Universidade Federal de Campina Grande
Centro de Ciências e Tecnologia
Coordenação de Pós-Graduação em Informática

Dissertação de Mestrado

Integrando Agentes Móveis com Sistemas Legados para Gerenciamento de Redes ATM Heterogêneas

André Ribeiro Cardoso

Campina Grande - PB, agosto de 2002

Universidade Federal de Campina Grande
Centro de Ciências e Tecnologia
Coordenação de Pós-Graduação em Informática

André Ribeiro Cardoso

Integrando Agentes Móveis com Sistemas Legados para Gerenciamento de Redes ATM Heterogêneas

Dissertação submetida à Coordenação de Pós-Graduação em Informática do Centro de Ciências e Tecnologia da Universidade Federal de Campina Grande, como parte dos requisitos para obtenção do título de Mestre em Informática.

Área de Concentração: Ciência da Computação
Sub-Área: Redes de Computadores

Orientador: Prof. Dr. Joaquim Celestino Júnior

Campina Grande - PB, agosto de 2002

Ficha Catalográfica

CARDOSO, André Ribeiro
C268I

Integrando Agentes Móveis com Sistemas Legados para Gerenciamento de Redes ATM Heterogêneas

Dissertação de Mestrado – Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande – PB, agosto de 2002

124 p. Il.

Orientador: Joaquim Celestino Júnior

1. Redes de Computadores
2. Gerenciamento de Redes
3. Agentes Móveis
4. Sistemas Legados

CDU 621.391

"INTEGRANDO AGENTES MÓVEIS COM SISTEMAS LEGADOS PARA GERENCIAMENTO DE REDES ATM HETEROGÊNEAS"

ANDRÉ RIBEIRO CARDOSO

DISSERTAÇÃO APROVADA EM 02.08.2002

PROF. JOAQUIM CELESTINO JÚNIOR, Dr.
Orientador

PROF. ELMAR UWE KURT MELCHER, Dr.
Examinador

PROF. ANILTON SALLES GARCIA, Dr.
Examinador

CAMPINA GRANDE – PB

A meus filhos Gabriel e Juliana
a minha esposa Georgiana
a meus irmãos Paulo, João e Mirtinha, Daniel e Aléxia
a meus pais
e a minha tia Violante

Agradecimentos

A Dra. Dirsa Nogueira, pelos ensinamentos de vida tão sabiamente transmitidos.

Ao Professor Maurício, Diretor do Departamento de Informática da UECE, pelo importante apoio oferecido durante todo o desenvolvimento deste trabalho.

A todos do DI, Paulo Marcelo, Abrahão, Lapa, Ladislav, Ivonildo, Odaelza, Odenize, Elcides e Otacílio, pela colaboração e compreensão dispensadas.

A Georgiana, minha querida esposa e grande companheira, pelo incentivo, dedicação e carinho ao longo deste trabalho. Não posso esquecer de agradecê-la pela companhia e apoio nas viagens de apresentação das publicações oriundas desta dissertação.

Aos meus sogros, Dr. Giordano e Dona Georgina, que desde o início de tudo confiaram a felicidade de sua filha, tão especial, a mim.

Ao Desembargador Nogueira Sales e Dona Prenda, e a Dona Marina (*in memorian*), avós de minha esposa, que sempre acreditaram no meu sucesso.

Aos meus tios, em especial o tio Evandro, sempre apostando nas minhas conquistas.

Ao Pedro Celso, meu grande amigo, pelo fiel companheirismo e lições de equilíbrio.

Ao Celestino, meu orientador, pela constante presença, visão de pesquisa e de vida, agilidade nas soluções de problemas (que problemas...) e incrível espírito cooperativo, característica marcante de sua personalidade. Discussões ocorreram. Porém, todas responsáveis pelo engrandecimento deste trabalho. E, após todo nosso empenho, recuperei meu mestrado, fui encaminhado ao doutorado e ganhei um amigo. Um grande amigo.

Finalmente, agradeço a DEUS, por me proporcionar oportunidade e sabedoria para concluir, com sucesso, meu mestrado.

Resumo

Esta dissertação tem como objetivo apresentar proposta de integração da tecnologia de agentes móveis com sistemas legados de gerenciamento de redes, como solução aos seguintes problemas por nós identificados: aumento de tráfego nas conexões da arquitetura "cliente/servidor" resultante do crescente número de equipamentos conectados na rede; centralização dos principais sistemas de gerenciamento de redes existentes e; finalmente, desafio de gerência de redes heterogêneas ATM. O sistema Concordia, criado pela *Mitsubishi Electric Information Technology Center America*, foi utilizado como plataforma de desenvolvimento e execução de agentes móveis e SNMP, proposto pela IETF, o sistema legado empregado para gerenciamento de redes de dados. Concordia foi escolhido, porque, escrito completamente em Java, proporciona portabilidade e execução de seus componentes em todos os lugares, agregando segurança, mobilidade, gerência e monitoramento de todo o sistema. SNMP, tendo como simplicidade de projeto sua principal vantagem, foi empregado devido ao seu amplo uso e suporte pela maioria de fabricantes de equipamentos, tornando-o indicado para execução em ambientes multiplataformas. Bibliotecas de classe Java do pacote AdventNet API foram usadas para habilitar agentes móveis Concordia a realizar comandos GETs e SETs do SNMP e, portanto, acessarem informações de gerenciamento. Este trabalho está incluído no Projeto AMI "Uma Plataforma Distribuída CORBA com Agentes Móveis Inteligentes para Gerenciamento de Redes e Serviços", do Laboratório de Redes de Computadores e Sistemas de Software – LARCES da Universidade Estadual do Ceará – UECE. O Projeto AMI propõe, num primeiro momento, ambiente com implementação através de plataformas já existentes, como é o caso do Concordia aqui empregado, e num segundo momento, construção de plataforma para execução de agentes móveis desenvolvidos no próprio LARCES/UECE. Inicialmente, seguindo o Projeto AMI, foram implementados agentes móveis Concordia para gerenciar PVCs entre dispositivos pertencentes a Rede Metropolitana de Alta Velocidade de Fortaleza (REMAV-For) e, posteriormente, de outras REMAVs do Brasil.

Abstract

This thesis presents the integration of mobile agent technology with network management legacy systems as a proposal for the solution of the following problems: the overload of information in the bandwidth because of the continuous increase of the servers connected to the network; centralization aspects of the main network management legacy systems and finally, the challenge of ATM heterogeneous network management. Concordia System, created by Mitsubishi Electric Information Technology Center America, was employed as a development platform for mobile agents. SNMP from IETF was used as a management legacy system for data network. Concordia was chosen because, written completely in Java, provides portability and allows the execution of its agents anywhere and at anytime with security, mobility, management and monitoring – all the necessary characteristics to achieve the perfect environment for the mobile agents. SNMP that has the simplicity of the project as its main advantage was chosen because of its wide use and support by great part of equipment manufactures. Java class libraries from AdventNet API package were used to allow mobile agents to execute GET and SET SNMP commands and therefore access management information. This thesis is part of the AMI Project "A CORBA Distributed Platform with Intelligent Mobile Agents to Manage Networks and Services" of Laboratory of Computer Network and Software Engeneering – LARCES of State University of Ceara – UECE. First, AMI Project proposes the implementation of an environment throughout platforms that already exist such as Concordia, and second the development of its own platform for mobile agents developed at LARCES/UECE. Initially, based on AMI Project, Concordia mobile agents were developed to manage PVCs among devices that belong to "High Speed Metropolitan Network of Fortaleza (REMAV-For)" and, afterward, to other REMAVs throughout Brazil.

Sumário

Índice de Figuras

Índice de Tabelas

Lista de Acrônimos

AAL	ATM Adaptaion Layer
AdmMg	Adminitration Manager
AM	Agente Móvel
AMg	Agent Manager
ANSI	American National Standards Institute
AOM&P	Operation, Administration, Maintenance and Provisioning
API	Application Program Interface
ASN.1	Abstract Syntax Notation One
AT API	Agent Transport API
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network
CBR	Constant Bit Rate
CES	Circuit Emulation Service
CMIP	Common Management Information Protocol
CR	Componente de Rede
DMg	Directory Manager
DPI	Distributed Protocol Interface
EMg	Event Manager
FCAPS	Fault, Configuration, Acounts, Performance and Security
FIPA	Foundation for Intelligent Physical Agents
GC PVC	Gerente de Configuração de PVC
IACMg	Inter-Agent Communication Manager
IDL	Interface Description Language
IETF	International Engineering Task Force
ILMI	Integrated Local Management Interface
IME	Integrated Management Entity
IP	Internet Protocol

ISO	International Standard Organization
ITU	Internation Telecommunication Union
JVM	Java Virtual Machine
LAN	Local Area Network
MASIF	Mobile Agent System Interoperability Facilities
MIB	Management Ingormation Base
NE	Network Element
NMS	Network Management System
OO	Orientado a Objeto
OSI	Open System Interconnection
PDH	Plesiochronous Digital Hierarchy
PDU	Protocol Data Unit
PLCP	Physical Layer Convergence Process
PMg	Persistence Manager
PNNI	Private Network-Network Interface
PVC	Permanent Virtual Connection
QMg	Queue Manager
QoS	Quality of Service
RAdm APIq	Remote Administration API
RDPI	Reverse Distributed Protocol Interface
REMAV	Rede Metropolitada de Alta Velocidade
RP	Remote Programming
RPC	Remote Procedure Control
SB API	Service Bridge API
SC	Servidor Concordia
SGMP	Simple Gateway Monitoring Protocol
SMI	Structure of Management Information
SMg	Security Manager
SLA	Service Level Agreement
SVC	Switched Virtual Connection
SNMP	Simple Network Management Protocol
TCP	Transport Control Protocol
TMN	Telecommunication Management Network

UDP	User Datagram Protocol
UNI	User Network Interface
UPC	Usage Parameter Control
VBR	Variable Bit Rate
VC	Virtual Channel
VCC	Virtual Channel Connection
VCI	Virtual Channel Identification
VCL	Virtual Channel Link
VL	Virtual Link
VP	Virtual Path
VPC	Virtual Path Connection
VPI	Virtual Path Identification
VPL	Virtual Path Link
WAN	Wide Area Network

Capítulo 1. Introdução

1.1. Introdução

Nos últimos anos a Internet e, em particular, a World Wide Web (WWW) ou, simplesmente Web, tem se tornado a mídia preferida para a transmissão e disseminação de informações, acesso a dados e comunicação pessoal. É crescente o número de novas aplicações distribuídas e multi-plataforma, como comércio eletrônico, educação a distância etc. ocupando um espaço cada vez mais importante na Internet.

O número de informações disponível pelo mundo, o custo de movimentação de dados e a pequena experiência em computação de muitos usuários são motivos de preocupação para a comunidade de redes e para os implementadores de aplicativos. A capacidade do usuário de procurar, selecionar e recuperar essas informações na rede, não está acompanhando o crescimento do volume de dados disponibilizados nos vários *sites*. Acrescido a isso, temos uma enorme movimentação de registros através da rede, a um custo significativo, sendo, cada vez mais, acessado por pessoas com pouca experiência em computação.

Tecnologias tais como Modo de Transferência Assíncrono (*Asychronous Transfer Mode – ATM*) foram apresentadas no sentido de incrementar velocidade de transmissão de informações, garantir qualidade de serviços (QoS) etc. O estabelecimento de Conexões Virtuais Permanentes (*Permanent Virtual Connections– PVCs*), vastamente empregadas nesta tecnologia, é uma tarefa importante, porém complexa, dada às peculiaridades de cada fabricante de dispositivos ATM.

1.2. Motivação

Com o aumento de servidores ligados na rede, a arquitetura "cliente/servidor", utilizada para conectar os computadores, se torna ineficiente e carente de mudanças. O uso de soluções paliativas, apenas transfere a problemática do congestionamento de informações para a rede. Neste trabalho, identificamos problemas relacionados ao comportamento do paradigma "cliente/servidor", ao gerenciamento de redes e, finalmente, à tarefa de gerenciamento de PVCs em redes ATM.

Relacionado ao paradigma "cliente/servidor", o número de aplicações ocupando espaço importante na Internet tem crescido substancialmente. Portanto, esta arquitetura tornou-se gargalo do volume do tráfego de informações causado pelo "*request/response*" característico deste modelo. Torna-se essencial e urgente o desenvolvimento de novas tecnologias de operação e gerência de conexões entre os diversos nós da rede. Como solução para estes problemas, utilizam-se programas agentes, ajudando o usuário a realizar suas tarefas. Se esses agentes executarem suas funções em diferentes computadores, serão denominados de agentes móveis. Agentes móveis estão sendo propostos para superar as limitações inerentes da arquitetura "cliente/servidor" e, juntamente com eles, diversas plataformas para seu desenvolvimento. Estes agentes podem mover-se para o lugar onde os dados estão armazenados e, utilizando inteligência, selecionar e processar informações necessárias ao usuário, economizando largura de banda, tempo e dinheiro.

Com relação ao gerenciamento de redes, atualmente, a grande maioria de aplicações de gerência é baseada em um dos seguintes protocolos, para resolver o problema de interoperabilidade em ambientes heterogêneos: *Simple Network Management Protocol (SNMP)* [1], nas redes de dados, ou, *Common Management Information Protocol (CMIP)* [2], nas redes de telecomunicações. Infelizmente, estes protocolos são baseados em soluções estáticas e centralizadas do modelo "cliente/servidor", onde, cada elemento da rede envia todos os dados a uma estação central, que os processa e provê interface com o

operador. Por isso, aplicações de gerenciamento apresentam problemas de escalabilidade e produzem muito tráfego na rede.

Finalmente, sobre o problema de gerenciamento de PVC, sabemos que PVC é considerada uma das mais importantes tarefas de gerenciamento de redes ATM. Não existe padrão nem uniformidade para o operador criar PVC numa rede heterogênea. Por exemplo, criar uma conexão fim-a-fim (PVC) entre diferentes locais, requer a configuração de todos os comutadores (*switches*) intermediários ao longo da rede. Tal operação envolve a criação de conexões cruzadas (*cross-connects*) em cada comutador. Porque comutadores são oriundos de diferentes fabricantes, cada um com interface própria de gerenciamento, é grande o incômodo de criar tal PVC. Portanto, é notável tratar-se de uma operação complexa necessitando da criação de ferramentas para torná-la tarefa fácil, isenta de complicações.

Essa dissertação tem como objetivo apresentar proposta de integração da tecnologia de agentes móveis com sistemas legados de gerenciamento de redes, característicos da tecnologia "cliente/servidor", como solução aos problemas enumerados nos parágrafos anteriores. O sistema Concordia [3], criado pela *Mitsubishi Electric Information Technology Center America*, foi a plataforma de desenvolvimento e execução dos agentes móveis empregada e SNMP, empregado para gerenciamento de redes de dados, o sistema legado considerado. Para integrar essas duas tecnologias, classes do AdventNet [4] foram incluídas nos agentes móveis Concordia, tornando-os aptos a acessarem informações de gerenciamento SNMP.

1.3. Trabalhos Relacionados

SNMP não é diretamente suportado pelas plataformas comerciais de desenvolvimento e execução de agentes móveis. Entretanto, há diversas pesquisas buscando essa integração. Os trabalhos relacionados abaixo tratam da interoperabilidade de agentes móveis com as

soluções centralizadas existentes, baseadas na arquitetura "cliente/servidor", de gerenciamento de redes. Algumas delas criam aplicações de estabelecimento de PVCs de forma heterogênea. A integração de SNMP com agentes móveis é fortemente considerada em nossa pesquisa.

Mobile AGENT environment for distributed Applications (MAGENTA) [5] é uma proposta de gerenciamento de rede da IRISA. MAGENTA introduz o conceito de um único Gerente de Rede Móvel (*Mobile Network Manager – MNM*) com o fim de prover a figura do gerente de redes independente de localização e sistema. Assim, a parte de aplicação de gerenciamento do clássico Sistema de Gerenciamento de Redes (*Network Management System - NMS*) é separado da parte do *kernel*, oferecendo a possibilidade de existirem múltiplos gerentes simultaneamente e, portanto, a distribuição dos sistemas de gerenciamento de redes. MAGENTA, escrito em Java, é composto de *lieus* e *agents*. *Lieu* dá suporte à recepção, execução, armazenamento e movimento dos agentes MAGENTA (*agents*), bem como serviços de manutenção da informação sobre outros *lieus* e agentes MAGENTA (*agents*) no ambiente. Cada l*ieu* possui único nome global e pode ser criado e apagado dinamicamente, acompanhando o surgimento/finalização de novos gerentes de redes. Agentes MAGENTA (*agents*) são programas com capacidade de se moverem entre *lieus*, seguindo itinerário e executando tarefas predefinidas. Segundo o projeto MAGENTA, agentes móveis são usados não apenas para facilitar o modo de funcionamento desconectado, mas também procurando descentralizar as funcionalidades do gerenciamento de redes.

Intelligent Network Control Architecture (INCA) [6], do *C&C Research Laboratories – NEC Europe Ltd*, propõe uma arquitetura aberta para gerenciamento distribuído de redes e sistemas de aplicação. Agentes estacionários e móveis são empregados para executar monitoramento e controle de rede e componentes dos sistemas, suportando gerenciamento integrado de redes e serviços. A arquitetura provê capacidade de transação para controlar transporte e mobilidade de agentes, priorização de agentes e esquema de transferência de código de múltiplos agentes. A priorização de agentes é necessária para suportar tarefas críticas em tempo de execução. Objetos gerenciados são

usados para acesso de recursos dos elementos de rede e novas funcionalidades do sistema podem ser criadas, distribuídas e substituídas dinamicamente. Porém, a plataforma de gerenciamento de redes, baseada no projeto INCA, não está ligada a nenhum ambiente de computação ou protocolo existente, o que significa não haver integração com SNMP e CMIP.

Asynchronous Message Transfer Agent System (AMETAS) [7], do *Dept. of Computer Science – Frankfurt/Main Germany*, apresenta a integração dos dois paradigmas de gerenciamento de redes: protocolos de gerenciamento legados, tais como SNMP, com gerenciamento descentralizado de redes usando agentes móveis. A idéia principal está na forma da coleta de dados SNMP, que é feita localmente, em vez de remotamente. O grande número da troca de mensagens na rede entre gerentes e agentes SNMP é substituído pela migração de agentes móveis AMETAS, habilitados ao SNMP, executando tarefas de gerência localmente. Assim, a carga produzida pelas aplicações de gerenciamento é largamente reduzida. Baseado no AMETAS, o sistema NetDoctor [8] permite o monitoramento de computadores em redes heterogêneas. NetDoctor apresenta agentes móveis que consultam agentes SNMP, processam dados de status e executam ações corretivas. Nossa pesquisa explora a interação local de agentes móveis Concordia com dados do SNMP. Para isso AMs Concordia são habilitados ao SNMP através da importação de classes Java [4] com este propósito.

Distributed Intelligent Agents for Network Administration (DIANA) [9] apresenta o foco no projeto de agentes inteligentes habilitados a incrementar seus comportamentos através da aquisição de habilidades. A arquitetura do agente é dividida em dois grandes tipos de componentes: a Inteligência (*Brain*) oferece as facilidades necessárias para operação do agente e as Habilidades (S*kills*) provêem os agentes com capacidades e comportamentos. Agentes estão aptos a adquirir novas habilidades (skills) sem interrupção de suas operações. Apresentam-se hierarquicamente organizados podendo compartilhar e/ou delegar atividades e responsabilidades. Cada agente possui um domínio de gerenciamento de rede, o qual representa um conjunto de elementos de rede com atribuições de uma gerência específica. Assim, os domínios dependem das suas tarefas de

segurança, gerenciamento de falhas etc. Embora não apresente preocupações com mobilidade de agentes, DIANA apresenta interessante trabalho de provimento automático de PVCs para redes heterogêneas ATM em [10].

No projeto *Perpetuum Mobile Procura PMP* [11] [12], da Universidade de Carleton - CA, desenvolveu-se uma infra-estrutura com ambiente capaz de prover mobilidade do código. Em [13], encontra-se pesquisa sobre integração de agentes móveis com SNMP na plataforma PMP. A integração é realizada através da Interface de Protocolo Distribuído (*Distributed Protocol Interface - DPI*) [14] e da Interface de Protocolo Distribuído Reverso (*Reverse Distributed Protocol Interface – RDPI*). SNMP DPI é uma extensão dinâmica para agentes SNMP, a qual permite adição, exclusão e alteração dinâmica de variáveis da MIB, sem necessidade de re-compilação dos agentes. O protocolo RDPI dá permissão a processos, normalmente executando no mesmo nó do agente, a lançarem comandos de gerência SNMP de forma mais leve. A idéia é capacitar os agentes móveis a executarem um conjunto completo de funções de gerenciamento, bem como prover novas informações para a estação central de gerenciamento de redes. Em [15] [16], encontram-se implementações de metodologias de configuração de PVCs para redes ATM heterogêneas em ambientes definidos pelo projeto PMP. Nossa pesquisa seguiu as principais idéias desse projeto, onde agentes móveis são capacitados a interagir, localmente, com agentes SNMP. As metodologias de estabelecimento de conexões fim-a-fim foram adotadas como referência.

Distributed Proactive Self-Adjusting Management (DPSAM) [17] é um ambiente genérico facilitador da incorporação de agentes inteligentes, computação distribuída e tecnologias baseadas na *Web* a fim de desenvolver sistemas de gerenciamento de redes. *PVC Management System (PMS)* [18] é baseado no DPSAM e trata do estabelecimento de PVCs. Possui uma arquitetura de três níveis e utiliza CORBA como *midleware* distribuído. Sua arquitetura também emprega protocolos padrões de gerenciamento tais como SNMPv1, SNMPv2, CMIP. Embora PMS não seja usuário da tecnologia de agentes móveis, ele representa trabalho interessante para nossa pesquisa por tratar de aplicação escalável e heterogênea para criação de PVCs em redes ATM.

