPLs e VCLs;
- Configuração automática do NE e notificação de alterações;
- Configuração de UNIs, B-ICIs e B-ISSIs;
- Configuração de VPL e VCL;
- Configuração de conexões ATM ponto-a-ponto ou ponto-a-multiponto;
- Ativação e desativação de funções nos comutadores.

Percebe-se que o BDA possibilita uma abordagem inteligente e pró-ativa para a filtragem e correlação de eventos, e a especificação de processos que regulam o desempenho de uma rede heterogênea. Maiores detalhes sobre a estrutura e o funcionamento de um banco de dados ativo podem ser encontrados no anexo B.

7.4 EIA (Elemento Inteligência do Agente)

Atualmente, vem se acentuando a utilização de técnicas de Inteligência Artificial em aplicações de gerenciamento [Rocha96]: sejam Sistemas Baseado em Conhecimento (SBC), sejam Sistemas Especialistas [Artola96], etc. A técnica escolhida para a implementação do elemento Inteligência do Agente foi a de Agentes Inteligentes (AIs). Esta técnica difere das outras técnicas "tradicionais" inteligentes, em especial, a de Sistemas Especialistas, dada as seguintes características adicionais: *modularidade dos agentes, distribuição do conhecimento* e *adaptação às mudanças exigidas pelo sistema*.

Na tarefa de melhor gerenciar os recursos disponíveis no ÁTILA, utilizou-se a abordagem de agentes inteligentes pelo fato dos agentes terem um comportamento autônomo. Isto facilita a resolução de

problemas locais, e possibilita o comportamento pró-ativo do ÁTILA, devido os agentes terem uma base de conhecimento [Gaiti93][Sichman92][Demazeau90]. O anexo C fornece maiores informações sobre a estrutura e funcionamento dos agentes inteligentes.

7.5 ER (Elemento de Rede)

As funções de gerência desempenhadas pelo elemento de rede, restringem a escolha desta solução em dois ambientes : SNMP e CMIS/CMIP. Será utilizado o SNMP por ser um padrão de mercado, ou seja, a maioria dos equipamentos atualmente suporta este protocolo. Maiores detalhes sobre SNMP são encontrados na seção 2.2.

7.6 Plataforma de Distribuição

O CORBA apresentou-se como a solução mais adequada para a implementação dos mecanismos de distribuição do ÁTILA. Dentre os motivos merecem destaque: suporte à comunicação transparente entre objetos distribuídos e utilização de um modelo de interface único, descrito pela linguagem IDL. Para disponibilizar as operações desses objetos, CORBA fornece mecanismos para a distribuição em sistemas heterogêneos indispensáveis ao ambiente do protótipo. Um outro aspecto considerado na opção pelo CORBA, diz respeito à disponibilidade de um ambiente para o desenvolvimento rápido e fácil de aplicações complexas, o ÁBACO [Souza96]. Maiores informações sobre a plataforma de distribuição CORBA são apresentadas no anexo D.

Capítulo 08 – Prototipagem

8.1 Introdução

Este capítulo apresenta as ferramentas utilizadas no protótipo. Em seguida, o protótipo é descrito através da modelagem e especificação das interfaces de seus componentes.

As ferramentas utilizadas para implementar as tecnologias escolhidas no protótipo são:

- A linguagem definida para a implementação da interface foi JAVA [JAVA];
- O Banco de Dados Ativo é o ODE (*Object Database Environment*) [ODE4.0] desenvolvido pela AT&T Bell-Labs;
- A implementação da plataforma de distribuição CORBA utilizada foi o ÁBACO [Souza96];
- Para a implementação dos agentes inteligentes foi escolhido o ambiente JATLite [JATLite];
- O acesso aos agentes SNMP é feito através da API-SNMP/JAVA Advent, da AdventNet [Advent].

8.2 Ferramentas

8.2.1 Linguagem Java - WWW (World Wide Web)

A escolha da tecnologia WWW para implementar o elemento acesso ao usuário (EAU), levou a decisão entre HTML/CGI e *"Applets Java"*. Embora HTML/CGI não seja uma ferramenta, ela foi introduzida nesta seção devido ao costume desta tecnologia ser comparada com ambientes JAVA.

Abaixo, são citadas as características de cada uma destas "ferramentas" e implicitamente é definida a escolha por "*Applets Java*".

CGI provê um método para gerar páginas WEB dinamicamente. Entretanto, a carga de processamento fica toda restrita ao lado do servidor WWW. Já que o cliente não pode controlar a execução dos binários CGI, seu escopo é usualmente limitado a transações simples como consultas a banco de dados.

Em 1995, a *"Sun Microsystems"* introduziu uma linguagem de programação chamada JAVA. JAVA é independente de arquitetura, ou seja, pode ser interpretado em qualquer máquina que possua um sistema em tempo de execução JAVA (virtual machine). Isto permite a implementação de código móvel como parte de código HTML (Hyper Text Markup Language) (*Applets Java*). Web browsers que possuem uma máquina virtual podem fazer um *"download"* e processar *"Applets Java"*. Uma vez feito o *"download"*, "*applets*" são executadas e controladas no lado do cliente sem mais comunicação com o servidor.

Outro fator importante para a escolha de JAVA, deve-se ao fato de outras ferramentas utilizadas no protótipo, no caso API SNMP da Advent e JATLite também se utilizarem de JAVA para implementar suas funcionalidades.

8.2.2 ODE (Object Database Environment) - Banco de Dados Ativo

A ferramenta escolhida para a implementação do banco de dados ativo no protótipo foi o ODE [Lieuwen96] [Agrawal89]. Dentre os motivos que levaram a sua escolha temos: a ferramenta é *freeware*, é baseada no paradigma de orientação a objeto, amplamente utilizada em entidades de pesquisa do mundo inteiro. A experiência já obtida com esta ferramenta em trabalhos anteriores [Vasconcelos97], é outra motivação.

ODE é um sistema de banco de dados baseado no paradigma de objeto. Ele utiliza a linguagem O++ [Agrawal93], que é uma extensão da linguagem C++, para definir, consultar e manipular seus objetos. O++ provê facilidades para se criar objetos persistentes, visualizando assim a memória em duas partes: volátil e persistente. ODE possui facilidade para associarmos restrições e *triggers* a objetos. Restrições e *triggers* são associados as definições das classes, o que faz com que elas sejam fáceis de ler, implementar e de se utilizar da funcionalidade de herança disponibilizada na linguagem O++.

As funcionalidades *triggers* e restrições (*Constraints*) caracterizam ODE como um banco de dados ativo. Através destas *triggers* podem-se definir facilidades para os bancos de dados orientados a objeto especificarem restrições complexas e de alto nível. ODE suporta três tipos de *triggers*: *"once-only"*, *"perpetual"* e *"timed-triggers"*. Tanto as *triggers* como as restrições são associadas a objetos, mas podem ser parametrizadas e ter múltiplas chamadas ativas ao mesmo tempo.

ODE utiliza um modelo de regras do tipo E-A [Gehani91], que é mais simples do que o modelo E-C-A. Este modelo além de eliminar a necessidade de uma parte C (condição) separada – a condição é parte de um evento neste modelo – elimina também a necessidade de se ter tipos especiais de acoplamento como os propostos no modelo E-C-A.

Neste banco de dados pode-se especificar eventos compostos como expressões de eventos, usando uma linguagem de composição de eventos [Jagadish92]. Esta linguagem é equivalente, em termos de expressividade, a expressões regulares sobre "strings" ou eventos lógicos. Esta ferramenta compila especificações de eventos compostos arbitrários em autômatos finitos, tornando a detecção destes eventos eficiente.

8.2.3 JATLite - Agentes Inteligentes

O JATLite [JATLite] é um ambiente que fornece facilidades na construção de sistemas multi-agente usando um conjunto de APIs Java. Ele está estruturado em quatro camadas: *Abstract Layer, Base Layer, KQML Layer, Router Layer*. A partir destas camadas é possível implementar agentes com vários níveis de abstração.

A partir da camada *Router*, é permitida a comunicação entre os agentes, e são disponibilizadas aos agentes o conhecimento das características dos demais. A partir da camada KQML, o JATLite provê toda a funcionalidade para que agentes possam se comunicar em alto nível.

Baseada em TCP/IP, esta ferramenta apresenta um Múltiplo Servidor de *Socket* e um

Múltiplo Receptor de Mensagens. Para o desenvolvimento do ambiente baseado em agentes foi escolhida a camada de Base (*Base Layer*). Ela disponibiliza um roteador de mensagem, e a comunicação dá-se através de interface browser.

O JATLite também apresenta Conexão Persistente sem *timeout*, e segurança a nível de conexão com nome do agente e senha. Estas características permitem a implementação apresentada na seção 8.3.3, no que se refere ao modelo de objetos ativos.

Dentre os fatores que nos levaram a utilizar a ferramenta JATLite, temos : a sua disponibilidade ("*freeware*"), experiência com sua utilização em trabalhos anteriores e devido ao fato dela se basear na linguagem JAVA.

8.2.4 Advent API SNMP/JAVA - SNMP

Advent [Advent] é um pacote que contém um conjunto de classes JAVA. Ele suporta a versão 1 do SNMP e inclui um analisador que permite carregar múltiplos módulos da MIB em qualquer tempo, tanto para aplicações quanto para *applets*. O Advent foi planejado, principalmente, para o desenvolvimento de gerentes SNMP e aplicações de gerenciamento. O pacote está designado a habilitar o desenvolvimento de *applets* JAVA orientado a objetos, e aplicações JAVA que usam SNMP para acessar os nós gerenciados. No caso de *applets*, suporte especial é provido para garantir as restrições de segurança.

Advent provê um programa JAVA para o servidor WWW chamado "SNMP APPLET SERVER" (SAS), que permite o *applet* enviar e receber pacotes SNMP para e de qualquer dispositivo gerenciado acessível a partir de um *applet host.*

O pacote é composto por 4 categorias de classes: variáveis SNMP, comunicação SNMP, MIB SNMP relacionadas e classes mistas.

A seguir temos uma descrição de cada categoria:

Classes de variáveis SNMP

O antecessor de todas as classes de variáveis SNMP é uma classe abstrata chamada SnmpVar. Esta classe contém métodos abstratos para imprimir, codificar/ decodificar ASN.1, etc. Algumas das classes que compõem este conjunto têm as seguintes funções: representar variáveis inteiras SNMP, variáveis SNMP de valor nulo, variáveis de endereço IP e usar vetor de octetos SNMP. A classe SnmpOID tem construtores e métodos para ajudar na interface com classes relacionadas com a MIB. Nem todas as classes são públicas, isto é, alguns métodos são usados somente por classes do pacote.

Classes de comunicação SNMP

O pacote usa a classe SnmpAPI para administrar sessões criadas pelo usuário da aplicação, módulos da MIB que estão carregados, e armazenar alguns parâmetros chaves para a comunidade SNMP. Uma aplicação SNMP (gerente ou agente) freqüentemente precisa administrar múltiplas sessões por causa das interações com múltiplos pares SNMP.

A variável *binding* é uma combinação de um Identificador de Objeto e uma variável SNMP, que são comumente usadas em interações gerente-agente. A classe SnmpPDU irá ser usada para prover as variáveis e os métodos para criar e usar o SNMP PDU. Os métodos incluem adição de valores nulos às variáveis *bindings* e impressão de todas as variáveis *bindings*.

Classes de MIB SNMP relacionadas

Os usuários obtém informações sobre a estrutura e formato dos dados disponíveis pelos agentes, através dos módulos da MIB que fornecem informações aos agentes SNMP. Os módulos da MIB são usualmente especificados em um arquivo, que precisa ser analisado para se entender a sintaxe e a estrutura dos dados disponíveis no agente. A classe MibModule provê um modo de analisar e usar as variáveis no arquivo. Cada instância de um módulo da MIB é criada de um arquivo, e pode-se carregar e descarregar módulos da MIB pela criação e liberação dessas instâncias. A instância contém todos os nós da árvore da MIB, tão bem quanto define *traps* e convenções textuais.

Classes mistas

Existem três classes que não estão incluídas nas categorias acima, são elas: SnmpClient; MibException; SnmpException. Estas classes estão relacionadas com a mudança de procedimentos default, decodificação de erros, etc.

8.2.5 ÁBACO - CORBA

O ÁBACO, é um ambiente com suporte gráfico que serve de apoio ao desenvolvimento de aplicações complexas no CORBA. Esse ambiente fornece uma interface amigável na qual aplicações distribuídas podem ser construídas rapidamente e sem a necessidade de nenhum conhecimento do CORBA por parte do desenvolvedor. Toda e qualquer funcionalidade relativa a comunicação é deixada totalmente de lado, cabendo ao desenvolvedor a tarefa de implementar somente a lógica da aplicação. Afora os motivos citados acima, a experiência adquirida com a utilização deste ambiente, foram as principais razões da sua escolha na implementação do protótipo.

Duas linguagens de programação foram desenvolvidas para a implementação do ÁBACO: a CDL (*Component Description Language*), que tem como objetivo principal criar objetos distribuídos configuráveis, denominados componentes no ÁBACO, a partir de objetos distribuídos ordinários, e a linguagem CCL (*Component Configuration Language*), utilizada na descrição da configuração de aplicações complexas a partir de objetos configuráveis programados em CDL.

8.2.5.1 Arquitetura Lógica

Modelo de Componentes

No ÁBACO, os objetos tem uma estrutura na qual serão contempladas tanto as características dos objetos distribuídos, quanto as características dos objetos do paradigma de configuração. A utilização desta estrutura híbrida, provém da necessidade desses objetos serem reconhecidos pelas plataformas de origem. Desta forma estes objetos continuam utilizando todos os mecanismos dessas plataformas, além de passarem a utilizar também as vantagens do paradigma de configuração.

A figura 8.1 ilustra um objeto do ambiente ÁBACO, denominado **componente.**

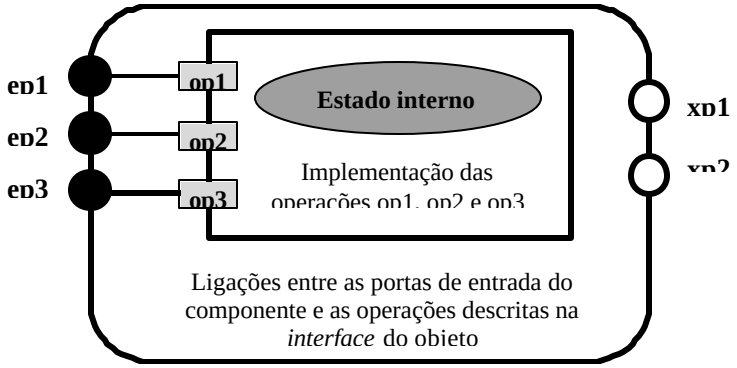


Figura 8.1 : Estrutura do Componente do Ambiente ÁBACO

Um componente do ambiente ÁBACO tem a seguinte estrutura:

Estrutura interna do componente :

- uma *interface* com a descrição das operações implementadas pelo objeto.

Estrutura externa do componente :

- um conjunto de portas representando as operações *requeridas* pelo componente (portas de saída);
- um conjunto de portas representando as operações *oferecidas* pelo componente (portas de entrada) e
- um conjunto de ligações entre as portas de entrada do componente e as operações descritas na *interface* do objeto.

8.3 Descrição do Protótipo

Nesta seção são apresentadas as especificações do ambiente do protótipo. Primeiramente, é detalhada a interface gráfica, onde são descritas as telas e as funções disponibilizadas por estas. Também são descritos os objetos relativos ao banco de dados ativo (BDA) e as respectivas interfaces IDL responsáveis pelo acesso e manipulação das informações de gerência que ele disponibiliza. A seguir, é descrito a especificação de um agente genérico, cuja estrutura é herdada por todos os agentes implementados no protótipo. Para finalizar, é definida a estrutura do agente SNMP e a suas interfaces IDL e o funcionamento do ambiente ÁBACO.

8.3.1 Elemento Acesso ao Usuário

A interface gráfica do protótipo do ÁTILA, apresentada na figura 8.2, foi implementada em JAVA. Esta interface se utiliza de um ambiente gráfico, onde são definidos mapas correspondentes à rede a ser gerenciada. Esses mapas são especificados através da edição de objetos que representam os elementos da rede física, como: *hosts*, comutadores, roteadores, etc.)

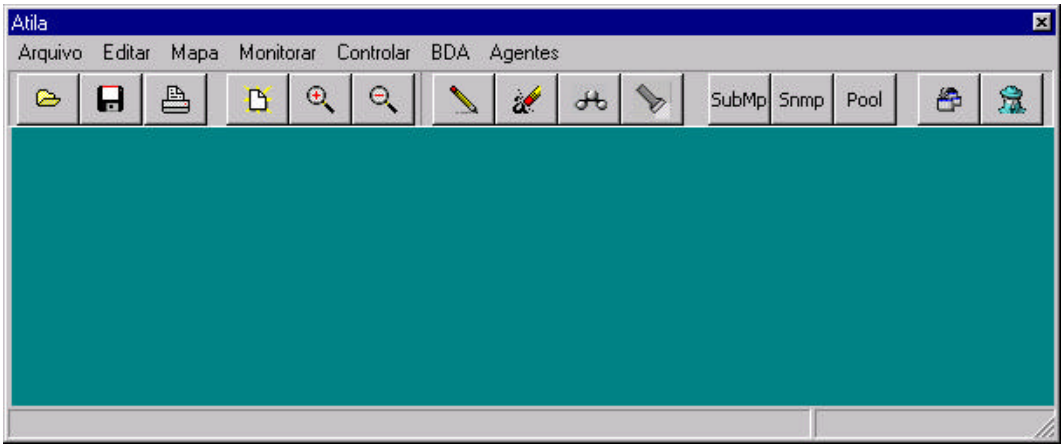


Figura 8.2 : Interface Gráfica do Protótipo

A interface do protótipo (Figura 8.2), está dividida em dois componentes básicos:

- Janela de Mapa da Rede;
- Barra de Atalho.

Janela de Mapa da Rede

A Janela de Mapa da Rede, é o local onde o mapa da rede a ser gerenciada é apresentado. Nessa janela, as topologias das redes são desenhadas. Os elementos desses mapas, são utilizados como abstrações dos elementos físicos reais. Ações nesses elementos, como a instanciação de um agente inteligente, são executadas na rede real de forma transparente. Eventos (alarmes e notificações) e migração de agentes, são mostrados nessa janela, em tempo de execução.

Barra de Atalho

Algumas das funções mais utilizadas na construção de mapas e na elaboração de atividades de gerência, estão disponíveis em uma barra de atalho no topo da Janela de Mapa da Rede. Funções como: criação de submapas, criação de objetos gerenciáveis e instanciação de agentes, estão disponíveis nessa barra, permitindo uma maior agilidade no processo de gerência da rede.

Dentre as funções gerais desta interface, pode-se citar :

- Ativação/desativação/monitoramento do estado e configuração dos Agentes Inteligentes associados a cada objeto (elemento de rede);
- Ativação/desativação/monitoramento das regras associadas ao Banco de Dados Ativo (BDA) e Agentes Inteligentes;
- Monitoramento e controle a partir das funções disponibilizadas pelo BDA e pelos Agentes Inteligentes em cada objeto.

As seções a seguir detalham a criação de uma mapa no protótipo e apresentam as funções disponíveis na sua interface.

8.3.1.1 Objetos de um MAPA

Para realizar a criação de um mapa (figura 8.2) vários objetos são utilizados. Abaixo, esses objetos são descritos.

- **Obj_SubMp**: representa um nível abaixo na estrutura da rede. Objetos deste tipo revelam segmentos adicionais no mapa.
- **Obj_Snmp**: representa um elemento gerenciável da rede, ou seja, que contém agentes SNMP. Cada objeto deste tipo deve possuir um nome único, que é mostrado no mapa, e um endereço de rede, que deve ser um endereço IP válido.
- **Obj_Pooling**: representa um dispositivo de rede que suporta IP, mas não suporta agente SNMP. O protótipo pode gerar "*pooling*" para estes objetos. O endereço IP também é requerido.

Os mapas especificados no protótipo poderão conter os três tipos de objetos citados acima. Estes mapas contém geralmente dois níveis hierárquicos, isto é, deverão conter submapas com o objetivo de modularizar o ambiente de gerenciamento. Abaixo são descritos os passos para a definição destas hierarquias.

No mapa de primeiro nível, denominado *mapa principal*, existem vários objetos do tipo "Obj_SubMp". O(s) mapa(s) de nível 2, podem conter objetos do tipo "Obj_Snmp" e "Obj_Pooling" que representam os equipamentos da rede definida no primeiro nível, e objetos do tipo "Obj_SubMp", caso se queira especificar outro nível hierárquico no mapa principal.

8.3.1.2 Construção de um MAPA

Na criação de mapas, deve-se primeiro criar o *mapa principal* (com objetos do tipo "Obj_SubMp") e logo após criar os submapas (com objetos do tipo "Obj_Snmp", "Obj_Pooling" e "Obj_SubMp").

- **Mapa Principal**

Para criar o *mapa principal* da rede, selecione no menu posterior da tela a opção **Arquivo/Novo Mapa**. Neste momento uma tela onde deve-se definir os objetos que compõe este mapa aparece (Figura 8.2).

Criação dos Objetos "Obj_SubMp"

1. Nesta seção são definidos todos os objetos do tipo "Obj_SubMp" contidos no *mapa principal.*

2. Para a criação de um objeto do tipo "Obj_SubMp" não deve ser selecionado nenhum objeto no mapa PRINCIPAL e no menu posterior da tela a opção **Editar/Objeto/ "Obj_SubMp"** deve ser selecionada. Após isto, aparecerá a tela abaixo (figura 8.3), onde devem ser preenchidos algumas informações a respeito do referido objeto.

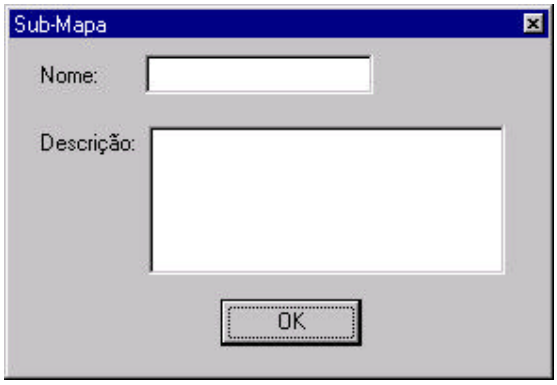


Figura 8.3 : Interface de Definição de um Objeto "Obj_SubMp"

3. Na opção *Nome* deve ser entrado o nome do submapa. Ou seja, um nome genérico que corresponde ao submapa, por exemplo, nome da cidade, de um bairro, de um laboratório, etc.

4. No campo *Descrição* deve ser entrada uma pequena descrição do objeto.

5. Após isto, deve-se pressionar o botão <OK> para adicionar este objeto ao mapa.

6. Os passos 1 a 5 devem ser repetidos para se incluir outros objetos do tipo "Obj_SubMp" ao *mapa principal.*

- **Submapas**

Nesta seção são definidos os submapas relativos aos objetos "Obj_SubMp" referenciados no *mapa principal.* Para criar um submapa deve-se selecionar o objeto "Obj_SubMp" do *mapa principal* relativo ao submapa que se quer definir, e acessá-lo. Após isto, aparecerá um mapa em branco onde os objetos (equipamentos) referentes ao objeto "Obj_SubMp" selecionado, serão configurados.

Criação dos Objetos "Obj_Snmp"

1. Nesta seção são definidos todos os objetos do tipo "Obj_Snmp" (equipamentos) contidos neste submapa. Objetos do tipo "Obj_Snmp" , como dito acima, representam os objetos gerenciáveis da rede, ou seja, equipamentos que suportam SNMP. Em virtude disto, antes de configurar um equipamento como sendo um objeto do tipo "Obj_Snmp" deve-se ter certeza de que o mesmo suporta SNMP;

2. Para a criação de um objeto do tipo "Obj_Snmp" não deve ser selecionado nenhum objeto neste submapa e deve-se selecionar no menu posterior da tela a opção **Editar/Objeto/"Obj_Snmp"** . Após isto, aparecerá a tela a seguir (figura 8.4), onde são preenchidas algumas informações a respeito do referido objeto (equipamento);

3. Na opção *Nome* deve-se entrar com o nome que identificará o objeto no submapa;

4. O campo *Endereço* deve conter o endereço IP do equipamento;

5. No campo *Descrição* deve-se entrar com uma pequena descrição do objeto (equipamento);

6. Para configurar as interfaces deste objeto (equipamento), o botão <Portas> deve ser selecionado, abrindo a tela abaixo (figura 8.5);

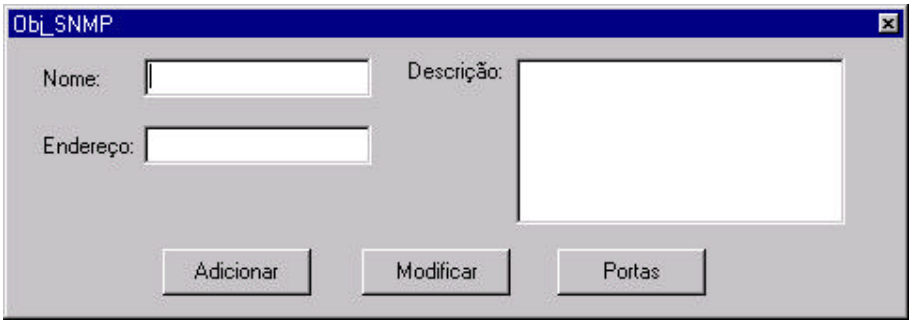


Figura 8.4 : Interface de Definição de um Objeto “Obj_Snmp”

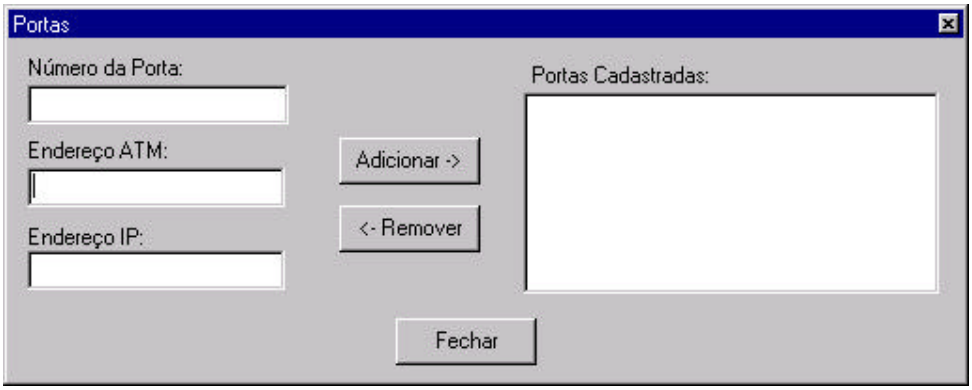


Figura 8.5 : Interface de Definição das Interfaces de um Objeto “Obj_Snmp”

7. Nesta tela deve-se preencher de forma seqüencial o campo *Número da Porta* que identifica cada interface física ;

8. O campo *Endereço ATM* deve conter o endereço ATM (se existente);

9. No campo *Endereço IP* deve conter o endereço IP (se existente);

10. Após isto, o botão <Adicionar> deve ser pressionado para adicionar a interface ao objeto (equipamento).

11. Os passos 7 a 10 devem ser repetidos para incluir todas as interfaces associadas ao respectivo objeto (equipamento).

12. O botão <OK> deve ser pressionado para finalizar a inclusão destas interfaces.

13. Os passos acima de 2 a 12 devem ser repetidos para se criar todos os objetos (equipamentos) do tipo “Obj_Snmp” do respectivo submapa.