A plataforma JAMES [19] propõe ambiente de desenvolvimento e execução de agentes móveis. Apresenta importantes aspectos de migração de código, tolerância a falhas, flexibilidade de distribuição de código com facilidade de atualização, mecanismos para controle de recursos, persistência necessária para operações desconectadas, portabilidade e, principalmente, interoperabilidade com tecnologias legadas SNMP existentes. Tem como principal alvo o gerenciamento de redes e aplicações de telecomunicações. JAMES adota implementação própria de SNMP, baseada em Java, integrada à sua plataforma. Com isso, ela provê interoperabilidade necessária entre a tecnologia de agentes móveis e dispositivos/aplicações usuárias de SNMP.

O projeto AMI "Uma Plataforma Distribuída CORBA com Agentes Móveis Inteligentes para Gerenciamento de Serviços" tem como principal objetivo explorar a tecnologia dos agentes móveis em aplicações ligadas ao gerenciamento de redes e serviços, propondo, num primeiro momento, um ambiente com implementação através de tecnologias já em uso, como é o caso do Concordia aqui utilizado e, num segundo momento, com a construção de uma plataforma para execução de agentes móveis desenvolvidos no próprio LARCES/UECE. A arquitetura AMI está dividida em cinco níveis (Figura 1.1):

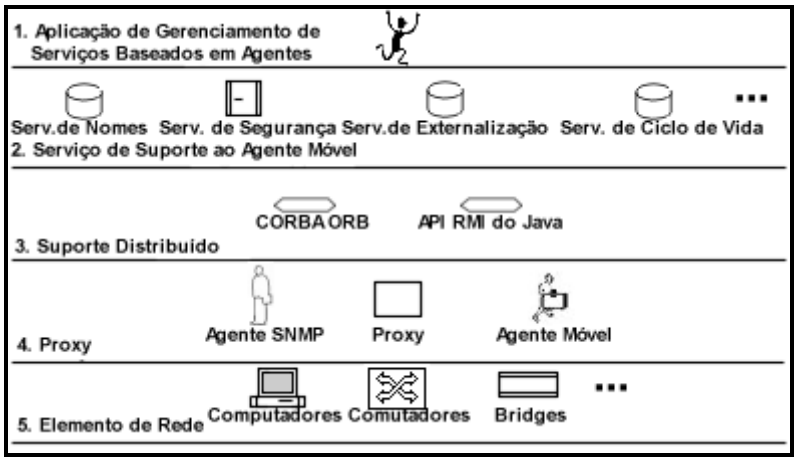


Figura 1.1. Níveis da Arquitetura AMI

- Nível de Aplicação de Gerenciamento de Serviços Baseados em Agentes;
- Nível de Serviço de Suporte ao Agente;

- Nível de Suporte Distribuído: CORBA, API RMI do Java;
- Nível *Proxy*;
- Nível de Elemento de Rede.

Inicialmente, seguindo o projeto AMI, criamos agentes móveis Concordia para configurar PVCs entre dispositivos da REMAV-For e, posteriormente, de outras REMAVs do Brasil.

1.4. Organização desta Dissertação

Nosso trabalho está assim dividido: Capítulo 2 apresenta *background* de ATM, enfatizando terminologias necessárias para o entendimento desta dissertação. É dada especial atenção ao estabelecimento de conexões virtuais dentro de redes ATM. Capítulo 3 aborda o gerenciamento de redes ATM e seus desafios. Capítulo 4 descreve sumariamente SNMP, voltado para gerência ATM. Por isso, enfatizamos a versão 2 do SNMP, suas tabelas conceituais e focalizamos na MIB AToM, responsável pelo gerenciamento de PVCs. Capítulo 5 introduz a tecnologia de agentes móveis, descreve seus principais conceitos, relaciona suas principais vantagens e finalmente, apresenta proposta de padronização MASIF. Capítulo 6 trata do sistema Concordia, utilizado para desenvolvimento dos agentes móveis deste trabalho, abordando seus conceitos, técnicas, vantagens e cuidados requeridos para sua execução. Nossa proposta de integração entre agentes móveis e SNMP é apresentada no capítulo 7. Para isso, é mostrado o resultado do estudo de um caso prático onde agentes móveis Concordia, desenvolvidos no LARCES / UECE, configuram PVCs de dispositivos ATM ao longo da rede REMAV.

Capítulo 2. Modo de Transferência Assíncrono (*Asychronous Transfer Mode - ATM*)

2.1. Introdução

ATM surgiu como grande e promissora tecnologia de rede devido a sua velocidade, escalabilidade, flexibilidade e garantia de qualidade de serviços (QoS). ATM oferece boa combinação de técnica e chaveamento de pacotes.

ATM tem sido escolhida como tecnologia para Banda de Freqüência Larga da Rede de Serviços Digitais Integrados (*Broadband Integrated Services Digital Network - B-ISDN*). A padronização B-ISDN foi definida pelo Setor de Padronização de Telecomunicações da União Internacional de Telecomunicações (*International Telecommunication Union - ITU-T*).

ATM é uma tecnologia que opera no modo de chaveamento de pacotes orientados a conexão. Como X.25, ATM utiliza conexões virtuais para transportar dados do usuário ao longo da rede. O Canal Virtual (*Virtual Channel - VC*), em ATM, é similar ao circuito virtual em X.25. A diferença está no uso de tecnologia de conexões virtuais de dois níveis pelo ATM, onde é introduzido o nível Caminho Virtual (*Virtual Path - VP*). Outra diferença chave está na utilização, pelo X.25, de pacotes de tamanho variável, enquanto ATM utiliza células fixas de 53 bytes, permitindo maior eficiência no chaveamento.

Há dois tipos de mecanismos de estabelecimento da conexão: Conexão Virtual Chaveada (*Switched Virtual Connection - SVC*) e Conexão Virtual Permanente (*Permanent*

Virtual Connection - PVC). O primeiro, estabelecido automaticamente, é baseado nos protocolos de sinalização Interface Rede-Usuário (*User Network Interface - UNI*) e Interface Rede-a-Rede Privada (*Private Network-Network Interface - PNNI*) [20]. O segundo, adotado neste trabalho, é pré-estabelecido pelo operador em cada dispositivo ao longo da rede. PVCs são indicados em diversas situações. Primeiro, existe a limitação onde SVCs não podem ser oferecidos com todos os parâmetros de tráfego padrões pela maioria dos comutadores. Segundo, por serem permanentes, PVCs garantem continuidade de QoS e disponibilidade imediata de utilização da conexão. Terceiro, PVCs são muito eficientes em conexões onde a comunicação entre dispositivos ATM é feita com freqüência e, portanto, não há o gasto de tempo com estabelecimentos de conexão. Finalmente, um dispositivo ATM pode não suportar completa implementação do protocolo de sinalização ATM. Implementações correntes de mecanismos de sinalização não são completamente compatíveis entre dispositivos de diferentes fabricantes [21]. Portanto, SVCs não podem ser estabelecidos em redes ATM heterogêneas, ficando como alternativa o uso de PVCs.

2.2. Comutações de Células ATM

O processo de comutação ATM trata da recepção de células em uma porta, e sua posterior transmissão em outra, mantendo a ordem em cada conexão. O dispositivo responsável por este processo é denominado comutador. Abaixo, seguiremos com os detalhes das operações de comutação de células ATM, necessárias ao entendimento do nosso trabalho.

O conceito de VP surgiu nas redes de alta velocidade, como resposta ao crescimento da relevância do controle de custos da rede. Agrupando conexões que compartilham o mesmo caminho através da rede, aplicações de gerenciamento podem tratar com menor número de grupos de conexões, evitando grande quantidade de conexões individuais. Isso resulta no aumento de desempenho e confiabilidade da rede e, em alguns casos, os tempos de conexão e de processamento podem ser reduzidos. Ao reservar uma Conexão VP (*VP Connection - VPC*), novas Conexões VC (*VC Connection - VCC*) podem

ser estabelecidas através da execução de simples funções de controle nos sistemas finais do VPC; nenhum processo é requerido nos sistemas intermediários.

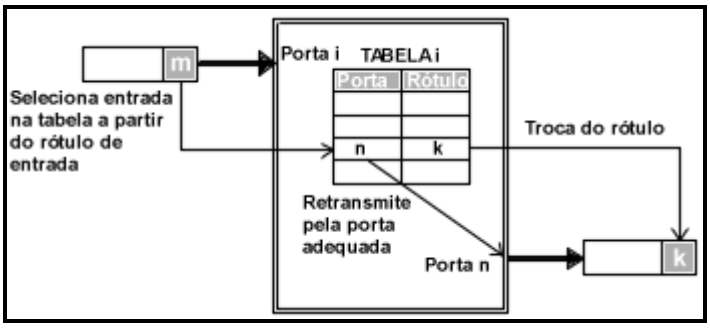


Figura 2.1. Comutação Através do Rótulo

Ao chegar no comutador, o Enlace de Conexão Virtual (*Virtual Channel Link - VCL*) da célula ATM é identificado e, de posse da porta de entrada, o comutador consulta uma tabela contendo a relação de cada VCL e porta de entrada para o próximo VCL e respectiva porta de saída, a ser utilizada no caminho estabelecido pelo VCC. O rótulo da célula é atualizado pelo comutador, que a retransmite pela porta de saída especificada na tabela. (Figura 2.1) [22].

Uma VCC consiste da concatenação de VCLs identificadas pelos rótulos e portas contidos nas tabelas dos nós de comutação. Exemplo, uma VCC entre duas estações (*A* e *B*) ligadas, é formada por 5 VCLs (rótulos *a,b,x,y* e *z*). Quando *A* solicita conexão com *B* recebe da rede o rótulo *a* para ser utilizado na transmissão das informações, e devolve a *B* o rótulo *z* como identificador da VCL de chegada das informações de *A* (Figura 2.2) [22]. Durante o estabelecimento da conexão, as tabelas de rota são atualizadas internamente na rede. VCCs são unidirecionais, portanto, duas VCCs, com sentidos opostos, devem ser criadas simultaneamente, entre dois computadores finais, a fim de obterem comunicação *full-duplex.*

A conexão virtual está associada a um conjunto de descritores de tráfego especificando suas características, incluindo parâmetros de tráfego e classe de QoS.

Enlaces virtuais herdam características de tráfego da conexão virtual a qual fazem parte. Antes das células serem transportadas de um endereço para outro, uma VCC deve ser estabelecida, permitindo à rede reservar os recursos necessários.

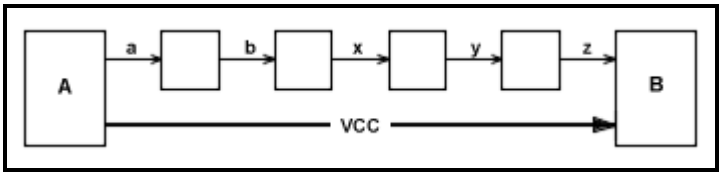


Figura 2.2. Conexão com Canal Virtual

São comuns várias VCCs roteadas pelos mesmos caminhos em determinadas partes da rede. Desta forma, o processamento em alguns nós é bastante reduzido porque as tabelas de rotas não precisam conter uma entrada para cada VCC, mas sim para um conjunto de VCCs. Conexão de Caminho Virtual (*Virtual Path Connection – VPC*) é composta de VCCs comutadas em conjunto. VPCs são formadas a partir da concatenação de Enlaces de Caminho Virtual (*Virtual Path Links - VPLs*), correspondendo aos diferentes enlaces que, juntos, formam o caminho entre dois pontos.

Existe o Identificador de Caminho Virtual (*Virtual Path Identifier - VPI*) de 8 bits e o Identificador de Canal Virtual (*Virtual Channel Identifier - VCI*) de 16 bits. VPI e VCI são únicos para células pertencentes à mesma conexão virtual no meio de transmissão compartilhado.

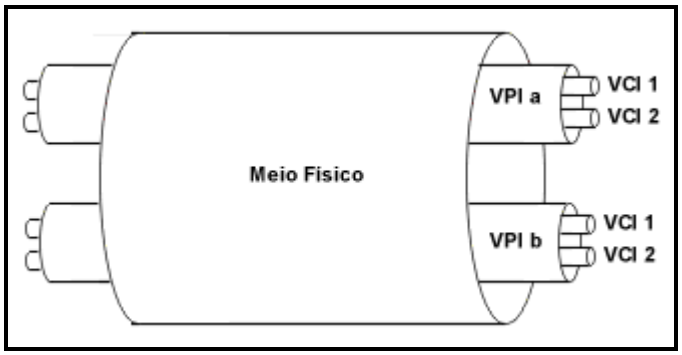


Figura 2.3. Meio Físico com duas VPLs e três VCLs.

Vários VPs, cada um composto de vários VCs, podem ser entendidos como um cabo, em cada enlace, composto de vários cabos internos mais finos (os VPLs) que, por sua vez ainda são compostos de cabos mais finos (identificadores de uma VCL dentro de uma VPL). A Figura 2.3 [22] ilustra essa analogia.

Como VPI é apenas parte do rótulo da célula utilizado para seu encaminhamento, pode-se imaginar duas camadas de comutação: camada inferior onde apenas o VPI é examinado, e camada superior, onde dado um VCI contido naquele VPI, a comutação poderá ser efetuada. Alguns nós da rede poderão efetuar a comutação baseando-se apenas no VPI, enquanto outros farão a comutação baseando-se no rótulo completo (VPI + VCI). A Figura 2.4 [22] apresenta operação de rede com comutadores de VP e VC. A presença de VPCs entre *A* e *T* e entre *T* e *B* pode ser visualizada no exemplo ilustrado.

A VCC entre *A* e *B* é identificada, em *T* e seus extremos *A* e *B*, por:

- Quando *A* transmite: VPI=*x1* e VCI=*a1*.
- Quando *T* recebe: VPI=*x3* e VCI=*a1*.
- Quando *T* transmite: VPI=*y1* e VCI=*a2*.
- Quando *B* recebe: VPI=*y3* e VCI=*a2*.

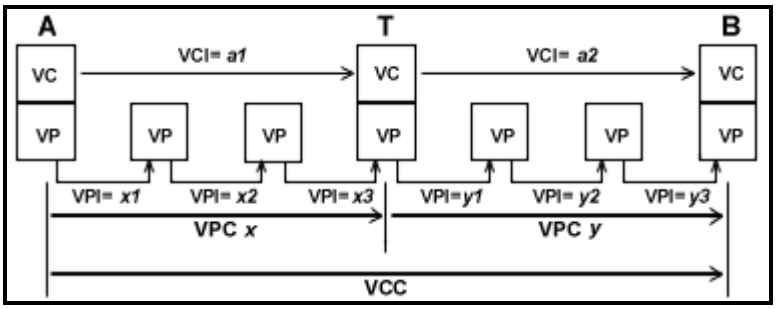


Figura 2.4. Exemplo de Funcionamento de Comutadores de VP e VC.

O funcionamento de comutadores de VP é igual ao ilustrado na Figura 2.6, só que apenas o VPI é utilizado (Figura 2.5) [22].

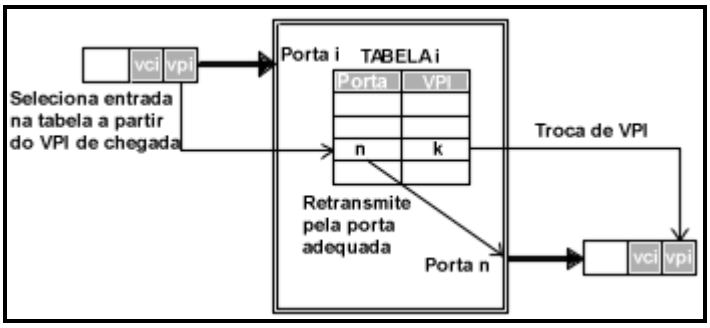


Figura 2.5. Funcionamento de Comutadores de VP.

Os comutadores que comutam VPs e VCs utilizam VPI como entrada para uma tabela de VPI associada à porta de entrada da célula. Cada porta do comutador tem uma tabela de VPI associada, na qual cada entrada representa um VPI diferente. Associada a cada entrada desta tabela, encontra-se um ponteiro para uma nova tabela: a tabela de VCIs associada àquele VPI. As tabelas de VCIs contêm então os novos identificadores de porta, VCI e VPI utilizados para transmitir a célula. O processo é ilustrado na Figura 2.6 [22].

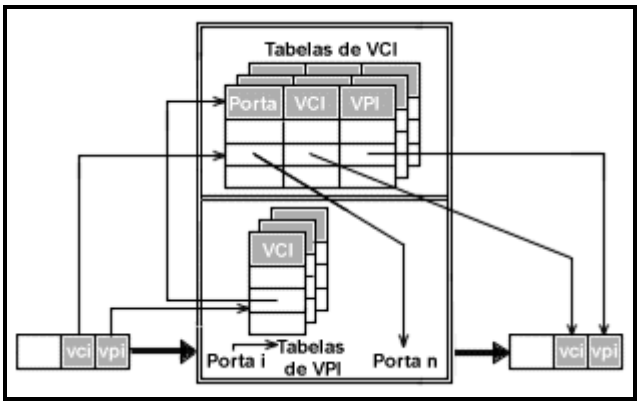


Figura 2.6. Funcionamento de Comutador com duas Camadas VP e VC.

Podemos verificar que, através de uma VPC, formada por comutadores VP, o VCI é passado de forma transparente. A Figura 2.7 apresenta exemplo de funcionamento das camadas de comutação de VP e de VC [22].

Células, utilizando a camada VP ou as camadas VP e VC, podem ser chaveadas pelo mesmo elemento comutador. A Figura 2.8 ilustra um exemplo de comutador onde as conexões da porta 1 são chaveadas pelas duas camadas, enquanto conexões da porta 4 são

chaveadas apenas na camada VP, sem necessidade de alteração de VCI. As tabelas para as portas 1 e 4 são apresentadas na Figura 2.9 [22].

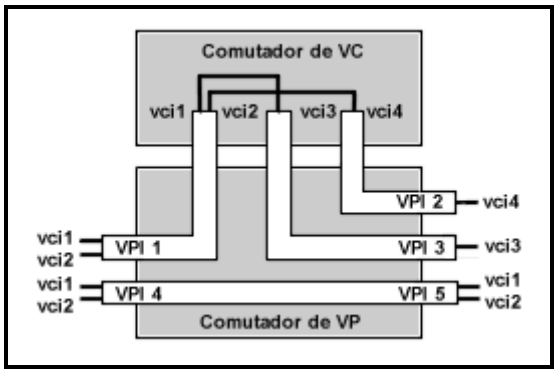


Figura 2.7. Comutadores de VC e de VP.

Células ATM transportam informações do usuário e informações de sinalização. Informações de controle e gerenciamento também são transportadas pelas células ATM, ficando, portanto, alguns valores de VPI e VCI reservados para esse transporte.

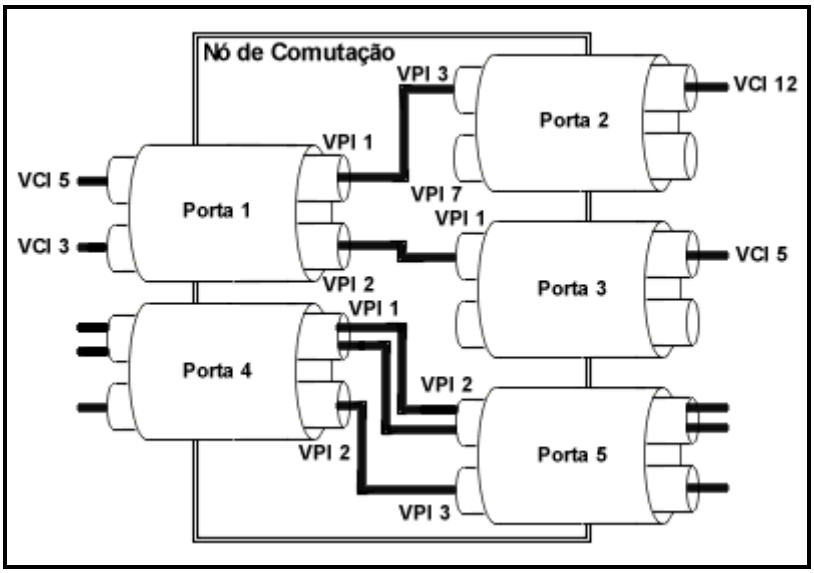


Figura 2.8. Exemplo de Configuração em Comutador.

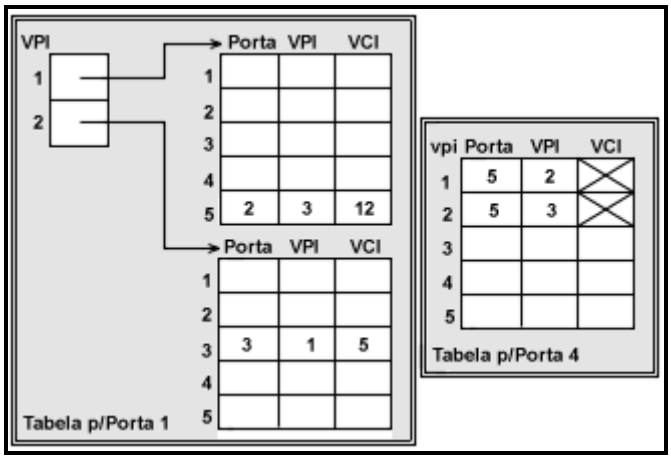


Figura 2.9. Tabelas para Portas 1 e 4 do Comutador.