Criação dos Objetos “Obj_Pooling”

A criação dos objetos do tipo “Obj_Pooling” é equivalente a configuração dos objetos “Obj_Snmp”. O que diferencia um objeto do outro é o tipo de informação que se pode acessar e manipular a partir destes elementos.

8.3.1.3 Utilização da Interface

Com o mapa da rede criado, o gerenciamento pode ser, de fato, iniciado. Para se utilizar a interface do protótipo no monitoramento e controle da rede, tanto os agentes como o BDA devem estar definidos em algum ponto da rede.

É através deste mapa, que os agentes são ativados para realizarem suas funções de monitoramento e controle. Para a instanciação de um agente em um determinado equipamento da rede, basta selecionar o objeto que corresponda a esse equipamento no mapa, e selecionar no menu a opção **Agentes/Instanciar**. É valido lembrar que só faz sentido instanciar agentes inteligentes em objetos do tipo “Obj_Snmp”. Nesse momento uma tela (figura 8.6) contendo uma lista de todos os agentes disponíveis é mostrada. Basta escolher o agente que se deseja instanciar e pressionar o botão <OK>. O agente escolhido será automaticamente ativado no elemento de rede físico correspondente ao objeto selecionado no mapa.

Os agentes podem ser monitorados, de modo a se observar seu *status* atual de funcionamento (ativo/inativo). Esse *status* também pode ser alterado através das opções **Agentes/Status** . Novos agentes podem ser introduzidos no protótipo através do menu na opção **Agentes/Inserir**.

O BDA, previamente instalado e configurado em uma determinada estação da rede, também pode ser manipulado através da interface. Operações relativas às Regras, aos dados e ao *Baseline*, podem ser facilmente acessadas através do menus nas opções

BDA/Regras , **BDA/Dados** e **BDA/*Baseline***, respectivamente.

Figura 8.6 : Interface de Instanciação dos Agentes Inteligentes

Em cada submenu do BDA (Regras, Dados e *Baseline*), novos menus podem ser acessados, de forma a se controlar as informações relativas a cada item. A tabela 8.1 abaixo, mostra toda a hierarquia a partir do item BDA do menu, para a manipulação do banco de dados ativo.

Menu	**Submenu**	**Descrição**
Regras	Deletar	Apaga uma regra do BDA
	Criar	Cria uma nova regra no BDA
	Atualizar	Atualiza uma regra do BDA
	Ativar	Ativa uma regra do BDA
	Desativar	Desativa uma regra do BDA
Dados	Insere	Insere dados do BDA
	Consultar	Consulta dados do BDA
	Atualiza	Atualiza dados do BDA
Baseline	**Consultar**	Consulta o *"Baseline"*
	Alterar	Altera o *"Baseline"*
	Criar	Cria limite para algum atributo

Tabela 8.1 : Hierarquia de "menus" do Banco de Dados Ativo

8.3.2 Elemento Inteligência do Gerente

- **Interfaces de Acesso/Manipulação**

Com a intenção de possibilitar o acesso e a manipulação das informações do banco de dados, um conjunto de classes foram definidas para implementar as operações necessárias para a realização destas tarefas.

Classe *Banco_de_Dados*: Abrir, Fechar e Renomear;
Classe *Regra* : Criar, Apagar, Habilitar, Desabilitar e Consultar;
Classe *Dado* : Consultar (Simples (Objeto), Cruzar(Join)), Incluir_Objeto, Atualizar_Objeto e Apagar_Objeto.

Para tornar disponíveis essas operações de forma distribuída, interfaces em IDL foram definidas. Através dessas interfaces, qualquer elemento da arquitetura pode interagir com o BDA para manipular as informações do banco de dados. Para facilitar as especificações, um módulo chamado BDA foi criado, e dentro desse módulo, todas as interfaces de manipulação do banco de dados foram especificadas. Abaixo são definidas as interfaces IDL relativas às classes *Banco_de_Dados* e *Regra*. As IDLs relativas à classe *Dado*, são definidas levando-se em conta a estrutura do banco de dados, ou seja que informações ele armazena. Em virtude disto, as interfaces para esta classe só são definidas quando da definição desta estrutura, ou seja, serão implementadas nos estudos de caso.

```
module BDA {

// Definição da interface da classe Banco de Dados
interface Banco_de_Dados {
        boolean Abrir (in string Nome_BD, in long Pemissao);
        boolean Fechar (in string Nome_BD);
        boolean Renomear (in string Old_Nome_BD, in string New_NomeBD);
};

// Definição da interface da classe Regra
```

```
interface Regra{
        typedef string ListaRegras[100];
        boolean Criar (in string Classe, in string Nome_Regra, in string Regra);
        boolean Ativar (in string Classe, in string Nome_Regra, in long Duracao);
       boolean Desativar (in string Classe, in string Nome_Regra, in long Duracao);
        string Consultar (in string Classe, in string Nome_Regra);
       ListaRegras Listar (in string Classe);
       };
}
```

O banco de dados também modela o conjunto de limiares ("*thresholds*") correspondentes a cada atributo, os quais formam o "*baseline*", a ser definido abaixo.

- **Estrutura da Base de Informações de Gerência**

A bases de dados contida no Gerente de uma aplicação de gerenciamento, deve possuir informações gerais da rede. Esta base de dados geralmente é gerada a partir de informações obtidas de um conjunto de MIBs disponibilizadas pelos recursos a serem gerenciados. Estas MIBs são integradas de maneira a disponibilizar uma visão ampla do ambiente gerenciado.

Para realizar esta integração é necessário uma metodologia que realize o tratamento de informações redundantes, de tipos de dados distintos que possuem o mesmo significado, realizar o mapeamento entre atributos da base de dados e informações contidas nas MIBs, etc.

No protótipo apresentado, a modelagem do banco de dados foi realizada com base em uma única MIB, a MIB ILMI [ILMI4.0]. A modelagem da base de dados do gerente, baseada em uma MIB única, simplifica a definição da base de informações, liberando o administrador dos detalhes das informações contidas nas MIBs, e se concentrando na estrutura do banco de dados.

Abaixo é apresentado um modelo para a geração da base de dados do gerente a partir de uma MIB-SNMP específica.

- **Modelo para a Geração da Base de Dados do Gerente a partir de uma MIB-SNMP**

O protótipo proposto se utiliza de um banco de dados baseado no paradigma de orientação a objeto, o ODE. Em função disto, temos que realizar a tradução de uma MIB-SNMP especificada em ASN.1, para um modelo orientado a objeto.

Para realizar esta tradução foram utilizados os passos abaixo. Posteriormente, estes passos são descritos em detalhes.

1. MIB-SNMP em ASN.1 → Modelo Entidade/Relacionamento
2. Modelo Entidade/Relacionamento → Modelo Orientado a Objeto
3. Modelo Orientado a Objeto → Base de Informações de Gerência

Uma MIB-SNMP é constituída de variáveis e tabelas. No caso das tabelas o mapeamento é direto para o modelo entidade-relacionamento. Cada tabela da MIB torna-se uma entidade neste modelo e as variáveis definidas nas tabelas tornam-se atributos de uma entidade específica. No caso das variáveis, estas devem formar uma nova entidade dependendo do grupo a qual pertencem. Ou seja, todas as variáveis de um determinado grupo se transformam em uma entidade com o nome do grupo, onde estas variáveis representam os atributos. Os relacionamentos foram identificados a partir dos atributos contidos em uma entidade que referenciam uma outra. (Ex. Na tabela "*atmVplTable*" temos uma variável de nome "*atmVplReceiveTrafficDescrIndex*", que referencia a tabela de descritores de tráfego "*atmTrafficDescrParamTable*".). O mapeamento do modelo descrito no modelo entidade-relacionamento para o modelo orientado a objeto foi realizado como definido em [Navathe94].

A MIB modelada na metodologia orientada a objeto ainda não contém todas as informações necessárias para se tornar uma base de dados de gerenciamento. Esta MIB se restringe a armazenar informações de um único elemento de rede. Alterações devem ser realizadas com o objetivo de se obter uma maneira de armazenar informações de diversos elementos de rede. A seguir são descritas em detalhes as alterações na MIB

modelada no modelo orientado a objeto, com o objetivo de fornecer suporte a vários elementos de rede e à outras funções.

- Uma classe, denominada ER, de nível mais alto deve ser criada para que as informações a serem armazenadas, não se limitem a um só elemento de rede, como em uma MIB ASN.1, mas sejam relativas a vários elementos de rede;
- Modelar na base de dados de gerência, informações sobre os limiares dos atributos;
- Métodos e Regras associadas as novas classes.

Classe ER

A necessidade de se ter informações de diversos elementos de rede armazenadas na base de dados, fez surgir a necessidade de criação de uma classe mais ampla que armazenasse estas informações e se relacionasse com as outras classes já definidas na MIB e que passam a conter informações de vários elementos de rede. Para isto, foi utilizada a definição da estrutura do grupo *"System"* especificado na MIB II [MIB-II], como base para a especificação desta nova classe, denominada ER. Isto se deu, devido a abrangência que este grupo fornece sobre um elemento de rede. Algumas variáveis existentes neste grupo foram tratadas e outras criadas para que fosse obtida uma visão mais geral da rede. Abaixo segue a estrutura dessa nova classe.

```
#ifndef ER_h
#define ER_h

#include <displaystring.h>
#include <timeticks.h>
#include <IpAdress.h>
#include <BD_Integer32>

persistent class ER {
public:

        // Identificador do elemento de rede, dentre os registrados na base de dados.
        indexable BD_Integer32 ER_ObjectID;

        // Descrição : Endereço de IP de acesso ao elemento de rede;
        indexable IpAdress ER_Address;

        // Descrição do elemento de rede. Contém especificações de hardware/software
```

```
        DisplayString ER_Descr;

        // Tempo desde a última atualização/(liga/desliga) do elemento de rede.
        TimeTicks ER_UpTime;

        // Identificação da pessoa de contato responsável pelo elemento de rede.
        DisplayString ER_Contact;

        // Nome administrativo do elemento de rede.
        DisplayString ER_Name;

        // Localização física do elemento de rede.
        DisplayString ER_Location;

        ER();

       ~ER();

        ER(BD_Integer32 ObjectID, IpAdress Address, DisplayString Descr, TimeTicks
        UpTime, DisplayString Contact, DisplayString Name, DisplayString Location);

};

#endif
```

"Baseline"

Como citado acima, o Banco de Dados Ativo do protótipo, foi modelado com o objetivo de armazenar tanto informações da rede quanto informações sobre os "*thresholds*" de algumas variáveis interessantes.

A modelagem do "*baseline*" foi implementada da seguinte forma : cada tipo primitivo de dado encontrado numa MIB (Integer, Gauge, OctetString, ...) foi transformado dentro do banco de dados ativo num tipo abstrato de dados (*struct*). Este tipo contém na realidade dois atributos do mesmo tipo primitivo, um destes armazena o valor da variável e outro o seu "*threshold*". Esta modelagem foi realizada somente em tipos primitivos numéricos, já que esta informação não se aplica a atributos de outros tipos. Os nomes dos atributos do tipo abstrato são : valor e limite. Ou seja, se eu tenho uma variável em uma determinada MIB com nome *NumCellLost* e o seu tipo primitivo é Integer32, no banco de dados ativo deve haver um atributo de mesmo nome com tipo BD_Integer32, por exemplo, que é um tipo abstrato de dados. Quando necessito acessar

o valor do atributo uso *NumCellLost.Valor*, quando quero o limite o referencio como *NumCellLost.Limite*.

Os tipos de dados descritos abaixo estão presentes nas MIBs modeladas no anexo A. Como citado acima, os tipos numéricos devem ser transformados dentro do BDA para tipos abstratos que contenham dois atributos : Valor e Limite. Assim temos :

```
// O tipo "INTEGER" foi alterado para tipo BD_INTEGER
typedef struct BD_INTEGER
{       INTEGER Valor;
        INTEGER Limite;
};

// O tipo "Integer32" foi alterado para tipo BD_Integer32.
typedef struct BD_Integer32
{       Integer32 Valor;
        Integer32 Limite;
};

// O tipo "Counter32" foi alterado para tipo BD_Counter32
typedef struct BD_Counter32
{       Counter32 Valor;
        Counter32 Limite;
};

// O tipo de dado "Gauge32" foi alterado para tipo BD_Gauge32
typedef struct BD_Gauge32
{       Gauge32 Valor;
        Gauge32 Limite;
};
```

A utilização de "*thresholds*" não se aplica aos tipos "*OctetString*", "*Timestamp*" e a outros que são definidos a partir de um *Object Identifier*.

8.3.3 Elemento Inteligência do Agente

O protótipo proposto utiliza um modelo genérico de agentes inteligentes, ou seja, todos os agentes serão subclasses do agente **ag-atila**, cuja hierarquia é ilustrada na figura 8.7. Este modelo discutido nesta implementação foi proposto em [Franca97], sendo baseado nos modelos apresentados em [Sichman92]. Este modelo está organizado em uma hierarquia de agregação de classes. Estas classes estão estruturadas da seguinte forma:

- **ClasseAtilaControle :** Esta classe abstrata apresenta as regras do comportamento básico de relacionamento com outros agentes (*métodos de políticas de comportamento*); o processamento de mensagens (*métodos de política de mensagens);* o controle e a funcionalidade dos serviços disponíveis apresentados em cada agente (*métodos de política de serviço e política de decisão).*
- **ClasseAtilaInformacoes :** Esta classe abstrata define as informações locais para cada agente (modelo interno) e faz o tratamento das informações do ambiente (modelo externo). Esta classe é responsável pela base de conhecimento dos agentes, relacionada ao conhecimento local e ao conhecimento global.
- **ClasseAtilaMensagens :** Esta classe é responsável pela comunicação entre os agentes na plataforma utilizada em nosso ambiente. Esta classe também suporta a comunicação entre os agentes e os outros elementos do protótipo. Ela utiliza o mecanismo de troca de mensagens entre objetos (*caixa de mensagem*) e as definições das interfaces (*portas comunicação*). Esta classe faz a interface entre o ambiente de distribuição (ÁBACO) e os agentes inteligentes.

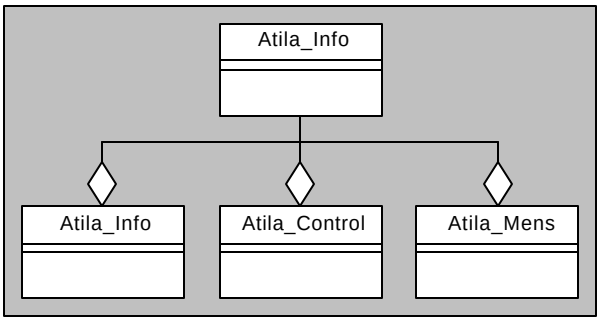


Figura 8.7 : Hierarquia do Agente *ag-atila*

O agente Átila apresenta os seguintes métodos gerais:

- **atila_control:** esta classe tem os seguintes métodos visíveis: *AtivaAgente, DesativaAgente.*
- **atila_info:** esta classe apresenta os seguintes métodos: *AdicionaRegra, RemoveRegra, RealizaConsulta, SubmeteRequisicao.*
- **atila_mens:** esta classe tem os seguinte métodos : *EnviaMensagem, RecebeMensagem.*

Especificação em JATLite:

```
import router.*;
class ag-atila extends router {
    private atila_info object_info;
    private atila_control object_ control;
    private atila_mens object_mens;
    public AtivaAgente() : Boolean ;
    public DesativaAgente() : Boolean;
    public AdicionaRegra (Regra: AtRegra);
    public RemoveRegra;
    public RealizaConsulta;
    public SubmeteRequisicao (TpR: AtRequisicao; DescricaoR : AtC AtRequisicao) :
    AtRequisicao;
    public EnviaMensagem (TipoM : AtMensagem; DescricaoM : AtCMensagem) :
    AtMensagem;
    public RecebeMensagem (TipoM : AtMensagem; DescricaoM : AtCMensagem);
}
```

8.3.4 Elemento de Rede

A interação direta junto ao elemento de rede com o objetivo de realizar monitoramento e controle, é realizada no protótipo do ÁTILA pelos agentes SNMP. Para que essas operações pudessem estar disponíveis tanto para o acesso dos agentes inteligentes como do BDA, elas foram descritas em IDL.

```
// Definição em IDL das interfaces dos agentes SNMP
interface SNMP{
    string get(in string host, in string OID);
    boolean set(in string community, in string host, in string OID, in string Val);
};
```

A implementação dessas operações foram baseadas nas funções disponíveis no Advent, para acesso e manipulação de informações de gerenciamento via SNMP.

8.3.5 Plataforma de distribuição

O ÁBACO, utilizado como ferramenta para a distribuição de objetos no protótipo, possui duas linguagens:

- a linguagem CDL (*Component Description Language*)
- a linguagem CCL (*Component Configuration Language*)

A linguagem CDL, utiliza a estrutura de objetos distribuídos e cria componentes a partir do encapsulamento desses objetos. Dessa forma, essa linguagem é utilizada para modelar os objetos a serem configurados. Isso é conseguido pela definição de portas para as operações requeridas e oferecidas pelo componente. A figura 8.8 mostra a especificação de um componente em CDL.

```
COMPONENT controle {
  USE tControle;
  ENTRYPORT   cmd,
              NA;
  EXITPORT    CB;
}
```

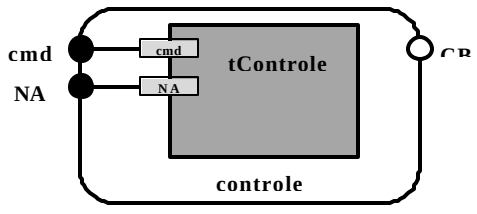


Figura 8.8 : Especificação de um Componente em CDL

A linguagem CDL, permite também a criação de componentes compostos no ÁBACO.

Assim, novos componentes podem ser criados pela união de outros componentes mais simples (figura 8.9).

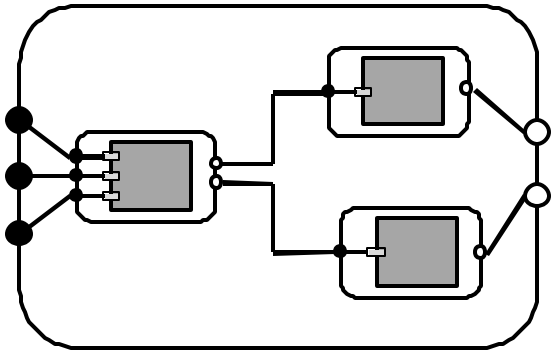


Figura 8.9 : Composição de componentes no ÁBACO

Modelo de Configuração

Com a estruturação dos objetos de aplicação na forma de componentes, o ÁBACO utiliza uma linguagem de configuração a CCL (*Component Configuration Language*) para a manipulação desses componentes. Essa linguagem fornece todas as características das linguagens de configuração, permitindo a criação de instâncias e a remoção de componentes de um sistema, a conexão desses componentes e sua alocação nos nós físicos da rede.

A linguagem CCL, por sua vez, utiliza os objetos descritos em CDL para fazer a configuração de aplicações, através da criação, remoção e alteração de instâncias desses objetos e das conexões entre eles. De uma forma geral, uma especificação de configuração CCL segue a seguinte estrutura:

- definição do nome do sistema;
- definição dos objetos que fazem parte do sistema (contexto);
- criação das instâncias a partir dos objetos;
- conexão das portas das instâncias dos objetos;

A figura 8.10 a seguir, apresenta o formato de uma especificação CCL.

Qualquer comunicação a ser realizada no ambiente ÁBACO, limita-se a invocações de

chamadas às portas locais de cada componentes, essas chamadas são realizadas através da primitiva *call*, onde são passados o nome da porta e os parâmetros da operação desejada. Essas chamadas são mapeadas em chamadas a objetos remotos de acordo com a configuração da aplicação. Esse mapeamento é realizado pela infra-estrutura de apoio à execução do ambiente ÁBACO.

```
SYSTEM  Sistema_Exemplo {
USE   Comp1, Comp2;
INSTANTIATE
      Inst1  FROM  Comp1 AT  NóX,
      Inst2  FROM  Comp2 AT  NóY;
LINK
     Inst1.port1 TO Inst2.porta1;
START   Inst1, Inst2;
}
```

Figura 8.10 : Formato de uma Especificação CCL

Modelo de Comunicação

A comunicação entre componentes deve ser realizada exclusivamente por meio de uma *interface* local, definida pelas portas destes componentes. Comunicação por meio de chamada direta a outras entidades limita a flexibilidade de configuração lógica dos sistemas. ÁBACO fornece uma primitiva de comunicação, o **call**, pela qual os componentes podem executar chamadas às suas portas locais.

A primitiva **call** é definida da seguinte forma:

CALL <nome_da_porta_local> [(<parâmetros>)] [**WAIT** <variável>]
[**FAIL** <mensagem>]

O termo <parâmetros> indica os parâmetros passados à operação ligada à porta definida em <nome_da_porta_local>. A variável de retorno, opcional, esta definida em <variável>. E a mensagem a ser dado em caso de falha da chamada, também opcional, é definida em <mensagem>.

A utilização dessa primitiva de comunicação, impõe que as chamadas dos métodos dos objetos distribuídos, realizadas pela utilização de referências aos objetos que

implementam as operações, sejam substituídas pela primitiva **call**.

Arquitetura Física

A arquitetura física do ÁBACO (figura 8.11), foi criada de modo a implementar todas as funcionalidades necessárias para suportar a criação e o controle de aplicações com objetos distribuídos configuráveis. Suportada por um ORB CORBA, essa arquitetura descreve todas as ferramentas utilizadas na criação de aplicações com o ÁBACO.

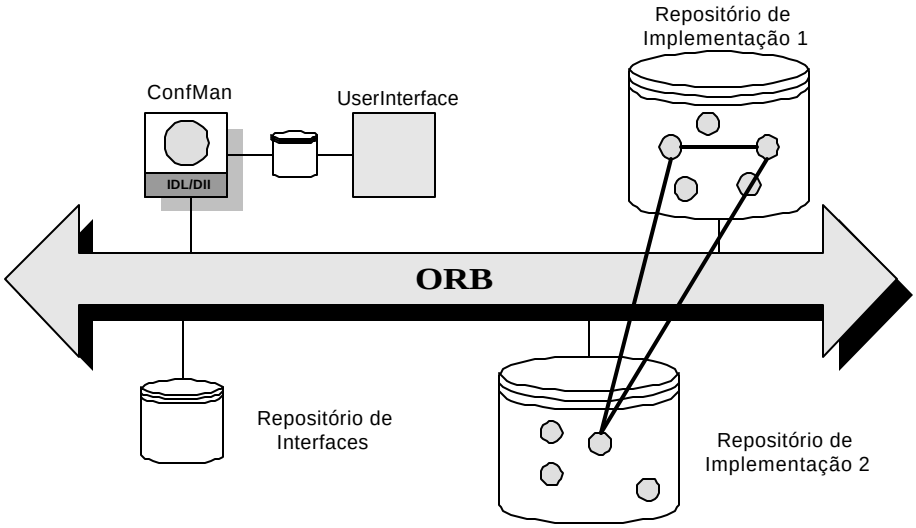


Figura 8.11 : A arquitetura física do ÁBACO

Esta arquitetura é composta pelas seguintes partes:

- ***UserInterface:*** essa é a interface visual da ferramenta. Através dela o desenvolvedor de aplicações distribuídas tem acesso a todas as funcionalidades do ÁBACO;
- ***ConfMan (Configuration Manager)*****:** o gerente de configuração tem como função principal gerenciar o mapa de configuração das aplicações. Esse módulo responde a interrogações do tipo “quem está ligado à porta X ?”.
- *Repositório de Interfaces*: para cada conjunto de portas de entrada em um componente, o ÁBACO gera automaticamente uma interface em IDL correspondente. Essas interfaces são armazenadas no repositório de interfaces do CORBA de forma a permitir a invocação das operações implementadas pelas portas através de chamadas dinâmicas;

- *Repositórios de Implementação:* esses repositórios guardam os objetos que são gerados pelo ÁBACO, que ficam disponíveis para acesso através do ORB.

Os módulos UserInterface e ConfMan foram desenvolvidos exclusivamente para o ÁBACO, esses módulos implementam as funcionalidades desejáveis para o pleno funcionamento dessa ferramenta. Os repositórios de interface e implementação fazem parte da arquitetura CORBA e são de grande importância para o ÁBACO, visto que a técnica escolhida para a implementação das rotinas de comunicação dessa ferramenta foi a de invocação dinâmica.

8.3.6 Funcionamento Geral do Protótipo

O Banco de Dados Ativo utilizado no protótipo, o ODE, contém uma base de dados estruturada, com informações de gerenciamento de uma rede ATM, implementada a partir dos passos descritos na seção 8.3.2. Além destas informações, o ODE contém regras que implementam a sua inteligência e disponibiliza ao protótipo regras "padrões". Estas podem ser referenciadas pelas aplicações de gerenciamento ou serem reutilizadas com o objetivo de se definir regras mais sofisticadas.

O ODE também disponibiliza um conjunto de operações de manipulação de banco de dados, que podem ser utilizadas tanto pelos desenvolvedores de aplicação quanto pelos agentes inteligentes (AIs). Estas operações são especificadas em IDL e estão acessíveis via ÁBACO.

No caso do desenvolvimento das aplicações, os usuários podem especificar as funções de gerência como regras E-C-A, provendo um mecanismo unificado para gerência de dados e eventos e automatização dos processos de monitoramento e controle. Em relação aos AIs, as operações permitem a interação com o ODE, objetivando a atualização/consulta das informações globais de gerenciamento.

O gerenciamento a nível de elemento de rede é realizado pelos AIs. Estes cooperam entre si e com o ODE trocando informações de gerenciamento com o objetivo de diagnosticar e solucionar problemas que possivelmente ocorram na rede. Exemplos de AIs utilizados no estudo de caso são descritos nas seções 9.3 e 9.4.

A interação entre os agentes inteligentes e o ODE tem como objetivo realizar monitoramento e controle inteligentes de forma distribuída. Através de regras especificadas no ODE, agentes inteligentes podem ser acionados para realizar as funções de gerenciamento junto aos elementos de rede. Estas funções são desempenhadas de forma inteligente através do conhecimento individual e autônomo presente em cada agente. Outra função importante é o acompanhamento contínuo do comportamento dos agentes, ou seja, realizar um monitoramento do desempenho das funções realizadas por estes.

A interação entre os agentes inteligentes é realizada através da plataforma JATLite. Esta interação é realizada em função da necessidade de haver cooperação tanto de informações como de inteligência.

Sendo a única forma de atuar junto aos recursos gerenciados, o agente SNMP continua tendo uma função passiva nesta arquitetura. O comportamento ativo a ser desempenhado localmente aos elementos de rede é realizado pelos AIs, os quais são auxiliados pelo ODE.

Os agentes SNMP disponibilizam interfaces IDL de acesso, as quais permitem o ODE e os agentes inteligentes interagirem diretamente com os elementos de rede com o intuito de realizar monitoramento e controle. Isto pode ser realizado sem a intervenção do Gerente.

Os agentes do protótipo são implementados como objetos do ÁBACO. Desse modo um conjunto de interfaces IDL é disponibilizado para permitir a comunicação entre os agentes e entre estes e os outros elementos do protótipo.

Capítulo 09 – Estudo de Caso

9.1 Introdução

Este capítulo está dividido em três seções. A primeira seção apresenta a modelagem do banco de dados do gerente referente a MIB ILMI [ILMI4.0]. Esta base de dados é utilizada pelos dois estudos de casos apresentados na duas próximas seções: um relativo à área de contabilização e outro à área de falha. A figura 9.1 apresenta uma visão geral do estudo de caso, onde são descritos todos os componentes das aplicações detalhadas nesse capítulo.

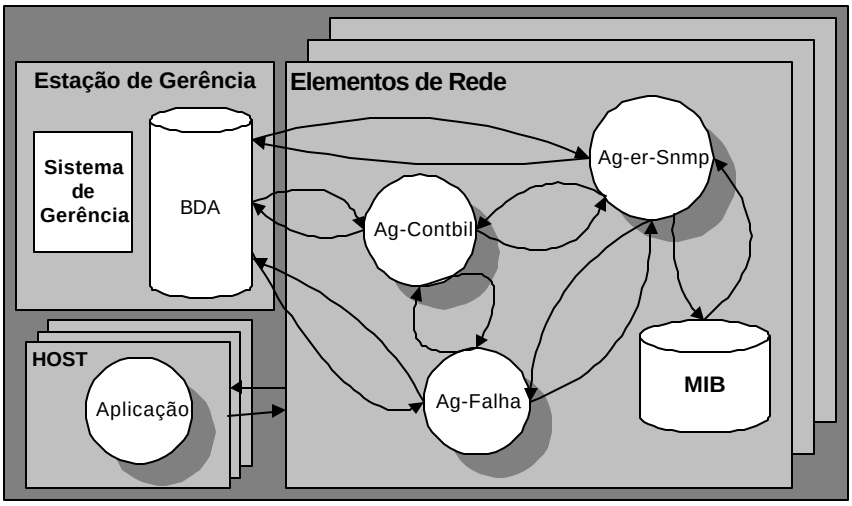


Figura 9.1 : Visão Geral do Estudo de Caso

9.2 Modelagem da Base de Informações do Gerente

A figura 9.2 apresenta a modelagem da MIB ILMI. Esta modelagem foi baseada no modelo de objetos da metodologia OMT (*Object Modeling Technique*) [Rumbaugh91]. Abaixo são apresentados códigos implementados na linguagem O++ do ODE, relativos a algumas classes e regras utilizadas nos estudos de caso. Maiores detalhes sobre a modelagem da MIB ILMI, podem ser encontradas no anexo A, onde foi realizado um estudo detalhado da estrutura de três MIBS-SNMP baseadas na tecnologia ATM. São elas além da MIB ILMI, a MIB AToM [RFC1695] e uma MIB Suplementar [atommib96]. Além disso, este anexo torna-se uma importante fonte de consulta, já que

apresenta de forma simplificada a estrutura destas MIBs, livrando o leitor das especif icações em ASN.1.

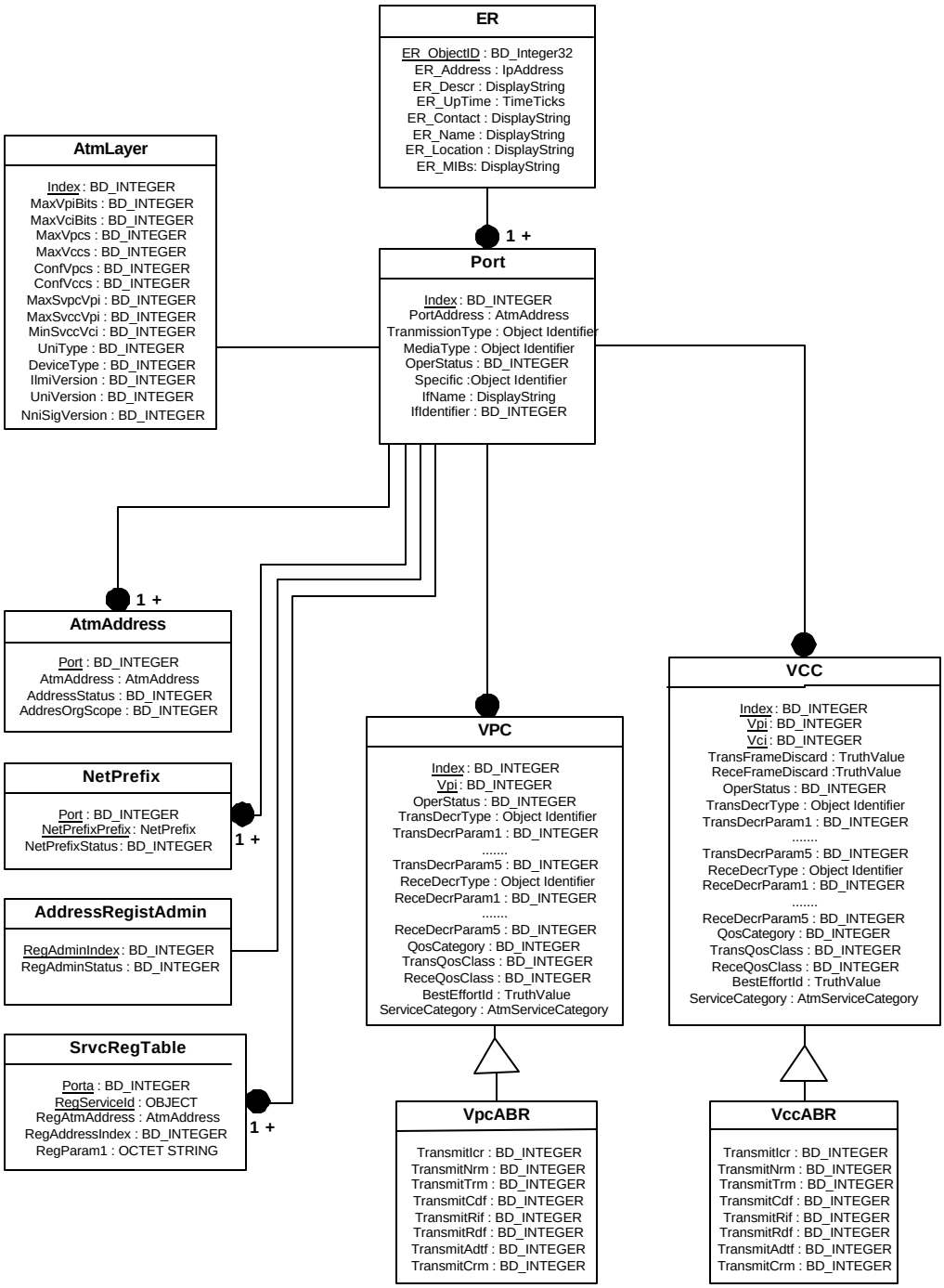


Figura 9.2 : Modelo de Objetos da MIB-ILMI (OMT)

Abaixo é apresentado o trecho de código referente a classe VPC.

```
#include <ode.h>

persistent class VPC {

public:

    BD_Integer Index;
    BD_Integer Vpi;
    BD_Integer OperStatus;
    ObjectID TransDescrType;
    BD_Integer TransDescrParam1;
    BD_Integer TransDescrParam2;
    BD_Integer TransDescrParam3;
    BD_Integer TransDescrParam4;
    BD_Integer TransDescrParam5;
    ObjectID ReceDescrType;
    BD_Integer ReceDescrParam1;
    BD_Integer ReceDescrParam2;
    BD_Integer ReceDescrParam3;
    BD_Integer ReceDescrParam4;
    BD_Integer ReceDescrParam5;
    BD_Integer QoSCategory;
    BD_Integer TransQoSClass;
    BD_Integer ReceQoSClass;
    TruthValue BestEffortId;
    AtmServiceCategory ServiceCategory;

    VPC();

    VPC( BD_Integer Index, BD_Integer Vpi, BD_Integer OperStatus, ObjectID
    TransDescrType, BD_Integer TransDescrParam1, BD_Integer TransDescrParam2,
    BD_Integer TransDescrParam3, BD_Integer TransDescrParam4, BD_Integer
    TransDescrParam5, ObjectID ReceDescrType, BD_Integer ReceDescrParam1,
    BD_Integer ReceDescrParam2, BD_Integer ReceDescrParam3, BD_Integer
    ReceDescrParam4, BD_Integer ReceDescrParam5, BD_Integer QoSCategory,
    BD_Integer TransQoSClass, BD_Integer ReceQoSClass, TruthValue BestEffortId,
    AtmServiceCategory ServiceCategory);

    persistent VPC * VPC::FindVPC(BD_Integer Id_ER, BD_Integer Vpi);

    void VPC::UpdateOperStatus(BD_Integer Status);

     void VPC::UpdateTransDescrParam (BD_Integer TransDescrParam1, BD_Integer
    TransDescrParam2, BD_Integer TransDescrParam3, BD_Integer
    TransDescrParam4, BD_Integer TransDescrParam5);

    void VPC::FindQoSCategory(BD_Integer Id_ER, BD_Integer Vpi);

    void VPC::FindBestEffortId(BD_Integer Id_ER, BD_Integer Vpi);
```

```
    /* Declarando Eventos */
    event after UpdateOperStatus, after VPC, before UpdateTransDescrParam;

    /* Regras */

    trigger Autoriza(TransDescrParam1, TransDescrParam2, TransDescrParam3,
    TransDescrParam4, TransDescrParam5): perpetual
    before UpdateTransDescrParam &
    (!Avalia_Recursos(TransDescrParam1, TransDescrParam2, TransDescrParam3,
    TransDescrParam4, TransDescrParam5)) ==>
    {
        ativa_agente_configuração;
        registra_log;
        ecout << "***** Não há recurso disponível" << endl;
    }


    trigger FalhaConexão(persistent VPC *VC):
    after UpdateOperStatus ==>
    {
        analisa_status (VC);
        ativa_agente_falha (VC);
        registra_log;
        ecout << "***** FALHA NA CONEXÃO" << endl;
    }


    trigger Inicia_processo_contabilizacao(persistent VPC *VC) :
    after VPC ==>
    {
        ativa_agente_contabilizacao (VC);
        registra_log;
        ecout << "***** NOVA CONEXÃO A SER CONTABILIZADA" << endl;
    }

}
```

9.3 Estudo de Caso : Contabilização

9.3.1 Visão Geral

Contabilização é uma área funcional importante para as redes corporativas, principalmente em ambientes ATM. Informações sobre usuários e serviços, motivam as concessionárias a implementar uma rede e obter informações de utilização dos recursos da rede. Diferentes categorias de serviço, tais como, voz, vídeo e dados necessitam ser avaliadas em conjunto para darem subsídios à análise detalhada da utilização da

rede.

Dentre as abordagens de gerenciamento ATM citadas no capítulo 04, nenhuma delas tem como objetivo principal o gerenciamento de contabilização [Alexander95]. O tratamento das funções de contabilização em ambientes ATM, apesar de suas peculiaridades, continua a ser tratado como em redes tradicionais.

A contabilização em redes ATM é realizada a partir de informações obtidas das MIBs, que são as informações negociadas junto ao usuário (aplicação) no momento do estabelecimento da conexão, ou seja, o cálculo da tarifação é realizado a partir dos parâmetros de qualidade de serviço (*QoS*) e contrato de tráfego. Além disso, informações obtidas junto às notificações enviadas pelos elementos de rede também podem ser utilizadas.

Portanto, as funções de contabilização aplicadas atualmente em ambientes ATM são caracterizadas pela forma estática de calcular as tarifas ao longo das conexões. Ou seja, os cálculos de tarifação são realizados apenas com base nos valores inicias dos parâmetros relativos à conexão citados acima, não correspondendo à real utilização dos recursos da rede.

9.3.2 Gerência de Contabilização e o Protótipo

As funções relativas à área de contabilização em ambientes ATM são uma boa forma de demonstrar o caráter distribuído do ÁTILA. Estas funções são tratadas através do banco de dados ativo ODE (BIG) e dos agentes inteligentes (BIA), visando o aumento da obtenção de informações sensíveis.

O ODE (BIG) funciona como elemento centralizador das informações. Isto tem como objetivo fornecer um ponto de gerência de informações onde são realizadas análises sobre a utilização e conseqüente tarifação de um conjunto de elementos de rede (BER). Outra importante função do ODE (BIG) é o armazenamento e controle das informações relativas à cotas e tarifas de um determinado serviço, aplicação e/ou usuário.

Os agentes inteligentes (BIA) tem a função de realizar o tratamento das informações junto ao elemento de rede (BER) . Isto permite ter-se um melhor acompanhamento da real utilização dos serviços em cada conexão. Além disso, estes agentes (BIA) são especializados tendo como base cada tipo de serviço que a rede disponibiliza. Outro fator importante é a troca de informações entre eles, o que permite uma melhor avaliação do cálculo de tarifação.

A interação entre o ODE (BIG) e os agentes inteligentes (BIA) junto aos elementos de rede (BER) permite uma melhor aproximação da tarifação real através das funções de contabilização, sendo realizada de uma maneira inteligente e distribuída. Abaixo são citadas algumas destas funções desempenhadas pelas tecnologias utilizadas no protótipo da arquitetura proposta.

- A tarifação e o estabelecimento de cotas pode ser determinado de forma dinâmica, isto é, a partir de informações de utilização dos recursos da rede fornecidos pelos agentes inteligentes (BIA), o ODE (BIG) tem a capacidade de avaliar/analisar através das regras o custo e consequentemente os valores associados à tarifação e cota. Ou seja, dependendo do grau de utilização dos recursos da rede, a tarifação de um determinado recurso aumenta ou diminui.
- A nível de controle, o ODE (BIG) através da interação com os agentes inteligentes (BIA), pode efetuar, baseado na cota do usuário, o bloqueio da prestação de serviços e o tratamento de prioridades em termos de aceitação de conexão (junto ao "CAC") ou descarte de células (junto ao "Balde Furado").

Estas funções dificilmente podem ser desempenhadas utilizando o modelo tradicional de gerenciamento "Gerente x Agente", devido ao caráter dinâmico e distribuído das aplicações de contabilização. Dessa maneira, o ÁTILA facilita a resolução desta problemática.

Outro fato importante é que ATM é uma plataforma que suporta vários tipos de serviço e os cálculos de tarifação dependem disso. Através do ÁTILA tem-se regras especializadas em cada tipo de serviço, tanto nos agentes inteligentes (BIA), quanto no ODE (BIG) com o objetivo de "cobrar o que é utilizado e não o que é combinado".

Detalhamento do Estudo de Caso

Em função das limitações das abordagens atuais em relação aos problemas de

contabilização, é apresentado neste estudo de caso uma tentativa de melhor aproximar a tarifação realizada da "tarifação verdadeira". Para isto será utilizado um parâmetro adicional : a taxa de utilização real dos recursos em função do serviço fornecido ao usuário.

A seguir, são apresentadas três visões do comportamento do protótipo em relação ao problema citado acima, denominadas Passo-a-passo, Diagrama de Seqüência e Pseudo-Código. Estas visões se baseiam na figura 9.1, e mostram as interações entre os elementos do protótipo, o comportamento destes elementos e o tipo de informação trocada.

- **Visão 01: Passo-a-passo**

Os passos a serem seguidos para realização das funções de gerência de configuração citadas acima, são apresentados abaixo.

1. O ODE (BIG) ativa os agentes *ag-contabil* (BIA) em determinados elementos de rede com o objetivo de obter informações tratadas sobre a utilização dos recursos.
2. Uma aplicação “X” qualquer interage com o CAC (Controle de Admissão de Conexão) para negociar o estabelecimento de uma conexão;
3. O CAC (Controle de Admissão de Conexão) a partir dos parâmetros de QoS e de tráfego, fornecidos pela aplicação, analisa a possibilidade de estabelecer a conexão;
4. A partir da resposta afirmativa do CAC, a aplicação juntamente com o elemento de rede diretamente interconectado estabelecem uma conexão;
5. A definição desta conexão é inserida na MIB do respectivo elemento de rede. Ou seja, do elemento de rede que está diretamente conectado ao *host* ou rede que gerou o pedido de estabelecimento da conexão;
6. O *ag-contabil* (BIA), caso esteja ativado neste elemento de rede, toma conhecimento desta nova conexão a partir de consultas contínuas à MIB realizadas através no *ag-er-snmp* (BER);
7. Em seguida, o *ag-contabil* (BIA) inicia o monitoramento e tratamento (cálculos) de informações estatísticas para realizar um estudo sobre a “real” utilização dos recursos por uma conexão;
8. O *ag-contabil* (BIA) faz o tratamento estatístico das informações das conexões e informa ao ODE (BIG);
9. O ODE (BIG) a partir destas informações realiza o cálculo dos custos incorridos

pelo uso dos recursos.

- **Visão 02 : Diagrama de Seqüência**

A figura 9.3 apresentada à seqüência a ser seguida para realizar a função de contabilização descrita acima.

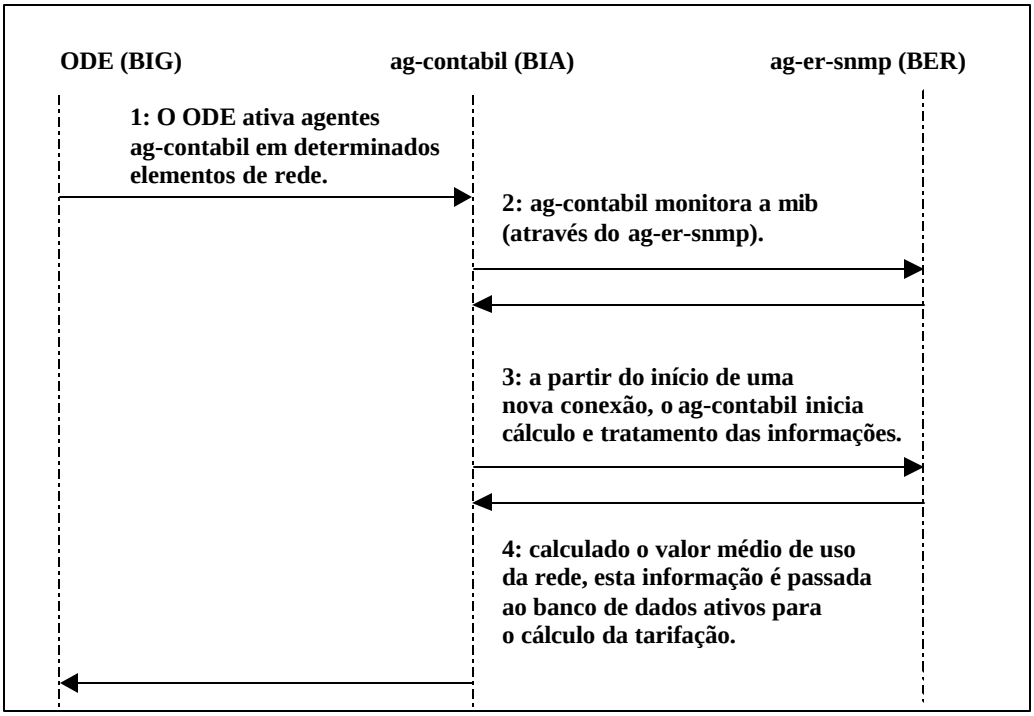


Figura 9.3 : Diagrama de Seqüência do Protótipo em Contabilização

- **Visão 03 : Pseudo-Código**

O Pseudo-Código a seguir, informa de uma maneira geral quais funções são chamadas no momento de inicialização do processo de contabilização. Ele informa também quais os passos necessários de execução para o cálculo de tarifação.

Function Processo_Contabil{

/ O ODE (BIG) ativa o ag-contabil (BIA) em determinados elementos de rede (BER) de acordo com o requisitado a ele pelas aplicacoes de gerência (BAU) */*

ativa_agentes_contabil(id_elemento de rede);

```
/*enquanto o processo continuar */
while not-sleep{
        if informacao_mib-nova_conexao {
                cria_sessao_monitora_conexao (id_aplicacao, id_elemento_rede,
                id_tipo_servico, taxa_de_transmissao);
        } /* fim if */

        /* a partir destes parametros e realizado um calculo medio continuo de
        utilizacao para cada conexao ativa */
        calcula_uso(id_aplicacao, id_elemento_rede, id_tipo_servico,
        taxa_de_transmissao);

        /* apos o calculo do uso das conexoes esta informacao e passada ao
        ODE (BIG) */
        inf_passada_bda(lista_sessao_conexao);

}/* fim while*/
}/* fim do processo*/
```

9.3.3 Especificação do Agente *ag-contabil*

O agente *ag-contabil* (BIA) é uma subclasse do *ag-atila*. Ele fica localizado em cada elemento de rede ATM, onde funções de gerenciamento de contabilização precisam ser executadas.

O *ag-contabil* tem a seguinte estrutura em sua implementação:

```
class ag-contabil{

    /* atributos */
    /* este atributo recebe o valor a partir do acesso ao ag-er-snmp (BER) tanto de
    configuracao como dados estatisticos da conexao */
    public AtDadosConexao DadosConexao;
```

/ este atributo contem os dados tratados que correspondem as informacoes de utilizacao real dos recursos da rede */*
public AtDadosAnalisados DadosAnalisados

/ metodos*/*
/ os metodos especificados abaixo tem como funcao permitir a interacao (acesso/manipulacao) de outros elementos da arquitetura com o ag-contabil (BIA) */*
/ os metodos "acesso_agentes_conexao" e "acesso_agentes_servico" permite o acesso ao ag-contabil (BIA) por outros agentes inteligentes (BIA), com o objetivo de trocar informacoes sobre o estado/configuracao das conexoes da rede */*

/ informacoes de uma conexao especifica */*
int acesso_agentes_conexao (int id_conexao, int interface, int elemento_rede)

/ informacoes sobre um tipo de servico especifico*/*
int acesso_agentes_servico (int tipo_servico, int interface, int elemento_rede)

/ o metodo "acesso_inf_agentes" permite ao ODE (BIG) obter informacoes gerais de estado/configuracao de um conjunto de conexoes da rede a partir do agente ag-contabil (BIA) */*

/ informacoes sobre uma ou mais conexoes */*
int[] acesso_inf_agentes(int[] grau_rede, int[] interface, int[] vpi_conexao, int[] vci_conexao, int elemento_rede)

/ informacoes sobre um ou mais tipos de servico */*
int[] acesso_inf_agentes(int[] grau_rede, int[] interface, int[] tipo_servico, int elemento_rede)

/ os metodos definidos abaixo dizem respeito ao processamento interno dos agentes ag-contabil (BIA) */*
/ o metodo "calcula_grau_uso_rede" retorna o grau de utilizacao dos recursos da redes obtido (avaliado) a partir de um conjunto de informacoes, tais como: parametros de QoS, parametros de trafego, media de celulas transmitidas, media de celulas transmitidas com sucesso, tempo da conexao */*

int calcula_grau_uso_rede (int reg_ncelulas_trans_csucesso, int reg_banda_conexao, int reg_qos_conexao);

/ este metodo a partir de um temporizador, calcula em uma determinada frequencia a utilizacao da rede */*
tempo_analise_dados(time tciclo)

/ o metodo "informacao_mib-nova_conexao" monitora a MIB com o objetivo de saber quando uma nova conexao foi estabelecida. O acesso a MIB e feito atraves do agente-er-snmp (BER)*/*
informacao_mib-nova_conexao;.

/ o metodo "inf_uso_conexao" retorna a utilização de uma determinada conexao, com base nas seguintes informacoes : nr. de celulas transmitidas, nr. de celulas transmitidas com sucesso, celulas descartadas, parametros de QoS */*
int inf_uso_conexao (int id_conexao, int intervalo_tempo, int interface, int elemento_rede);

/ o metodo "inf_uso_servico" retorna a utilização de um determinado servico, com base nas seguintes informacoes : nr. de celulas transmitidas, nr. de celulas transmitidas com sucesso, celulas descartadas, parametros de QoS, todas estas informacoes sendo obtidas de um conjunto de conexoes que possuem um tipo de servico especifico */*
int inf_uso_servico(int tipo_servico, int intervalo_tempo, int interface, int elemento_rede);

}

9.4 Estudo de Caso : Falha

9.4.1 Visão Geral

A gerência de falha se propõe em manter a operação de uma rede complexa. Para realizar estas funções, cuidados devem ser tomados no sistema como um todo, e individualmente em cada componente essencial. As funções de falha necessitam lidar com altas velocidades e terem a habilidade de trabalhar com vastos volumes de dados gerados a partir de diversas categorias de serviço, dando assim outra dimensão aos problemas/soluções da rede.

As funções de gerência relativas a área de falha em um ambiente ATM são realizadas através dos sistemas de gerência baseados no modelo tradicional Gerente x Agente x MIB e especificamente em relação a camada ATM, também pelas células OAM de falha [Minoli97]. Estas células provêem mecanismos de testes fim-a-fim disponibilizando ao sistema de gerência informações globais que não poderiam ser obtidas a partir das MIBs.

As funções de gerência de falha podem ser agrupadas em duas categorias [Minoli97]: funções de acompanhamento de alarmes (*alarm surveillance*), que inclui monitoramento e a notificação de falhas; e funções de localização e testes de falha, que tem propriedades como: habilitar o sistema de gerência de redes, seccionalizar uma falha, analisar características de circuito e equipamento, e permitir um elemento de rede ATM diagnosticar seu próprio estado interno.

Baseado nas funções citadas acima, quando da ocorrência de uma falha, é importante tão rapidamente quanto possível :

- Determinar a localização e realizar o isolamento da falha;
- Reconfigurar ou modificar a rede de tal maneira a minimizar o impacto da operação mesmo sem o(s) equipamento(s) com falha;
- Reparar ou trocar os equipamentos com falha para retornar a rede ao seu estado inicial.

As informações obtidas através das gerências de configuração e de desempenho devem ser utilizadas de forma pró-ativa para evitar falhas previsíveis.

9.4.2 Gerência de Falha e o Protótipo

O tratamento dos eventos local não resolve o problema de redundância de eventos, pois os mesmos podem ser gerados a partir de vários elementos de rede. Em função disto, a cooperação entre os agentes inteligentes (BIA), juntamente com as funções e informações disponibilizadas pelo banco de dados ativo ODE (BIG), são essenciais para um tratamento final dos eventos de falha gerados pela rede.