VPCs e VCCs podem ser permanentes (também chamados dedicados) ou chaveadas (estabelecidas dinamicamente através de procedimentos de sinalização). Estabelecimento de VPCs e/ou VCCs permanentes, tratado neste trabalho, é descrito na seção seguinte.

2.3. Estabelecimento de PVC

O estabelecimento de PVCs é uma tarefa complexa e requer cuidado especial do administrador de redes [9]. Inicialmente, uma rota fim-a-fim de dois sistemas finais deve ser selecionada. A rota consiste de uma lista de nós intermediários conectados aos sistemas finais. Cada nó contém a tripla que identifica o dispositivo, a porta de entrada e a porta de saída que serão utilizadas. Uma rota física deve ser selecionada dentre as várias disponíveis.

Identificando-se a rota, o passo seguinte é decidir quais VPs serão utilizados no PVC. A forma mais simples é sempre selecionar VP com identificador 0. VP 0 é criado e mantido em cada porta do dispositivo ATM, logo que iniciado. Entretanto, VPI igual a 0 não pode ser usado indefinidamente, pois VP tem capacidade limitada e suporta número determinado de valores para VCIs.

Outra forma consiste em estabelecer a VPC de maneira a otimizar o número de VCCs aceitas pela rede simultaneamente. Uma VCC pode ser roteada dentro de muitas VPCs, onde cada VPC cobre um trecho da rota física. Pode-se observar que se trata de processo bastante complexo. Em [23], encontramos várias referências desta metodologia de atribuição de VPIs/VCIs.

Porém, a pesquisa adotada por operadores de ATM consiste em criar VPs aleatórios entre comutadores consecutivos na rota. Embora pareça solução pouco otimizada em termos de números de VCCs suportadas, possui vantagens relevantes como facilidade e rapidez de estabelecimento de PVC.

O passo final é atribuir valores VCIs a cada comutador e criar uma entrada de rota VC. Em cada comutador, é necessária um VCI para porta de entrada e um para porta de saída. O valor do VCI de saída de determinado comutador deve ser o mesmo de entrada do seguinte. O mesmo acontece quando VPs são criados nos comutadores para veicular o PVC. Se algum destes passos não for satisfeito, dados transmitidos ao longo deste PVC não alcançarão seu destino. Esses são problemas de erros de configuração difíceis de serem diagnosticados. Outra dificuldade encontrada está na definição do Controle de Parâmetros do Usuário (*Usage Parameters Control - UPC*), que deve usar os mesmos parâmetros em cada comutador da rede. Sendo configurado erroneamente, todo o tráfego ao longo do PVC estabelecido poderá ser afetado.

O principal obstáculo que dificulta a tarefa de estabelecimento de PVCs resulta da heterogeneidade de fabricantes. Cada fabricante de dispositivos ATM provê interface própria de gerenciamento. Na maioria dos casos, a configuração é realizada através do "*telnet*" ou da interface de gerenciamento oferecida. O operador de redes ATM deve conhecer estas interfaces de gerenciamento para criar um PVC. Portanto, qualquer aplicação de gerenciamento que permita a automatização de estabelecimento do PVC, pode ajudar administradores de rede ATM a proverem serviços mais eficientes a seus clientes.

2.4. Conclusão do Capítulo

Para concluir este capítulo, seguimos os princípios básicos do ATM definidos pela Recomendação I.150 do CCITT, onde podemos observar que:

- ATM é considerado como modo básico específico de transferência orientada a pacotes, baseado em células de tamanho fixo. Cada célula consiste de um campo de informação e um cabeçalho, que determina o canal virtual para realizar o roteamento apropriado. A integridade da seqüência da célula é preservada pelo canal virtual;
- ATM trata-se de uma tecnologia orientada a conexão. Valores do cabeçalho são designados para cada seção da conexão em toda conexão;
- Campo de informação da célula ATM é carregado transparentemente pela rede. Nenhum processo, como controle de erro, é realizado dentro da rede; Todos os serviços (voz, vídeo, dados) podem ser transportados via ATM, incluindo serviços sem conexão.

Muito utilizada em ATM, a tarefa de estabelecimento de conexões permanentes fim-a-fim requer peculiaridades características de cada fabricante. Obstáculos provenientes da heterogeneidade de estabelecimento de PVCs dificultam sua operacionalização. No sentido de criar aplicação para automatizar tal tarefa, propomos, ao final deste trabalho, integração de sistemas de gerência existentes com AMs.

Capítulo 3. Gerenciamento de Redes ATM

3.1. Introdução

A expressão gerenciamento de redes é usada para traduzir o controle e monitoramento de infraestrutura de redes. Pode ser feito em alto nível, gerando dados para medida de Contrato de Nível de Serviços (*Services Level Agreements - SLAs*), ou em nível mais detalhado, realizando provisão de equipamentos ou localizando seu mal funcionamento. A infraestrutura de redes refere-se não somente a roteadores, comutadores e distribuidores, usualmente encontrados em redes corporativas, como também a servidores de aplicação, servidores de arquivos, servidores de impressão etc. Sistemas de Gerenciamento de Redes (*Network Management Systems - NMSs*) surgem para controlar tais equipamentos e softwares, oferecendo completa visão da infraestrutura e habilitando estatísticas precisas do desempenho da rede.

Fabricantes suprem seus dispositivos com softwares agentes para monitorar e coletar dados operacionais (p.ex. estatísticas) em bases de dados locais e/ou detectar eventos de exceção (p.ex. taxas de erro). Estações de gerenciamento recebem dados ou obtém notificações de eventos dos dispositivos gerenciados através de protocolos de comunicação. Plataformas de gerenciamento suportam ferramentas para mostrar dados graficamente, interpretá-los e controlar operações.

No sentido de padronizar as exigências de gerência, há diversos órgãos tratando destes aspectos. Para padronização do protocolo temos:

- IETF (*International Engineering Task Force*) descreveu o protocolo SNMP usado na infraestrutura TCP/IP;
- ISO (*International Standard Organization*) descreveu o protocolo CMIP usado na infraestrutura OSI (*Open System Interconnection*).

Para padronização da arquitetura temos:

- ITU-T (*International Communication Union*) publicou uma arquitetura básica com alguns padrões genéricos, os quais tem sido conhecido como TMN (*Telecommunication Management Network*).

A arquitetura TMN é definida em arquitetura física, funcional e de informação, e podem ser vistos em [24]. TMN suporta gerenciamento e implantação de serviços de telecomunicações dinâmicos, provendo gerente de funções e comunicações para operação, administração, manutenção e provisão (*Operation, Administration, Maintenance and Provisioning – AOM&P*). Como ATM é tecnologia capaz de implementar LANs e WANs, existem debates sobre qual sistema é melhor para gerenciá-la. Tradicionalmente, TMN, tendo CMIP especificado como protocolo, é escolhido para gerenciamento de WANs, enquanto SNMP para LANs.

Tabela 3.1. Funções de Gerenciamento de Rede FCAPS

Categoria	Funções de Gerenciamento
Gerenciamento de Configuração	- Monitoramento do status da rede - Gerenciamento de facilidades - Topologia da rede - Controle de dispositivos de rede - Gerenciamento de clientes - Gerenciamento de serviços - Gerenciamento de SLAs - Planejamento
Gerenciamento de Falhas	- Gerenciamento de logs - Gerenciamento de tickets - Notificação de eventos - Estados operacionais - Correlação de alarmes e diagnósticos - Auto testes - Reconfiguração e Gerenciamento de degradações - Animações e sinopses gráficos
Gerenciamento de Desempenho	- Medidas e estatísticas de: tempos de resposta largura de banda taxas de erro serviços e dispositivos disponíveis - Matriz de Tráfego
Gerenciamento de Segurança	- Gerenciamento de direito de acessos - Gerenciamento de senhas - Autenticação - Gerenciamento dados criptografados - Gerenciamento de logs de segurança - Distribuição de chaves
Gerenciamento de Contas	- Avaliação da rede e uso de serviços geração, coleta de informação, relato de contas e política de descontos

Funções de gerenciamento de redes são identificadas pela ISO, no modelo OSI [25]. Estas funções são classificadas em 5 grandes áreas conhecidas pelo acrônimo FCAPS – Falha (*Fault*), Configuração (*Configuration*), Contabilidade (*Accounts*), Desempenho (*Performance*) e Segurança (*Security*). O sumário das funções de gerenciamento de rede é apresentado na Tabela 3.1. A descrição detalhada de cada função pode ser encontrada em [26]. Nosso trabalho está vinculado às funções de Configuração.

3.2. SNMP versus CMIP

SNMP e CMIP provêem padrões de plataforma para permitir interoperabilidade de gerenciamento de redes em ambientes heterogêneos de diversos fabricantes. Elas são bastante similares nos termos de funcionalidade e operação básica.

SNMP e CMIP fazem uso do modelo "cliente/servidor" onde o cliente é o sistema de gerenciamento e o servidor o sistema gerenciado. O sistema gerenciado (servidor) assume o papel de agente e, ao receber requisições de gerenciamento, executa comandos e transmite eventos de notificações não solicitadas – os *traps*. O sistema de gerenciamento (cliente) assume o papel de gerente, invoca operações e recebe notificações/*traps*. Entretanto, há diferenças essenciais entre esses protocolos, conforme apresentaremos nas seções 3.2.1 a 3.2.4 [21].

3.2.1. Modelo de Informação

O sistema de gerenciamento OSI, CMIP, baseia-se no princípio da Orientação a Objeto (OO) para representar informações de gerenciamento. Recursos gerenciados são representados em termos de objetos e classes gerenciadas. Com conceitos de OO tais como herança e alomorfos, informações de gerenciamento podem ser ordenadas de forma estruturada, permitindo assim, alto grau de abstração.

Por sua vez, SNMP apresenta estrutura de armazenamento de informação relativamente plana. Operações de gerenciamento ficam restritas a objetos escalares. Único tipo de estrutura suportada é tabela conceitual bidimensional, descrita no próximo capítulo. Através deste esquema de representação, nenhum objeto genérico pode ser definido. Convém ressaltar que, embora recursos gerenciados também sejam referenciados como objetos, SNMP não é OO.

3.2.2. Protocolo

SNMP e CMIP são protocolos do nível de aplicação. SNMP está em conformidade com a pilha de protocolos de 4 níveis da Internet, enquanto CMIP, é baseado no modelo de referência da ISO-OSI de 7 níveis.

Mensagens do protocolo SNMP são UDP (*User Datagram Protocol*), ou seja, não são orientadas a conexão. O principal motivo desta escolha está na continuidade de operação do SNMP, mesmo em redes com baixa qualidade de comunicação. Usando protocolo de transporte não orientado a conexão, SNMP assume responsabilidade da transmissão confiável de mensagens. Em contraste, CMIP faz uso de serviço confiável orientado a conexão, provido pela camada de transporte do modelo de referência.

Operações SNMP têm sido deliberadamente específicas no sentido de torná-lo simples. Objetos gerenciados SNMP aceitam somente operações GET e SET, enquanto comandos imperativos de criação/exclusão em tabelas são emulados através do SET. CMIP provê rico conjunto de operações, permitindo sofisticados acessos aos objetos gerenciados. Assim, comandos imperativos são diretamente executados. Além disso, mecanismos de *scoping* e filtragem reduzem consideravelmente o tráfego no gerenciamento da rede. Através do *scoping,* CMIP lança uma simples requisição para obter todas informações necessárias. Utilizando mecanismo de filtragem, valores de base no agente podem ser reunidos eliminando a transmissão de informações de rotina para a estação de gerenciamento. Entretanto, esses mecanismos penalizam o desempenho do agente por

causa da sua complexidade. No SNMP não existe nenhum destes mecanismos. Como resultado, SNMP precisa de uma série de requisições, a partir da estação de gerenciamento, para transmissão de dados.

3.2.3. Filosofia de Gerenciamento e Interoperabilidade

SNMP e CMIP desempenham papel de gerenciamento em toda arquitetura distribuída, embora de diferentes formas. A idéia atrás do sistema de gerenciamento da ISO é distribuir inteligência em toda rede, isto é, entre entidades gerenciadas e gerentes. A solução resultante é maior flexibilidade. Entretanto, requer mais recursos em termos de capacidade de memória e processamento nos agentes. Conseqüentemente, o custo inicial para primeira aplicação torna-se muito elevado, tendendo a cair com adição de novos objetos.

SNMP tem como estratégia de projeto transferir a carga de trabalho para estações de gerenciamento e, portanto, o reflexo da adição de elementos gerenciados é mínimo. Agente SNMP pode ser implementado em sistemas pequenos e baratos. Qualquer complexidade necessária ao sistema é realizada pelo gerente SNMP; portanto, o papel operacional e o desempenho de um sistema não são comprometidos com a inclusão de um agente SNMP. A complexidade do sistema total, bem como seus custos são otimizados porque o número de estações gerentes é muito menor do que de agentes.

Como *traps* SNMP são lançadas pelo elemento gerenciado de forma assíncrona para estações de gerenciamento, elas não são, por natureza, confiáveis. Conseqüentemente, gerentes de rede não devem confiar nelas. Alternativamente, muitas implementações usam *polling* para checar o status do agente. Isso limita muito a escalabilidade do SNMP em redes muito grandes, devido ao grande número de processamento e tráfego gerados pelos *pollings* para elementos gerenciados.

Mecanismos de reportagem de eventos CMIP são mais sofisticados, permitindo controle flexível de notificações emitidas. Gerenciamento de redes baseado em CMIP adota mecanismo dirigido a eventos, visto que empregam serviço OSI de transporte confiável e os agentes são inteligentes. Reportagem de eventos fornecem informações de desempenho e são gerados para relatar eventos espontaneamente. Desta forma, *overhead* de rede é consideravelmente reduzido, estendendo a escalabilidade do NMS. Entretanto, *pollings* regulares são necessários para garantir operação apropriada da rede.

3.2.4. Adequabilidade para Gerenciamento ATM

Característica OO e escalabilidade do sistema de gerenciamento de rede OSI são importantes vantagens no manuseio de grandes redes. Por exemplo, se uma interface para, todos protocolos executados naquela interface também param. Mecanismos de *scoping* e filtragem são especialmente necessárias para redes ATM, uma vez que muitas condições de alarme ocorrem devido à falha do enlace. Este sistema é mais atrativo com a introdução de TMN, onde CMIP é especificado como protocolo de gerenciamento.

Entretanto, o custo para implementação de agentes CMIP é bem mais alto do que de agentes SNMP, porque são mais complexos e requisitam muito mais recursos computacionais para sua execução. Como resultado, o desempenho do agente é bastante penalizado. Isso, efetivamente, impede CMIP de ser usado em outras áreas, como equipamentos de comunicações de dados. SNMP, por outro lado, oferece baixo custo, simplicidade de implementação e rápido processo de padronização.

SNMP não foi projetado para tratar grandes volumes de tráfego de rede gerados pelo ATM. Muitas adaptações podem ser necessárias para gerenciamento ATM. Exemplo, pode-se empregar AMs para tratar dos alarmes gerados por comutadores numa grande rede, reduzindo o tráfego do gerenciamento. Muitas MIBs já foram definidas para ATM. Estas MIBs cobrem quase todos aspectos do ATM. SNMP é adequado em termos das funções de gerenciamento e recursos solicitados, provendo ótimo custo benefício e solução

de interoperabilidade para o desafio da gerência de redes ATM. Quase todos comutadores ATM são gerenciáveis pelo SNMP, assim como muitas aplicações executadas sobre TCP/IP também são. Como resultado, o escopo do gerenciamento ATM baseado no SNMP pode ser facilmente estendido para outros dispositivos e aplicações que não sejam ATM.

3.3. Desafios da Gerência de Redes ATM

Devido sua sofisticação e flexibilidade, ATM é considerada tecnologia complexa. O desenvolvimento de redes ATM requer um *overlay* altamente complexo da infraestrutura de protocolo de software, conduzindo a um nível de complexidade, sem precedentes, de gerenciamento de redes. Soluções adequadas para gerência de outras tecnologias são insuficientes para ATM, particularmente para aplicações. Conseqüentemente, o gerenciamento efetivo ATM direciona-se para suas características, tais como ambiente orientado a conexão, variação das classes de serviço, alto volume de tráfego, configuração de redes virtuais e múltiplos tipos de tráfego.

O gerenciamento de dispositivos ATM exige alto consumo de recursos computacionais de máquina, por exemplo, obter estatísticas de desempenho de comutadores. A funcionalidade de software e hardware em *dispositivos* ATM deve ser dividida para permitir melhor desempenho de processamento. Por causa do alto custo das redes de alta velocidade e do alto fluxo de informações gerenciáveis através dos enlaces existentes, o gerenciamento efetivo da rede tornou-se crítico [21].

ATM suporta um número quase ilimitado de conexões virtuais de alta velocidade. Simples falha na rede pode resultar numa imensidão de mensagens de alarme, tornando impossível para o administrador ler, analisar e agir na tentativa de solucionar cada um. Conseqüentemente, é necessária pesquisa eficaz e inteligente para controlar eventos e desempenho de processos nos dispositivos ao longo da rede heterogênea.

Redes ATM provêem diferentes níveis de QoS para múltiplos tipos de tráfego, tais como voz, vídeo e dados. Portanto, o gerenciamento desse tráfego representa importante aumento de regras de sistemas de gerenciamento de rede. Garantir nível de serviço desejado através da configuração de parâmetros, prioridades de tráfego, controle de volume e velocidade do fluxo de informações torna-se importante prioridade.

Enquanto serviços sofisticados são oferecidos a usuários finais, *hackers* podem explorar esse benefício, caso medidas de segurança não sejam devidamente implementadas. Freqüentemente, a capacidade das ferramentas de gerenciamento de segurança é inadequada aos atuais aspectos de segurança ATM, existentes.

Diferentes tipos de classes de serviço e atributos de QoS tornam o gerenciamento de contabilidade tarefa muito complexa. Para ajudar o controle de alocação de recursos, mecanismos de contabilidade devem ser instituídos através do sistema de gerenciamento. Gerenciamento de Contabilidade é requisito emergente para redes ATM.

Gerenciamento de redes ATM deve ser capaz de suportar o controle de aplicações executadas sobre sua tecnologia, permitindo integração de redes e sistemas. Deve, também, ser concordante com padrões existentes e aceitar interoperabilidade entre ambientes de diferentes fabricantes.

Este trabalho fundamenta-se no gerenciamento de ATM baseado no SNMP. SNMP oferece integração com aplicações executadas sobre ATM, levando-se em consideração que a maioria destas aplicações é hoje e, provavelmente no futuro próximo, orientada para Internet. Muitos esforços têm sido dedicados na padronização de objetos SNMP voltados para ATM, oferecendo bons níveis de flexibilidade e interoperabilidade de redes, sistemas e aplicações de gerência.

3.4. Aplicação das 5 Áreas de Gerenciamento OSI em ATM

As funções de gerenciamento de redes são identificadas pela ISO no conhecido modelo de gerência da OSI [27]. Essas funções podem ser classificadas em cinco grandes áreas de gerenciamento. Estas cinco áreas e suas aplicações à gerência ATM são apresentadas a seguir.

3.4.1. Configuração

Envolve a determinação do que está na rede e a manipulação dos componentes para manter interconexões e serviços ativos e executando. Aplicados ao gerenciamento de redes ATM, os requisitos relacionados ao gerenciamento de configurações incluem:

- Estabelecimento, liberação e reconfiguração de conexões ATM (VCCs e VPCs);
- Obtenção de informações de estados das conexões;
- Determinação do número máximo de conexões, do número de conexões pré-configuradas e do número de conexões ativas em uma interface;
- Configuração do número de bits VPI/VCI suportados;
- Configuração e obtenção de informações de endereçamento em uma interface.

3.4.2. Falhas

Trata da determinação, isolamento e correção de comportamentos anormais da rede. Devido ao crescimento de importância das redes nas empresas, qualquer interrupção dos serviços computacionais pode afetar profundamente toda operacionalização da organização. No contexto de gerência ATM, os requisitos relacionados ao gerenciamento de falhas incluem:

- Notificação de ocorrência de falhas em dispositivos, interfaces e conexões;

- Notificação e isolamento de falhas detectadas, com base na execução de testes realizados sob demanda;
- Manutenção de um registro ("*log*") das falhas detectadas.

3.4.3. Desempenho

Avalia o comportamento e produtividade. Ajuda a entender como a rede opera sob condições normais e anormais. Pelo monitoramento das estatísticas dos componentes de rede, a performance da rede pode ser melhorada. Aplicados ao gerenciamento de redes ATM, os requisitos ao gerenciamento incluem:

- Coleta de dados estatísticos que permitem avaliar a capacidade de um dispositivo em encaminhar e processar com sucesso células ATM;
- Monitoramento de violações dos parâmetros de tráfego negociados para uma conexão;
- Determinação dos limites aceitáveis para diversos tipos de contadores e o envio de notificações ou alertas, quando um limite estabelecido é ultrapassado.

3.4.4. Segurança

Trata do monitoramento, controle e proteção das informações gerenciais na rede, a partir de acessos acidentais e desautorizados. É responsável pelo gerenciamento das facilidades de segurança, incluindo seu controle e monitoramento. Ameaças ou brechas de segurança são reportadas.

3.4.5. Contabilidade

Consiste da alocação de custos entre os usuários da rede. Inclui coleta de informações usuais da rede, agrupando limites de contas e reportando custos envolvidos.

3.5. Conclusão do Capítulo

SNMP possui a simplicidade de projeto como principal vantagem, tornando fácil sua implementação em toda rede. Devido ao seu amplo uso, ele é suportado pela maior parte de NMSs e fabricantes de equipamentos, tornando indicada sua execução em ambientes multi-plataformas. Porém, apresenta duas grandes desvantagens: falta de escalabilidade e segurança.