Dentre as funcionalidades existentes nos agentes inteligentes (BIA), com o objetivo de tratar problemas em falha, destacam-se :

- Filtragem e tratamento local de eventos (alarmes e notificações);
- Avaliação de informações das MIBs (BER), do ODE (BIG) e de informações obtidas a partir de células OAM;
- Capacidade para a resolução de problemas “pontuais”.

Os agentes (BIA) podem realizar esta correlação de alarmes e eventos localmente a um elemento de rede e também aprender a partir de novos eventos gerados. Estas correlações e filtragem de eventos e alarmes podem ser baseadas em informações, como limiares, obtidas do ODE (BIG).

A localização de falhas é um dos grandes problemas a serem enfrentados pelas funções de gerenciamento. Desta maneira, as informações disponibilizadas pelos agentes inteligentes (BIA) e pelo ODE (BIG) auxiliam na resolução do problema citado acima.

Quando da ocorrência de um problema específico os agentes (BIA) podem salvar o estado da rede naquele momento com o objetivo de analisá-lo, e posteriormente criar modelos que permitam a solução ou a prevenção de futuros problemas (mecanismo de aprendizagem).

Detalhamento do Estudo de Caso

Em função das limitações das abordagens atuais em relação aos problemas de falha, é apresentado neste estudo de caso uma tentativa de se atuar com maior rapidez aos eventos, através dos agentes inteligentes (BIA) que estão junto aos elementos de rede. Além disso, através do elemento centralizador, o ODE (BIG), pode-se ter funções e informações gerais que auxiliem estes agentes (BIA) no tratamento dos problemas.

Baseado na figura 9.1, é apresentado abaixo o comportamento do protótipo em relação as funções citadas acima, de três formas diferentes. Estas formas apresentam as interações entre os elementos do protótipo, o comportamento destes elementos e o tipo de informação trocada. As três formas são denominadas: Passo-a-passo, Diagrama de Seqüência e Pseudo-Código.

- **Visão 01 : Passo-a-passo**

Os passos a serem seguidos para realização das funções de gerência de falhas citadas acima, são apresentados abaixo. Esta seqüência indica o funcionamento de um processo

de ação do ODE (BIG) e dos agentes inteligentes (BIA).

1. ODE (BIG) ativa os agentes *ag-falha* (BIA) para o monitoramento e solução do problema de falha;
2. A partir de um alarme de falha provocado pela rede, o *ag-falha* (BIA), utilizando da informação disponibilizada pelo *ag-er-snmp* (BER), analisa a severidade de falha e tenta resolver o problema;
3. O *ag-falha* (BIA) não conseguindo resolver o problema em questão, busca na plataforma outros agentes (BIA) que possam auxiliá-lo;
4. Os agentes inteligentes (BIA) não solucionando o problema de falha, requisitam o auxílio do ODE (BIG);
5. O ODE (BIG), caso tenha as informações para a análise e solução da falha, age diretamente no elemento de rede através do *ag-er-snmp* (BER);
6. O ODE (BIG) informa o problema solucionado juntamente com sua solução (modelo evento-condição-ação) aos agentes *ag-falha* (BIA).

- **Visão 02 : Diagrama de Seqüência**

A figura 9.4 apresenta o Diagrama de Seqüência do problema de falha a ser resolvido, citado acima.

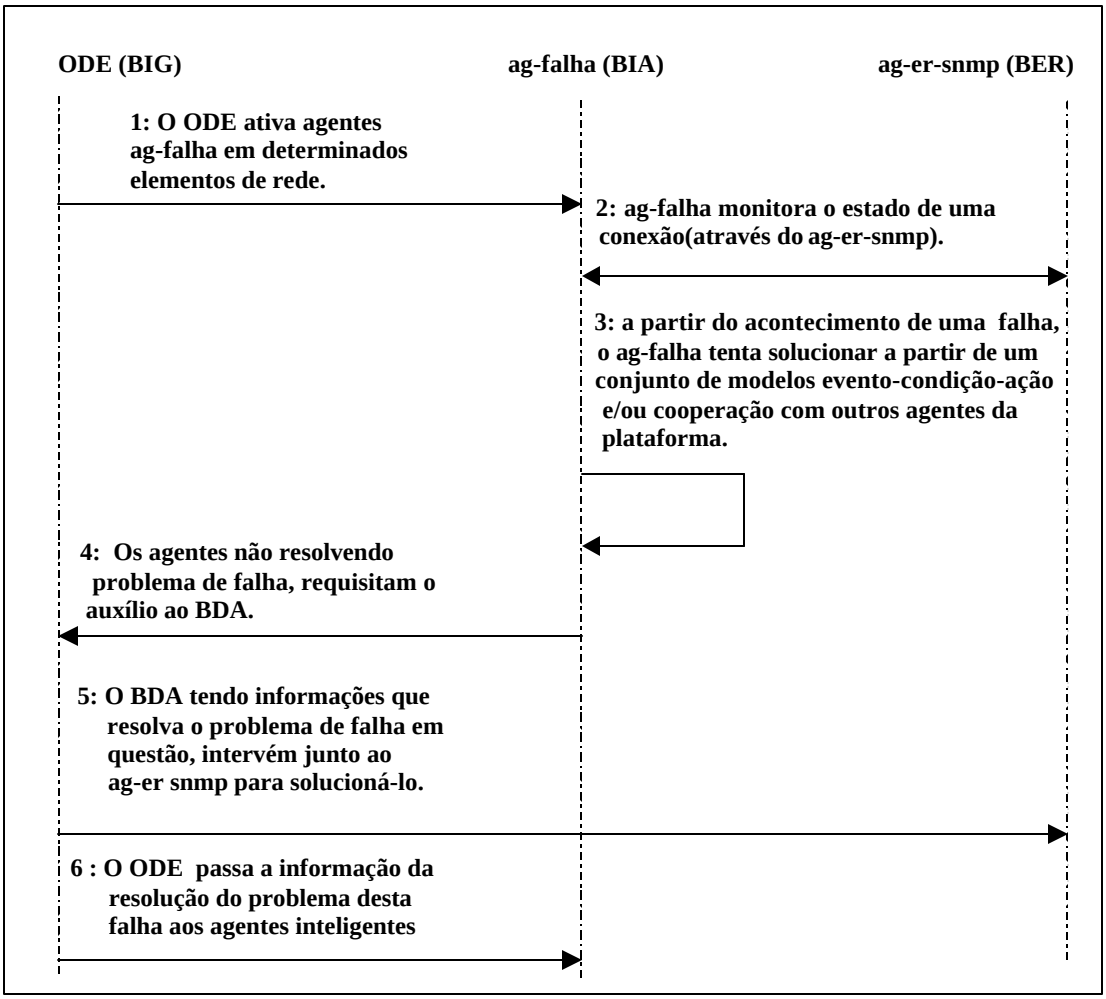


Figura 9.4 : Diagrama de Seqüência do Protótipo em Falha

- **Visão 03 : Pseudo-Código**

Este Pseudo-Código informa de uma maneira geral as funções chamadas no momento de inicialização do processo de falha.

Function Processo_Falha{

/ O ODE (BIG) ativa o ag-falha (BIA) em determinados elementos de rede de acordo com o requisitado a ele pelas aplicacoes de gerenciamento */*

ativa_agentes_falha(id_elemento de rede);

*/*enquanto o processo continuar */*

```
while not-sleep{

    /* esta funcao analisa todas as possiveis falhas a nivel de interface */
    cria_sessao_monitora_interfaces ( id_elemento_rede);

    /* a partir da criacao de uma nova conexao o ag-falha (BIA) fica
    monitorando continuamente esta conexao* /
    if informacao_mib-nova_conexao {
        cria_sessao_monitora_falha_conexao (id_elemento_rede,
        id_conexao);
    } /* fim if */

    if_acontece_falha {

        /* o ag-falha (BIA) analisa todos os parametros de severidade de falha, obtidos
        atraves do tratamento das informacoes monitoradas, na tentativa de localizar o
        problema de falha refererente*/
        analisar_severidade_falha;

        if analisar_modelo_falha(id_elemento_rede, id_falha);

        else if busca_agentes(id_elemento_rede, id_falha);

        else if busca_bda (id_elemento_rede, id_falha);

        acao_resolucao_falha;

    }//*fim if falha */
  }/* fim while*/
}/* fim do processo*/
```

9.4.3 Especificação do Agente *ag-falha*

O agente *ag-falha* (BIA) é uma subclasse do *ag-atila*. Ele fica localizado em cada

elemento de rede ATM, onde funções de gerenciamento de falha precisam ser executadas.

O *ag-falha* tem a seguinte estrutura em sua implementação:

```
public class ag-falha{

    /* atributos*/
    /*este atributo contém uma lista de modelos que representam niveis de
    conhecimento de um determinado ag-falha (BIA)*/
    public modelo modelo[];

    /* metodos*/
    /*  metodo que busca a cooperacao de outros agentes  inteligentes (BIA) para a
    resolucao do problema de falha em questao */
    boolean requisita_ajuda_agente (int id_elemento_rede, int Id_falha);

    /* metodo que busca cooperacao de ODE (BIG) */
    boolean requisita_ajuda_bda (int id_elemento_rede, int Id_falha);

    /*  metodo que filtra os eventos gerados pela rede*/
    filtragem_alarmes(alarmes[] alarmes)

    /* metodo que analisa a severidade da falha */
    analisar_severidade_falha();

    /* metodo que a partir da criação de uma nova conexao estabelece uma sessao de
    monitoramento para a mesma */
    cria_sessao_monitora_falha_conexao (int id_elemento_rede, int id_conexao);

    /* este metodo analisa todas as interfaces físicas associadas a um elemento de
    rede*/
    cria_sessao_monitora_interface (int id_elemento_rede);
```

}

Capítulo 10 – Conclusão / Trabalhos Futuros

• ATM e Redes de Telecomunicações e de Computadores

As chamadas redes de telecomunicações, idealizadas para o serviço de voz, e as redes de computadores, projetadas para o transporte de dados, têm experimentado, nesta mudança de século, o apelo de seus usuários quanto a necessidade de suportar aplicações com diversas mídias, sincronizadas e interativas. Baseadas na comutação de circuito e na comutação de pacotes, respectivamente, as redes de telecomunicações e as redes de computadores têm evoluído no sentido de satisfazer as novas exigências deste tráfego dito isócrono, no novo milênio.

Quanto às redes de telecomunicações, estas resolveram a integração de mídias, migrando em direção às RDSIs Faixa Larga, cuja tecnologia de base é o ATM. Do ponto de vista do usuário, esta migração em direção à outras tecnologias não oferecerá grandes traumas às Operadoras, uma vez que toda a inteligência da comutação esta localizada mais internamente à rede do que nos equipamentos dos usuários. Já nas redes de computadores, ao contrário das redes de telecomunicações, a existência de inteligência nos equipamentos usuários (aplicações e restante da pilha de protocolos) promove uma certa dificuldade em uma mudança de tecnologia, mesmo que esta mudança se justifique devido a uma adequação a novos requisitos do tráfego. Por exemplo, é impensável, abandonar-se de imediato a tecnologia de pacote baseada no IP, com vistas a uma tecnologia mais adequada a aplicações multimídia. Se de uma lado o IP é um "gargalo" para o tráfego isócrono, o parque instalado mundial da Internet é extraordinariamente grande suficiente para não se cogitar, pelo menos a curto prazo, esta opção. Daí terem surgido tentativas, como o RSVP (*Resource Reserved Protocol*), em dotar o IP de mecanismos de reserva de recursos, com o interesse em oferecer uma qualidade de serviço (*QoS*) ao novo usuário.

• ATM e o Futuro da Internet

Para conciliar estes dois fatos (ATM e Redes IP), não necessariamente excludentes, a Internet do futuro próximo fará uso do "IP over ATM", já iniciado no projeto Internet II. Por ser ATM, uma tecnologia baseada em conexão, portanto comutada, e o IP baseado em datagramas é roteado estabelece-se aparentemente uma incompatibilidade na convivência entre estas tecnologias. Este problema é minimizado se admitirmos a atuação do IP apenas no momento inicial de identificação das entidades cooperantes, ficando o ATM com a responsabilidade pelo restante da comunicação de dados na transação. Esta é a solução proposta em [Pujolle97].

Mas não é só em WANs que ATM tem se feito presente como alternativa de transporte de mídias a alta velocidade. A tecnologia ATM tem também feito suas incursões como solução de LANs e, principalmente, de MANs. O embate ATM x Ethernet tem sido recentemente vencido por este último, devido ao barateamento de placas de rede e de *switches* que segmentam barramentos, garantindo, senão a

isocronia da aplicação, pelo menos o desempenho na taxa de transmissão. Já como solução em redes metropolitanas, ATM tem desbancado a tecnologia FDDI que imperou até meados desta década.

- **ATM e o ÁTILA**

Devido a este contexto, ATM tornou-se, decididamente, uma tecnologia com grandes chances de consenso em projetos de "backbones" de longa distância e de redes de alcance metropolitano, pelo menos para os próximos anos. Em conseqüência, o convívio de ATM com outras diferentes tecnologias tem transformado os sistemas de comunicação em estruturas bem mais complexas, principalmente no que diz respeito às atividades de gerência. Por não ter um comportamento linear, a complexidade no gerenciamento resultada do crescimento e da heterogeneidade das redes tem exigido soluções não convencionais no monitoramento, análise e controle dos elementos de rede. Naturalmente, os modelos de gerência formalizados e já maduros, usados em redes de pacotes tem sido adaptados na tentativa de se gerenciar estas novas estruturas complexas, baseadas em ATM.

ÁTILA, o ambiente proposto neste trabalho, apresenta-se como uma nova arquitetura na tentativa de solucionar, por meio da pró-atividade, problemas de gerência mal resolvidos ou não resolvidos pelos modelos convencionais baseados em Gerente x Agente x MIB. Assim, ÁTILA pode também ser visto como uma evolução das três abordagens atualmente usadas em gerência ATM (ILMI, Células OAM e M R-ATM Forum) a medida que incorpora elementos inteligentes e distribuídos.

- **Abstrações e limitações do ÁTILA**

São três os níveis de abstração com que ÁTILA se apresentam neste trabalho. As arquiteturas funcional e física descritas no capítulo 5 da Parte II, independem da tecnologia adotada na implementação do protótipo. O fluxo de informação do ÁTILA, representado por um diagrama dinâmico no capítulo 6 da Parte II, bem demonstra a versatilidade dessa arquitetura no tratamento pró-ativo do sistema gerenciado. Em um segundo nível de abstração tem-se o ÁTILA visto através das tecnologias escolhidas (capítulo 7 da Parte III) para a implementação do protótipo. Banco de dados Ativo, Agentes Inteligentes e a plataforma CORBA conferem ao ÁTILA um caráter inovador. A terceira visão do ÁTILA refere-se a sua implementação propriamente dita. As ferramentas utilizadas no protótipo são ODE, JATlite e o ÁBACO, que correspondem as tecnologias acima citadas, respectivamente. Portanto, um outro protótipo completamente diferente do apresentado neste trabalho, é possível de ser implementado, tanto no que se refere a tecnologia quanto a ferramentas a serem utilizadas.

Dentre as limitações do protótipo implementado vale destacar a linguagem de composição de eventos do ODE. Ela possui restrições em relação a definição de eventos temporais, limitando, por conseguinte, a definição de regras utilizadas no monitoramento e controle dos elementos de rede.

Embora planejado inicialmente, não foi implementada a distribuição dos agentes na plataforma CORBA. Isto teria exigido a definição de uma estrutura de controle dos agentes. Optou-se, então, pela utilização do ambiente JATlite, o qual já incorpora mecanismos de controle e distribuição dos agentes. Portanto, o JATlite atua na interoperação entre agentes. No entanto, para um agente "conversar" com os outros componentes do protótipo, ele o faz por intermédio do ambiente ÁBACO.

A utilização do ÁBACO no protótipo deve-se a disponibilidade desta ferramenta, além do interesse em se analisar a influência que mecanismos de configuração de componentes existentes nesta plataforma teriam no desempenho do ÁTILA.

Por fim, vale dizer que não faz parte do escopo deste trabalho o estudo dos objetos disponibilizados das MIBs. Este assunto é complexo e deve ser considerado para estudos futuros pois, em virtude de ATM ser uma tecnologia relativamente nova.

- **Trabalhos Futuros**

A inexistência da disponibilização de um ambiente de experimentação ATM, impediu uma completa realização dos objetivos previstas com a arquitetura ÁTILA. Infelizmente, a experimentação planejada para um laboratório ATM associado ao LARC (Laboratório Nacional de Redes de Computadores) não foi levado a efeito.

Embora várias implementações tenham sido realizadas no contexto do ÁTILA, comprovando a aplicabilidade dessa arquitetura no gerenciamento pró-ativo de redes, um primeiro trabalho a ser perseguido, seria a transposição dessas implementações para um ambiente ATM de grande porte, onde se poderia analisar com mais precisão a eficácia do ÁTILA nesse ambiente.

Em relação ao protótipo, a eliminação do JATlite, deixando todo o aspecto de distribuição a cargo do ÁBACO seria um trabalho de relevância. Certamente, uma análise mais acurada sobre a influência do uso da linguagem de definição de componentes do ÁBACO, no desempenho da plataforma seria recomendável como tarefa a ser perseguida no futuro.

Uma análise de desempenho do ÁTILA sobre os sistemas convencionais de gerência, tanto de uma forma geral ou em determinadas tipo de gerência, seria extremamente recomendável, como um trabalho futuro.

Apesar de todas as limitações citadas, o ÁTILA é, sem dúvidas, um ambiente original na sua arquitetura proposta que põe em questão o modelo de gerência existente, criativo ao fazer uso de tecnologias não convencionais, tais como, Banco de Dados Ativo e Agentes Inteligentes, audacioso ao ser implementado com ferramentas que testam a eficiência de novos elementos em ambientes de gerência, ex. configuração de componentes do ÁBACO.

Espera-se que o ÁTILA, em sendo um produto acadêmico, possa servir como fonte de outras pesquisas na área.

Referências Bibliográficas

[Advent] “Advent SNMP Package Version 1.2”.

[Agrawal89] Agrawal, R., e Gehani, N. H. - “ODE (Object Database and Environment): The Language and the Data Model”, ACM-SIGMOD, Portland, Oregon - Maio 1989 - Pág. 36-45.

[Agrawal93] Dar, S., Agrawal, R., e Gehani, N. H. - “The O++ Database Programming Language: Implementation and Experience”, IEEE 9th Int'l Conf. Data Engineering - 1993.

[Alexander95] Alexander, Peter e Carpenter, Kacey – “ATM Net Management: A Status Report” - Data Communications - September 1995.

[Alles95] Alles A. - “ATM Internetworking” - Maio 1995.

[APM93] “AN OVERVIEW OF ANSAWARE 4.1”, *Architecture Projects Management Ltd.*, Cambridge, UK, 1993.

[Artola96] Artola - “Um Sistema Especialista para Gerência Pró-ativa Remota”, 14º. SBRC, Fortaleza-Ce, Brasil - Maio 1996.

[ATM-Forum] “ATM Forum” - URL: http://www.atmforum.com.

[atommib96] IETF <draft-ietf-atommib>, “Definitions of Supplemental Managed Objects for ATM Management” - Abril 1996.

[Bruegge92] Bruegge, Bernd - “Object-Oriented System Modeling with OMT” - Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA’92), Vancouver, ACM Press, pag. 359-376, Outubro, 1992.

[CNM94] ATM-FORUM <af-nm-0019.000>, “Customer Network Management (CNM)

for ATM Public Network Service" - Outubro 1994.

[CORBA91] "THE COMMON OBJECT REQUEST BROKER: ARCHITECTURE AND SPECIFICATION". Object Management Group, Dezembro 1991.

[Datta96] Datta, A., Viguier, I. e Orer L. – "Database for Active, Rapidly Changing data Systems" - Depto. of MIS, Universidade do Arizona, Janeiro 1996.

[Dayal89] McCarthy, D. R. e Dayal, U. - "The Architeture of An Active Database Management System", ACM-SIGMOD, Portland, Oregon - Maio 1989 - Pág. 215-224.

[Demazeau90] Demazeau – "Descentralized Artificial Intelligence" – Elsevier Science Publishers – Holanda – 1990.

[FIPA] "Foundation for Inteligent Physical Agents"- URL: http://drogo.cselt.stet.it/fipa/.

[Franca97] França, Mardônio – "Agentes SIM: Uma Abordagem de Gerenciamento Pró-Ativo de Redes de Computadores utilizando Sistemas Multi-Agentes" - IV Encontro de Iniciação Científica, Universidade Estadual do Ceará - Novembro 1997.

[Gaiti93] Gaiti, Dominique - "Distributed Artificial Intelligence as an AIP technique for Network Management", IFIP Transactions C: Communication Systems. Advanced Information Processing Techiniques for LAN and MAN Management. Versailles, France - Abril 1993.

[Gehani91] Gehani, N. H. e Jagadish, H. V. - "Ode as an Active Database: Constraints and Triggers". 17th Int'l Conf. Very Large Data Bases, Barcelona, Espanha – 1991 - Paginas 327-336.

[Gehani92] Gehani, N. H., Jagadish, H. V. e Shmueli, O. - "Event Specification in an Active Object-Oriented Database", AT&T Bell Labs Technical Memorandum - 1992.

[Greaves94] Greaves, David J., McAuley, Derek, French, Leslie J. e Eoin Hyden -

"Protocol and Interfaces for ATM LANs", - 1994.

[Hasan95] Hasan, Masum Z. - "An Active and Temporal Model for Network Management Databases" - IFIP/IEEE Fourth International Symposium on Integrated Network Management - Maio 1995 - Pág. 524-535.

[Hasan96] Hasan, Masum Z. - "The Management of Data, Events and Information Presentation for Network Management". Phd Thesis, Department of Computer Science, University of Waterloo, Ontario, Canada - 1996.

[HiPac] U. Dayal, B. Blaustein, A. Buchmann, U. Chakravarthy, M. Hsu, R. Ledin, D. McCarthy, A. Rosenthal, S. Sarin, M. J. Carey, M. Livny, R. Jauhari – "The HiPAC Project: Combining Active Databases and Timing Constraints". SIGMOD - Record 17(1): 51-70 (March 1988).

[Hott96] Hott, Robert W. - "Technical Review of ATM Specifications" - Julho, 1996.

[IETF] "Internet Engineering Task Force (IETF)" – URL: http://www.ietf.org

[ILMI4.0] ATM-FORUM <af-ilmi-0065.000>, "Integrated Local Management Interface (ILMI) Specification Version 4.0" - Setembro 1996.

[ITU-T] "International Telecommunication Union (ITU-T)" – URL: http://www.itu.ch.

[Jagadish92] Gehani, N. H., Jagadish, H. V. e Shmueli, O. - "Composite Events Especification in an Active Databse: Model & Implementation", AT&T Bell Labs Technical Memorandum - 1992.

[JATLite] "JATLite" - URL: http://java.sanford.edu/JATLiteDoc.html.

[JAVA] "JAVA" - URL: http://www.javasoft.com/.

[Juha92] Heinänen, Juha - "Routing and Addressing in ATM Networks", Telecom Finland - 1992.

[LANE1.0] ATM-FORUM <af-lane-0021.000> "ATM Forum LAN Emulation 1.0 (LANE) Specification" – Janeiro 1995.

[Lieuwen96] Lieuwen, D., Gehani, N., e Arlein, R. - "The Ode Active Database: Trigger Semantics and Implementation", Proc. Data Engineering - Março 1996.

[M4-94] ATM-FORUM <af-nm-0020.000>, "M4 Interface Requirements and Logical MIB" - Outubro 1994.

[M4-96] ATM-FORUM <af-nm-0058.000>, "M4 Public Network view" – Março 1996.

[Marshak91] Marshak, David S. - "ANSA - A MODEL FOR DISTRIBUTED COMPUTING". Network Monitor - Guide to Distributed Computing, Novembro 1991.

[MIB-II] Rose, Marshall T. e McCloghrie, Keith - IETF RFC1213 "Management Information Base for Network Management of TCP/IP-based Internets: MIB-II", 1991.

[Miller97] Miller, Mark A. - "Analyzing Broadband Networks (ISDN, Frame-Relay, SMDS, ATM)", segunda edição, M&T Books - 1997.

[Minoli97] Minoli, Daniel e Golway, Thomas – "Planning & Managing ATM Networks", Prentice Hall - 1997.

[MPOA1.0] ATM-FORUM <af-mpoa-0087.000> - "Multi-Protocol Over ATM Specification v1.0", Julho 1997.

[M.3000] "M.3000 Tutorial Introduction to TMN" – TMN M.3000 Series.

[Navathe94] Navathe e Elmasri, Ramez - "Fundamentals of Database Systems" - Segunda Edição, 1994.

[ODE4.0] - Gehani N., Arlein R., Gava J. e Lieuwen D. - "Ode 4.0 User Manual" - AT&T Bell Laboratories.

[OMG] "Object Management Group"- URL : http://www.omg.org.

[Partridge93] Partridge, Craig – "Gigabit Networking", Addison-Wesley Publishing Company - 1993.

[Pujolle97] Pujolle, Guy - "Les nouvelles technologies des réseaux haut -débits : ATM versus Internet" - II SFBSID'97 - Novembro 1997.

[RFC1695] IETF <RFC 1695>, "Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2" - Agosto 1994.

[Rocha96] Rocha, Marco; Fernandez, Luis e Westphall, Carlos - "Gerência Pró-ativa de Redes de Computadores usando Agentes e Técnicas de Inteligência Artificial"- 14º. SBRC, Fortaleza -Ce, Brasil - Maio 1996.

[Rose95] Rose, Marshall T. e McCloghrie, Keith - "How to Manage your Network Using SNMP", Prentice Hall - 1995.

[Rumbaugh91] J. Rumbaugh, M. Blaha, E. Eddy, and W. Lorensen.: *"OBJECT-ORIENTED MODELING AND DESIGN"*, Prentice-Hall, 1991.

[SAMOS] Stella Gatziu, Klaus R. Dittrich "SAMOS: an Active Object-Oriented Database System". Data Engineering Bulletin 15(1-4): 23-26 (1992).

[Sichman92] Sichman – "When Can Knowledge-based Systems Be Caleed Agents ?" – Simpósio Brasileiro de Inteligência Artificial – Rio de Janeiro, Brasil – 1992.

[Schuhknecht95] Schuhknecht, Anja e Dreo, Gabi -"Preventing Rather Repairing - A New Approach in ATM Network Management" - 1995.

[Siegel96] Siegel, J. - "CORBA FUNDAMENTALS AND PROGRAMMING", Editora Wiley Computer.

[Signalling] ATM-FORUM <af-sig-0061.000> "UNI Signaling 4.0" - Julho 1996.