CMIP conta com a vantagem de poder atuar nos Elementos de Rede (*Network Components - NEs*), além de realizar consultas de informações. CMIP resolve alguns problemas do SNMP, incluindo furos de segurança. Sua segurança suporta autorização, controle de acesso e arquivos de *log*. Em linhas gerais, as definições de seus objetos gerenciados são: modelo orientado a objeto, heranças múltiplas, mecanismo de nomes, polimorfismos e pacotes condicionais. Infelizmente, CMIP apresenta duas desvantagens. Primeiro, o processamento executado nos NMS CMIPs é grande, muito superior ao executado nos NMS SNMPs. Isto, aplicado também aos NEs, os quais podem facilmente aumentar o preço da implementação. Segundo, CMIP é difícil de programar devido sua complexidade.

Aproveitando as vantagens do protocolo SNMP, procuramos superar suas desvantagens, tais como centralização de procedimentos, escalabilidade e flexibilidade buscando a integração com a tecnologia de agentes móveis. Agentes móveis, apresentados no Capítulo 5, possuem importantes características que podem ser empregadas, em conjunto com sistemas legados, para potencializar o gerenciamento de redes heterogêneas.

Capítulo 4. Usando SNMP para Gerência ATM

4.1. Introdução

O problema da gerência ATM é composto pelo confronto de duas culturas independentes de indústrias de desenvolvimento de rede: telecomunicações e comunicações de dados [21]. O primeiro, TMN, é adotado pelos provedores de telecomunicações no gerenciamento de redes de telecomunicação e o segundo, marcado pelo protocolo SNMP, é adotado pelas empresas de comunicação de dados. Existe a disputa de padronização entre o sistema de gerenciamento de rede da OSI / ISO e os sistemas de gerenciamento da Internet.

Indiferente a muitos projetos de pesquisas, os primeiros comutadores comercialmente disponíveis são quase exclusivamente gerenciados pelo SNMP da IETF. Comparado ao modelo de gerenciamento da OSI, o qual não ganhou suporte suficiente, com exceção do TMN, SNMP tornou-se a principal escolha de sistema de gerência. Muitos documentos sobre Bases de Informação de Gerenciamento (*Management Information Base*) e aplicações de gerenciamento foram produzidos. SNMP oferece completo suporte a aplicações executadas em redes ATM, considerando-se que a maioria das aplicações ATM está orientada para Internet. Vários esforços estão sendo feitos para padronizar objetos SNMP gerenciáveis voltados para aspectos ATM, oferecendo um nível sem precedentes de gerência e interoperabilidade.

SNMP tem alcançado bons resultados provendo soluções de interoperabilidade à gerência de redes, habilitando monitoramento efetivo e controle de dispositivos heterogêneos. Atualmente, há três versões do SNMP: SNMPv1, SNMPv2 e SNMPv3. A Figura 4.1 [1] descreve o histórico da evolução do SNMP. O ponto de partida se deu com Protocolo de Monitoramento de Roteador Simples (*Simple Gateway Monitoring Protocol* -

SGMP) em novembro de 1987. SGMP provia monitoramento de roteadores nos sistemas de três níveis do modelo OSI. Em 1988, ficou clara a real necessidade do gerenciamento de rede envolvendo, a curto prazo, o nível de complexidade existente e, a longo prazo, o possível aumento do seu ambiente. SNMP, consistido dos documentos RFC 1155, RFC 1212 e RFC 1157, provia mínimo, mas potente, conjunto de facilidades para monitoramento e controle dos elementos conectados. Porém, com dificuldades na gerência das redes. Completa e única melhora funcional ao SNMP foi conseguida com a definição dos objetos de gerenciamento referenciados na MIB RMON - Monitoramento de Redes Remotas (*Remote Network Monitoring - RMON*).

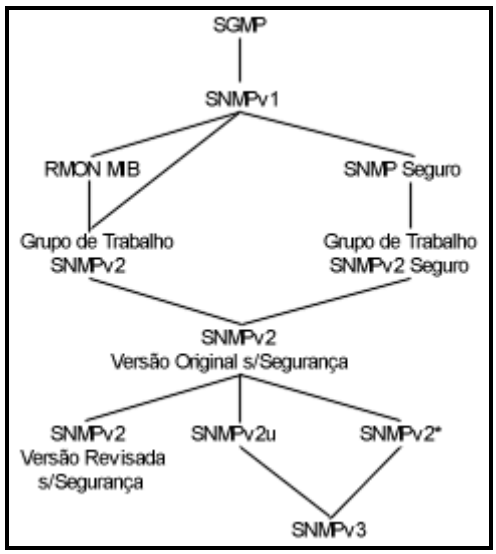


Figura 4.1. Evolução do SNMP

Outra notável deficiência do SNMP se refere à completa falta de segurança. Para remediar este problema, um conjunto de documentos, o SNMP Seguro (S*ecure SNMP – S-SNMP*) foi lançado como proposta de padronização em julho de 1992. No mesmo mês, surgiu o Protocolo de Gerenciamento Simples (*Simple Management Protocol - SMP*), fora da estrutura padrão da Internet. SMP provia melhora nas funcionalidades do SNMP e incorporava, com algumas modificações, seguranças do S-SNMP. SMP também incorporava conceitos do RMON, incluindo especificações de alarmes, eventos e uso de objeto *status columnar*, o qual facilitava criação e exclusão de linhas em tabelas das MIBs. A partir do SMP, foi desenvolvida a versão 2 do SNMP (SNMPv2), ficando a primeira conhecida como SNMPv1.

Para produzir o SNMPv2, dois grupos de trabalho foram criados. O primeiro, baseado no SMP e, portanto, sem preocupação com aspectos de segurança, encarregou-se pelo SMI, MIB, protocolo, comandos de adaptação e estratégias de coexistência com SNMPv1. Teve seus trabalhos concluídos em dezembro de 1992. O segundo, baseado no S-SNMP, conhecido como grupo de segurança SNMPv2, concluiu seus trabalhos em janeiro de 1993, mas com alguns assuntos não resolvidos. O trabalho total de criação do SNMPv2 foi encerrado com os RFCs 1441-1452 em março de 1993.

Depois de vários anos de experiência com SNMPv2, IETF resolveu revisá-lo e, após algumas modificações, relançá-lo com documentos RFCs 1901-1908 em 1996. Como resultado, surgiu novo SNMPv2, com modestas alterações da especificação e sem considerações dos aspectos de segurança. Esta nova versão fez uso do conceito do conceito de comunidade do SNMPv1, sendo chamado de SNMP Baseado em Comunidade (*Community Based SNMPv2*) ou SNMPv2C.

Para solucionar a falta de segurança, dois grupos independentes começaram a trabalhar na melhora do SNMPv2: SNMPv2u e SNMPv2*. Estas duas pesquisas serviram de entrada para novo grupo de trabalho, o SNMPv3, iniciando suas atividades em março de 1997. Em janeiro de 1998, este grupo publicou conjunto de documentos RFCs 2271-2275, onde se definiu um sistema incorporando funcionalidades do SNMPv1 e SNMPv2. SNMPv3 descreve arquitetura com estrutura específica de mensagens e características de segurança, sem alterar o formato do PDU SNMP. A idéia era PDUs SNMPv1 e SNMPv2 poderem ser usados dentro da nova arquitetura. A implementação do SNMPv3 consiste de características de arquitetura e segurança definidas pelas RFCs 2271-2275 mais formato do PDU e funcionalidade definida nos documentos do SNMPv2 ou seja: “SNMPv3 = SNMPv2 + Segurança + Administração”.

4.2. O Modelo de Gerenciamento

SNMP é baseado num conjunto reduzido de conceitos (Figura 4.2):

- Agentes de Gerenciamento (programas SNMP) em nós gerenciados, provendo acesso remoto para gerência;
- Estação de Gerenciamento de Rede (*Network Management System - NMS*), também conhecida como gerente, concentra as aplicações de gerenciamento;
- Protocolo de Gerenciamento de Rede, SNMP, usado para trocas de informações de gerenciamento entre Estação de Gerenciamento e Nó Gerenciado;
- Base de Informações de Gerenciamento (*Management Information Base - MIB*), especificada pela Estrutura da Informação de Gerenciamento (SMI) e definição de objetos gerenciados.

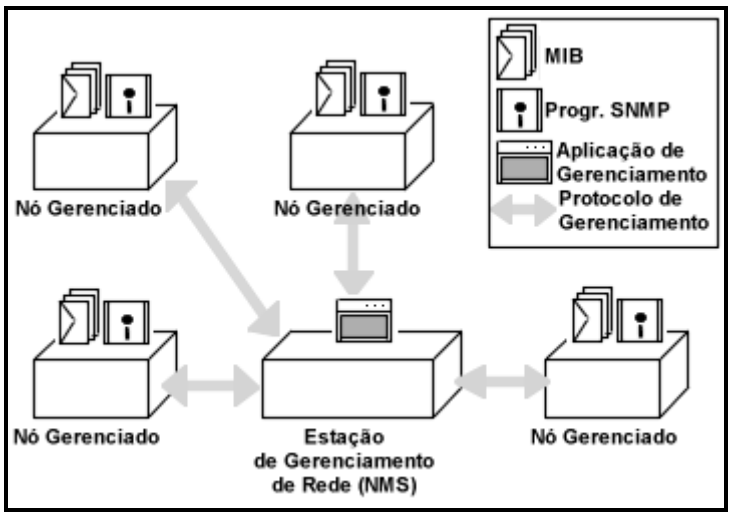


Figura 4.2. Modelo Clássico de Gerenciamento de Redes

O ambiente SNMP consiste de três tecnologias centrais: estrutura da informação e base de informação (SMI e MIB, respectivamente) e da maneira como esses recursos são resgatados (protocolo de gerenciamento SNMP) [21].

4.2.1. Estrutura da Informação de Gerenciamento (*Structure of Management Information – SMI*)

SMI define a forma, dentro da qual MIBs são construídas. Descreve estruturas comuns e esquemas de identificação para definição das informações de gerenciamento, particularmente o modelo de informação de objetos, com conjunto de tipos genéricos usados nas informações de gerenciamento. Faz uso da Notação de Sintaxe Abstrata 1 (*Abstract Syntax Notation One – ASN.1*).[1] Atualmente, existem duas versões de SMI. SMIv1, definida pelas RFCs 1155, 1212 e 1215, contém informações de gerenciamento do padrão Internet. SMIv2, definida pelas RFCs 1902-1904, é a versão compatível e atualizada do SMIv1. Possui um tipo de dados, o *RowStatus*, que, uma vez excluído, MIBs especificadas de acordo com SMIv2 são semanticamente idênticas a SMIv1.

4.2.2. Protocolo de SNMP

O protocolo é usado para transportar informações da gerência entre agentes e estações de gerenciamento, bem como diferentes estações de gerenciamento. Tarefas como interpretação, correlação e medidas corretivas devem ser especificadas por aplicações de gerenciamento.

4.2.3. Base de Informações de Gerenciamento (*Management Information Base - MIB*)

MIB é a coleção estruturada de objetos gerenciados, representando recursos nos elementos de rede. Definindo MIBs para novas aplicações e tipos de rede, SNMP pode facilmente estender suporte a grande variedade de dispositivos. MIBs compreendem a coleção de conhecimentos de especialistas sobre estratégias de gerência e dados necessários ao monitoramento e controle de determinado equipamento.

[1] ASN.1 é uma linguagem formal usada para definir sintaxes abstratas de aplicações de dados. É importante porque, através dela, são definidas a Unidade de Dados do Protocolo (*Protocol Data Unit - PDU*) e as MIBs.

Detalhes da MIB AToM e tabelas conceituais exploradas neste trabalho são apresentados na próxima seção. Entretanto, antes de prosseguirmos com suas peculiaridades, nas subseções seguintes nos deteremos ao conceito, operações de acesso e manipulação de tabelas conceituais pertinentes ao SMIv2.

4.2.3.1. Tabelas Conceituais

Assim como no SNMPv1, operações de gerenciamento em SNMPv2 só se aplicam a objetos escalares. Informações mais complexas podem ser representadas como tabelas conceituais. Uma tabela pode ter zero ou mais linhas, cada qual contendo um ou mais objetos escalares. SNMPv2 melhora as convenções usadas na RFC 1212 e especificação RMON (RFC 1757) para facilitar criação, exclusão e acesso de linhas. Essencialmente, duas tabelas conceituais são permitidas em SNMPv2 [1].

- Tabelas que proíbem criação e exclusão de linhas pelo gerente. Estas tabelas são controladas completamente pelo agente. O máximo nível de acesso permitido em qualquer objeto é ler-escrever. Em muitos casos, a tabela inteira consistirá apenas de objetos somente-leitura. Estas tabelas são úteis quando o número de linhas corresponde a atributos fixos ou a quantidade é controlada apenas pelo agente.
- Tabelas que permitem criação e exclusão de linhas pelo gerente. Estas tabelas podem ser iniciadas sem linhas, onde gerente pode criá-las e excluí-las. É também possível para o número de linhas na tabela variar por ação do gerente ou do agente.

Ambos os tipos provêem convenções e facilidades para acessar linhas na tabela, através da indexação. Porém, tabelas que permitem criação e exclusão de linhas possuem características adicionais que facilitam estas funções.

4.2.3.2. Criação e Exclusão de Linhas Conceituais

De todos assuntos envolvidos em SNMPv1 e SNMPv2, nenhum é mais controverso, consome mais tempo, ou requer mais atenção do que criação e exclusão de linhas conceituais. Duas estratégias são consideradas.

- Definir dois novos PDUs, *Create* e *Delete*, para serem usados na criação e exclusão de linhas, respectivamente.
- Embutir na semântica de criação e exclusão de linhas da MIB nova convenção contextual, chamada **RowStatus**. Com isso, criação e exclusão se tornam possíveis com uso dos comandos **SET** e **GET**.

Há problemas com as duas propostas em termos de complexidade e *overhead* de comunicação. Entretanto, a segunda estratégia é adotada e será descrita a seguir.

Uma tabela conceitual pode ser definida de forma a permitir criação de novas linhas ou exclusão das existentes. Para suportar criação e exclusão, deve haver um objeto **columnar** na tabela com valor **RowStatus** para cláusula **SYNTAX**, e valor **read-create** para cláusula **MAX-ACCESS**. Exemplo, objeto **atmVclRowStatus** (.1.3.6.1.2.1.37.7.1.13) da tabela **atmVclTable** (Tabela 4.1) na MIB AToM.

Tabela 4.1. Definição do Objeto atmVclRowStatus

```
atmVclRowStatus          OBJECT-TYPE
     SYNTAX       RowStatus
     MAX-ACCESS   read-create
     STATUS       current
     DESCRIPTION
      "This object is used to create, delete or
       modify a row in this table.  To create
       a new VCL, this object is initially set
       to 'createAndWait' or 'createAndGo'…

     DEFVAL { active }
           ::= {atmVclEntry 13}
```

Por convenção, este termo é designado de **status column** para linha conceitual e possui seis valores, listados na Tabela 4.2.

Tabela 4.2. Valores para o Objeto RowStatus

active	Linha conceitual disponível para uso pelo dispositivo gerenciado
notInService	Linha conceitual existe, mas indisponível para uso pelo dispositivo gerenciado
notReady	Linha conceitual existe, mas s/informação p/habilitá-la p/uso pelo disp.gerenciado
createAndWait	Estação gerente deseja criar nova instância da linha conceitual sem disponibilizar
createAndGo	Estação gerente cria e disponibiliza linha conceitual automaticamente
destroy	Estação gerente exclui instância da linha conceitual

Segundo a definição de **RowStatus** na RFC 1442, existem dois métodos utilizados na criação de linhas conceituais: **createAndWait** e **createAndGo**.

No método **createAndWait** (Tabela 4.3) o gerente inicia o processo instruindo o agente a criar nova linha com instância do identificador (valor do índice) fornecida. Em caso de sucesso, o agente cria nova linha e assinala valores *default* para o restante dos objetos da linha. Se todos os objetos **read-create** possuem valor *default*, a linha é substituída pelo estado **notInService**, indicando que a linha está criada, porém não ativa. Se alguns objetos **read-create** não possuem valores *default*, a linha é substituída pelo estado **notReady**, indicando que não pode ser criada por faltarem alguns valores. O gerente deve enviar um comando **GET** a fim de determinar o status de cada objeto **read-create** da linha. Em seguida, o agente responde com o valor *default* de cada objeto: **noSuchInstance** quando suportado pelo agente e **noSuchObject** quando, embora definido na MIB, não é suportado pelo agente. O gerente usa o comando **SET** para assinalar valores *default* para todos os objetos **noSuchInstance**. Uma vez criados todos objetos com acesso **read-create**, o gerente ativa a linha conceitual alterando o valor de status para **active,** através do comando **SET**.

Tabela 4.3. Esquema do Método createAndWait

createAndWait(5)

- Gerente instrui agente a criar nova linha
- Se sucesso
  - Agente cria linha
  - Assinala valores *default* ao restante dos objetos
  - Se todos objetos *read-create* possuem valores *default*
    - Linha substituída pelo valor **notInService** (criada e não ativa)
  - Senão
    - linha substituída pelo valor **notReady**
    - Gerente envia GET p/determinar status de cada objeto *read-create*
    - Agente responde com: **noSuchInstance** (suportado pelo agente) e **noSuchObject** (não suportado pelo agente)
    - Gerente envia SET com valores *default* para objetos noSuchInstance
  - Gerente envia SET para ativar a linha (**active**)

O método **createAndGo** (Tabela 4.4) é mais simples, porém bem mais restrito por não oferecer ao gerente possibilidade de acompanhamento automático dos valores *default*. O gerente inicia processo selecionando instância identificadora (valor do índice). Ele envia comando **GET** para determinar quais objetos **read-create** são **noSuchInstance**, caso não possua estas informações a partir do agente. Em seguida, o gerente envia comando **SET**, criando nova linha e assinalando valores para os objetos restantes da linha. Se o comando **SET** obtiver sucesso, a linha será criada e assinalada com estado **active**.

Tabela 4.4. Esquema do Método createAndGo

createAndGo(4)

- Gerente envia GET p/determinar quais os objetos **noSuchInstance**
- Gerente envia SET p/criação de nova linha e assinala os valores dos objetos restantes
- Se sucesso
  - Linha é criada, já ativada (a**ctive)**

O método **createAndWait** sobrecarrega o agente por estar habilitado a manter linhas conceituais no estado **notInService**. Entretanto, é mais eficiente por permitir acompanhamento e maior flexibilidade. Por outro lado, o método **createAndGo** envolve somente uma ou duas trocas de PDUs, sendo, portanto, mais eficiente nos recursos de

comunicação, tempo de gerenciamento da estação e complexidade do agente. As duas opções estão disponíveis no SNMPv2 e permitem ao implementador decidir qual opção usar em cada tabela.

Para excluir linhas conceituais, estações de gerenciamento lançam operações **SET** substituindo o estado do objeto **RowStatus** para **destroy**. Em caso de sucesso, o agente remove imediatamente a linha da tabela conceitual.

A estação de gerenciamento também pode suspender uma linha correntemente ativa. Para isso, basta lançar operação **SET** com estado **notInService**. Em caso de sucesso, a linha especificada é retirada do serviço e o agente retorna com resposta **noError**. Caso contrário, o valor **wrongValue** é retornado pelo agente.

4.3. Gerenciamento de Redes ATM com SNMP

Segundo [28] três requisitos básicos devem ser preenchidos para tornar uma rede ATM gerenciável através do SNMP:

- Dispositivos devem conter agentes SNMP e informações de gerenciamento, denominadas MIB;
- Cada dispositivo é responsável pelo comportamento do sistema, registrado na sua MIB.
- Gerentes estão habilitados a trocar *Protocol Data Units* (PDUs) com entidades agente. O protocolo é usado para transportar informações da gerência entre agentes e estações de gerenciamento, bem como diferentes estações de gerenciamento.

Diante da necessidade de suprir os requisitos para as redes ATM serem gerenciadas pelo protocolo SNMP, foram criadas várias MIBs específicas. Dentre essas

MIBs, as que mais se destacam são MIB AToM [29] e MIB ILMI [30]. As informações usadas nessas MIBs são baseadas no SMIv2 e têm acesso permitido pelo protocolo SNMP.

A MIB ILMI contém objetos que representam atributos das interfaces ATM ILMI. A interface ILMI, padronizada pelo ATM Forum, possibilita que dois dispositivos ATM adjacentes configurem automaticamente os parâmetros de operação de uma conexão ATM comum a eles, com troca de informações de gerência. Essa interface permite ainda controle de configuração da interface da camada física, interface da camada ATM e das Conexões de Caminho Virtual (*Virtual Path Connections VPCs*) e Conexões de Canal Virtual (*Virtual Channel Connections VCCs*) de um sistema adjacente.

Cada entidade de gerenciamento de interface (*Interface Management Entity – IME)* é associada a uma interface ATM que suporta as funções ILMI. Entende-se por IME, um agente de gerenciamento da interface.

A MIB AToM, definida pelo IETF, especifica um conjunto de objetos de gerência ATM. Esta MIB define objetos para gerenciar interfaces ATM, conexões virtuais, *cross-connect*s, entidades AAL5 e conexões suportadas por computadores, comutadores e redes ATM. Seu principal objetivo é gerenciar Conexões Virtuais Permanentes (*Permanent Virtual Connections PVCs*) e, por isso, foi adotada em nosso trabalho. A MIB AToM será detalhada na próxima seção.

4.31. MIB AToM da RFC 1695

Sucessivos aumentos no número de informações gerenciadas levaram a um grande crescimento do número de MIBs. Entretanto, as diferenças entre variações de versões do SNMP possuem pequeno impacto referentes às MIBs. A RFC 1695 [29], surgiu para especificar uma MIB, baseada no SNMPv2, com finalidade de gerenciar redes ATM. Essa MIB, também conhecida como MIB AToM (Figura 4.3), trata da combinação de duas

partes: ATM e "o" minúsculo, que significa SONET [31]. SONET, ligada ao gerenciamento de interface física, foi definida pelo Instituto Nacional Americano de Padronização (*American National Standards Institute - ANSI*) e é tecnologia usada pela ATM para transmissão de células. MIB AToM define objetos de gerência das interfaces ATM, enlaces virtuais, cruzamentos de conexões e entidades e conexões AAL5 suportadas por sistemas finais e intermediários de redes ATM. Ela é definida de acordo com SNMPv2.

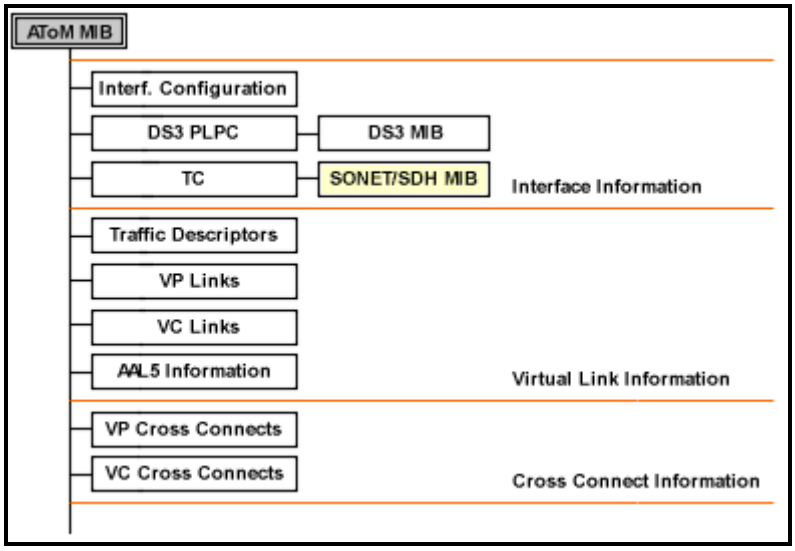


Figura 4.3. Estrutura da IETF MIB AtoM (RFC 1695)

Assim como em diversas MIBs do SNMP, cada grupo de objetos gerenciados é representado na MIB AToM por tabelas conceituais, descritas no SMI v2 [32]. A Figura 4.4 [28] apresenta esquema da MIB AToM.

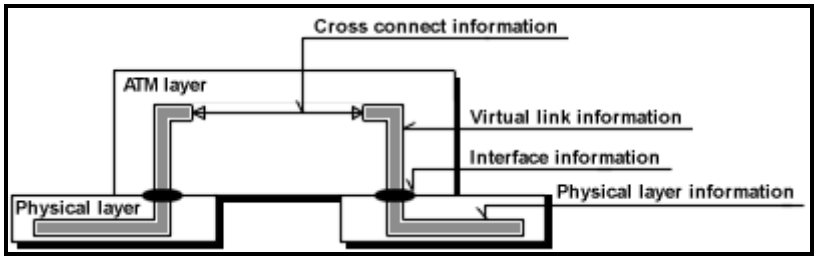


Figura 4.4. Esquema da MIB AToM

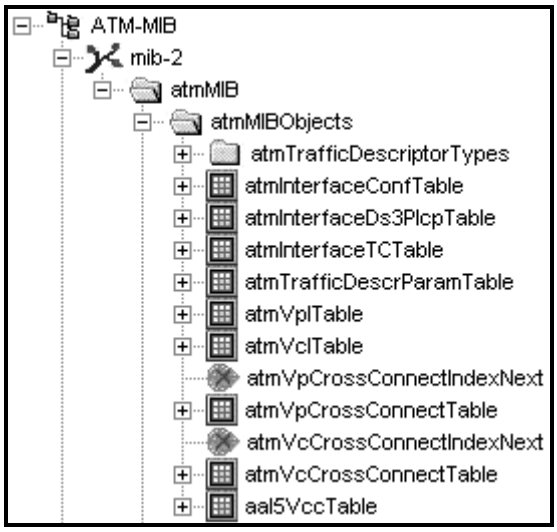


Figura 4.5. MIB AToM

MIBs devem ser pequenas e eficientes. O resultado da MIB AToM (Figura 4.5) [4], apesar de grande, é satisfatório, tendo em vista a complexidade de elementos ATM gerenciáveis. Sua proposta principal é gerenciar PVCs. Embora SVCs possam ser representadas nela, o completo gerenciamento de conexões chaveadas requer capacidades adicionais, que vão além do escopo da MIB AToM. Com finalidade específica de descrever estes objetos gerenciados suplementares para SVCs, IETF publicou um documento interno em novembro de 1995 [33].

4.3.1.1. Interface Information

Informações no Nível Físico ATM estão contidas em três tabelas conceituais:

- A tabela **atmInterfaceConfTable** (Figura 4.6.) [4] contém informações ATM específicas, disponíveis nas interfaces ATM, acrescentadas às informações gerais já presentes na Tabela **ifTable** da MIB II [34]. É, portanto, extensão da **ifTable**. A tabela de uma MIB SNMP pode ser estendida indexando-se a nova tabela com objetos da antiga. Assim, usando-se o mesmo valor do índice nas duas tabelas, pode-se acessar os dois objetos (Figura 4.7) [28]. A tabela **atmInterfaceConfTable** contém as seguintes informações:

- Número máximo de VPCs e VCCs suportados pela interface (PVCs e SVCs);
- Número de VPCs e VCCs configurados, em uso;
- Número máximo de bits ativos VPI (8) VCI (16);
- VPI e VCI usado para transporte de gerenciamento ILMI;
- Endereço ATM da Interface;
- Informações de vizinhança, incluindo nome da interface e endereço IP do vizinho para o qual esta interface se conecta;

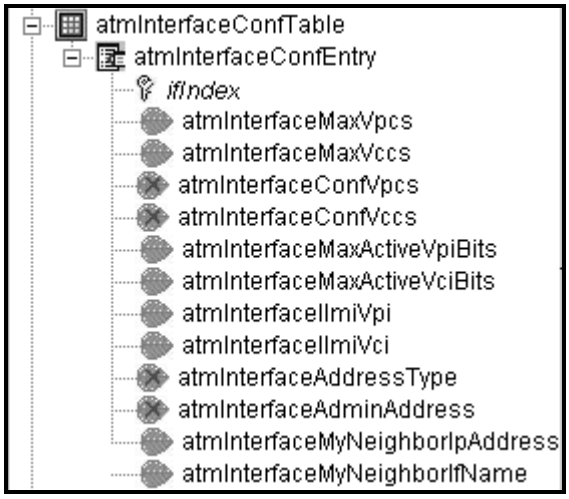


Figura 4.6. Tabela atmInterfaceConfTable

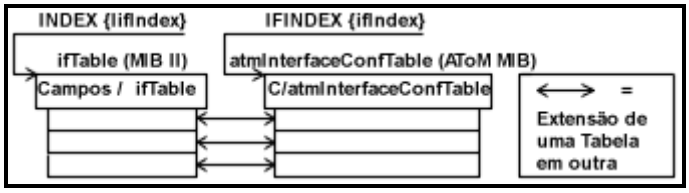


Figura 4.7. Extensão da ifTable através da atmInterfaceConfTable

- A tabela **atmInterfaceTCTable** (Figura 4.8) [4] também estende a tabela **ifTable**, com dois objetos de erro para a subcamada Convergência de Transmissão (TC) da camada Física. Ela só é válida se a interface ATM estiver usando a subcamada TC, como SONET [35] ou DS3 [36]. Esses objetos indicam a existência de problemas de delineamento de células.

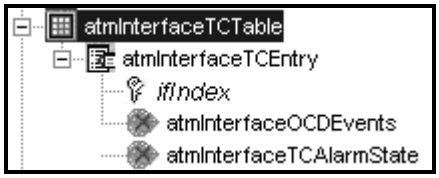


Figura 4.8. Tabela atmInterfaceTCTable

- A tabela **atmInterfaceDs3PlclTable** (Figura 4.9) [4] estende a tabela **ifTable** com objetos específicos para DS3 PLCP. Assim como a tabela anterior, somente é válida se DS3 estiver em uso. DS3 é um tipo específico da camada Física, e o Processo de Convergência da Camada Física (PLCP) é um método para transportar células no *Plesiochronous Digital Hierarchy* (PDH) [37], meio físico baseado em janelas. DS3 é um nível de multiplexação PDH particular com taxa de transmissão de 44.726 Mbits/seg [28].

Figura 4.9. Tabela atmInterfaceDs3PlclTable

4.3.1.2. Virtual Link Information

Informações dos enlaces virtuais estão contidas em três tabelas que, juntas, descrevem enlaces dos SVCs e PVCs presentes no dispositivo ATM gerenciado. Essas tabelas estão presentes nos sistemas finais (computadores) e intermediários (comutadores). Informações sobre VPLs e VCLs encontram-se em tabelas distintas. Informações sobre contrato de tráfego para cada enlace ficam armazenadas na terceira tabela.

As tabelas **atmVplTable e atmVclTable** são similares (Figura 4.10 e 4.11) [4]. Elas armazenam as seguintes informações:

- Identificador do enlace. VPI para VPL e, uma combinação VPI x VCI para VCL.
- Informações de status. O status do enlace virtual é representado por objetos AdminStatus, OperStatus e LastChange. AdminStatus é implementado apenas para VPLs/VCLs terminais de VPCs/VCCs, onde não há conexão cruzada (*cross-connect*) para outros VPLs/VCLs. Seus valores especificam o estado administrativo desejado, ou seja, tráfego habilitado (*up*) ou desabilitado (*down*). OperStatus indica o status operacional corrente do VPL/VCL. Ocorrendo uma falha no enlace, OperStatus deve apresentar estado "*down*" , enquanto AdminStatus deve permanecer "*up*". Finalmente, LastChange indica a hora em que o VPL/VCL tornou-se operacional.

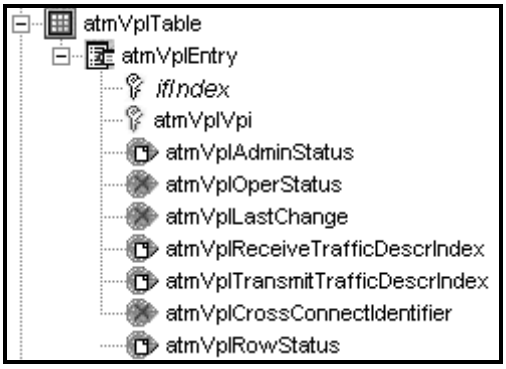


Figura 4.10. Tabela atmVplTable

- Índices descritores de tráfego. Um descritor de tráfego descreve o tráfego do enlace virtual, qualitativamente e quantitativamente. Descritores são resultantes da negociação de criação da conexão. Um objeto é criado para cada direção do fluxo do tráfego. A direção refere-se ao dispositivo ATM conectado no enlace. O tráfego recebido é registrado no objeto Receive e o tráfego transmitido no objeto Transmit. Objetos TrafficDescrIndex estão indexados na tabela **atmTrafficDescrParamTable** discutida a posterior. É possível descrever fluxos de tráfego assimétricos, tendo em vista que cada direção possui descritores próprios.

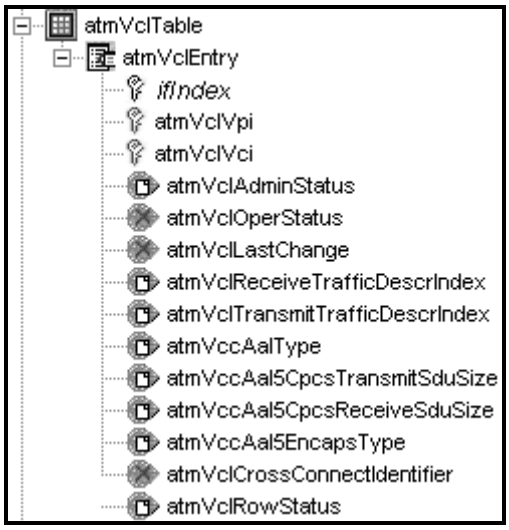


Figura 4.11. Tabela atmVclTable

- Identificador de conexão cruzada. Este objeto é implementado apenas para VPL/VCL que possui conexão cruzada com outro VPL/VCL pertencente ao mesmo VPC. Será descrito melhor na seção seguinte.

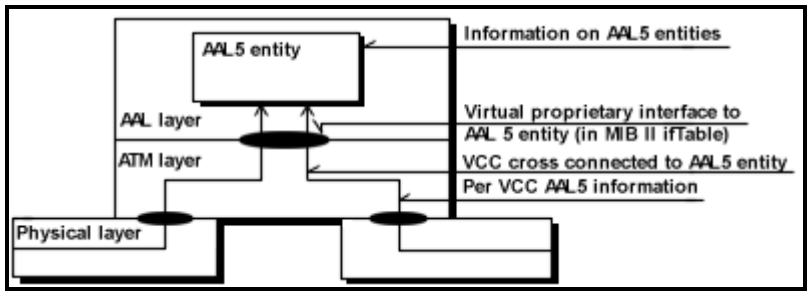


Figura 4.12. Esquema da AAL5 Modelada na MIB AToM

- Uma coluna *RowStatus*;
- Além destes objetos comuns à entrada VPLs e VCLs, há objetos na tabela **atmVclTable** relacionados à AAL. Estes objetos, específicos da AAL, especificam o tipo de AAL usado na VCC, e para AAL5, determina o tamanho máximo do SDU, em octetos, suportado na direção trasmite/recebe da VCC. A Figura 4.12 [28] ilustra como a MIB AToM modela AAL5. Uma entidade AAL5 possui interface virtual própria com uma entrada na

tabela **ifTable** da MIB II. Informações gerenciadas para a entidade AAL5 são mantidas na MIB II.

A tabela **atmTrafficDescrParamTable** contém informações dos tipos de descritores de tráfego ATM e os parâmetros associados (Figura 4.13) [4]. A Figura 4.14 [28] ilustra como os objetos descritores de tráfego nas tabelas **atmVpl/VclTable** apontam para entradas específicas da tabela **atmTrafficDescrParamTable**. Cada entrada desta tabela consiste de:

- Um índice para identificar a entrada da tabela;
- Uma coluna (*Type*) que descreve o tipo de tráfego;
- Cinco colunas c/ parâmetros de tráfego;
- Uma coluna indicando a Classe de QoS (1,2,3 e 4) e 0 para *besteffort*
    - Service Class A: CBR *video Service* e CES
    - Service Class B: VBR *video/audio Service*
    - Service Class C: *Connection-oriented data Service*
    - Service Class D: *Connectionless data*
- Uma coluna de *RowStatus.*

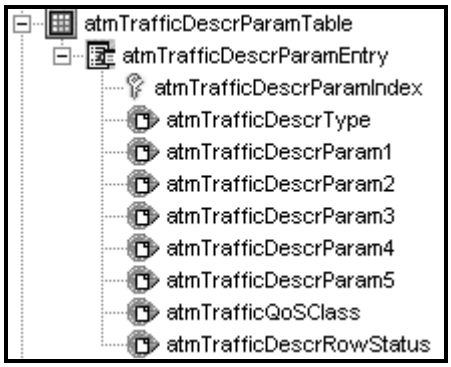


Figura 4.13. Tabela atmTrafficDescrParamTable

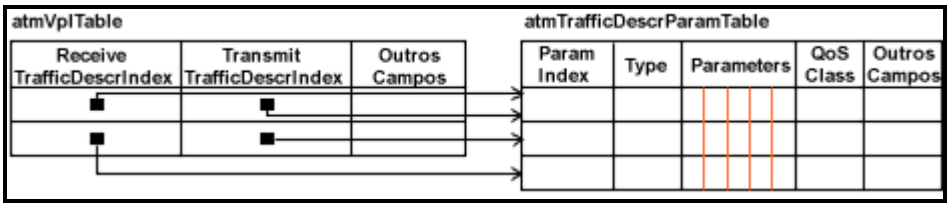


Figura 4.14. Enlaces da atmVpl/VclTable para atmTrafficDescrParamTable

4.3.1.3. Cross-connect Information

A informação de conexão cruzada descreve como enlaces virtuais são conectados entre si. Essas informações somente estarão presentes em dispositivos ATM que suportam a funcionalidade de conexão cruzada, tais com comutadores. Através deste grupo, pode-se descrever conexões ponto-a-ponto, ponto-a-multiponto e multiponto-a-multiponto.

VPLs e VCLs possuem tabelas separadas de conexão cruzada, **atmVpCrossConnectTable** e **atmVcCrossConnectTable** (Figura 4.15 e 4.16) [21] respectivamente, e, portanto, apresentam estrutura similar. Uma entrada na tabela de conexão cruzada sempre conecta dois enlaces virtuais. Conexões ponto-a-multiponto e multiponto-a-multiponto utilizam múltiplas entradas nesta tabela, com mesmo indexador. Cada entrada tem os seguintes componentes:

- O índice da conexão cruzada. VLs usam esse valor para identificar-se a uma conexão cruzada particular. Todas as conexões cruzadas possuem único valor de identificação. A MIB AToM possui mecanismo de identificadores, caracterizado por objetos com permissão de leitura chamados **CrossConnectIndexNext** para cada tabela de conexão cruzada. A cada leitura do objeto, um único valor é retornado para aplicação de gerenciamento. Isso garante valores unos para os índices, mesmo em caso onde múltiplos gerentes criam conexões cruzadas simultaneamente. Um gerente pode localizar informações da conexão cruzada usando esses valores para **ifIndex**, VPI, VCI e identificadores encontrados em **Vpl/VclTable** como índice da **Vp/VcCrossConnectTable**.
- Uma referência para VPL e VCL conectados por essa conexão cruzada. VPL é unicamente identificada pelo seu **ifIndex** e VPI, assim como VCL é unicamente identificado pelo seu **ifIndex**, VPI e VCI. Vpl/VclTable são indexadas pelos valores **ifIndex** e VPI/VCI. Com isso, gerentes de redes podem acessá-las diretamente. Para distinguir os dois VLs da conexão cruzada, eles são denominados VLs de baixo nível e VLs de alto nível, com a ordem determinada pelo valor numérico do **ifIndex**.

- Informação de status consiste de um **AdminStatus**, contendo o status da conexão cruzada, e um par **OperStatus** e **LastChange** para cada direção da conexão cruzada.

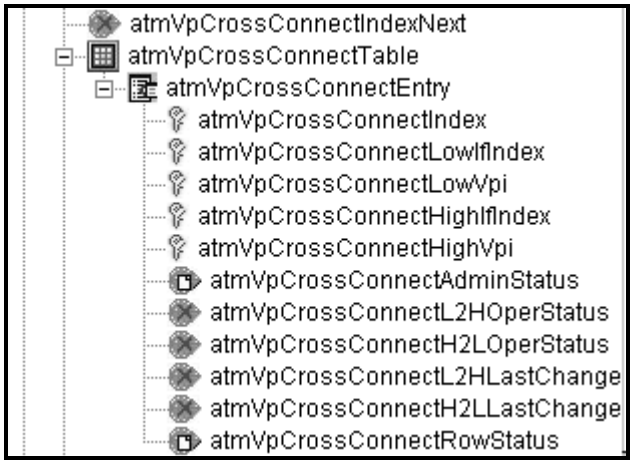


Figura 4.15. Tabela atmVpCrossConnectTable

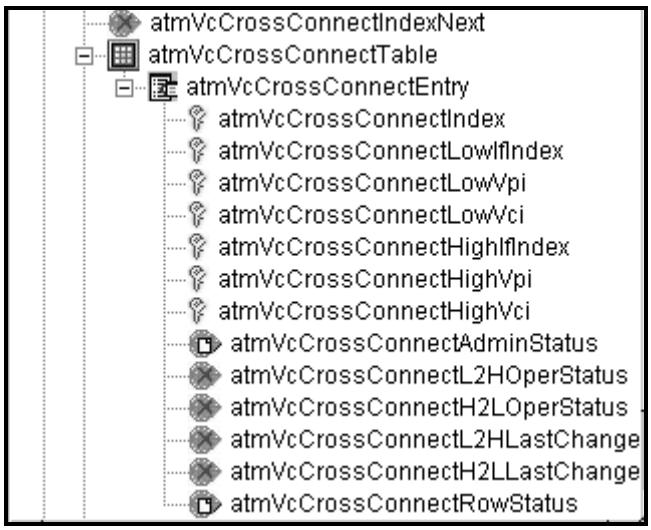


Figura 4.16. Tabela atmVcCrossConnectTable

A Figura 4.17 [28] apresenta uma conexão cruzada ponto-a-ponto entre dois VPLs. As setas conectando diferentes entradas das tabelas **atmVpCrossConnectTable** e **atmVplTable** são interpretadas a seguir. **AtmVpCrossConnectTable** é indexada pelo seu índice de conexão cruzada e os valores **Low ifIndex**, **Low VPI**, **High ifIndex** e **High VPI**. O identificador da conexão cruzada, e valores **ifIndex** e VPI de uma entrada na tabela

VplTable identificam unicamente uma entrada particular na tabela **VpCrossConnectTable**. **AtmVpCrossConnetTable** é indexada pelos valores **ifIndex** e valores VPIs. Usando aqueles dois objetos da **atmVpCrossConnectTable** a entrada correspondente na **atmVplTable** pode diretamente ser encontrada.