[Soares95] Soares, L.F. G.: "REDES DE COMPUTADORES : DAS LANSS E WANSS ÀS REDES ATM". Editora Campus, 1995.

[Souza96] Souza, Cidcley T. de – "Um Ambiente para o Desenvolvimento de Aplicações Orientadas à Configuração Utilizando Objetos Distribuídos", Dissertação de Mestrado – Depto. de Computação, Universidade Federal do Ceará, Fortaleza – Ceará –Brasil, 1996.

[Stallings93] Stallings, W. – "SNMP, SNMPv2 and CMIP, The Pratical Guide to Network Management Standards", Addison-Wesley Publishing Company,Inc. - 1993.

[Stiller] Stiller, Burkhard - "A Survey of UNI Signaling Systems and Protocols for ATM Networks", Universidade de Cambridge.

[Tanenba um95] Tanenbaum, Andew S.- "Distributed Operating Systems", Prentice Hall - 1995.

[Tanenbaum96] Andrew S. Tanenbaum - "Computer Networks" - 1996.

[Traffic4.0] ATM-FORUM <af-tm-0056.000>, "ATM Forum Traffic Management Specification Version 4.0" - Abril 1996.

[UNI4.0] ATM Forum <af-sig-0061.000>, "ATM User-Network Interface (UNI) Signalling Specification Version 4.0" - Julho 1996.

[Vasconcelos97] Vasconcelos, Marcelo V. - "Utilizando Banco de Dados Ativo no suporte ao Gerenciamento Pró-ativo de Redes ATM", II SFBSID'97 - Novembro 1997 - Pág. 347-358.

[Vieira96] Vieira, M. e Sari, Solange - "Prototipação de um Sub-agente Adaptativo Baseado em Redes Neurais", 14º. SBRC, Fortaleza-Ce, Brasil - Maio 1996.

[Windom94] Windom, Jennifer – "Active Database Systems", Stanford University - Setembro 1994.

ANEXO A - Modelagem de MIBs ATM

1. Introdução

Este anexo apresenta as estruturas de três MIBs, que se relacionam com a tecnologia de redes ATM, e são definidas pelo IETF e ATM Forum. Como será visto, as MIBs detalhadas se baseiam no padrão de gerenciamento de redes SNMP (Simple Network Management Protocol).

As MIBs a serem descritas são:

- MIB AToM (RFC 1695 - "Definition of Managed Objects for ATM Management, Version 8.0 using SMIv2") [RFC1695];
- IETF DRAFT - "Definitions of Supplemental ATM Managed Objects for ATM Management" [atommib96];
- MIB ILMI 4.0 (Integrated Local Management Interface) [ILMI4.0].

As duas primeiras especificações citadas acima são definidas pelo IETF e se destinam ao suporte do uso de redes ATM no ambiente Internet. A MIB ILMI foi especificada pelo ATM Forum juntamente com a especificação da UNI 4.0.

As seções a seguir apresentam uma descrição detalhada de cada uma das variáveis destas MIBs, juntamente com suas modelagens no modelo entidade/relacionamento e no modelo de objetos, este último baseado na metodologia OMT (Object Modeling Technique) [Rumbaugh91].

2. MIB AToM - IETF

Esta especificação da MIB AToM é de responsabilidade do grupo de trabalho "MIB ATM", que é o encarregado pelas questões de gerenciamento ATM dentro do IETF. Ela se preocupa com a definição de objetos que auxiliam no gerenciamento de interfaces, circuitos virtuais, cruzamento de conexões, entidades AAL5 e conexões AAL5 em um ambiente ATM. A especificação atual desta MIB é baseada na SMIv2, ou seja, tem como base o protocolo de gerenciamento SNMPv2.

Esta MIB tem como principal propósito o gerenciamento de circuitos virtuais permanentes (PVC – Permanent Virtual Circuit) em ambientes ATM. Apesar de oferecer objetos que possuem informações sobre circuito virtuais comutados (SVC – Switched Virtual Circuit), este serviço requer a definição de capacidades adicionais não representadas nesta MIB. Estas capacidades estão sendo especificados em MIBs suplementares como a descrita na seção 3 deste anexo.

Com o objetivo de realizar o gerenciamento de interfaces, circuitos virtuais e cruzamento de conexões, outras MIBs são necessárias. Dentre elas temos: MIB II para o gerenciamento de interfaces e do sistema com um todo, MIBs relacionadas com interfaces físicas (DS3 e SONET), e MIBs que representam aplicações que se utilizam de ATM, como SMDS e Frame-relay.

Os objetos de gerenciamento definidos nesta MIB são representados através de tabelas e divididos de acordo com suas funções, como descrito no diagrama da figura A.1 abaixo.

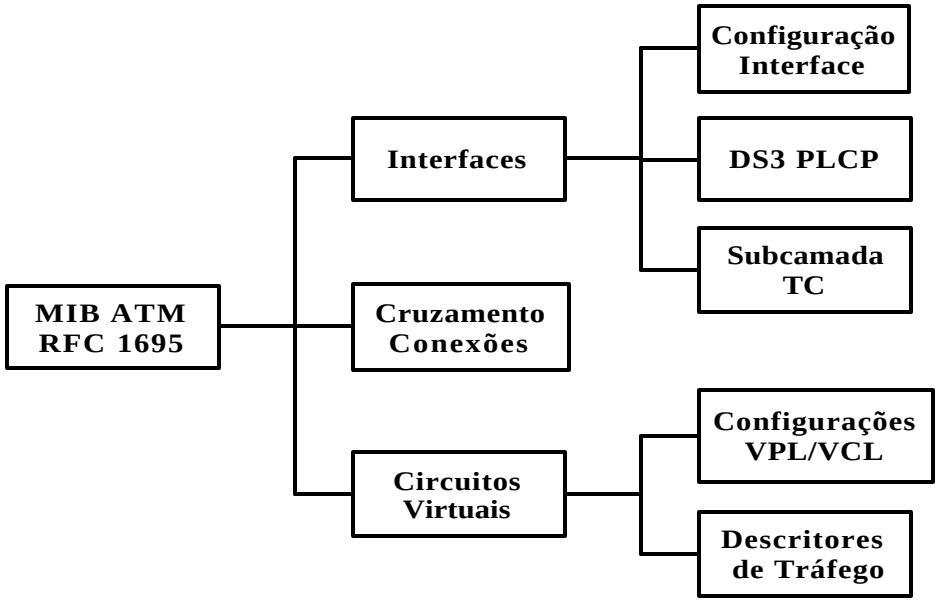


Figura A.1 : Estrutura da MIB AToM

2.1 Interfaces

Configuração de interfaces (atmInterfaceConfTable)

Este grupo contém informações de configuração sobre as interfaces ATM, além das encontradas na tabela de interfaces "ifTable", da MIB II [MIB-II].

Cada entrada se relaciona com uma interface ATM presente no dispositivo e é composta de parâmetros de configuração como: endereço ATM da interface (atmInterfaceAdminAddress), numero máximo de VPCs e VCCs suportados (atmInterfaceMaxVpcs e atmInterfaceMaxVccs), número de VPCs e VCCs configurados (atmInterfaceConfVpcs e atmInterfaceConfVccs) (incluindo SVCs e PVCs), VPI e VCI utilizado pelo protocolo ILMI, etc.

Interfaces DS3 PLCP (atmInterfaceDs3PlcpTable)

Este grupo é também uma extensão da tabela "ifTable" e contém parâmetros de configuração, e de estado do DS3[1] PLCP[2].

Esta tabela possui uma entrada por interface, que se utiliza de DS3 PLCP para transportar células sobre DS3. Cada entrada possui informações sobre o número de eventos de erro (atmInterfaceDs3PlcpSEFSs), existência de alarmes (atmInterfaceDs3PlcpAlarmSate) e de períodos de tempo em que a interface não esteve disponível (atmInterfaceDs3PlcpUASs).

Subcamada TC (atmInterfaceTCTable)

Este grupo possui parâmetros de estado e de configuração da subcamada TC. Também é uma extensão da tabela "ifTable".

1. Nível de multiplexação de PDH (Plesiochronous Digital Hierarchy) que opera a taxas de transmissão de 45Mb/s.

Cada entrada se relaciona com uma interface que utiliza a subcamada TC para transportar células. As entradas possuem dois atributos (atmInterfaceOCDEvents e atmInterfaceTCAlarmState) que contém informações sobre a existência de problemas na delineação de células. Exemplos destas interfaces são aquelas que possuem como camada física SONET ou DS3.

2.2 Circuitos virtuais ATM

Configuração de VPLs e VCLs (atmVplTable e atmVclTable)

Estes grupos contém informações de configuração e de estado de um circuito VPL/VCL bidirecional que são definidos a partir de duas tabelas: atmVplTable e atmVclTable, respectivamente. Como seus parâmetros são similares, iremos detalhá-las juntamente. Estas tabelas são definidas em terminais e comutadores e podem ser utilizadas para se criar, atualizar ou liberar um VPL/VCL.

Os parâmetros comuns à estas tabelas são: identificador de caminho virtual, que é o VPI (atmVplVpi) para conexões VPL e a combinação de VPI e VCI (atmVclVpi e atmVclVci) para VCL; os atributos que caracterizam o estado do circuito, AdminStatus (implementado para um VPL/VCL final de um VPC/VCC e descreve o estado administrativo do VPL/VCL indicando se o fluxo de tráfego está habilitado ou não), OperStatus (especifica o estado operacional do VPL/VCL) e LastChange (‘Timestamp’ que indica a última alteração do estado do circuito). Os outros parâmetros são índices para a tabela de descrição de tráfego (atmTrafficDescrParamTable - um para cada direção de tráfego dentro do circuito), e um índice para a tabela de cruzamento de conexões (CrossConnectTable).

O grupo VCL possui mais alguns parâmetros em sua tabela. Estes só são aplicados no caso em que o fim de um VCL é também o fim de um VCC. São eles: atmVccAalType (identifica o tipo de camada de adaptação utilizada neste circuito), atmVccAal5Cpcs*Transmit*SduSize (descreve o tamanho máximo da AAL5 CPCS SDU

2. PLCP (Physical Layer Convergence Procedure) é um método para transportar células sobre PDH,

em octetos que é suportado na direção de transmissão deste VCC), atmVccAal5Cpcs*Receive*SduSize (Igual o anterior na direção de recepção) e atmVccAal5EncapsType (especifica o tipo de encapsulamento de dados usado sobre a camada AAL5 SSCS).

Descritores de tráfego ATM (atmTrafficDescrParamTable)

Este grupo possui um conjunto de parâmetros que caracterizam o tráfego ATM, incluindo a classe da Qualidade de Serviço (QoS). Esta tabela é definida tanto em terminais como em comutadores.

Cada entrada descreve o tráfego que é transportado sobre um circuito virtual, tanto quantitativamente quanto qualitativamente. Os descritores de tráfego estão de acordo com os resultados da negociação, quando do estabelecimento da conexão. Quando é criada uma nova entrada nesta tabela seus parâmetros são checados para garantir a consistência.

Os atributos definidos em cada entrada da tabela podem ser divididos em dois subconjuntos:

- parâmetro atmTrafficDescrType descreve o tipo de tráfego e como interpretar outros cinco parâmetros, que caracterizam fluxo de tráfego. Os atributos definidos neste subconjunto são utilizados pelo serviço UPC (Usage Parameter Control);
- parâmetro atmTrafficQoSClass caracteriza a classe de serviço sendo utilizada pelo circuito virtual.

2.3 Cruzamento de conexões

Cruzamento de conexões VP/VC (atmVp(Vc)CrossConnectTable) - PVC

Estes grupos descrevem informações sobre estado e configuração de todos os cruzamentos de conexão VP/VC relacionados com PVC (Permanent Virtual Circuit), sejam eles ponto a ponto, ponto a multiponto e multiponto a multiponto. Estas

um meio físico baseado em frames.

informações são disponibilizadas somente em comutadores, onde temos a funcionalidade de cruzamento de conexões. Os grupos baseiam-se respectivamente nas tabelas atmVpCrossConnectTable e atmVcCrossConnectTable, que serão descritas juntas em virtude de suas similaridades.

Um conjunto de entrada nesta tabela representa o cruzamento de conexões VPC/VCC bidirecionais. Para uma conexão ponto a ponto temos uma entrada, ponto a mutiponto com 'N' nós folhas temos 'N' entradas e para conexões multiponto a multiponto entre 'N' nós temos, N(N-1)/2 entradas. Cada uma dessas entradas referencia duas entradas nas tabelas de circuitos virtuais (VPL e VCL).

2.4 Visualização em diagramas da estrutura da MIB

Modelo Entidade/Relacionamento

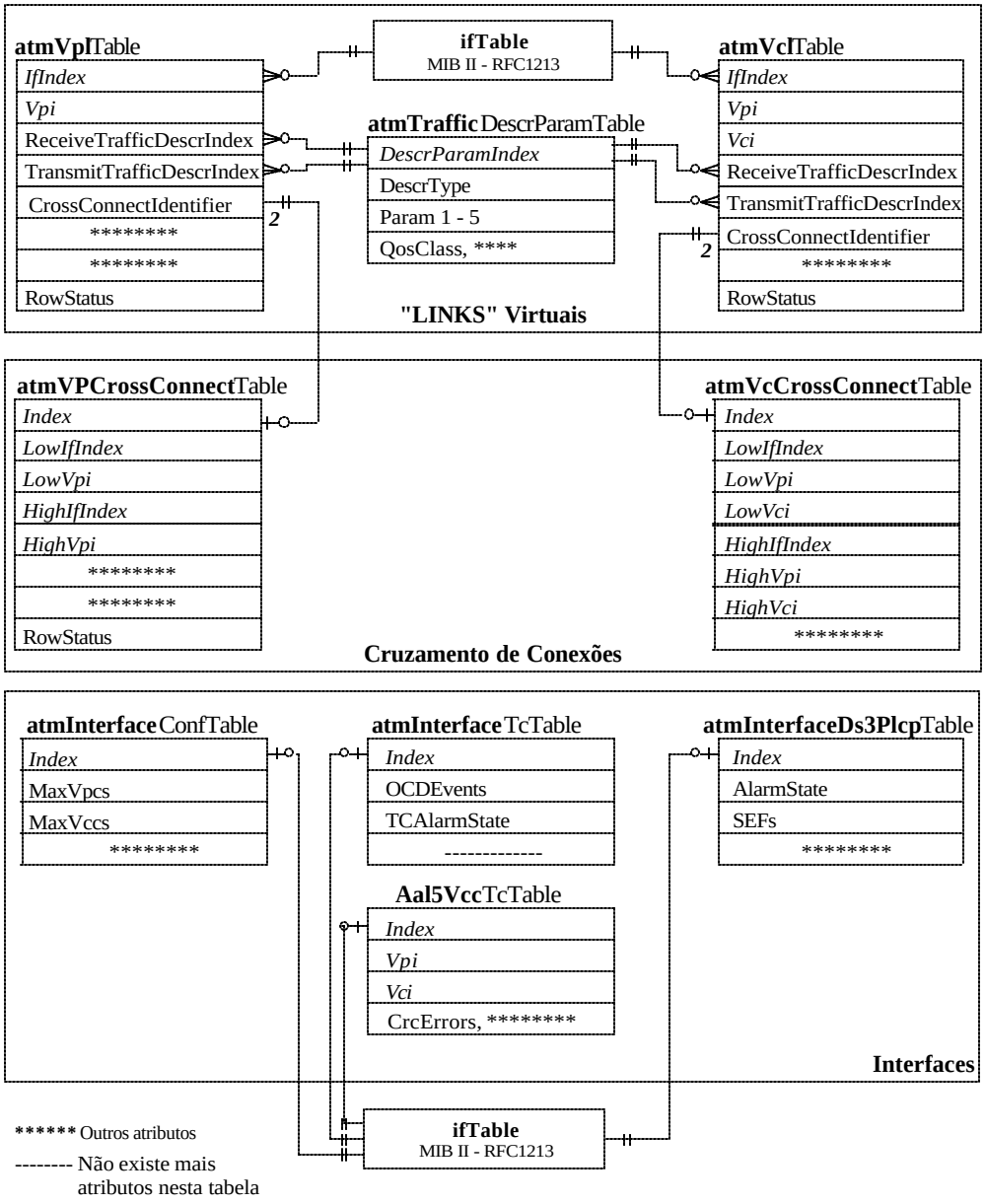


Figura A.2 : Modelo Entidade/Relacionamento : MIB AToM

Modelo Orientado a Objeto

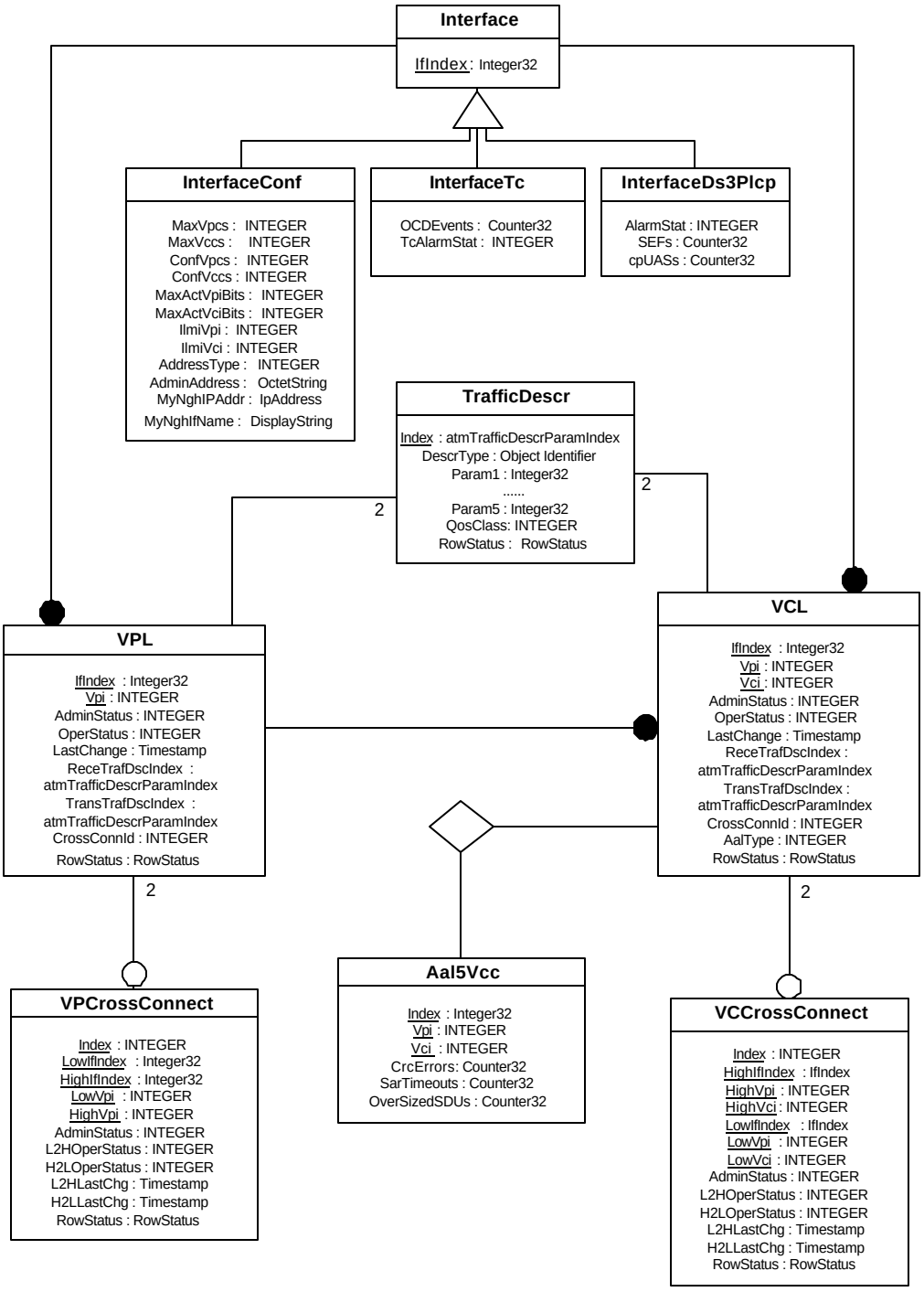


Figura A.3 : Modelo Objeto (OMT) : MIB AToM

3. MIB ILMI 4.0 - ATM Forum

A MIB ILMI se utiliza do protocolo SNMP, para prover à qualquer dispositivo ATM (Comutadores, etc.) informações sobre o seu estado e a sua configuração. Dados disponíveis nesta MIB se relacionam com: VPC (Virtual Path Connection), VCC (Virtual Channel Connection), prefixo de redes ATM, endereços ATM e serviços.

As especificações desta MIB a dividem em três outras MIBs que são relacionadas com: circuitos, endereçamento e serviço. A figura A.4 abaixo, descreve a estrutura geral desta MIB e em seguida é detalhado o seu conteúdo.

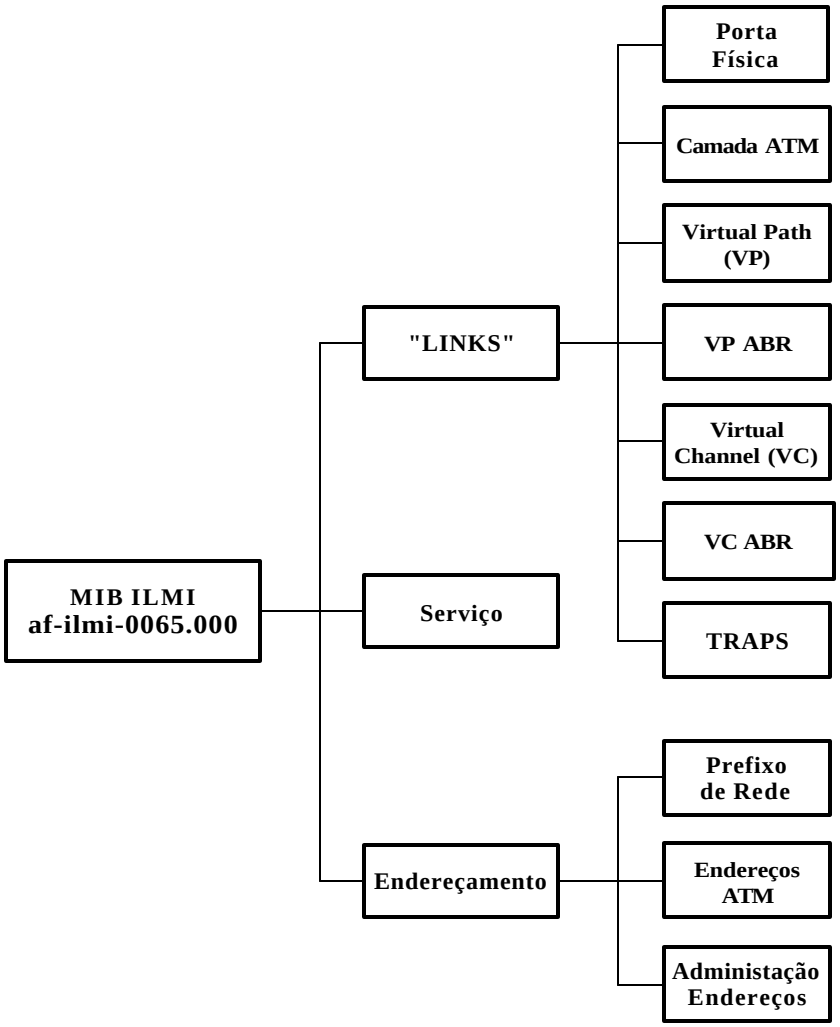


Figura A.4 : Estrutura da MIB ILMI

3.1 Enlaces (“ LINKs”)

A parte de enlaces definidas nesta MIB é composta de sete grupos, detalhados a seguir.

Camada Física (atmfPortTable)

Este grupo contém informações de estado e de configuração da camada física de interfaces físicas ATM.

As informações relacionados nesta tabela são:

- identificação implícita da interface física ou virtual sobre a qual mensagens ILMI são recebidas (atmfPortIndex).
- atributos que permitem que o sistema de vizinhança mantenha uma tabela de sistemas adjacentes para facilitar a auto-descoberta e os vestígios de conexões ATM. São eles: atmfPortMyIfName (nome da interface), atmfPortMyIfIdentifier (valor único para cada interface ATM, ex. IfIndex), atmfMyIpNmAddress (endereço IP da interface), atmfMyOsiNmNsapAddress (endereço NSAP da interface) e atmfMySystemIdentifier (endereço MAC).

Camada ATM (atmfAtmLayerTable)

Esta tabela contém informações sobre o estado e configuração da camada ATM em interfaces ATM.

Os atributos definidos neste grupo são (atmfAtmLayer + [3]):

- Index: identifica a interface ATM local;
- MaxVpiBits: número máximo de bits VPIs que podem estar ativos nesta interface. Para interfaces virtuais possui valor zero;

---

3. Este padrão será utilizado em todo o restante deste documento significando que todos os atributos definidos em relação a uma determinada tabela iniciam com a string definida antes do símbolo “+”, neste caso “atmfAtmLayer”.

- MaxVciBits: número máximo de bits VCIs que podem estar ativos nesta interface. Se uma célula é recebida em uma interface local de um comutador VP, este atributo é ignorado e todos os bits VCI devem ser comutados transparentemente;
- MaxVPCs: número máximo de VPCs, permanentes e comutados, que uma interface local pode suportar. Limitações podem restringir este número como menor ou igual a dois elevado ao atributo MaxVpiBits. Para interfaces virtuais possui valor zero;
- MaxVCCs: número máximo de VCCs, permanentes e comutados, que uma interface local pode suportar. Limitações podem restringir este número como menor ou igual a dois elevado ao somatório dos atributos MaxVpiBits e MaxVciBits;
- ConfiguredVPCs: número corrente de VPCs permanentes que uma interface local é configurada para processar;
- ConfiguredVCCs: número corrente de VCCs permanentes que uma interface local é configurada para processar;
- MaxSvpcVpi: valor máximo de VPI a ser utilizado numa alocação de VPIs para uma VPC comutada. Isto quer dizer que a entidade de sinalização desta interface só poderá se utilizar de VPIs com valores entre 1 (0 é usado pelo ILMI e sinalização de VCCs) e o valor deste atributo quando do estabelecimento de um VPC. Para interfaces virtuais ou se a mesma não suporta VPC comutado seu valor é zero;
- MaxSvccVpi: valor máximo de VPI a ser utilizado numa alocação de VPIs para uma VCC comutada. Isto quer dizer que a entidade de sinalização desta interface só poderá se utilizar de VPIs com valores entre 0 e o valor deste atributo quando do estabelecimento de um VCC. Para interfaces virtuais ou se a mesma não suporta VCC comutado seu valor é zero;
- MinSvccVpi : identifica o valor mínimo de um VCI a ser utilizado numa VCC comutada (Svcc). Este valor é usado por todos os VPI utilizados numa VCC comutada;
- UniType: indica se um dispositivo ATM é público ou privado;
- DeviceType: indica se o dispositivo ATM é do tipo "*user*" ou "*node*";
- atmfATMLayerIlmiVersion: é o valor da última versão da ILMI suportada nesta interface;
- UniVersion: indica a última versão da especificação da sinalização na UNI ATM pelo ATM Forum suportada nesta interface;
- NniSigVersion: indica a última versão da especificação de sinalização da PNNI pelo ATM Forum suportada nesta interface.

VP - Virtual Path (atmfVpcTable)

Esta tabela possui parâmetros de configuração e dados sobre o estado dos VPCs existentes numa interface ATM. Para interfaces virtuais esta tabela não se aplica.

As informações disponíveis neste grupo são (atmfVpc +):

- PortIndex : índice da interface, relacionado com uma interface definida na tabela "IfTable" da MIB II;
- Vpi : o valor do VPI relacionado com este VPC;
- OperStatus : o estado do VPC conhecido pela interface local;
- *Transmit*TrafficDescriptorType, Param1, ... : parâmetros que definem o tipo e a descrição do tráfego a ser transmitido por este VPC;
- *Receive*TrafficDescriptorType, Param1, ... : parâmetros que definem o tipo e a descrição do tráfego a ser recebido através deste VPC.

VP ABR - Available Bit Rate (atmfVpcAbrTable)

Cada entrada nesta tabela deve estar relacionada com uma entrada da tabela atmfVpcTable, onde neste caso o VPC deve estar definido como ABR.

Este grupo é indexado pelo índice da interface (atmfVpcAbrPortIndex) e pelo VPI (atmfVpcAbrVpi). Os outros atributos são especificados de acordo com os parâmetros ABR definidos em [Traffic4.0].

VC - Virtual Channel (atmfVccTable)

Somente VCCs permanentes que foram configurados para serem utilizados são descritos neste grupo, incluindo os VCCs permanentes padrões utilizados pelos protocolos de sinalização, pela ILMI, etc.

Os atributos definidos nesta tabela são (atmfVcc +):

- Vpi e Vci : índice da interface (PortIndex), os valores de VPI e VCI;
- OperStatus: o estado do VCC conhecido pela interface local;
- *Transmit*TrafficDescriptorType, Param1, ... : parâmetros que definem o tipo e a descrição do tráfego a ser transmitido por este VCC;
- *Receive*TrafficDescriptorType, Param1, ... : parâmetros que definem o tipo e a descrição do tráfego a ser recebido através deste VCC;
- *BestEffort*Indicator : indicador de melhor esforço;
- *Receive*FrameDiscard/ *Transmit*FrameDiscard : indicador de habilitação de mecanismos de descarte de "frames", sejam eles recebidos ou transmitidos;
- ServiceCategory : categoria do serviço disponibilizada neste VCC.

VC ABR - Available Bit Rate (atmfVccAbrTable)

Cada entrada nesta tabela deve estar relacionada com uma entrada da tabela atmfVccTable, onde o VCC é definido como ABR.

Este grupo é indexado pelo índice da interface (atmfVccAbrPortIndex) e pelo VPI/VCI (atmfVccAbrVpi/atmfVccAbrVci). Os outros atributos são especificados de acordo com os parâmetros ABR definidos em [Traffic4.0].

Traps (Notificações)

Os "traps" definidos nesta MIB são utilizados para prover informações sobre uma nova configuração, liberação ou alteração de VPCs ou VCCs *permanentes*. São eles atmfVpcChange que identifica o VPI da VPC modificada ou apagada e o atmfVccChange que identifica o VPI/VCI de uma VCC modificada ou apagada.

3.2 Endereçamento

Esta MIB é definida como uma extensão para a MIB ILMI, e possui três tabelas adicionais descritas abaixo.

Prefixo de Rede (atmfNetPrefixTable)

Esta tabela contém uma entrada para cada prefixo de rede e é localizada na IME (Interface Management Entity) do lado do usuário.

As informações descritas são (atmfNetPrefix +):

- Index : índice da interface;
- Prefix (prefixo da rede) : que no caso da estrutura de endereços de rede privada são os 13 primeiros octetos e no caso de endereços no padrão E.164 é o endereço todo codificado em 8 octetos, como se fosse um IDP (E.164) numa estrutura de endereçamento privado;
- Status : e o indicador de validade do prefixo de rede.

Endereços ATM (atmfAddressTable)

É uma tabela que pode ser implementada pela IME do lado da rede tanto em uma UNI privada, quanto pública. É indexada pelo índice da interface (atmfAddressPort) e além disso possui os seguintes atributos:

- atmfAddressAtmAddress : Identifica o endereço ATM, que no caso da estrutura de endereços de uma rede privada são os 20 octetos do endereço e na estrutura de rede pública é igual ao prefixo da rede;
- atmfAddressStatus : Indicador de validade do endereço ATM;
- atmfAddressOrgScope : Indica o escopo organizacional para os endereços associados.

Administração de registro de endereços (atmfAddressRegistrationAdminTable)

Esta tabela é obrigatória em todos os IMEs e é indexada pelo índice da interface. Outro atributo é o atmfAddressRegistrationAdminStatus que indica o suporte aos grupos atmfNetPrefixTable e atmfAddressTable. Os grupos citados acima só serão suportados

se ambos os lados da UNI o indicarem.

3.3 Serviço (atmfSrvcRegTable)

Esta MIB tem o propósito de se registrar os serviço de rede ATM (Lan Emulation Server (LES), ATM Name Server (ANS),..) disponibilizados para que estes fossem localizados mais facilmente.

Possui uma tabela chamada atmfSrvcRegTable que é implementada pelo IME no lado da rede de uma UNI ATM e contém todos os serviços disponibilizados pela rede ao IME do lado do usuário.

Os atributos disponíveis em cada entrada são (atmfSrvcReg +):

- Port : Identifica a interface ATM. O valor 0 indica a interface sobre a qual a mensagem foi recebida;
- ServiceId : Identifica unicamente um tipo de serviço no endereço ATM (ATMAddress) disponibilizado;
- ATMAddress : Indica o endereço completo do serviço. Utilizado pelo IME do lado do usuário para estabelecer uma conexão com um determinado serviço;
- AddressIndex : Inteiro arbitrário para diferenciar entradas que possuam o mesmo serviço numa mesma interface;
- Parm1 : Octeto onde o tamanho e o significado são determinados pelo valor do atributo ServiceId.

3.4 Visualização em diagramas da estrutura da MIB

Modelo Entidade/Relacionamento

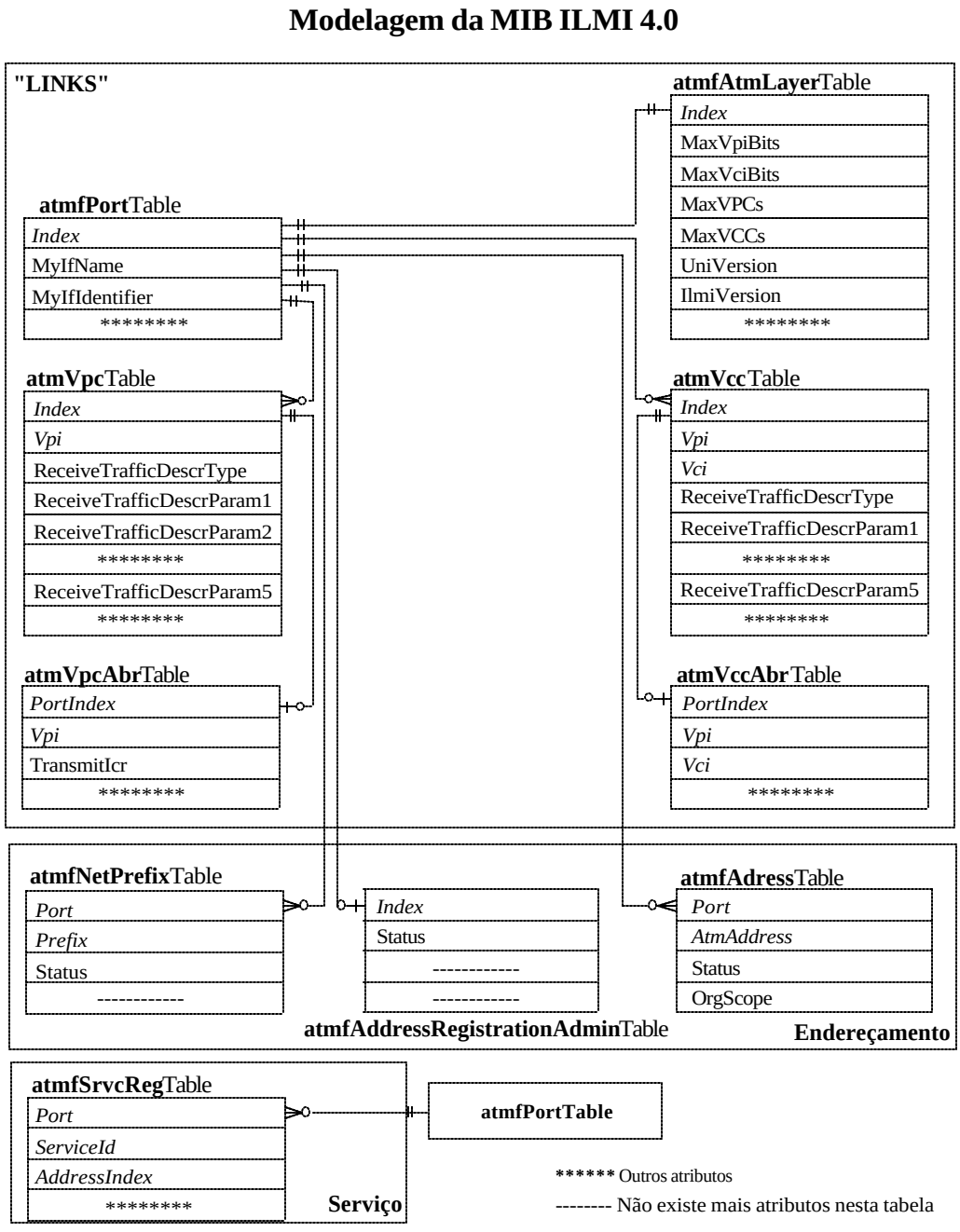


Figura A.5 : Modelo Entidade/Relacionamento : MIB ILMI

Modelo Orientado a Objeto

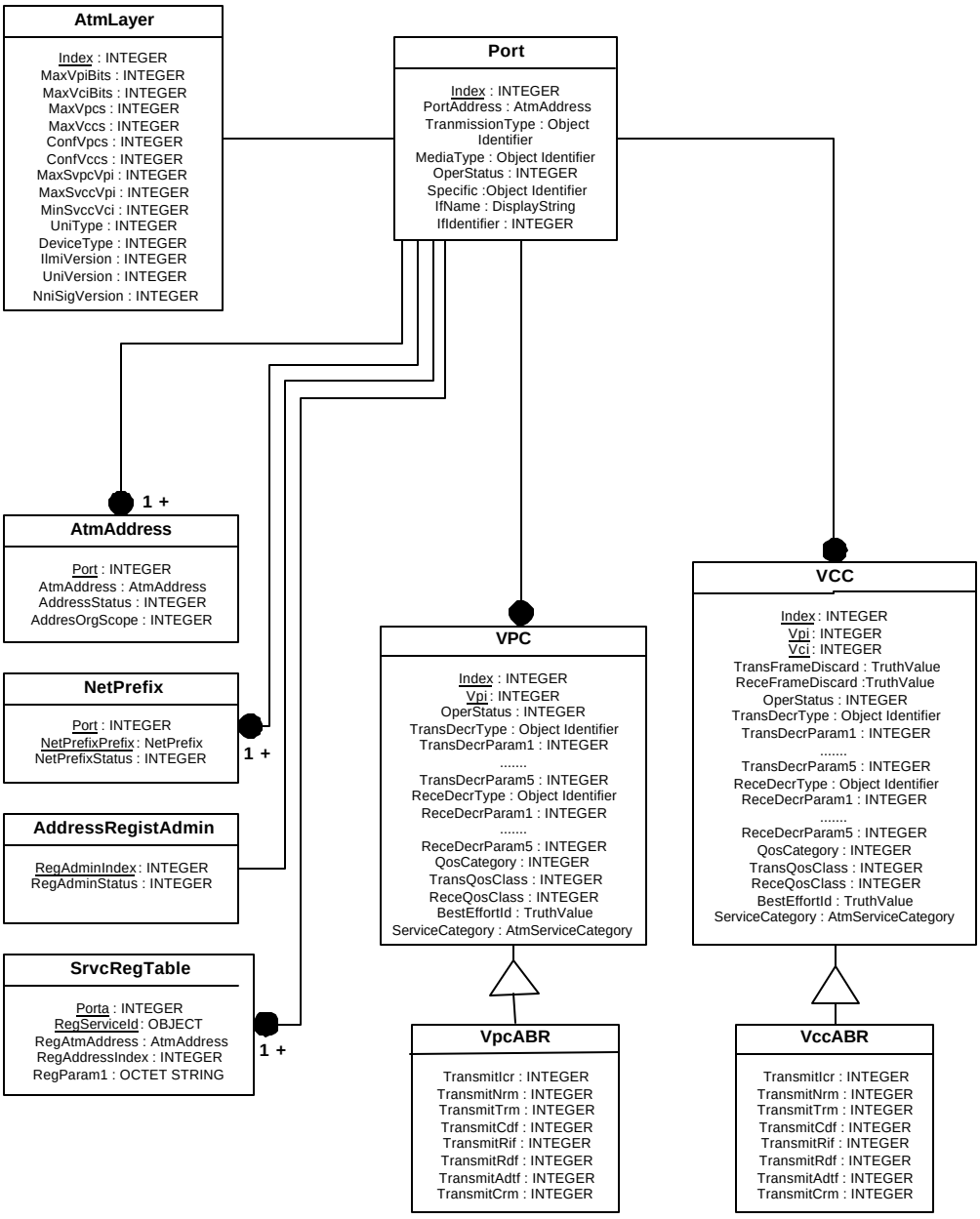


Figura A.6 : Modelo Objeto (OMT) : MIB ILMI

4. Objetos Suplementares para Gerenciamento ATM - IETF

Esta especificação, que é um *Internet Draft* [atommib96], foi proposta com o objetivo de auxiliar a MIB AToM no gerenciamento de redes ATM. Este documento provê extensões para o suporte ao gerenciamento de SVCs (Switched Virtual Circuit), o que não é oferecido pela MIB AToM.

Na figura A.7 a seguir é apresentada a estrutura desta MIB e logo após é detalhado o seu conteúdo.

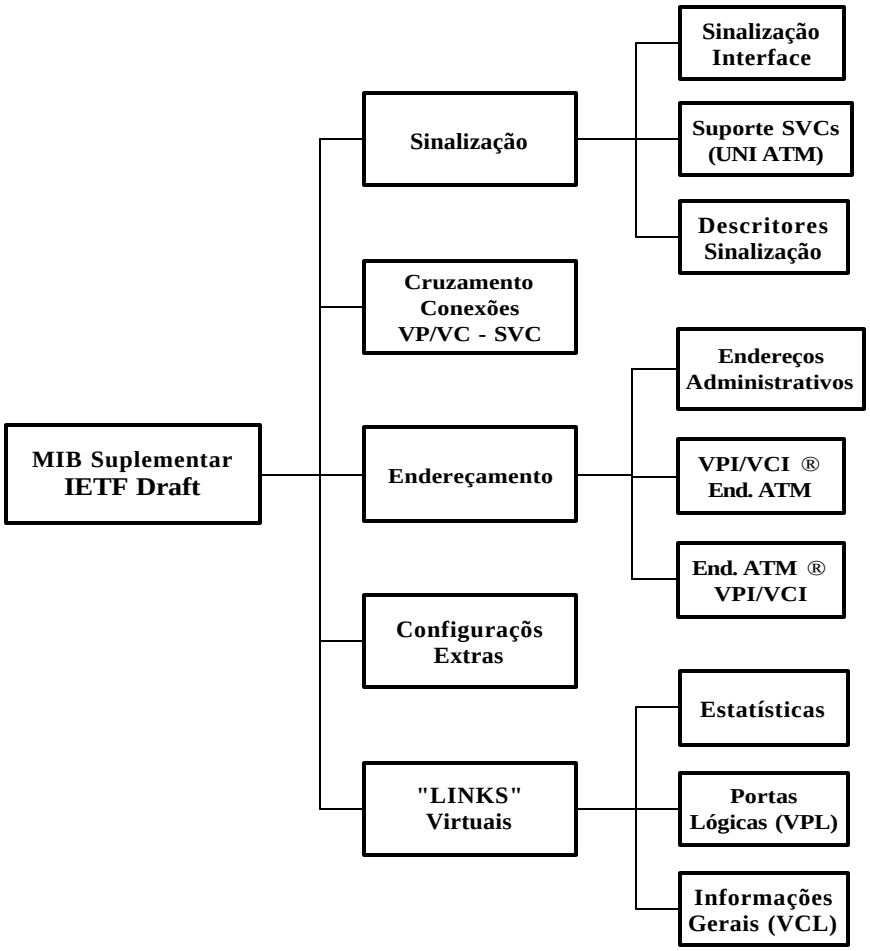


Figura A.7 : Estrutura da MIB Suplementar

4.1 Sinalização

Este grupo é composto de três tabelas onde a tabela "atmSigTable" possui informações sobre a configuração das entidades de sinalização, "atmSigSupportTable" provê informações que dão suporte ao processo de sinalização utilizado para o estabelecimento de SVCs, e "atmSigDescrTable" que possui dados sobre a sinalização das conexões VCCs existentes.

Sinalização numa interface ATM (atmSigTable)

É uma extensão da "IfTable", ou seja, as interfaces que suportam sinalização possuem uma entrada nesta tabela. Portas lógicas também podem se relacionar com esta tabela.

Os atributos mais importantes são divididos nos seguintes grupos:

- Tipo da sinalização: informa o tipo de sinalização configurado (atmConfigSigType) e o utilizado (atmActualSigType);
- Papel da entidade de sinalização: informa o papel da entidade de sinalização (atmConfSigSide) e o papel realmente sendo desempenhado (atmActualSigSide);
- SSCOP: este grupo possui informações sobre a camada de sinalização SSCOP, como: numero de eventos (desconexões, falhas na conexão, etc.) ocorridos nesta camada (atmSigSSCOPConEvents), número de PDUs com erro (atmSigSSCOPErrPdus);
- Falhas no serviço: contém informações sobre falhas que ocorreram quando do estabelecimento de um SVC, como: "time-out" em sinalizações emitidas a partir da interface (atmSig*Emitt*TimerExpires[4]), recursos indisponíveis (atmSig*Detect*UnavailResrcs[5]), rotas indisponíveis (atmSig*Emitt*UnavailRoutes) , etc.;
- Erros de protocolo na camada de sinalização quando estamos estabelecendo um

4. O *"Emitt"* existente nesta variável, relaciona-se com sinalizações que são emitidas a partir da interface relacionada.

5. O *"Detect"* existente nesta variável, relaciona-se com sinalizações que chegam na interface relacionada.

SVC. Contém dois atributos que definem o número de "Restarts" recebidos (atmSig*Detect*Restarts) e enviados (atmSig*Emitt*Restarts);

- Estatísticas: contém dois atributos que informam o número de conexões estabelecidas para SVCs entrando (atmSigInEstabls) na interface relacionada e para conexões saindo (atmSigOutEstabls) desta interface.

Suporte a SVCs na interface UNI ATM (atmSigSupportTable)

Esta tabela é utilizada para auxiliar no isolamento de problemas. Para que se possa isolar um problema temos que obter informações sobre quais objetos estão sendo suportados.

Alguns dos atributos citados abaixo devem existir em comutadores ou redes que suportam SVCs. Eles indicam se a entidade que iniciou o processo de sinalização ou a entidade que recebeu a requisição da sinalização disponibilizam ou não uma determinada informação.

- atmSigSupport*Clg*PtySubAddr : indica que o sub-endereço da entidade que requisitou o processo de sinalização será transferido até a entidade que recebeu esta requisição. AtmSigSupport*Cld*PtySubAddr : idem, mas o envio do sub-endereço é feito de maneira inversa, ou seja, da entidade requisitada à entidade requisitante.
- atmSigSupportAALInfo : parâmetros da camada AAL são enviados pela entidade que iniciou o processo de sinalização até a entidade que recebe esta requisição.
- atmSigSupportPerfCarrier : identifica o "carrier" para o qual chamadas entre "carrier" originadas desta interface são roteadas quando informações sobre a rede não é provida pela entidade requisitante da sinalização.

Parâmetros descritores de sinalização (atmSigDescrTable)

As informações contidas nesta tabela são disponibilizadas para a rede no momento da criação de um novo VCC, ou seja, quando ocorre a negociação de um conjunto de parâmetros entre o usuário e a rede. Informações adicionais são disponibilizadas pelas aplicações RDSI-FL, que além de fornecerem informações sobre a conexão, informam

sobre parâmetros de contrato de tráfego. As informações sobre contrato de tráfego estão contidas na tabela atmTrafficDescrParamTable descrita na seção 1.2.

Os dados contidos nesta tabela podem ser utilizados pela camada ATM para garantir a qualidade de serviço (QoS) desejada. Os principais atributos são divididos nos seguintes grupos (atmSigDescrParam +):

- AAL : são descritos aqui o tipo de AAL (AalType), o modo da AAL (AalMode) e a sub-camada de convergência (ex. SSCOP)(AalSscsType) utilizada pela conexão.
- Camadas acima da AAL : informa-se o tipo da camada sendo utilizada (Ex. ISO, proprietária) (BhliType) e dados adicionais (BhliInfo) dependendo do seu tipo.
- Broadband Bearer Capability (BBC) : atmSigDescrParamBbcConConf.
- Camadas abaixo da AAL: contém informações sobre os protocolos de nível 2 (BlliLayer2), nível 3 (BlliLayer3), tamanho "default" do pacote (BlliPktSize), identificador do SNAP (BlliSnapId) e codificação OUI/PID (BlliOuiPid).

4.2 Cruzamento de conexões

Cruzamento de conexões VP/VC (atmSvcVp(Vc)CrossConnectTable) - SVC

As informações sobre cruzamento de conexões relacionadas nesta MIB falam a respeito de SVCs. As informações sobre PVCs são especificadas na MIB ATM, descrita na seção 1.3.

Este grupo se baseia em duas tabelas, atmSvcVpCrossConnectTable e atmSvcVcCrossConnectTable, onde são armazenadas as informações sobre a configuração e o estado de todos os SVCs. A função destas tabelas é fazer a associação entre circuitos virtuais. Elas permitem acesso somente de leitura e podem ser utilizadas para monitorar o cruzamento de conexões entre os VPLs/VCLs em um comutador ATM.

Como na MIB ATM, no caso de uma conexão ponto a ponto temos uma única entrada nestas tabelas, para uma conexão ponto a multiponto (N nós destino) temos N entradas e

para conexões multiponto a multiponto (N membros) temo N(N-1)/2 entradas. Uma diferença entre a tabela de cruzamento de conexões relativas a PVCs e a relativa a SVC, é que a última não possui dados sobre o estado dos cruzamentos enquanto a primeira possui esta informação para cada uma das direções.

Como as informações sobre VPs e VCs são idênticas iremos descrevê-las ao mesmo tempo. Os termo "Low" e "High" representam a ordenação de duas interfaces associados com um cruzamento de conexões VPC/VCC. A interface com o termo "Low" é a que possui menor IfIndex[6] e "High" significa maior IfIndex.

Cada entrada (AtmSvc*Vp*CrossConnect + ou AtmSvc*Vc*CrossConnect +) possui um identificador de linha (Index), possui informações que identificam as interfaces e as conexões que serão cruzadas (LowIfIndex, LowVpi, HighIfIndex, HighVpi), no caso da tabela de VCs temos mais dois atributos (LowVci e HighVci) e o atributo RowStatus que especifica a validade da entrada da tabela.

4.3 Endereçamento

Este grupo é composto de dois tipos de informações sobre endereçamento divididas em três tabelas. A tabela atmIfAdminAddrTable possui uma lista de todos os endereços validos por interface e outras duas tabelas referem-se ao mapeamento de endereço. São elas: atmVclAddrBindTable que provê informações sobre o mapeamento de VPI/VCI em endereços ATM e atmAddrVclTable que possui dados sobre mapeamento de endereços ATM para VPI/VCI.

Endereços administrativos de uma interface (atmIfAdminAddrTable)

Esta tabela é aplicada somente para comutadores ou redes e somente para interfaces que possuam mais de um endereço.

6. IfIndex é o índice de uma interface na tabela IfTable (tabela de interfaces), especificada na MIB II.

Cada entrada nesta tabela informa: o endereço ATM de uma interface em um comutador ou rede (atmIfAdminAddrAddress), o tipo da fonte de um endereço dado a um endereço ATM (ex. dinâmico(3) no caso de se usar ILMI)(atmIfAdminAddrAddressSource) e a validade da entrada na tabela (atmIfAdminAddrRowstatus).

Mapeamento de VPI/VCI em endereços ATM (atmVclAddrBindTable)

Esta tabela possibilita o mapeamento entre a tabela atmVclTable e os endereços ATM locais e remotos. Ou seja, tendo o VPI/VCI de um VCL podemos encontrar os endereços ATM das duas entidades que se conectam através deste VCL. É bom notar que podemos ter nesta tabela várias entradas para uma mesma VCL, no caso de conexões ponto a multiponto.

Cada entrada possui: o endereço ATM de um lado do VCL (Este dado é sempre informado no momento da criação do VCL, sendo que no caso de SVC quem informa é o agente e no PVC este valor é informado pelo gerente) (atmVclAddrBindAddr), tipo do endereço representado (ex. Local ou Remoto)(atmVclAddrBindType) e a validade da entrada na tabela (atmVclAddrBindRowStatus).

Mapeamento de endereços ATM para VPI/VCI (atmAddrVclTable)

Esta tabela se comporta da mesma maneira que a atmVclAddrBindTable, a diferença é que enquanto esta tabela é indexada pelo endereço ATM a outra é indexada pelo índice da interface na tabela IfTable.

Os atributos informados em cada entrada são: o índice da interface que eqüivale ao definido na tabela "IfTable" da MIB II (atmAddrVclAtmIfIndex), o valor do VPI para o VCL (atmAddrVclVpi), o valor do VCI para o VCL(atmAddrVclVci) e o tipo de endereço ATM representado pelo objeto (atmVclAddrBindAddress) (ex. local ou remoto)(atmAddrVclAddrType). Os três primeiros parâmetros são utilizados em conjunto como índice na tabela atmVclTable definida na MIB ATM descrita na seção 1.2.

4.4 Circuitos virtuais

Este grupo é composto de quatro tabelas onde atmVplStatTable possui todas as informações estatísticas de cada VPL, atmVclStatTable possui todas as informações estatísticas de cada VCL, atmVplLogicalPortTable define portas lógicas que serão utilizadas na criação dos chamados túneis VP e atmVplGenTable que possui informações gerais sobre cada VC.