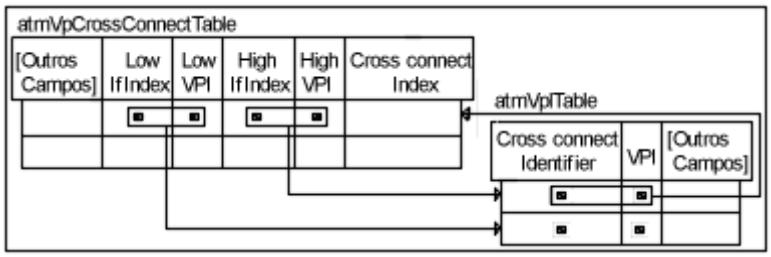


Figura 4.17. Conexões cruzadas VPLs

4.4. Modelos de Gerenciamento na MIB AToM

São dois os modelos de gerenciamentos na MIB AtoM: Gerenciamento de PVC e Gerenciamento AAL5 [21].

4.4.1. Gerenciamento de PVC

Um simples sistema final ou intermediário ATM não suporta toda dimensão fim a fim da conexão virtual. Normalmente, a conexão virtual é composta por múltiplos sistemas finais e intermediários, onde cada um suporta enlaces virtuais. Assim, cada sistema final possui a terminação da conexão virtual e os enlaces virtuais em suas interfaces externas, enquanto cada sistema intermediário por onde a passa a conexão virtual, suporta múltiplos enlaces virtuais nas interfaces externas e as conexões cruzadas daqueles enlaces virtuais pertencentes a ele. O gerenciamento fim a fim de uma conexão virtual é alcançado pela combinação do gerenciamento de suas partes individuais.

A gerência de conexões virtuais fim-a-fim (Figura 4.18) [21] é considerada a mais importante chave do gerenciamento de redes ATM. Tal gerenciamento é alcançado

através do controle apropriado de suas partes individuais, isto é, VLs. Uma VC está associada com um conjunto de descritores de tráfego que especificam características tais como parâmetros de tráfego, classe de QoS. VLs herdam as características das VCs, as quais fazem parte. Parâmetros de tráfego podem ser simétricos ou assimétricos em duas direções da conexão. Isto é, as duas direções podem, ou não, possuir as mesmas características. Nosso trabalho baseia-se neste tipo de gerenciamento provido pela MIB AToM.

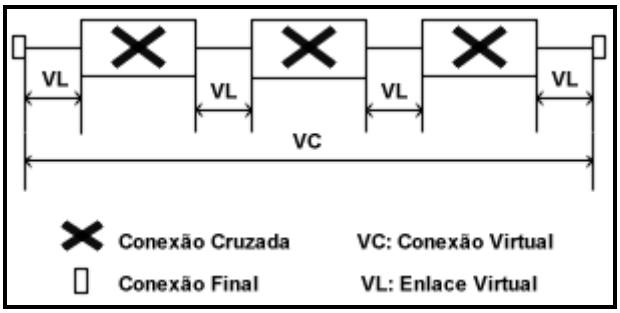


Figura 4.18. Modelo de Gerenciamento PVC

4.4.2. Gerenciamento AAL5

A função da camada AAL no ATM é mapear os protocolos de transferência de informações em ATM. AtoM MIB suporta o gerenciamento AAL5 utilizando dois modelos em computadores e comutadores ATM. O primeiro modelo, gerenciamento AAL5 em computadores, trata de VCCs terminadas em computadores onde a camada AAL5 está diretamente empilhada sobre a camada ATM (Figura 4.19) [21]. O segundo modelo, gerenciamento AAL5 em comutadores, é modelado usando ifTable através de uma interface virtual interna e o gerenciamento de desempenho AAL é suportado usando uma tabela adicional de conexão AAL5 na MIB AtoM. A associação entre o enlace virtual AAL5 na interface proprietária interna virtual e o enlace virtual ATM na interface ATM é derivada a partir da atmVcCrossConnectTable e da atmVclTable. O gerenciamento AAL5 no copmutador ATM é descrito na Figura 4.20 [21]. AAL5 trata somente de conexões virtuais que transmitem AAL5 e terminadas em entidades AAL5 dentro de comutadores que suportam canais de sinalização. VCCs dentro de UNIs ATM transmitindo AAL5 são

chaveadas pelo comutador ATM (entidade ATM) para VCs numa interface proprietária interna associada com o processo AAL5 (entidade AAL5) (Figura 4.21).

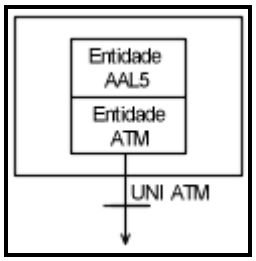


Figura 4.20. AAL5 p/computadores

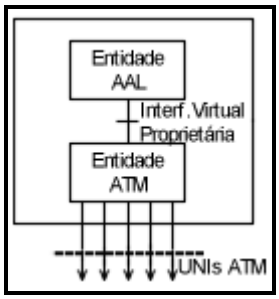


Figura 4.21. AAL5 p/comutadores

4.5. Conclusão do Capítulo

A padronização do gerenciamento ATM é necessária. Gerenciar redes ATM consiste de operação envolvendo equipamentos de diferentes fabricantes, o que dificulta tal tarefa, pois diferentes dispositivos suportam diferentes informações de gerenciamento.

SNMP pode ser usado como protocolo de gerenciamento de redes ATM. Isto pode ser observado através da análise citada anteriormente. SNMP apresenta vantagem da simplicidade de implementação e, devido à sua vasta utilização, é suportado pela maioria de dispositivos de rede, incluindo ATM.

Porém, gerenciamento de redes baseado em SNMP leva a arquitetura centralizada, apresentando desvantagens como falta de escalabilidade, desempenho e ineficiência de utilização dos recursos de rede. Gerenciamento por delegação e gerenciamento com AMs são alternativas para resolver estes problemas. Buscando a integração com AMs acreditamos alcançar solução eficiente para gerenciamento de redes ATM heterogêneas, fazendo uso das vantagens do SNMP, mas superando suas limitações.

Capítulo 5. Tecnologia de Agentes Móveis

5.1. Introdução

O conceito de AMs surgiu a partir do exame crítico da comunicação de computadores. Estimulado pela dificuldade da atual arquitetura da Internet em se adaptar ao ritmo de crescimento exponencial de seus usuários, uma nova pesquisa se tornou necessária, para satisfazer necessidades aparentemente contraditórias: o crescente aumento de tipos sofisticados de comunicação, sem estrangular os conteúdos que trafegam na Internet, na largura de banda disponível.

Historicamente, aplicações distribuídas são criadas fazendo uso da arquitetura "cliente/servidor". Nesse modelo, uma operação é dividida em duas partes na rede. O cliente faz requisições, a partir de um computador do usuário, a um servidor, que executa essas requisições, num grande e centralizado sistema. Um protocolo é seguido e, ambos, cliente e servidor, são programados para implementá-lo. Uma conexão de rede é estabelecida entre eles e um protocolo cumprido.

O modelo "cliente/servidor" apresenta vantagem de reduzir clientes a pequenas máquinas situadas remotamente e trabalhar bem com algumas aplicações. Porém, não funciona adequadamente em situações envolvendo grandes sistemas distribuídos, conexões de redes com baixa qualidade de serviço e, especialmente, volumosas aplicações de troca.

No sistema com um servidor central e vários clientes, o problema de escalabilidade é pequeno. À medida que múltiplos servidores são envolvidos, o problema se agrava rapidamente, ficando cada cliente com a tarefa de gerenciar e manter as

conexões com os diversos servidores. O uso de sistemas de dois níveis (*two-tier systems*) [38] ou de procuradores (*proxies*) apenas transfere essa problemática para a rede.

O emprego do modelo "cliente/servidor" exige boas conexões de rede. Primeiro, o cliente precisa estar ligado de maneira confiável ao seu servidor, porque somente mantendo-se conectado, ele pode ser autêntico e seguro. Segundo, o cliente necessita assegurar uma resposta previsível, já que suas requisições ao servidor precisam ser completadas. Terceiro, é necessária boa largura de banda pois, por natureza, o modelo "cliente/servidor" copia dados através da rede. Finalmente, o protocolo utilizado pelo cliente e pelo servidor é muito especializado e estático. Freqüentemente, procedimentos específicos do servidor são codificados no protocolo e se tornam parte da interface. Certas classes de tipos de dados são amarradas a esses procedimentos e o resultado final é uma versão específica da rede de uma interface de aplicação. Essa interface pode ser extensível, porém, com alto custo de recodificação da aplicação, para fornecer ao protocolo versão compatível e atualizada de software etc. Como as aplicações crescem e são incrementadas, a programação "cliente/servidor" rapidamente se torna obsoleta, necessitando de alternativas.

AMs solucionam algumas limitações inerentes ao modelo "cliente/servidor". Com AMs, o fluxo do controle se move através da rede, em vez de usar o esquema "*request/response"* do "cliente/servidor". De fato, todo nó é um servidor na rede de AMs, e o agente (programa) se move para o local, onde vai executar seus serviços. Por exemplo, o mesmo AM interage com o usuário via GUI para obter as chaves de requisições, viaja para o servidor de banco de dados e realiza sua consulta.

O relacionamento entre usuário e servidor está codificado dentro do AM, sem precisar estar espalhado através de clientes e servidores. O próprio AM cria o sistema, em vez da rede ou do administrador. A administração do servidor se transforma em matéria simples de gerência e monitoramento local de armazenamento.

O problema do sobrecarregamento de redes é diminuído por várias razões. O tempo de conexão é reduzido ao tempo da viagem do AM. A conexão é simplesmente um canal, pois o AM leva suas próprias credenciais, não ficando o usuário sujeito ao engano (*spoofing*). Nenhuma mensagem do tipo "*request/response"* flui através da conexão, o AM se move levando os procedimentos e os argumentos em cada viagem [39]. Isso permite eficiência e otimização em vários níveis.

Finalmente, e mais importante, nenhum protocolo da camada de aplicação é criado com o uso de AMs. Logo, a compatibilidade é provida por qualquer aplicação baseada em AMs. O aumento da compatibilidade se transforma em norma, e não em problema a ser resolvido. Atualizar ou reconfigurar uma aplicação pode ser feito de maneira sem precisar estender ao cliente. Servidores podem ser atualizados, serviços movidos, balanços de armazenamento interpostos e políticas de segurança reforçadas, sem interrupções ou revisões da rede e dos clientes.

5.2. Perspectiva Histórica

O princípio central de organização das redes de comunicação de computadores, Chamada de Procedimento Remoto (*Remote Procedure Calling* - RPC), foi concebido em 1970 e utilizado na comunicação de computadores, para habilitá-los a realizarem chamadas em outras máquinas [39]. Cada mensagem transportada na rede utiliza uma requisição (*request)* ou reconhecimento (*acknowledge*). Uma requisição inclui dados com os argumentos de processamento e a resposta (*response)* com seus resultados. O procedimento é interno à máquina que o realiza (Figura 5.1) [39].

Dois computadores se comunicando via RPC utilizam procedimento de acesso remoto e tipos de argumentos e resultados. Isso constitui um protocolo. Um cliente efetua seu trabalho através de uma série de chamadas remotas. Cada chamada é constituída pelo envio de uma requisição do usuário ao servidor e pelo envio da resposta do servidor para o

usuário. Exemplo, para apagar arquivos com mais de dois meses de um servidor, o computador do usuário deve fazer duas chamadas: uma para pegar os nomes e as idades dos arquivos pertencentes aos usuários e uma segunda para cada arquivo a ser apagado. A análise que decide que arquivos podem ser apagados é feita no computador do usuário. Se ele decidir excluir *n* arquivos, o computador do usuário deve enviar ou receber um total de $2(n + 1)$ mensagens [39].

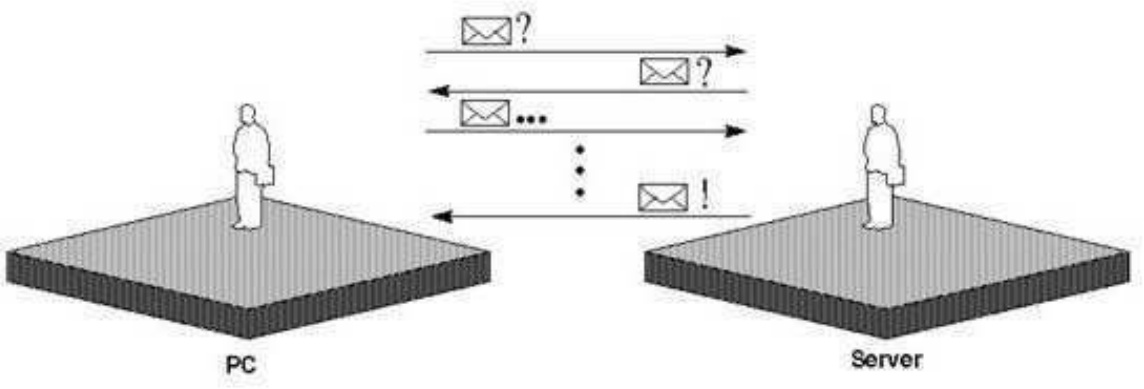


Figura 5.1. Chamadas request e response

RPC requer essencialmente que cada interação entre o computador do usuário e o servidor consista de dois atos de comunicação: um para pedir ao servidor para realizar o processo e outro para confirmar qual servidor fez. Por isso, numa interação cada série de envios requer uma série de confirmações tornando as chamadas RPCs em constantes necessitando da reciprocidade entre usuários e servidor.

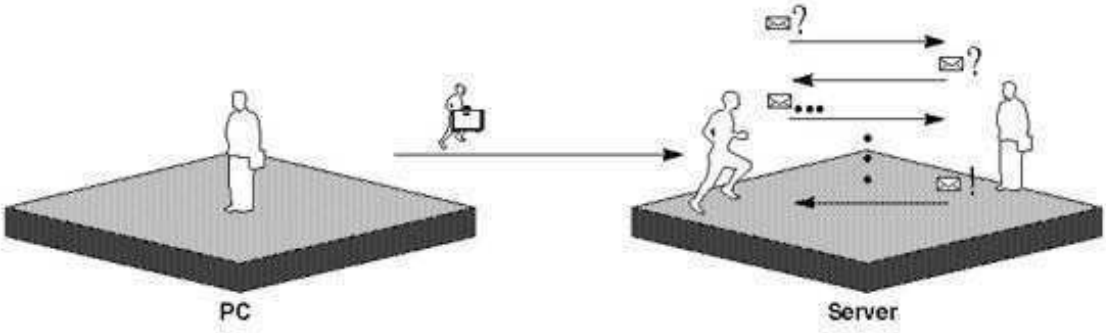


Figura 5.2. Definindo instruções

Uma alternativa para o RPC é a Programação Remota (*Remote Programming* RP) [39]. Para o paradigma RP, a comunicação entre computadores não é apenas a necessidade de habilitar uma máquina a fazer chamada em outra, mas também a de fornecer os procedimentos a serem realizados. Com RP, cada mensagem transportada na rede é composta de um procedimento e seus argumentos (dados) a serem processados no computador remoto. Portanto, o computador que envia a mensagem inicia o processamento enquanto o computador que recebe e processa os argumentos (dados) que definem situações particulares, ou seja, os estados correntes do procedimento.

Dois computadores se comunicando através do paradigma RP utilizam as instruções localizadas no procedimento e os tipos de dados que definem seu estado. Esse acordo constitui uma linguagem. A linguagem inclui instruções que permitem o procedimento tomar decisões, examinar seus estados, modificá-los e chamar os procedimentos fornecidos pelo computador que os recebe. Tais chamadas são preferencialmente locais, e não remotas. Esses procedimentos são denominados de agentes móveis para enfatizar, na máquina remota, que eles representam o computador que os envia.

Por exemplo, um servidor envia a outro um agente cujo procedimento executa localmente as requisições (p.ex. apagar) baseadas nos seus estados (p.ex. arquivos com mais de dois meses). Assim, para apagar os arquivos descritos no exemplo anterior de RPC, não importando a quantidade, basta uma mensagem transportando o agente entre computadores. O agente enviado realiza sua tarefa decidindo, no local, quais arquivos serão apagados (Figura 5.2) [39].

Usando RP, o computador do usuário interage com seu servidor sem sobrecarregar a rede, visto que o agente já tenha sido transportado entre eles. Esse tipo de operação surge alcançando novos passos de incrementação sem a sobrecarga na largura de banda. De fato, RP sugere a realização de muitas operações utilizando menos largura de banda.

A vantagem da utilização da RP sobre a RPC pode ser analisada de duas perspectivas diferentes: a primeira quantitativa e tática; a segunda qualitativa e estratégica.

5.2.1. Vantagem Quantitativa e Tática

A vantagem quantitativa e tática imediata da RP é o desempenho [39]. Quando um computador usuário tem um trabalho a ser feito no servidor, em vez de solicitar comandos pela rede, ele envia um agente ao servidor, para realizar sua tarefa localmente. A rede é utilizada apenas para transportar pequenas mensagens.

A vantagem de desempenho na RP depende, em parte, da rede: quanto mais baixa a disponibilidade ou vazão, quanto maior o custo ou a latência, melhor a vantagem. A conexão padrão por telefone apresenta maior oportunidade de visualização, da vantagem do uso desse novo paradigma do que a Ethernet. Na conexão sem fio, também, é grande a vantagem da utilização da RP.

Computadores pessoais são exemplos de computadores conectados na Internet apenas ocasionalmente. RP permite ao usuário, com tal sistema, delegar um agente para realizar uma ou mais tarefas. O sistema não necessita estar conectado, enquanto o agente trafega na rede. Dessa forma, RP permite aos computadores, conectados ocasionalmente na rede, realizarem tarefas consideradas impraticáveis com o uso do RPC.

5.2.2. Vantagem Qualitativa e Estratégica

A principal vantagem qualitativa e estratégica da RP é a customização [39]. Agentes Móveis despertam o interesse dos desenvolvedores de softwares a estenderem as funcionalidades oferecidas pelos fabricantes de softwares para servidores. Como registrado anteriormente, se o software de servidor fornece a possibilidade de listar arquivos de usuários e outro de apagar arquivos por nome, o usuário pode, efetivamente, adicionar um

procedimento para apagar todos os arquivos de determinada idade. Esse novo procedimento toma forma de agente adaptando um servidor para determinado usuário.

O paradigma RP não só muda a divisão do trabalho ao longo dos desenvolvedores de softwares, mas também facilita a instalação de qualquer software. Diferente de aplicações *standalone,* que popularizam o computador pessoal, computadores pessoais empregam aplicações de comunicação com componentes residentes nos servidores. Os componentes dos servidores de uma aplicação baseada em RPC para computadores pessoais são estaticamente instalados pelo usuário. Por outro lado, os componentes do servidor de aplicação, baseados em RP, são dinamicamente instalados a partir do computador do usuário, se cada componente for um agente.

A vantagem do RP já é significante na pequena rede de uma empresa. Nas redes públicas, onde servidores são operados e pertencem a provedores de serviços públicos, ela se torna ainda mais vantajosa. Introduzir uma nova aplicação, baseada em RPC, requer uma decisão difícil da parte do provedor de serviço. Com uma aplicação baseada em RP, todas decisões requeridas são individuais de cada usuário. A RP faz da Internet, assim como de um computador pessoal, uma plataforma.

5.3. Conceitos Básicos

Esta seção tem por objetivo apresentar os conceitos relevantes ao ambiente de agentes móveis. Para isso, nos basearemos na proposta de Facilidades de Interoperabilidade de Sistemas de Agentes Móveis (*Mobile Agent System Intereoperability Facilities – MASIF*) [40], considerando os aspectos gerais e os fatores de interoperabilidade entre os sistemas existentes.

5.3.1. Agente

Agente é um programa de computador que age autonomamente em benefício de uma pessoa ou organização. Atualmente, a maioria dos agentes é programada através de uma linguagem como Java devido sua portabilidade. Cada agente tem seu processo de execução (*thread*). Assim, tarefas podem ser realizadas a partir de sua própria iniciativa.

5.3.2. Agente Estacionário

Agente Estacionário é executado apenas no sistema onde ele inicia sua execução. Se o agente precisa de informações, não constantes naquele sistema, ou necessita interagir com um agente num sistema remoto, ele usa o mecanismo de transporte de comunicação tal como RPC. A comunicação necessária para agentes estacionários pode ser encontrada sistemas de objetos distribuídos, tais como CORBA, DCOM e RMI.

5.3.3. Agente Móvel

Agente Móvel não está necessariamente ligado aos sistemas onde ele inicia sua execução. Tem a habilidade de se transportar, a partir de um sistema conectado na rede, para outro. Esta característica permite ao agente se mover até o sistema contendo os objetos utilizados na interação. Além disso, o agente pode usar os serviços do sistema destino.

Portanto, Agente Móvel é o programa executável que pode migrar, durante sua execução, de uma máquina para outra em redes heterogêneas. Em cada máquina, o agente móvel pode interagir com outros agentes, estacionários ou móveis, e outros recursos para realizar sua tarefa.

5.3.4. Estado do Agente

Quando o agente viaja, seu estado e código são transportados com ele. Dentro deste contexto, o estado do agente pode ser seu estado de execução, ou seus valores dos atributos que determinam sua execução quando resumida no sistema de agente destino. Os valores dos atributos de agentes incluem o estado do sistema de agentes associado com o agente (p.ex. tempo de vida).

5.3.5. Estado de Execução do Agente

O estado de execução do agente é seu estado de execução, incluindo contadores e pilha de janelas.

5.3.6. Autoridade do Agente

A autoridade do agente identifica a pessoa ou organização para quem o agente atua. Uma autoridade deve ser autenticada. Por isso, o servidor deve conhecer a autoridade de qualquer procedimento de tratamento de arquivos. Essa necessidade, importante em qualquer segurança de rede, é a mesma para procedimentos estáticos ou móveis [40].

Um sistema de agente pode fornecer ou negar serviços personalizados a agentes, baseados na sua autoridade.