Estatísticas (atmVplStatTable e atmVclStatTable)

Estas tabelas são utilizadas para fazer o monitoramento da utilização de VPLs/VCLs em termos de células que entram e que saem.

Os atributos definidos são (atmVplStat + ou atmVclStat +):

- TotalCellIns : número total de células ATM válidas recebidas por este VPL/VCL, incluindo células com CLP=0 ou CLP=1;
- Clp0CellIns : número de células ATM válidas recebidas por este VPL/VCL com CLP=0;
- TotalDiscards : número total de células ATM válidas descartadas neste VPL/VCL pela entidade de policiamento de tráfego, incluindo células com CLP=0 ou CLP=1;
- Clp0Discards : número total de células ATM válidas recebidas com CLP=0 por este VPL/VCL, e que foram descartadas pela entidade de policiamento de tráfego;
- TotalCellOuts : número total de células ATM válidas enviadas por este VPL/VCL, incluindo células com CLP=0 ou CLP=1;
- Clp0CellOuts : número total de células ATM válidas com CLP=0, enviadas por este VPL/VCL;
- TaggedOuts : número total de células ATM válidas “tagged” pela entidade de policiamento de tráfego de CLP =0 para CLP=1 neste VPL/VCL.

VPL em Portas lógicas (atmVplLogicalPortTable)

As portas lógicas, como citado acima, são utilizadas na criação de túneis VP. Túneis VP tem como principal utilização a interligação entre duas redes ATM privadas através de uma ou mais redes públicas mesmo que estas não suportem sinalização. As redes públicas só devem fornecer uma conexão permanente VP e as redes privadas se utilizam da sinalização para criar VCCs dentro deste VP. Os túneis VP terminam em cada uma das pontas em interfaces lógicas que são definidas na tabela "IfTable" da MIB II.

A tabela atmVplLogicalPortTable é utilizada para fazer a conexão entre um dos VPLs definidos numa interface física ATM com a interface lógica criada. Esta tabela é uma extensão da tabela atmVplTable na MIB AToM (seção 1.2) e possui um atributo que identifica para cada VPL se este pertence a uma porta lógica ATM (atmVplLogicalPortDef). Se pertence, o outro atributo é utilizado para definir na tabela IfTable qual é a interface lógica que esta relacionada com este VPL (atmVplLogicalPortIndex).

Informações gerais sobre VCL (atmVclGenTable)

Como falado acima, esta tabela contém informações gerais a respeito de cada VC.

4.5 Configurações Extras

Extensões de configurações das interfaces (atmInterfaceExtTable)

Esta tabela contém informações adicionais não contidas na tabela atmInterfaceConfTable da MIB AToM definida na seção 1.1.

Os atributos definidos são separados em grupos:

- Dados configurados (atmInterface*Conf*Min): SvpcVpci (Mínimo VPCI que uma pilha de sinalização numa interface ATM é configurada para suportar, na alocação de VPCs comutadas), SvccVpci e SvccVci (Idem ao anterior, mas em relação ao valor de

VPCI e VCCI na alocação de VCCs comutadas.

- Dados reais (atmInterface*CurrentMin*): SvpcVpci (Mínimo VPCI que uma pilha de sinalização numa interface ATM pode atualmente alocar para VPCs comutadas), SvccVpci (Idem ao anterior, mas em relação ao valor de VPCI na alocação de VCCs comutadas.) e SvccVci (Idem ao anterior, mas em relação ao valor de VCCI na alocação de VCCs comutadas.).

4.6 Visualização em diagramas da estrutura da MIB

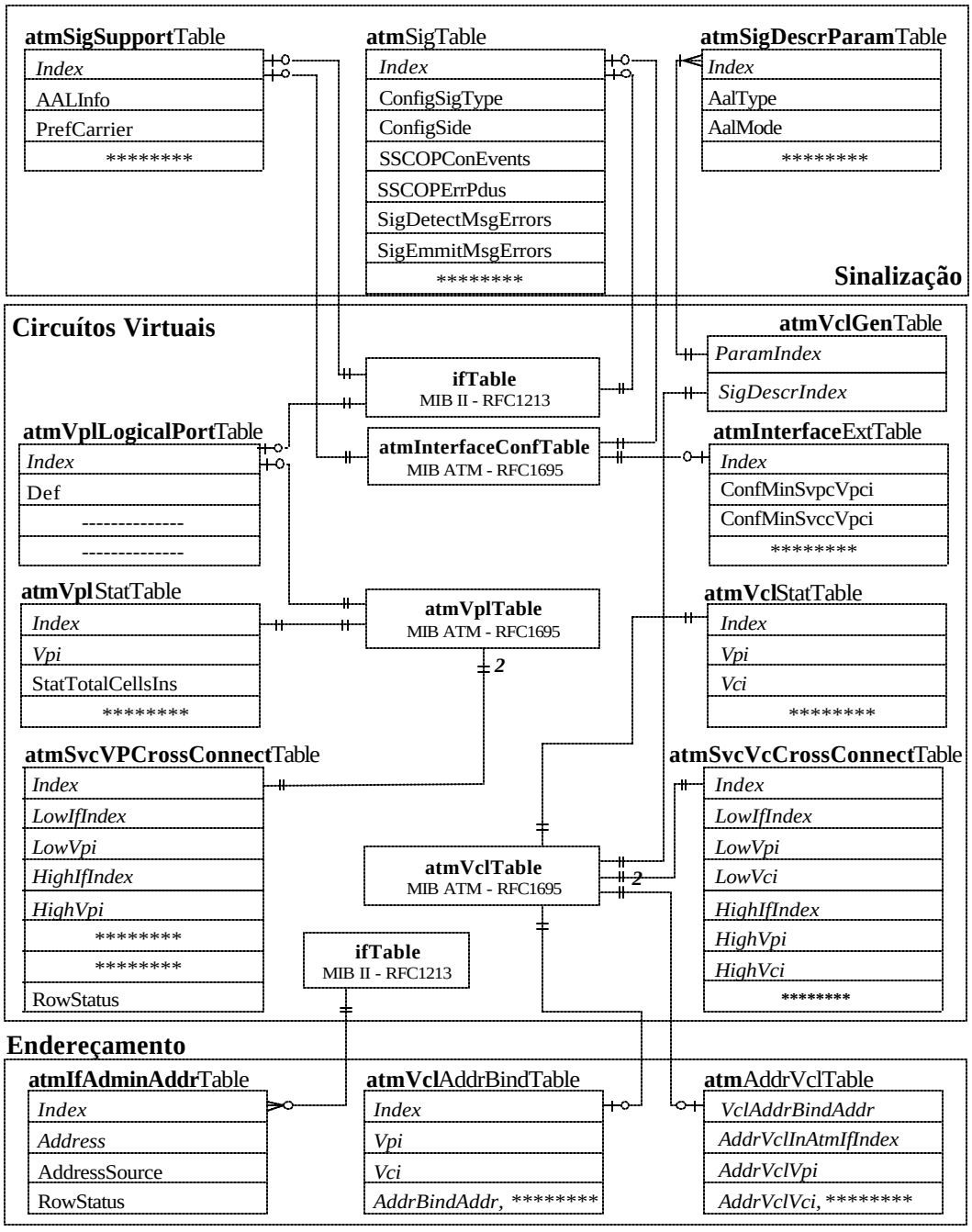


Figura A.8 : Modelo Entidade/Relacionamento : MIB Suplementar

ANEXO B – Banco de Dados Ativo

1. Introdução

Este anexo descreve as principais funcionalidades de um banco de dados ativo [Windom94] [Dayal89], e suas virtudes comparadas a banco de dados convencionais. É dado ênfase, na descrição do que é a unidade básica do ambiente proposto, a REGRA. São apresentadas diversas funcionalidades disponibilizadas pelo banco de dados ativo, que são essenciais ao gerenciamento de rede.

Bancos de dados ativos se baseiam nos sistemas de banco de dados convencionais, possuindo diversas outras funcionalidades. Algumas das capacidades inerentes a esta tecnologia são: a detecção da ocorrência de eventos, monitoramento de condições especificadas a partir do estado do BD e a execução de ações correspondentes. Isto sem que haja qualquer interferência do usuário (aplicação) ao sistema.

Comportamento ativo neste sistema é especificado através de regras de produção, também chamadas E-C-A (Evento-Condição-Ação), que são definidas e armazenadas no banco de dados. A semântica destas regras é a seguinte: o sistema monitora continuamente os eventos especificados; uma vez detectada a ocorrência de um evento relevante para uma regra, a condição associada a esta regra é avaliada e, se esta for satisfeita, o sistema executa a ação apropriada. A condição é definida sobre o estado do banco de dados e seu ambiente. Uma ação pode ser um tanto uma operação interna ao banco de dados, ou uma operação no próprio ambiente conforme observado na figura B.1 [Hasan96].

O objetivo principal da pesquisa em bancos de dados ativos tem sido obter respostas apropriadas a situações pré-especificadas sem sacrificar a modularidade do software. Os bancos de dados ativos tentam realizar isto através de um modelo de conhecimento em que as regras E-C-A são especificadas num nível de abstração que tem uma semântica bem definida, e de um modelo de execução em que a monitoração de situações, isto é, eventos e condições, e o desencadeamento das ações decorrentes, são feitos sem a intervenção dos usuários ou das aplicações.

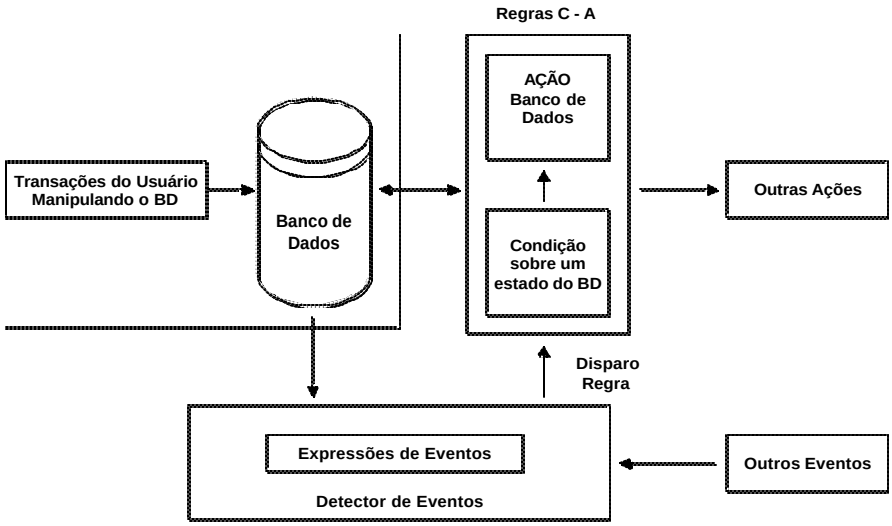


Figura B.1 : Estrutura de um Sistema Gerenciador de Banco de Dados Ativo

Integrar a facilidade de regras de produção em sistemas de banco de dados, fornece um mecanismo uniforme para um grande número de fatores avançados de banco de dados incluindo melhoramento dos recursos para garantir as restrições de integridade, manutenção de dados derivados, ações, alertas, proteção, controle de versão, asseguram restrições de acesso, conseguem estatísticas para otimização de consulta ou reorganização do banco de dados, dentre outras. Em adição, um sistema de banco de dados com a capacidade de processamento de regras provê uma plataforma útil para bases de conhecimento e sistemas especialistas grandes e eficientes.

2. Evento-Condição-Ação

2.1 Eventos

Um evento [Gehani92] é uma ocorrência num banco de dados que, teoricamente, não tem duração. Este especifica o fato causador do disparo de uma regra e pode ser básico (primitivo) ou composto.

Geralmente os bancos de dados ativos possuem os seguintes eventos primitivos:

- Eventos relacionados com operações de manipulação do banco de dados. Dentre elas estão: controle de transações (commit, etc.), execução de métodos relacionados a objetos, etc.
- Eventos temporais, que indicam o momento no qual a regra deve ser disparada. Estes podem ser dos seguintes tipos: absoluto, relativo ou periódico.
- Eventos externos, gerados fora do ambiente do banco de dados. São geralmente definidos por aplicações externas (ex. sensores indicando aquecimento, etc.).

Eventos primitivos podem ter argumentos, os quais são mapeados em valores reais no instante da detecção do evento. As partes Condição e Ação de uma regra E-C-A podem se utilizar destes parâmetros como base para teste e execução, respectivamente.

Apesar de eventos primitivos desempenharem um papel importante nestes sistemas, eles não possuem as funcionalidades necessárias para algumas aplicações. Em virtude disto alguns sistemas disponibilizam o que é chamado de eventos compostos. Composição de eventos são definidos através de uma álgebra que especifica o inter-relacionamento entre eventos primitivos ou compostos.

2.2 Condição

Em todas as linguagens de especificação de regras, a condição é definida como um predicado ou consulta sobre os dados do banco de dados. Os predicados podem se utilizar de métodos (funções) escritos em linguagem de programação de aplicação e as consultas usam operadores lógicos sobre o estado do banco de dados. Independente de como a condição é especificada, um mecanismo que possibilita referenciar-mos os parâmetros do evento que disparou a regra é disponibilizado. Ela deve ser avaliada após a regra ser disparada e antes da execução da ação.

A condição não é parte obrigatória de uma regra. Uma regra sem condição, significa que a ação deve ser sempre executada quando a regra associada é disparada.

2.3 Ação

A ação de uma regra E-C-A, descreve as operações a serem realizadas quando a regra é disparada e sua condição é satisfeita. Esta ação pode conter operações de banco de dados, operações de transação, operações de regra, procedimentos (métodos) escritos numa linguagem de programação que podem ou não acessar o banco de dados, etc.

Esta parte da regra é obrigatória. Geralmente se utiliza de parâmetros disponibilizados pelas partes Evento-Condição.

3. Composição de Eventos

É desejável que muitas aplicações reajam não somente a eventos primitivos, mas também a uma composição ou seleção de eventos ocorrendo em momentos diferentes. Uma álgebra de eventos permite a especificação de eventos compostos consistindo de outros eventos primitivos ou compostos, por meio de operadores definidos na álgebra. Uma expressão de composição de eventos tem como base a história de eventos, permitindo assim expressar relacionamentos entre eventos de uma maneira temporal. Redes de Petri e Máquinas de Estado Finito podem ser utilizados para modelar os operadores da linguagem e detectar a composição dos eventos expressadas como expressões de evento.

Os operadores definidos nestes sistemas permitem relações temporais entre eventos básicos e outras operações nos eventos, tais como, compressão, supressão, filtragem, agregação e cont abilização.

Linguagens de composição de eventos permite relacionar-se eventos básicos (ou compostos) ocorrendo em diferentes pontos no tempo, da mesma maneira que consultas temporais em BD temporais habilitam a especificação de modelos de valores em versões (história) sucessivas de relações. Assim especificação de eventos compostos são uma forma de consultas temporais.

4. Modelos de Execução

Apesar de parecer um processo simples, existem várias complicações com que o sistema de banco de dados ativo deve se preocupar, em virtude da diversidade de modelos de execução que se pode utilizar quando da implementação de regras. Alguns destas são: processo recursivo de regras onde a ação de uma regra pode causar o disparo de outros eventos, e a interação entre as transações "normais" do banco de dados e as geradas pela ação das regras.

Um modelo de execução que trabalha com transações aninhadas é o mais utilizado. Neste modelo, quando mais de uma regra são disparadas por um evento, as regras são executadas concorrentemente (transações serializáveis) como subtransações filhas do evento que as disparou.

O problemas da interação entre transações normais e transações de regras dependem de três características descritas abaixo.

5. Modos de Acoplamento

Os modos de acoplamento definem o relacionamento entre as partes Evento-Condição e Condição-Ação. Existem três modos que se aplicam para os dois modos.

- Imediato: a avaliação da condição/execução da regra ocorre na mesma transação que gerou o evento, imediatamente após a sinalização do evento/satisfação da condição.
- Retardado: a avaliação da condição/execução da regra ocorre no final da transação que gerou o evento, imediatamente antes do "commit".
- Desacoplado: a avaliação da condição/execução da regra é realizada numa transação separada da que gerou o evento.

5.1 Granularidade de mudanças no BD

A granularidade define o momento no qual uma regra disparada deve ser executada.

Existem duas possibilidades. Na primeira a regra (ação) é executada para cada instância do banco de dados que dispare a regra e satisfaça a sua condição. A segunda possibilidade é orientada a conjunto , ou seja, a regra (ação) é executada uma única vez para todas as instâncias que disparem a regra e satisfaçam a condição da regra.

	Modo C-A		
Modo E-C	**Imediato**	**Retardado**	**Desacoplado**
Imediato	Condição checada e ação executada após o evento.	Condição checada após evento, e ação executada no fim da transação.	Condição checada após evento, ação executada em uma transação separada.
Retardado	Inválida	Condição checada e ação executada no fim da transação.	Condição checada no fim da transação, ação executada em uma transação separada.
Desacoplado	Condição checada e ação executada em uma transação separada.	Inválida	Condição checada em uma transação separada, ação executada em outra transação separada.

Tabela B.1 : Modos de Acoplamento

5.2 Seleção de regras

Esta seleção define qual regra deverá ser executada a partir de um conjunto de regras disparadas. Existem diversas maneiras de resolvermos este conflito. A mais simples se resume à escolha aleatória de uma regra, o que introduz um grau de não-determinismo no processo. Geralmente, a escolha é feita baseada em prioridades especificadas nas definições das regras, no caso de termos regras de mesma prioridade então a escolha é aleatória.

Os seguintes eventos primitivos são geralmente suportado num SGBD Ativo: eventos

relacionados com operações de manipulação do banco de dados, eventos de transação, eventos de tempo absoluto e relativo, em banco de dados orientado a objetos, eventos de execução de métodos e funções e eventos explícitos e abstratos que surgem das aplicações.

6. Exemplos de Ferramentas que Implementam um Bancos de Dados Ativo

6.1 SAMOS (Swiss Active Mechanism-Base Object-Oriented Database System)

SAMOS [SAMOS] combina as características de banco de dados ativo e orientação à objeto. Ele ataca os três principais problemas de uma banco de dados ativo : especificação, execução e gerenciamento de regras. Especificação de regras é responsável pela natureza dos eventos, condições e ações e relacionamentos com o modelo de dados. Execução de regras refere-se ao processamento de regras, que deve ser integrada ao contexto do modelo de transação geral suportado pelo sistema de banco de dados. Gerenciamento de regras incorpora tarefas para representação interna das regras e eventos, para a detecção de eventos e a seleção de todas as regras que devem ser executadas.

SAMOS, analisa e transforma regras e eventos em uma forma interna por meio de um compilador de regras. Regras e eventos compilados se tornam uma parte persistente do banco de dados e formam a base de regras e eventos, respectivamente. Obviamente a base de regras e eventos pode ser alterada no decorrer do tempo. Estas bases são representadas de maneira orientada à objeto. Assim, regras e eventos são tratadas como objetos e todo evento definido pelo usuário pode ser representado como uma instância de uma classe.

6.2 HiPAC

HiPAC [HiPac] é um sistemas de banco de dados ativo baseado no paradigma de orientação à objeto. Neste banco de dados, regras são objetos de primeira classe, ou seja, podem ser organizadas em tipos como qualquer outro objeto. Tipos de regra podem participar em hierarquias de subtipo, elas podem ter atributos, e podem ser relacionados com outros objetos. Como outros objetos, regras podem ser incluídas em coleções, as quais podem ser explicitamente nomeadas ou definidas por consultas.

No HiPAC eventos podem ser operações genéricas no banco de dados (inserção, atualização, ...), invocações de métodos incluindo operações em objetos da classe regra, operações de transação, eventos temporais, eventos externos tais como mensagens e sinalização a partir de dispositivos, sinalizar a ocorrência de eventos definidos pelo usuário, chamadas a procedimentos de aplicações e vários eventos compostos deste eventos, incluindo disjunção, seqüência e repetição.

As condições especificadas nas regras são um conjunto de predicados ou consultas ao banco de dados no HIPAC. Condições transação também podem ser expressadas, usando o mecanismo de parametrização de eventos. No HiPAC o evento disparado de uma regra pode ser parametrizado, e estes parâmetros podem ser referenciados nas partes condição e ação das regras.

HiPAC difere consideravelmente da maioria dos sistemas de banco de dados ativo, na maneira que ele manipula múltiplas regras disparadas. Em vez de selecionar uma regra disparada para executar usando alguma forma de resolução de conflito, HiPAC executa todas as regras disparadas concorrentemente. Se durante a execução da regra, regras adicionais são disparadas, elas também são executadas concorrentemente. Para fazer isto, HiPAC usa a extensão do modelo de transações aninhadas. Regras no HiPAC possuem uma ordenação relativa, e esta ordenação é usada para influenciar a ordem de serialização da execução de sub-transações aninhadas concorrentes.

ANEXO C – Agentes Inteligentes

1. Introdução

Com a popularização das redes de computadores, surgiram diversos problemas onde o enfoque é a distribuição dos processos computacionais, sejam eles processos que envolvam gerenciamento, administração, ou comunicação. Foram necessárias soluções "inteligentes" para enfrentar estes problemas, soluções onde aspectos de portabilidade, interoperabilidade, performance são consistentes. Neste universo de problemática, surgiu uma área denominada "Distributed Solving Problem" (DPS), onde são apresentadas diversas soluções para problemas desta natureza.

Entretanto, a abordagem DPS, apresenta em geral, soluções particulares, inviabilizando este método para problemas mais genéricos. Face este questionamento, foi apresentado uma abordagem chamada "Multi Agent System" (MAS), onde é proposto um modelo para soluções que a distribuição de conhecimento tem um papel predominante.

A partir de disseminação da Internet, o conceito de Agentes Inteligentes foi popularizado de uma maneira extraordinária, causando um novo problema quanto ao enfoque e significado deste conceito. A exemplo do que aconteceu no início dos anos 80 com o conceito de objeto. O conceito de agente utilizado neste trabalho é baseado no conceito apresentado em [Sichman92], que têm as seguintes caraterísticas :

- Agentes são autônomos no sentido que eles podem definir suas próprias metas e planos internos;
- Eles são capazes de participar de complexas interações, utilizando de altos níveis de domínio em primitivas de comunicações independentes;
- Agentes são independentes de uma arquitetura particular. De fato, os agentes podem definir ou mudar a arquitetura tal qual a atividade de solução é executada;
- Eles são capazes de perceber no ambiente do qual estão inseridos, mudar seu comportamento e incorporar tais mudanças ao seu modelo interno;
- Eles são aptos a adicionar as habilidades e metas de outros agentes com a finalidade de resolver um problema cooperativamente;

A diferença entre a abordagem MAS e um Sistema Baseado em Conhecimento (SBC) é o fato do MAS ter facilidades de distribuição de conhecimento. Assim os "módulos inteligentes" podem cooperar para a resolução de um problema.

No contexto do ÁTILA, o MAS pode ser no entanto, abstraído como um SBC com características adicionais (*percepção*, de *comunicação*, de *adaptação* e de *distribuição*), como apresentado na figura C.1.

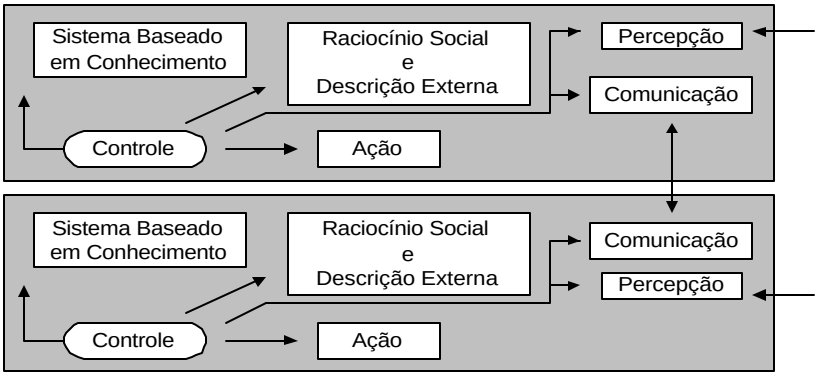


Figura C.1 : De Sistemas Baseados em Conhecimento a Agentes

Dessa maneira, distribui-se o conhecimento por toda a rede e ataca-se o problemas em diferentes módulos, ou seja, "agentes racionais", que apoiam, analisam e executam tarefas de gerência usando mecanismos de *cooperação*, *competição*, *coabitação* ou *distribuição* [Demazeau90].

Existem atualmente diversas implementações de aplicações baseadas em agentes, sendo referenciado neste trabalho JatLite [JATLite], KBS, e FIPA[FIPA]. Em paralelo à distribuição de conhecimento, cada vez mais ambientes de programação fornecem facilidades para a mobilidade de aplicações como JAVA, Tcl/Tk e ACL.

Na especificação do modelo de agente, utilizou-se neste trabalho o modelo proposto em [Franca97] que está estruturado da seguinte maneira:

- **Controle :** Esta parte do agente contém as regras do comportamento básico de relacionamento com outros agentes (*políticas de comportamento*). Neste modelo apresenta-se as seguintes políticas : *política de mensagens*, *política de serviço*, *política*

de destino e *política de decisão.*

- **Serviço de Comunicação :** Os protocolos de comunicação relativos aos agentes são tratados pela plataforma utilizada em nosso ambiente, e se dá pelo mecanismo de troca de mensagens entre objetos (*caixa de mensagem*) e as definições das interfaces (*portas comunicação*).
- **Base de Conhecimento :** A base de dados interna dos agentes é composta dos seguintes módulos : modelo do ambiente, modelo interno, objetos privativos (descrição interna), objetos públicos (descrição externa) .

Na implementação do modelo de agentes, utiliza-se o conceito de **Objetos Ativos** que tem as seguintes características :

- *estado público* de um agente é um lugar para encontrar as descrições externas, troca entre agentes de informações para comunicações ;
- *estado privado* de um agente é usado para comunicações interiores em um agente(um quadro-negro interno, instanciação).
- *atividades de comunicações e percepções* podem realizadas por método definidos nos agentes em especificações determinadas;
- *agentes* podem ser definidos adicionando *métodos de domínios independentes* para apoio a resolução de conflitos, decisões coletivas, negociações, etc. Alguns destes métodos serão responsáveis pelo controle de protocolos interativos.

2. Comportamentos Sociais Possíveis

Pode-se classificar os possíveis comportamentos de tais agentes em um ambiente de acordo com dois critérios :

- Localidade da tarefa global ou local. Uma tarefa global é uma tarefa que diz respeito a todos os agentes, e uma tarefa local que diz respeito a apenas um agente.
- Capacidade de realizar a tarefa capaz ou incapaz. Um agente é capaz de realizar uma tarefa se ele tem as habilidades necessárias para efetuá-la.