5.3.7. Nome de Agente

Agentes requerem nomes para serem identificados em operações de gerenciamento e podem ser localizados através do serviço de nomes. Agentes são nomeados pela sua autoridade, identidade e tipo de sistema de agentes. Sua identidade é o único valor dentro do escopo da autoridade, que identifica uma instância particular do agente. A combinação

da identidade, autoridade e tipo de sistema possui sempre um único valor global. Assim, o nome pode ser usado como chave em operações referentes a uma instância particular do agente.

5.3.8. Sistema de Agente

Sistema de agente é a plataforma que pode criar, interpretar, executar, transferir e terminar agentes (Figura 5.3) [40]. Como um agente, o sistema de agente está associado com uma autoridade identificadora da pessoa ou organização para quem atua. Um sistema de agente é unicamente identificado pelo seu nome e endereço. Um Computador pode conter um ou mais sistemas de agentes.

Figura 5.3. Sistema de Agente

5.3.9. Tipo do Sistema de Agente

Tipo de sistema de agente descreve o ambiente do agente. Por exemplo, o tipo de sistema de agente usado neste trabalho é o Concordia, implementado pela *Mitsubishi Eletric Information Technology,* suporta Java como linguagem de agente, usa itinerário para viagem e utiliza a serialização de objetos Java. Esta especificação reconhece o tipo de sistema de agente suportando múltiplas linguagens e, linguagens suportando múltiplos métodos de serialização.

5.3.10. Interconexão entre Sistemas Agentes

Toda comunicação entre sistemas agentes se dá através da infra-estrutura de comunicação. O administrador de região define os serviços para comunicação entre regiões (Figura 5.4) [40].

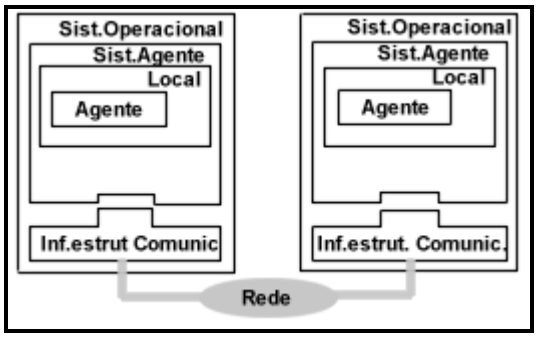


Figura 5.4. Interconexão de Sistemas de Agentes

5.3.11. Local

Quando um agente é transferido, viaja entre vários ambientes de execução chamados locais. Local é uma área dentro do sistema agente, na qual o agente pode executar. Esta área provê diversas funções, dentre elas o controle de acesso. Local de origem e local de destino podem residir, ou não, no mesmo sistema agente suportando ambientes de agentes iguais.

5.3.12. Região

Região é o conjunto de sistemas agentes contendo mesma autoridade, mas não necessariamente o mesmo tipo de sistema agente. O conceito de região permite mais de um sistema agente representar a mesma pessoa ou organização. Regiões permitem escalabilidade porque podem distribuir o armazenamento ao longo de múltiplos sistemas agentes.

A região provê nível de abstração para comunicação de clientes provenientes de outras regiões. Quando necessita contatar um agente ou um sistema agente, o cliente não precisa se preocupar com sua localização. O cliente possuindo o endereço para região e o nome do agente ou local, torna possível contatar ou fazer a comunicação.

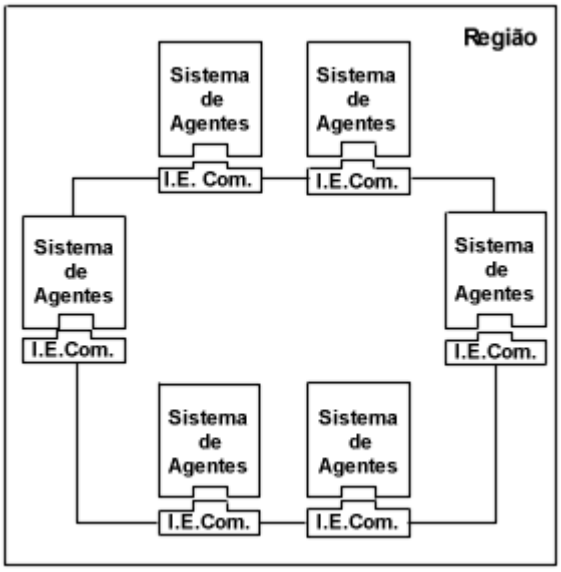


Figura 5.5. Arquitetura da Região

O agente pode ter a mesma autoridade da região na qual está residindo e executando correntemente. Significa que o agente representa mesma pessoa ou organização que a região. Normalmente, a configuração da região garante vasto conjunto de privilégios a tal agente, em relação a outro residente com diferente autoridade. Exemplo, um agente que possui autoridade da região pode garantir privilégio administrativo.

A região interconecta sistemas agentes dentro de suas fronteiras e habilita a transferência ponto-a-ponto de informações entre eles (Figura 5.5) [40]. Cada região contém um ou mais pontos de acesso, formando através de suas interconexões a rede.

5.3.13. Serialização

Serialização é o processo de armazenamento de agentes, de forma serial. A chave para armazenar e recuperar agentes está representando serialmente o estado do agente e é suficiente para reconstruí-lo. A forma serial deve estar habilitada a identificar e verificar as classes a partir das quais os campos foram salvos.

5.4. Padronização de Sistemas Agentes

Atualmente, grande número de plataformas de AMs tem sido desenvolvido baseado em diferentes tecnologias e linguagens de programação, podendo ser executado em diferentes sistemas operacionais. Novas linguagens estão sendo criadas com o fim específico de suportar AMs. Entretanto, este novo desenvolvimento apresenta fortes tendências para linguagens como Java, servindo de fundamento para construção da maioria das plataformas de AMs. Padrões e interoperabilidade entre plataformas de AMs são encontrados em [40] e [41].

Ao longo do tempo, vários requisitos têm sido identificados em função de pesquisas realizadas e desenvolvimento de atividades [42]. Esses requisitos, relevados por qualquer plataforma de desenvolvimento de agentes, incluem suporte ao gerenciamento, segurança, mobilidade, identificação única, transação e comunicação (Figura 5.6) [42]. Requisitos adicionais podem surgir, dependendo do nível da aplicação.

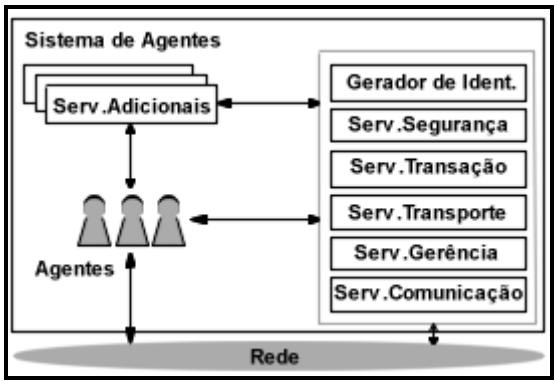


Figura 5.6. Capacidades Básicas das Plataformas de AMs

5.4.1. Linguagem de Descrição de Interface (*Interface Description Language – IDL*) MASIF

AMs apresentam bastante diferenças na arquitetura e implementação de seus sistemas. Esta diferença impede sua interoperabilidade e, portanto, rapidez de proliferação. Para contornar este problema, alguns aspectos da tecnologia de AMs devem ser padronizados.

Com a padronização, é possível obter a interoperabilidade entre os vários fornecedores de sistemas AMs. A interoperabilidade se torna mais alcançável se ações como transferência de AMs, transferência de classes e gerenciamento de AMs são padronizadas.

A padronização não se preocupa com a especificação de operações locais de AMs, tais como interpretação, serialização, execução etc. Estas ações são implementadas de maneira específica, não havendo razão para limitar a implementação de sistemas de AMs numa arquitetura singular.

A proposta MASIF [40] trata da interoperabilidade entre plataformas de AMs de diferentes fornecedores, não necessitando, para tanto, de mudanças radicais. As especificações são usadas como aditivos aos sistemas já existentes. A seguinte lista compreende os requisitos obrigatórios, identificados em RFP3 [43].

- Organização dos pacotes codificados (p.ex. agentes) para movimentação;
- Forma de codificação dos agentes para transporte, visto que utilizarão a infra-estrutura ORB;
- Transporte do AM a partir de um sistema de AMs para outro;
- Registro e invocação do sistema de AM em tempo de execução;
- Associação de nomes únicos e absolutos para identificar e localizar AMs;
- Segurança de AMs em tempo de execução.

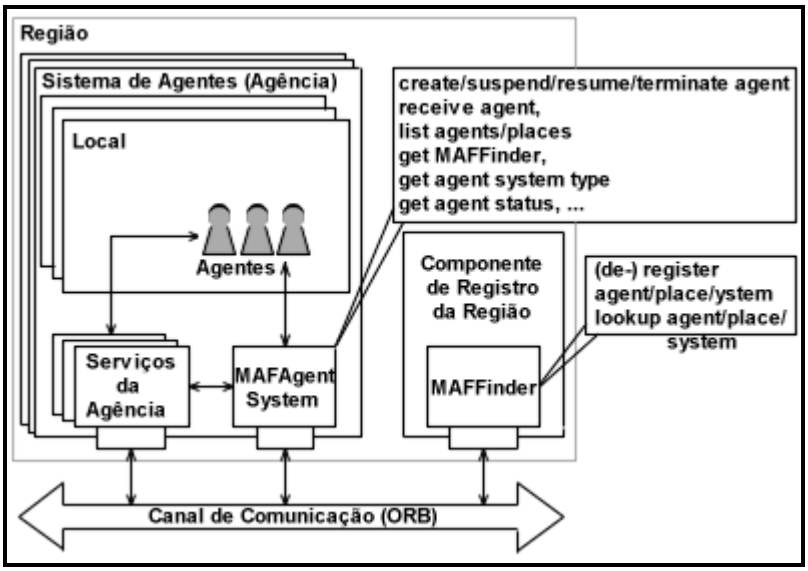


Figura 5.7. Arquitetura da Plataforma MASIF

Adicionalmente, requisitos opcionais foram definidos pela RFP, cobrindo entre outros, serviços inicialização, finalização e suspensão da execução de AMs, além de facilidades de monitoramento em tempo de execução do sistema de AMs.

Duas interfaces representam o coração da padronização MASIF: MAFAgentSystem e MAFFinder. São definidas, preferencialmente, no nível de sistemas de agentes, a fim de resolver problemas de interoperabilidade (Figura 5.7).

A interface MAFAgentSystem está associada com cada sistema de agentes concordantes com MASIF, provendo operações de gerenciamento e transferência de agentes. Define as operações de agentes, incluindo recebimento, criação, suspensão e finalização. Define, também, operações de registro e localização de agentes, sistemas de agentes e locais. Como parte do sistema de agentes concordante com MASIF, o objeto MAFAgentSystem interage internamente com serviços específicos e provê a interface CORBA associada para usuários externos.

A interface MAFFinder é um serviço de nomes. Está associada com a região, isto é, conjunto de sistemas de agentes. É parte do componente de registro da região que suporta a localização de agentes, agências e locais, no escopo da região. Antes de um cliente solicitar à MAFFinder para encontrar um objeto, deve obter sua referência.

As seguintes funcionalidades são cobertas pelas interfaces concordantes MASIF, devido aos requisitos definidos no RFP3 MASIF:

- O Gerenciamento de Agente permite a um sistema de agentes controlar agentes de outro. Compreende a criação, finalização, suspensão e reinício de agentes. A interface MAFAgentSystem provê diversos métodos para esse fim.
- A Perseguição de Agente permite a investigação de agentes registrados pela MAFFinder de diferentes sistemas de agentes. Agentes, agências e locais são registradas no componente de registro da região através da interface MAFFinder. Enquanto agências e locais são registrados somente durante seu tempo de vida, agentes móveis são desregistrados antes de cada migração e novamente registrados, no novo local, após a migração. Desta maneira, o componente de registro da região conhece a localização de cada agente a qualquer tempo.
- Para o Transporte de Agente, a interface MAFAgentSystem oferece dois métodos. O primeiro transfere o estado do agente e os dados requisitados e o segundo para restaurar o código do agente, se necessário.
- Atribuição de Nomes a Agências e Agentes padronizada habilita agências e agentes a se identificarem mutuamente. Quando uma operação de gerenciamento é invocada, o gerente deve ser identificado. Portanto, a sintaxe e a semântica devem ser padronizadas. Essa padronização permite um sistema de agente determinar se ele suporta a entrada de determinado agente, além de permitir a identificação mútua entre dois agentes.
- Tipo de Sistema de Agentes e Sintaxe de Localização provêem informação sobre aspectos importantes de sistemas de agentes específicos, tais como a linguagem de implementação utilizada. Antes que o agente possa migrar, deve determinar se o sistema de agentes de destino suporta sua execução.

5.5. Conclusão do Capítulo

Este capítulo tem foco principal nos AMs como tecnologia emergente para confrontar novos cenários estabelecidos pela rápida difusão de informações em sistemas distribuídos na Internet. Embora tecnologia ainda não totalmente madura, podemos afirmar que AMs têm impacto positivo no projeto de aplicações que requeiram larga disseminação de informações através da rede. A disponibilidade de código portável e aumento da eficiência na atuação são bastante satisfatórios e, conseqüentemente encorajadores do seu uso.

Entretanto, ainda há vários pontos em aberto, os quais devem ser logo abordados a fim de garantir rápida divulgação e utilização dos AMs, como padronização de plataformas, integração com sistemas legados, segurança etc. A proposta de padronização MASIF, conforme visto anteriormente, trata dos problemas de interoperabilidade de plataformas de AMs. A integração com sistemas legados existentes, abordada neste trabalho, é elementar para a sobrevivência desta emergente tecnologia. Várias são as pesquisas sobre segurança de AMs, porém fogem do escopo desta dissertação.

Capítulo 6. Concordia

6.1. Introdução

Hoje, o ambiente competitivo de negócios exige informação disponibilizada sem interrupções, em qualquer local e independente de computadores ou dispositivos de comunicação específicos. Usuários móveis podem estar trabalhando em casa ou em movimento, a caminho do escritório. Eles podem utilizar *notebooks*, *desktops* etc., ficando o acesso à informação ligado apenas à conexão.

A *Mitsubishi Eletric Information Technology Center America* criou o sistema Concordia [3], com o objetivo de desenvolver, executar e gerenciar aplicações de AMs, em redes, para acessar informações, a qualquer hora, local e dispositivos suportando Java.

Aplicações Concordia realizam acesso a dados, executam decisões e notificações, oferecendo ao usuário as informações e as respostas requisitadas.

O sistema Concordia é considerado uma infra-estrutura de *middleware*, para desenvolver aplicações distribuídas. Torna a programação da aplicação distribuída fácil e reduz, bastante, o tempo de desenvolvimento.

O Concordia provê todas as funções básicas do sistema de AMs:

- Comunicação eficiente;
- Fácil desenvolvimento de aplicações distribuídas;
- AMs assíncronos e autônomos;
- Desenvolvimento flexível e escalável de sistemas distribuídos.

Além do mais, os AMs Concordia possuem as seguintes características que os distinguem dos outros sistemas de AMs:

- Adaptam-se dinamicamente, mudando sua trajetória e execução, baseados na comunicação com outros agentes;
- São naturalmente heterogêneos, independentes de plataforma e nível de transporte;
- São robustos e tolerantes a falhas;
- Podem, facilmente, se integrar com aplicações legadas;
- Provêem suporte à segurança, pois, os AMs são criptografados e só podem acessar recursos permitidos pelos servidores.

Concordia consiste de múltiplos componentes, todos escritos completamente em Java, juntos provendo completo e robusto ambiente para aplicações.

O sistema Concordia é composto de uma Máquina Virtual Java (*Java Virtual Machine* - JVM), um servidor Concordia e pelo menos um AM. JVM pode estar em qualquer máquina: é um ambiente padrão. O servidor Concordia é um programa Java executado na JVM, e em qualquer outro nó da rede, por onde o AM necessite viajar. O AM é, também, um programa Java, gerenciado pelo servidor Concordia, incluindo seu código, dados e movimentos.

Normalmente, há vários servidores Concordia, um em cada nó da rede. Quando necessário, o servidor Concordia conecta-se sob demanda, com outro, para transferir AMs, de maneira segura e confiável. A transferência do AM é iniciada com a invocação de métodos do servidor Concordia, que cria imagem persistente do AM a ser transferido. O servidor Concordia analisa o objeto denominado Itinerário (*Itinerary*), criado pelo AM, para determinar o destino apropriado. Esse destino é contatado e a imagem do AM transferida, onde é armazenada, de forma persistente, antes de ser reconhecida. Desta maneira, a confiabilidade da transferência é garantida.

Depois de transferido, o AM é enfileirado para execução no computador destino. Isso acontece instantaneamente, mas, possivelmente, sujeito a certos obstáculos administrativos. Quando novamente executado, o AM é reiniciado no novo nó, de acordo com o método especificado no seu itinerário e carrega consigo os objetos, com as requisições do programador. Suas credenciais de segurança são transferidas, automaticamente, e seu acesso a serviços está sob controle administrativo local.

O trabalho executado pelo AM depende da sua proposta, ou seja, do código pelo qual foi programado para realizar. Geralmente, AMs tem vários componentes, assim como qualquer programa os tem. O AM pode ser iniciado interativamente, através do *prompt* do usuário, procurar a informação viajando pelo servidor e realizar sua função. Pode ser também, simplesmente, um tipo de *daemon* remoto, tal como um filtro de caixa de correio ou notificação de remetente. Como seus métodos são completos, os AMs movem-se através dos nós Concordia, com diferentes propostas e itinerários. Em todos os casos, o AM Concordia é autônomo e determina suas próprias operações.

As próximas seções deste capítulo são de grande importância para o entendimento deste trabalho, pois somente conhecendo o funcionamento do sistema Concordia, pode-se compreender o protótipo do nosso sistema de gerenciamento, proposto no capítulo posterior.

6.2. Componentes do Concordia

Abaixo serão ilustrados os diversos componentes do sistema Concordia. Todos os seus componentes são completamente codificados na linguagem Java.

O sistema Concordia contém inúmeros componentes, criando juntos, uma completa área de trabalho para os agentes móveis [44]. O servidor Concordia é o maior

bloco, dentro do qual, residem vários gerentes Concordia. Alguns componentes possuem interface e, em qualquer caso, cada um é responsável por uma porção do projeto, de maneira modular e extensível (Figura 6.1).

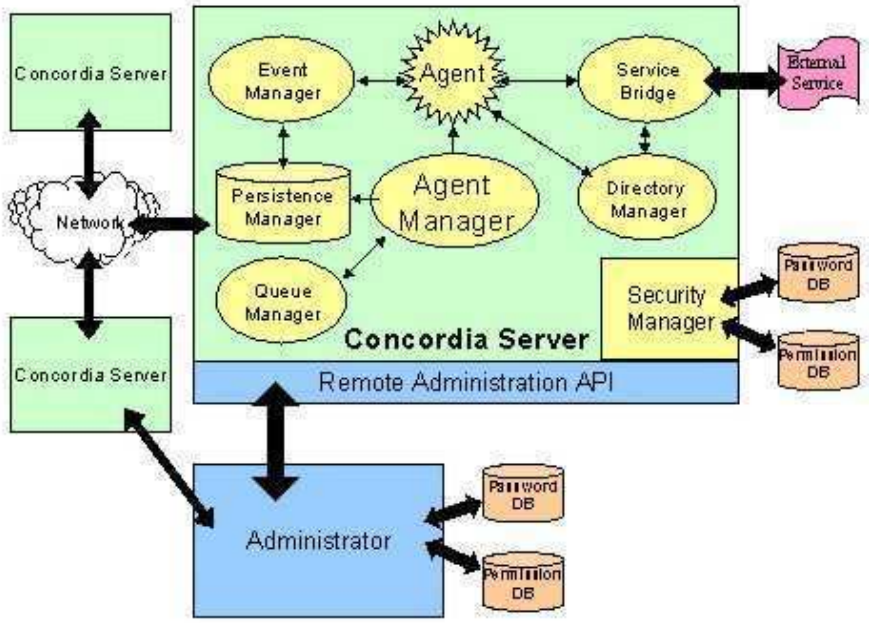


Figura 6.1. Visão do Concordia

.

O agente móvel Concordia é um pouco diferente do programa Java "não-móvel". A diferença vem em estruturar a aplicação, de maneira a tirar melhor proveito das facilidades do Concordia. O desenvolvimento, em ambiente Concordia, torna-se substancialmente fácil, com o uso do Java. A portabilidade da JVM, combinada com a administração avançada, faz do Concordia uma plataforma de execução de AMs bastante interessante de ser considerada.

São componentes do sistema Concordia: Concordia Server, Agent Manager, Administrator, Security Manager, Persistence Manager, Queue Manager, Directory Manager, Service Bridge e Agent Tools Library.

6.2.1. Concordia Server

O *Concordia Server* é o nome completo, para o componente instalado e executado numa máquina, pertencente a uma rede Concordia. Cada nó, no Sistema Concordia, consiste de vários componentes executados na JVM, oferecendo infraestrutura para desenvolvimento e gerenciamento de aplicações de AM. O principal objetivo é prover mobilidade, suporte a colaboração, persistência de estado, transmissão confiável e segurança para AMs. A arquitetura Concordia consiste do conjunto de classes Java para execução do servidor, desenvolvimento de aplicação e ativação do AM.