De acordo com esses critérios, os possíveis comportamentos ocorrem :

- Coabitação : um agente pode resolver suas próprias tarefas locais e assim faz. Pode-se notar que um agente não é obrigado a cooperar com outros agentes simplesmente porque eles estão emergidos no mesmo ambiente.
- Cooperação : um agente não pode resolver suas próprias tarefas locais(ou lhe falta habilidades suficientes ou ele não pode obter o desempenho necessário), e dessa forma pede aos outros para cooperar com ele.
- Colaboração : um agente pode resolver uma tarefa global sozinho. Se existem muitos agentes, um mecanismo de eleição deve ser executado.
- Distribuição : alguns objetivos globais devem ser realizados por uma ação coletiva. Nenhum dos agentes pode realizar a tarefa global sozinho, assim a tarefa deve ser dividida e alocada de acordo com de disponibilidade e especialidade de cada agente.

ANEXO D – Plataforma de Distribuição CORBA

1. Introdução

CORBA (Commom Object Request Broker) [Siegel96] [CORBA91] é uma plataforma de distribuição resultado da arquitetura OMA (Object Management Architecture) do OMG (Object Management Group) [OMG], uma organização internacional, fundada em maio de 1989 por 8 empresas: A plataforma CORBA fornece mecanismos pelos quais objetos, de forma transparente, fazem pedidos e recebem respostas em um ambiente distribuído.

A arquitetura OMA objetivava reduzir a complexidade, diminuir os custos e acelerar a introdução de novas aplicações distribuídas, sem que para isso seja preciso realizar grandes mudanças. Um modelo de referência OMA foi então especificado, caracterizando os componentes, as interfaces e protocolos, resultando daí a plataforma CORBA.

O intuito do OMG era ajudar o desenvolvimento e o crescimento da tecnologia de orientação a objetos. Seus princípios incluem o estabelecimento de diretrizes industriais e especificações de gerenciamento de objetos para prover uma base única para o desenvolvimento de aplicações. O OMG adota uma tecnologia denominada ORB (*Object Request Broker*), que fornece interoperabilidade entre aplicações em diferentes máquinas em ambientes heterogêneos distribuídos. O ORB tem as seguintes características:

- Prover mecanismos pelos quais os objetos fazem pedidos e recebem respostas de forma transparente.
- Prover interoperabilidade entre aplicações em diferentes máquinas em ambientes distribuídos e heterogêneos.

CORBA é, portanto, a tecnologia ORB adotada pelo OMG. Ela define uma estrutura para que diferentes implementações de ORBs possam prover serviços e *interfaces*

comuns para suportar clientes e implementações de objetos portáveis.

Dentre os aspectos relevantes para a caracterização da plataforma CORBA, destacam-se os seguintes:

- semântica dos objetos
- modelo de objetos

Semântica dos objetos

- **objetos**: entidade identificável e encapsulada que provê serviços que podem ser requisitados por clientes;
- **pedidos**: evento que acontece em um tempo definido. As informações associadas a um pedido são : operação, objeto alvo, parâmetros reais (que podem ser de entrada, de saída ou de entrada e saída) e um contexto de pedido opcional[7].
- **referências a objetos**: valor que identifica um objeto e que seguramente denota um (e somente um) objeto particular[8].
- **criação e destruição de objetos**: acontecem como resultados de pedidos. Do ponto de vista do cliente, não existe nenhum mecanismo especial para criação e destruição de objetos;
- **tipos**: conceito tradicional de tipos. Entidade identificável com um predicado, associado definido sobre os valores membros deste tipo.
- **interface**: descrição do conjunto de operações possíveis de um objeto que podem ser requisitadas por um cliente. Um objeto pode suportar múltiplas *interfaces* através de mecanismos como herança. As *interfaces* são especificadas em uma linguagem denominada IDL *(Interface Definition Language*);
- **operação**: entidade que denota um serviço a ser requisitado. É extremamente genérica, na medida em que é independente da implementação dos objetos e se utiliza de mecanismos de herança de *interfaces* em IDL.

7. Um pedido causa a execução de um determinado serviço, e retorna uma exceção se acontecer uma condição anormal durante a sua execução.

8. Em contrapartida, um objeto pode ser denotado por diversas referências.

Uma operação possui uma **assinatura**, a qual descreve todos os valores de parâmetros e resultados possíveis, através de:

- especificação de parâmetros necessários em chamadas a esta operação;
- especificação de resultados da operação;
- especificação das exceções que podem ocorrer durante a execução do serviço;
- especificação de informação adicional de contexto;
- indicação da semântica de execução que o cliente espera encontrar na ocasião de um pedido a esta operação.

A forma geral da **assinatura** de uma operação é:

[oneway] <op_type_spec> <identifier> (param1,...,paramL) [raises] (except1,...,exceptN)] [context(name1,...,nameM)]

Onde:

- **oneway** indica a semântica "*best-effort*" de execução, na qual a operação não retorna nenhum resultado, e não há sincronização entre o requisitante e o término da operação. O *default* é a semântica "*at-most-once*", na qual garante-se que se uma operação retorna com sucesso, esta foi executada exatamente uma vez. No caso de exceção, foi executada uma vez no máximo[9].
- **op_type_spec** é o tipo do retorno da operação, que é um parâmetro de saída distinto;
- **identifier** é o nome da operação;
- **parâmetros** são modificados para indicar a direção da informação: **in** significa que a informação passa do cliente para o servidor, **out** do servidor para o cliente e **inout** em ambas as direções.
- a expressão **raises** é seguida de uma lista de exceções definidas pelo usuário (as exceções padrão são incluídas implicitamente) que podem ser sinalizadas para terminar o pedido, indicando que a operação não foi executada com sucesso.
- **context** é uma expressão opcional que indica o contexto de pedido disponível na

9. A semântica de execução é associada a cada operação, o que garante que tanto o cliente quanto a implementação do objeto sempre assumirão a mesma semântica.

implementação do objeto, e pode afetar a performance do pedido.

- **atributos**: Analogamente aos conceitos de orientação a objetos pura, os atributos de uma interface são logicamente equivalentes a declaração de um par de funções de acesso: uma para recuperação do valor do atributo e outra para "setar" o seu valor (caso o atributo não seja apenas para leitura).

O modelo de objetos

Um modelo de objetos provê uma apresentação organizada dos conceitos e terminologia dos objetos, e define um modelo parcial para implementação que engloba as características fundamentais de objetos que devem ser consideradas pelas tecnologias.

O modelo de objetos definido pelo OMG [OMG] é um modelo abstrato, na medida em que não é diretamente realizado por uma tecnologia particular. Já o modelo de objetos do CORBA é um modelo concreto que foi derivado do modelo abstrato do OMG, e difere-se deste último por:

- ser mais específico (define-se aspectos como a forma dos parâmetros dos pedidos, linguagem de especificação de tipos, etc.).
- introduzir instâncias específicas de entidades definidas no modelo.
- restringir o modelo, eliminando entidades ou acrescentando restrições adicionais no seu uso.

No sistema de objetos do CORBA existe uma clara separação entre os clientes (requisitantes de serviços) e os servidores (provedores de serviços), através de uma *interface* de encapsulamento bem definida. Quanto aos clientes, o modelo de objetos é muito específico ao definir conceitos relevantes como criação e identificação de objetos, pedidos e operações, tipos e assinaturas. Quanto ao aspecto de implementação dos objetos, o modelo trata de conceitos como métodos, execução e ativação, porém de uma maneira mais sugestiva, dando máxima liberdade para que cada tecnologia possa implementar os objetos a sua maneira.

O modelo de objetos do CORBA é um modelo de objetos clássico, onde um cliente envia uma mensagem a um objeto, o objeto interpreta a mensagem e decide que serviço deve realizar. Como em um modelo clássico, uma mensagem identifica um objeto e zero ou mais parâmetros reais. O primeiro parâmetro é requerido e identifica a operação que deve ser realizada, e serve de base para que, durante a interpretação da mensagem, o objeto receptor (ou o ORB) possa selecionar o método adequado.

2. Object Request Broker

Qualquer objeto desenvolvido em conformidade com o padrão ORB deve garantir portabilidade e interoperabilidade de objetos sobre uma rede de sistemas heterogêneos. Basta que o ORB seja definido por suas *interfaces* e, então, qualquer implementação adequada às *interfaces* se torna aceitável. Tais *interfaces* são organizadas em 3 categorias:

- operações padrão para toda implementação de ORB;
- operações específicas a um tipo de objeto;
- operações específicas a um estilo de implementação de objetos.

Estrutura de um ORB

A figura D.1, mostra a estrutura de um ORB, cujos componentes serão detalhados a seguir:

- **ORB core:** parte do ORB que provê a representação básica dos objetos e comunicação de pedidos. Existem componentes acima do ORB Core que provêem interfaces para mascarar possíveis diferenças entre mecanismos de objetos para que estes possam coexistir;
- **Cliente:** um cliente de um objeto tem acesso à referência do objeto e chama operações sobre ele. O cliente de um objeto conhece a estrutura lógica do objeto, de acordo com a sua *interface*, e não sabe nada sobre a sua implementação, qual o adaptador de objetos usado ou qual o ORB usado para acessá-lo. É importante ressaltar que um cliente não é sempre um programa de aplicação: como exemplo, a implementação de um objeto pode ser cliente de outros objetos. Outro aspecto importante é a portabilidade: clientes devem ser capazes de trabalhar em qualquer ORB

que suporte o mapeamento de linguagem existente sem qualquer alteração no código fonte;

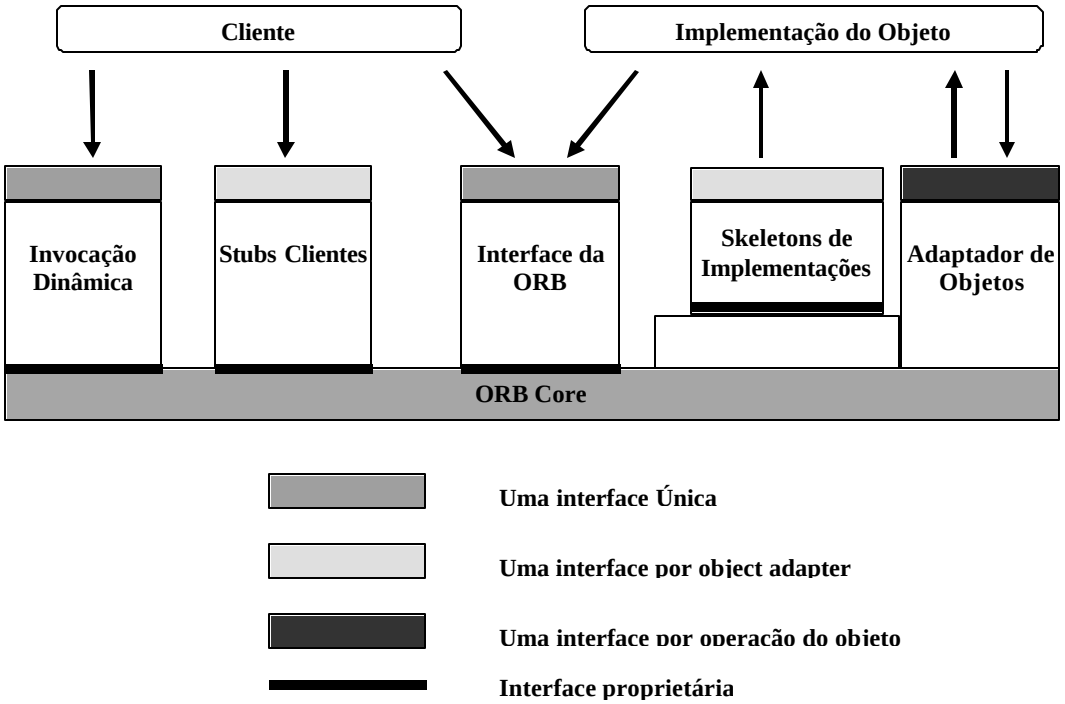


Figura D.1 : Estrutura de um ORB

- **Implementação de objetos:** Provê a semântica do objeto, definindo dados para a instância do objeto e código para os métodos. As diversas implementações de objetos que podem ser suportadas são: servidores separados, bibliotecas, um programa por método, aplicação encapsulada, SGBDOO, etc. Pode-se suportar diversos estilos de implementação com a definição de adaptadores de objetos adicionais. A portabilidade também é importante: implementações de objetos são portáteis entre quaisquer ORBs que suportem o mapeamento de linguagem adequado. A não dependência das implementações dos objetos em relação aos ORBs se dá através da existência dos Adaptadores de Objetos
- **Referências a objetos** Informação necessária para especificar um objeto em um ORB, que é dependente do mapeamento da linguagem e da implementação do ORB.
- **Esqueleto da implementação:** É através dele que o ORB faz as chamadas às rotinas existentes na *interface* para os métodos que implementam cada tipo de objeto. Porém, é possível escrever um adaptador de objetos que não se utiliza deste esqueleto para

invocar os métodos de implementação;

- **Adaptadores de objetos:** É um caminho básico pelo qual a implementação do objeto acessa os serviços do ORB, tais como: geração e implementação das referências para objetos, invocação de métodos, ativação e desativação dos objetos e suas implementações, registro das implementações, etc. Devido a grande diferença existente entre cada tipo de objeto (estilo de implementação, tempo de vida, políticas,...), com os adaptadores de objetos torna-se possível o ORB acessar um grupo de implementações que tenham requisitos semelhantes de uma maneira padrão, com *interfaces* comuns;
- ***Interface* ORB** É comum a todas as implementações de ORB, e contém operações (úteis tanto aos clientes quanto às implementações) que são comuns a todos os objetos, independentemente da interface do objeto ou do adaptador de objetos. Em função dessa independência, existem muito poucas operações disponíveis na interface ORB. Consiste na *interface* para as funções ORB que não dependem do *Object Adapter* utilizado. Essas operações são as mesmas para todos os ORB's e para todas as implementações de objetos e podem ser utilizadas por clientes dos objetos ou por suas implementações. As operações de criação de listas e determinação de objetos de contexto *default* referenciadas *Dynamic Invocation Interface* constituem também funções do ORB;
- **Repositório de implementações:** Contém informações necessárias para que o ORB localize e ative implementações.
- ***Stubs* de cliente:** Promovem acesso às operações definidas em IDL para um objeto de um modo que seja fácil para um programador que conheça IDL e o mapeamento para a linguagem de programação fazer previsões. As *stubs* fazem chamadas para o ORB utilizando-se de *interfaces* privadas ao núcleo do ORB que está sendo utilizado.
- **Invocação dinâmica de *Interfaces:*** Permite construção dinâmica para invocação de objetos. Com isso um cliente pode especificar, através de uma seqüência de chamadas, o objeto a ser invocado, a operação a ser executada e o conjunto de parâmetros para a operação. O próprio código do cliente pode obter do repositório de interfaces (ou de outra fonte em tempo de execução) informações sobre a operação e os parâmetros necessários, fornecendo-as para a invocação *Interface* para invocação dinâmica.

Permite criação e invocação dinâmica de pedidos à objetos, sendo capaz de distribuir qualquer pedido para qualquer objeto, através da interpretação em tempo de execução dos parâmetros e identificadores da operação. O resultado semântico obtido é o mesmo

que utilizando os *stubs* gerados em C, mas o número de chamadas que se tornam necessárias para executar uma operação é bem maior do que quando o pedido é "montado" em tempo de compilação.

Na invocação dinâmica os parâmetros são fornecidos como elementos de uma lista, serão checados em tempo de execução e devem ser fornecidos na mesma ordem em que foram definidos no Repositório de Interfaces *(Interface Repository).*

Repositório de interfaces

Trata-se de um serviço na estrutura de um ORB que provê objetos persistentes que representam a informação IDL em forma disponível em tempo de execução, e pode ser usado pelo ORB para execução dos pedidos, tornando possível que se encontre um objeto cuja *interface* é desconhecida, e determinando que operações são válidas para tais objetos. É o componente do ORB que possibilita a persistência das definições de *interface*, possibilitando distribuição e gerenciamento de uma coleção de objetos relacionados às *interfaces*.

Para que um ORB funcione corretamente ele precisa conhecer a definição dos objetos que ele "manuseia". Isso pode ser feito de duas formas :

- incorporando a informação proceduralmente nas rotinas embutidas;
- acessando dinamicamente o repositório de *interfaces*;

A definição das *interfaces* é mantida no repositório como um conjunto de objetos acessíveis através de um conjunto de definições de *interface* específicas em IDL. A definição de uma *interface* contém as operações que ela suporta, incluindo os tipos dos parâmetros, as exceções e a informação de contexto, se houver. Além disso, é armazenado nesse repositório valores de constantes e ***typecodes*** (que são valores que descrevem um tipo em termos estruturais). Este repositório é organizado em módulo, para facilitar a navegação por nome. Os módulos podem conter constantes, definição de tipos, exceções, definição de *interfaces* e outros módulos.

Um ORB pode ter acesso à vários repositórios de *interface*. A implementação de um

repositório de *interface* necessita de um mecanismo de persistência para os objetos. Normalmente o tipo de persistência utilizada irá determinar como às definições de *interface* serão distribuídas e/ou replicadas por um domínio de rede. Por exemplo, se for utilizado um sistema de arquivos existirá somente uma cópia do conjunto de *interfaces* mantida em somente uma máquina, por outro lado se for usado um banco de dados orientado à objetos, várias cópias podem ser mantidas distribuídas entre várias máquinas. Além disso, um mecanismo de segurança deve ser adotado para garantir o controle de acesso aos objetos.

3. Interface Definition Language (IDL)

IDL (Linguagem de definição de *Interfaces*) descreve as *interfaces* que são chamadas pelos clientes e fornecidas pelas implementações. Uma *interface* IDL provê a informação necessária para o desenvolvimento dos clientes que se utilizam das operações da *interface.* Os clientes não são escritos em IDL, mas sim em linguagens para a qual tenha sido definido mapeamento dos conceitos IDL. Ela define os tipos dos objetos, especificando suas *interfaces.* Por *interface* entende-se o conjunto de operações e parâmetros para tais operações. Pelas definições IDL, é possível mapear objetos CORBA em diferentes linguagens de programação

A linguagem C é a primeira linguagem para a qual foi estabelecido mapeamento IDL. IDL obedece as mesmas regras léxicas que a linguagem C++, com apenas algumas palavras reservadas a mais. A gramática IDL é um subconjunto da ANSI C++, com construções adicionais para suportar mecanismos de chamadas a operações. Por ser uma linguagem declarativa, IDL não possui nenhuma estrutura algorítmica ou variáveis. Para que se evitasse conflito de nomes da especificação CORBA com os da linguagem de programação, convencionou-se que os primeiros devem ser tratados como se tivessem definidos em um módulo denominado CORBA. Os nomes usados na *interface* devem ser referenciados, portanto, por seu nome completo (CORBA::<nome>).

4. Requisição e Tratamento de Pedidos

A figura D.2 mostra um pedido seguindo do cliente para a implementação do objeto. O cliente é a entidade que deseja realizar uma operação no objeto e a implementação do objeto é o código e os dados que realmente implementam o objeto.

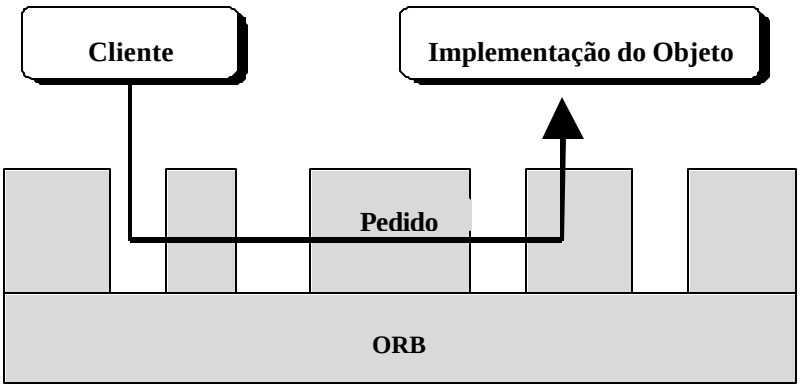


Figura D.2 : Pedido de Cliente via ORB

O ORB é responsável por todos os mecanismos necessários para procurar a implementação do objeto para o pedido, prepará-la para receber o pedido e fazer a comunicação dos dados para elaboração do pedido. A *interface* visível para o cliente é independente da sua localização, da linguagem de programação que o implementa ou de qualquer outro aspecto que não esteja descrito na *interface*.

A estrutura de um ORB é como descrita na figura D.1. Para fazer um pedido, o cliente pode utilizar a *interface* de invocação dinâmica (que é independente da *interface* do objeto alvo) ou uma *stub* IDL (específica para a *interface* do objeto alvo). A implementação do objeto recebe um pedido como uma chamada através do esqueleto gerado pela IDL (IDL *generated skeleton*). Durante o processamento de um pedido, a implementação do objeto pode chamar o Adaptador de Objetos e o ORB.

Para realizar um pedido, um cliente precisa ter acesso à referência do objeto e conhecimento sobre o tipo do objeto e da operação desejada. Feito isso, a inicialização do pedido se dá por chamadas a rotinas *stub* específicas ao objeto ou então pela construção dinâmica do pedido, sendo as duas abordagens transparentes ao receptor da mensagem.

Quanto ao atendimento do pedido, o ORB localiza o código de implementação do objeto, transmite parâmetros e transfere o controle para a implementação do objeto através do esqueleto da IDL. Durante a execução do pedido, a implementação do objeto pode requerer serviços do ORB através do adaptador de objetos. Ao final do atendimento, retornam-se o controle e os valores de saída para o cliente.

Estrutura de um cliente

O cliente de um objeto possui uma referência para o objeto. Essa referência é um *token* que pode ser invocado ou passado como parâmetro na chamada de um objeto diferente. A unidade que gerencia a transferência de controle e de dados entre os clientes e as implementações é o ORB. No caso de uma operação não ser executada com sucesso, gera-se uma exceção que deve ser tratada pelo cliente.

A invocação de uma operação é feita da seguinte maneira: o cliente acessa as *stubs* específicas ao tipo do objeto, as quais têm acesso à referência para o objeto, implementadas em uma linguagem de programação, que seguem como parâmetro para o ORB realizar a operação.

A referência para objetos pode ser convertida para uma *string*, armazenada em arquivo, preservada ou comunicada por diferentes meios, e depois transformada de volta em uma referência pelo ORB que gerou a *string*

Estrutura da implementação do objeto

A implementação de objetos define o comportamento de um objeto, procedimentos para ativar e desativar objetos, e se utiliza de outras facilidades para tornar um objeto persistente e controlar seu acesso.

Existe uma interação entre a implementação do objeto e o ORB, implementada via o adaptador de objetos, com vários objetivos: estabelecer a identidade do objeto, criar novos objetos, obtenção de serviços dependentes do ORB.

Quando ocorre um pedido, o núcleo do ORB, juntamente com o adaptador de objetos e

o esqueleto da *interface*, realizam uma chamada ao método apropriado para atender o pedido, fornecendo os parâmetros necessários.

Quando da criação de um objeto, o ORB é notificado para que saiba onde se encontra a implementação do novo objeto.

5. Estrutura do Adaptador de Objetos

O adaptador de objetos é o meio básico para que uma implementação de objetos possa acessar os serviços do ORB (como por exemplo geração de referências para objetos). Existe uma *interface* pública que é exportada para as implementações dos objetos, e uma privada que fica disponível para o esqueleto. Algumas das funções disponíveis nos adaptadores de objetos, que são executadas usando o núcleo do ORB, são:

- geração e interpretação de referências para objetos;
- invocação de métodos;
- segurança nas interações;
- ativação e desativação de objetos e suas implementações;
- mapeamento das referências para objetos para as correspondentes implementações;
- registro de implementações;

Os adaptadores de objetos estão implicitamente envolvidos nas chamadas aos métodos (serviços como autenticação), embora a *interface* direta seja através dos esqueletos da IDL.

Exemplos de adaptadores de objetos

Os adaptadores de objetos são responsáveis por definir a maioria dos serviços do ORB dos quais a implementação dos objetos podem depender. Com eles, é possível que a implementação dos objetos tenha acesso a um serviço que pode não estar implementado pelo núcleo do ORB (caso esteja, o adaptador fornece apenas uma interface para ele, senão o adaptador deve implementá-lo sobre o núcleo do ORB).

- **Básico**: implementações são geralmente programas separados. Pode haver um programa por objeto, ou um programa compartilhado por todas as instâncias de um tipo de objeto. Existe apenas uma pequena quantidade de armazenamento persistente para cada objeto;
- **Biblioteca**: usado no caso de objetos com implementações em biblioteca. Os dados persistentes estão em arquivos, e não existem os mecanismos de ativação e autenticação;
- **Orientado a objetos**: usa uma conexão a um banco de dados orientado a objetos para acessar seus objetos. Não é necessário que se guarde nenhum estado no adaptador de objetos e, além disso, os objetos são registrados implicitamente no ORB, já que o próprio banco de dados é responsável por armazenar dados e métodos dos objetos;

O adaptador de objetos básico

A *interface Object Adapter* é a principal utilizada pelas implementações para ter acesso às funções do ORB. O ***Basic Object Adapter*** (BOA) é a *interface* que pretende ser mais disponibilizada e que suporte o maior número de implementações de objetos. Ela inclui *interfaces* para gerar referências à objetos, registro de implementações formadas por um ou mais programas, ativação de implementações e autenticação de pedidos. além disso ainda fornece um armazenamento persistente limitado mas que pode ser utilizado em conjunto com outro maior.

A maior parte da *interface* do BOA pode ser expressa em IDL, desde que seja para operações no *Object Adapter.* Toda implementação de ORB deve possuir um ***Basic Object Adapter***. Apesar de sua implementação ser dependente do ORB, deve ser

possível à uma implementação de um objeto que usa o BOA rodar em qualquer outro ORB que suporte o mapeamento de linguagem usado na implementação das operações.

O BOA utiliza funções do sistema operacional para ativar e comunicar com os programas que implementam um objeto. Com isso obtemos um certo grau de não portabilidade do BOA, pois ele necessita de informações que não são comuns à todos os sistemas. Para resolver este problema, definiu-se o conceito de repositório de implementação que irá armazenar essas informações permitindo assim que cada sistema instale e inicia suas aplicações de acordo com seu sistema.

A forma de conexão do BOA com o ORB, bem como do *skeleton* (parte do BOA responsável pela execução dos métodos) não foi especificado por ser dependente do mapeamento de linguagem utilizado.

Entre as funções executadas pelo BOA podemos citar ativação e desativação de implementações: A ativação de implementação ocorre quando não existe nenhuma implementação de um objeto disponível para lidar com o pedido feito. Para que isso ocorre é preciso uma coordenação entre o BOA e os programas que contém a implementação.

O BOA não adota nenhum estilo específico de gerenciamento de segurança. Ele garante que para todo método ele irá identificar o "principal" sobre o qual o pedido foi feito. O significado do "principal" depende do ambiente de segurança sobre o qual a implementação está rodando. A decisão de permitir ou não a execução de uma operação é da implementação, que normalmente associa direitos de acesso com objetos "principais" e examina se eles podem executar a operação.

A persistência dos objetos é obtida em conjunto pelo BOA e o ORB Core, possibilitando assim um cliente acessar um objeto à qualquer momento, mesmo que a implementação tenha sido desativada ou o sistema reinicializado.