6.2.2. Agent Manager

O *Agent Manager* oferece a infra-estrutura de comunicação necessária para os agentes serem transmitidos e recebidos, pelos diversos nós da rede. A transferência do AM é iniciada invocando métodos deste componente. O *Agent Manager* envia AMs ao mesmo componente de outro Sistema Concordia. Ele simplifica a interface de rede, de forma que o programador de agentes não precisa conhecer detalhes específicos da rede nem programar qualquer interface de rede. Ele cria e destrói agentes e provê ambiente no qual os agentes podem ser executados. Concordia estende a interação do agente para suportar duas formas de comunicação: eventos assíncronos distribuídos e colaboração de agentes. Eventos assíncronos distribuídos são agendados e gerenciados pelo componente *Event Manager*, enquanto colaboração de agentes requer especificação de objeto para esse fim, através da programação da aplicação com utilização de classes próprias do Concordia [45]. Colaboração permite interação de agentes, bem como alteração de seus estados.

6.2.3. Administrator Manager

A administração da rede Concordia é feita através do *Administration Manager*, em cooperação com os serviços Concordia, executando nos vários nós da rede, sob a administração. O *Administration Manager* gerencia todos os serviços do Concordia e suporta administração remota, a partir de ponto central único, embora, mais de um possa

ser empregado. Possui uma interface com o usuário permitindo iniciar e interromper outros servidores na rede Concordia. Também monitora o progresso do AM em toda a rede e mantém estatísticas do sistema.

6.2.4. Security Manager

O *Security Manager* é responsável pela identificação dos usuários, autenticação de seus agentes, proteção dos recursos do servidor e garantia de segurança e integridade dos agentes e seus objetos de dados acumulados ao longo da viagem na rede. O *Security Manager* é também responsável pela autorização do armazenamento dinâmico de classes Java, que satisfazem a necessidade dos agentes. O *Security Manager* tem um componente de interface de usuário, para configurar e monitorar atributos de segurança, de vários usuários e serviços conhecidos pelo Concordia. Essa função de interface de usuário é integrada à interface do *Administration Manager*.

Credenciais seguras, usadas pelo *Security Manager*, são provenientes de várias origens. Por medida de segurança, para sistemas que cruzam redes públicas ou semipúblicas a criptografia pode ser requisitada, mas credenciais podem somente refletir a identidade do usuário, tal como nome ou grupo a que pertence. Para alguns sistemas completos de agentes empregados na Internet, a forte autenticação e a segurança podem ser providas por autoridades externas. Todos esses níveis de segurança são suportados pelo Concordia.

6.2.5. Persistent Manager

Mantém o estado dos agentes em trânsito na rede. Ele realiza a checagem e o reinício de agentes numa eventual falha do sistema. Além disso, pode checar objetos sob requisição de agentes, prover alta granularidade de garantia de confiabilidade, para procedimentos críticos. O *Persistent Manager* é completamente transparente em suas operações, p.ex.

nem os agentes nem os administradores necessitam controlar ou monitorar suas operações. De qualquer forma, o acesso ao gerenciamento é disponibilizado, se necessário.

6.2.6. Event Manager

Cuida do registro, envio e notificação de eventos de/para agentes. Ele pode fazer a notificação de eventos em vários nós da rede Concordia. O *Event Manager* trabalha, em conjunto com o *Concordia Server*, para distribuir os eventos necessários. Importante função do *Event Manager* é suportar a colaboração de agentes Concordia.

6.2.7. Queue Manager

É responsável pelo enfileiramento e movimentação dos agentes, entre os sistemas Concordia. Essas características incluem a manutenção de agentes, enquanto esperam a oportunidade de realizar suas tarefas e mantêm seu estado de persistência, quando entram e saem do sistema. O *Queue Manager* provê o mecanismo de priorização e gerenciamento de execução de agentes, na entrada para os nós Concordia.

6.2.8. Directory Manager

Provê o serviço de gerenciamento da rede Concordia. O administrador pode configurar o serviço de nomes, de várias maneiras, escolhendo de acordo com a necessidade dos programadores e serviços. O *Directory Manager* pode consultar serviço de nomes, localmente, ou passar requisições para outros servidores existentes.

6.2.9. Service Bridge

É responsável pela interface dos agentes, com os serviços disponíveis, nas várias máquinas da rede Concordia. É composto de um conjunto de extensões de programas que acessam

APIs nativas, assim como faz a interface deste com o *Directory Manager* e *o Security Manager.*

6.2.10. Agent Tools Library

É a biblioteca que provê todas as classes necessárias para o desenvolvimento de agentes móveis Concordia. Inclui a classe *Agent* e outras derivadas das classes Java, com interfaces para a infra-estrutura Concordia.

6.3. Redefinindo a Tecnologia de Agentes Móveis com Concordia

Atualmente, o termo agente significa um programa de software responsável pela automatização de uma série de benefícios computacionais de um usuário, mesmo não estando conectado na rede. Um agente realiza sua função definida pelo fornecedor e autorizada pelo usuário. Concordia expande a funcionalidade do agente para incluir: Mobilidade, Segurança, Persistência, Colaboração [46].

6.3.1. Mobilidade

Os agentes podem viajar de um dispositivo para outro, na rede, como determinado pela aplicação ou usuário, realizando várias sub-tarefas em cada dispositivo de seu caminho. Processamentos são realizados no local dos dados de origem, reduzindo o tempo de conexão de rede e o custo.

A mobilidade dos agentes Concordia pode ser estendida a várias redes locais (*Local Area Networks* - LANs), bem como para várias redes de longa distância (*Wide Area Network* - WANs). Para evitar problemas de desempenho e segurança associados à transmissão dos agentes em redes com diferentes características, o Concordia provê

suporte para transações de filas de agentes, entre o *Agent Manager* (AMg) residente em diferentes redes.

A infra-estrutura do Concordia provê suporte a transmissão confiável de agentes, através da rede, usando, como fundamento, um subsistema de enfileiramento de mensagens. O enfileiramento de mensagens, no Concordia, permite ao AMg submeter uma requisição de um agente da fila, ao *Queue Manager (QMg)* de maneira assíncrona. A fila de mensagens do agente serve como *buffer* de transmissão da requisição. Essa característica de enfileiramento de mensagens é a maneira natural do modo de operação desconectado dos agentes móveis, porque provê o mecanismo *"store and forward*". Agentes podem ser armazenados na fila de mensagens do servidor local, enquanto o servidor remoto está em manutenção ou simplesmente foi movido para um local diferente. Quando o servidor remoto é reconectado, o servidor local prossegue com o processo enviando o agente.

O subsistema de enfileiramento de mensagens provê segurança adicional, mantendo cópia do agente, a ser transmitido em uma fila, até o recipiente dessa transmissão reconhecer seu conteúdo, através de um protocolo de *handshaking*, denominado *Two Phase Commit Protocol* [45]. É conhecido nas empresas como enfileiramento de transação de mensagem. O QMg se comunica com seu AMg local e realiza o *handshaking* com outros QMgs remotos, para uma transmissão confiável.

O QMg gerencia filas de entrada e saída. O AMg submete uma requisição de um agente para fila de saída do QMg local, que é, então, retirada dessa fila e colocada na de entrada do QMg remoto. Por sua vez, o QMg remoto prossegue com a retirada desse agente da sua fila de entrada e posterior transmissão para seu AMg local.

O QMg é implementado como objeto Java *multi-thread* para permitir acesso concorrente à fila de armazenamento. A preservação de especificação da classe de um

objeto no disco é manipulada pela facilidade de serialização de objetos Java enquanto a comunicação do QMg conta com o pacote Java RMI [47].

O objetivo do QMg inclui a otimização da utilização do espaço de disco, operações de gravações rápidas, recuperação rápida de falhas de servidor e gerenciamento confiável de transmissão de agentes pendentes. A implementação da sua arquitetura de armazenamento de fila copia algumas idéias da pesquisa de sistemas de arquivos estruturados [48] [49], para empregar uma única arquitetura de dados, que garanta melhor desempenho sobre a arquitetura convencional do sistema de enfileiramento de mensagens [50] [51].

A arquitetura convencional de enfileiramento de mensagens geralmente não é otimizada para gravar operações sem requerer hardware extra para trabalhar de maneira eficiente. Tais sistemas parecem requerer otimização de desempenho especial de hardware e software no sentido de manipular a carga de trabalho com alta vazão e baixo tempo de permanência de mensagens, característica importante de um sistema de agentes móveis.

Além disto, arquiteturas de tratamento de filas de mensagens empregam separadamente filas de dados e de arquivos de *logs* os quais introduzem um nível extra de insegurança desde que haja dois pontos em potencial de corrupção e falhas. Normalmente, não há nenhum significado para o administrador do sistema de enfileiramento de mensagens em predefinir a quantidade de trabalho necessário para fazer a recuperação. A utilização do sistema automático de checagem através do *Handle Manager* trata esse problema de maneira eficiente.

No Concordia, a arquitetura de dados da fila consiste de uma combinação de arquivos estruturados para a fila de mensagens de dados e de registros de *logs*. Essa arquitetura utiliza uma fila circular e consiste de um arquivo criado pelo QMg quando inicializado. Cada entrada no arquivo de dados da fila contém o objeto agente em blocos

contíguos no disco. O arquivo de dados da fila é particionado em um número predefinido de segmentos lógicos. Cada segmento contém, no início, um número predefinido de blocos de controle. Esses blocos de controle contêm informações de controle para as entradas da fila e informações do registro de *log*. Objetos do tipo agente enfileirado e seus registros de *logs* são armazenados de forma misturada depois do bloco de controle. Quando um novo segmento é alcançado no arquivo de dados da fila, um novo conjunto de blocos de controle é escrito no início do segmento no disco. Desta maneira, segmentos lógicos servem como intervalos de pontos de checagem forçados no estado do sistema de filas inteiro e os dados são armazenados no arquivo de fila.

O QMg adota o projeto de atomicidade, consistência e propriedades de isolamento conhecidas como ACID. A propriedade de durabilidade de proteção contra falhas de hardware pode ser alcançada duplicando a combinação de filas de dados/arquivo de *log* em disco separado. Pode-se utilizar a tecnologia RAID existente para duplicar gravações de forma transparente. Essa decisão foi utilizada no projeto no sentido de prover flexibilidade aos desenvolvedores de aplicações Concordia com respeito a possíveis trocas entre custo e nível de confiabilidade requisitado [46].

6.3.2. Segurança

Agentes somente podem acessar recursos para os quais são autorizados. Concordia usa técnicas de criptografia para prover segurança aos agentes enquanto eles estão no dispositivo e trafegam pelas redes.

Segurança no sistema de agentes móveis geralmente aborda 4 problemas distintos: (1) segurança na transferência de agentes na rede, (2) proteção de um computador contra ataques ou uso indevido de agente mal intencionado, (3) proteção de um agente contra o ataque de outro e (4) proteção de um agente do ataque de um computador [52] e [53]. O modelo de segurança Concordia provê os três primeiros tipos de proteção. Dentro do sistema, ao ser identificado como servidor Concordia, o computador é

considerado ambiente confiável e o pacote de Segurança Concordia pode de fato seguir os passos para garantir a integridade do ambiente.

Dentro do sistema Concordia, o termo “proteção de agente” é usado para referenciar a proteção dos agentes durante sua transmissão ou armazenamento no disco. Proteção do agente durante sua transmissão é reconhecidamente um problema no sistema de agentes. Como medida de segurança complementar, Concordia usa o armazenamento persistente do objeto para salvar, periodicamente, o estado do agente. No caso de falha do sistema, esse servidor usa o objeto armazenado para recuperar a execução do agente e resumir a viagem dos agentes. Desde que o armazenamento do objeto salva um agente e suas informações de estado no disco, pode também se tornar um risco de segurança em potencial. A proteção do agente Concordia assegura bem isso no disco. Concordia não protege o agente quando presente na memória flexível, mas conta com a proteção oferecida pelo sistema operacional e a JVM.

Concordia usa o termo "proteção de recursos" para se referenciar ao processo de proteção dos recursos do servidor contra um acesso desautorizado. Essa área de proteção endereça o problema de segurança de um Computador a partir do ataque ou mau uso de agentes. Além disso, essa proteção é aplicada para resguardar os agentes do ataque de outros.

Detalhes sobre a proteção de agentes e proteção de recursos no Sistema Concordia podem ser vistos em [54].

6.3.3. Persistência

A infra-estrutura do Concordia incorpora servidores de alta confiabilidade salvando seu estado interno num armazenamento persistente e, durante uma falha, recuperando, devolvendo e reconstruindo qualquer dado solicitado para continuar o serviço de seus

clientes. O Concordia também salva o estado dos agentes no armazenamento persistente, através da habilitação do restabelecimento de falhas do sistema e do servidor continuando a execução. Aplicações e agentes Concordia podem utilizar o armazenamento persistente para checarem sua integridade e, depois de uma falha, reiniciar a partir do último ponto.

O *Persistent Manager PMg* é uma proposta utilizada pelos servidores Concordia para salvar os agentes móveis e seus estados. Tipicamente, cada servidor ou aplicação possui uma instância PMg e cada uma utiliza um arquivo diferente para o armazenamento persistente. O PMg exporta métodos para criar, excluir, atualizar e produzir objetos a partir do armazenamento persistente.

O PMg gerencia objetos salvos utilizando o acesso randômico construído pelo pacote de serialização de objetos Java. O código de serialização do objeto garante que quando um objeto seja armazenado, todos os objetos alcançados por ele também sejam armazenados. Similarmente, quando um objeto é devolvido, todos objetos alcançados também são devolvidos.

O AMg em cada máquina é responsável para enviar, receber e prover um ambiente de execução dos agentes. Quando o AMg recebe um agente, grava seu estado no armazenamento persistente antes de criar sua *thread* principal de execução [15]. Depois do agente encerrar a execução, o AMg atualiza seu estado no armazenamento persistente antes de transmiti-lo para o próximo destino. O agente permanece no armazenamento persistente até ser totalmente transferido para o próximo AMg. Quando um agente termina sua execução no destino final, o AMg o exclui do armazenamento persistente.

Depois de uma falha no sistema ou no servidor, o AMg devolve cada estado do agente a partir do armazenamento persistente e reinicia sua execução. O agente reiniciado pode repetir um trabalho já completado por ele. Assim, o armazenamento persistente garante somente a correção da operação dos agentes. Esse critério é suficiente para muitas

aplicações de agentes móveis, que podem ser freqüentemente classificadas como devolução e filtragem de informação. De qualquer forma, a necessidade de correção pode ser eliminada se o agente utilizar o PMg para checar seus estados internos e iniciar sua execução a partir do último ponto de checagem.

Se o processo de recuperação do AMg produz um agente que já finalizou sua execução, ele transfere o agente para o próximo destino (se existe) e o exclui do armazenamento persistente, uma vez que a transferência tenha se completado. Esse algoritmo pode resultar na duplicação da transferência de agentes se o sistema ou servidor falhar durante ou imediatamente após a transferência do agente. O problema é detectado e tratado pelo QMg, garantindo uma completa e segura transferência de agentes. A cada viagem de um agente para uma nova máquina, o AMg recebe cópia exata desse agente.

Muitos servidores Concordia empregam o PMg. O *Event Manager (EMg)* e o *Directory Manager (DMg)* salvam seus registros no armazenamento persistente. Agentes recebendo eventos distribuídos fazem o registro dos específicos com o EMg. Cada vez que o EMg recebe um registro, captura-o e grava sua informação de atualização no armazenamento persistente. Se o EMg é reiniciado, o processo devolve os registros a partir do arquivo do armazenamento persistente. Enquanto o EMg está indisponível, vários de seus registradores param. Assim, ele estará indisponível para notificar esses registradores de novos eventos e apagará seus registros.

O DMg utiliza o armazenamento persistente de maneira similar. Servidores de aplicações podem exportar seus serviços para agentes móveis registrando com o DMg. O DMg captura, salva no armazenamento persistente e devolve a informação quando reinicia.

Administradores de sistemas podem opcionalmente habilitar a persistência para o AMg, EMg e DMg. O Concordia é flexível. Permite a administradores e desenvolvedores pesarem os custos da persistência e a necessidade de confiança de suas aplicações. É um

contraste ao paradigma *Telescript* [55], no qual há o armazenamento de objetos persistentes desnecessários, implicando no alto custo em termos de utilização de recursos e performance. Muitos modelos recentes de sistema de agentes móveis não provêem alto grau de confiabilidade e, portanto, não incluem suporte a persistência.

6.3.4. Colaboração

O Concordia inclui sistema de colaboração habilitando múltiplos agentes a trabalharem juntos e coordenarem suas ações. Agentes dentro de uma aplicação podem formar uma ou mais unidades de colaboração, denominados grupos de agentes. Esses grupos de agentes coordenam o trabalho de algumas classes de agentes especializados, denominados agentes colaboradores. Concordia oferece classes básicas para os agentes colaboradores e grupos de agentes (p.ex., *CollaboratorAgent* e *AgentGroupImpl*, respectivamente). Grupos de agentes são implementados como objetos distribuídos os quais exportam simples interface para os agentes colaboradores que, em troca, guardam referências remotas para grupos de agentes.

Os agentes podem cooperar entre si para realização de tarefas. Uma aplicação pode ser composta de vários agentes com sub-tarefas específicas. O mecanismo de colaboração consiste de duas partes: mecanismo de colaboração genérico e aplicação específica de política de análise.

As vantagens da colaboração são várias. Primeira, a operação é distribuída. Se um servidor está indisponível, elas não irão falhar. Segunda, as operações acontecem em paralelo, aumentando a vazão. Finalmente, o agente pode ser programado para tomar decisões baseadas em resultados. Exemplo, o agente pode checar o melhor preço, a melhor disponibilidade e decidir dinamicamente a qual fornecedor fazer o pedido.

6.4. Vantagens do Concordia

São grandes as vantagens de utilização do sistema Concordia. Primeiro, Concordia é escrito em Java, permitindo sua portabilidade e execução em todos os lugares. É executado em grandes e pequenas plataformas e se integra facilmente com qualquer aplicação e sistema existentes.

Segundo, agentes Concordia provêem aplicações móveis, p.ex. os agentes suportam processamentos *off-line* e operações desconectadas. Essas aplicações são escritas com pouco ou quase nenhum conhecimento desse tipo de comunicação a ser empregado. Concordia esconde os detalhes de implementação dos programadores e usuários, permitindo um agente adaptar seu ambiente e administração.

Terceiro, os agentes Concordia são seguros. Cada agente carrega a identidade do usuário que o criou e as operações requisitadas estão sujeitas às permissões do mesmo usuário. Cada agente é seguramente transmitido através da rede, e nenhum código adicional é preciso para prover a segurança e operação distribuída.

Quarto, agentes Concordia são confiáveis. Todos os agentes são verificados antes da execução pelo PMg e podem retornar ao ponto de checagem, se necessário. A confiabilidade dos agentes é garantida a cada estágio de sua execução porque eles estão cobertos pelos serviços do QMg enquanto trafegam pelos diversos nós.

Finalmente, Agentes Concordia podem colaborar. O conceito de colaboração é importante e bastante funcional para o programador de agentes. Ele oferece vários benefícios, tais como habilitar operações paralelas em múltiplos servidores ou múltiplas redes. Ele pode dividir a tarefa em partes adequadas, e estas partes podem ser executadas nos vários locais que passam. O resultado destas sub-tarefas é montado pelo serviço de

colaboração. Assim, o sistema de colaboração permite uma tomada de decisão baseada nos resultados, que podem ser usados para determinar o destino, ação ou comportamento apropriado.

6.5. Usos do Concordia

O Concordia é um sistema completo para desenvolvimento e gerenciamento de aplicações distribuídas. Ele transforma a programação de aplicações distribuídas em tarefa fácil de ser cumprida, reduzindo consideravelmente o tempo de criação dos agentes. Analisando criteriosamente o sistema, pode-se enumerar as finalidades/características de sua utilização e dos seus agentes.

Sistema Concordia:

- Habilitar a mobilização de aplicações legadas;
- Programar dispositivos móveis como aplicações de clientes;
- Superar limitações provenientes da barreira cliente / servidor;
- Fazer integração com objetos distribuídos p.ex. CORBA;
- Fazer integração com sistemas legados p.ex. banco de dados;
- Facilmente executar de maneira *standalone* etc.
- Oferece protótipo rápido com fáceis caminhos de produção;
- Oferece operação robusta através de agentes persistentes;
- Provê segurança e integridade;
- Suporta operações *off-line*;
- Provê acesso a banco de dados heterogêneos;
- Agentes carregam código para plataformas remotas.

Agentes Concordia:

- Processam dados na origem;
- Levam dados na viagem;

- Podem literalmente executar em qualquer lugar: Web, desktop, palmtop etc.
- Permitem alta escalabilidade e programação paralela;
- Escondem o transporte de rede de aplicações, desenvolvedores e usuários;
- Escondem distribuição, escala e paralelismo da aplicação.

6.6. Gerenciamento do Concordia

A gerência dos servidores numa rede Concordia é provida pelo componente *Administration Manager* operando em conjunto com os serviços disponíveis em cada nó. Estes serviços incluem os servidores Concordia juntamente com vários gerenciadores, incluindo o AMg, SMg, QMg etc. O AdmMg oferece uma interface gráfica ao administrador para todos os nós do sistema.

A administração concentra-se em duas grandes áreas, gerenciamento do servidor e gerenciamento dos agentes.

O administrador Concordia pode realizar as seguintes operações nos servidores:

- Iniciar e parar os servidores e gerentes Concordia;
- Atualizar e instalar servidores Concordia;
- Monitorar a performance do servidor Concordia;
- Visualizar os *logs* do servidor Concordia e
- Gerenciar as filas Concordia.

O administrador Concordia pode também realizar as seguintes operações, individualmente nos agentes:

- Instalar e remover códigos e bibliotecas de agentes;
- Gerenciar o itinerário dos agentes;
- Lançar remotamente agentes;

- Terminar, suspender e resumir agentes;
- Monitorar individualmente as operações de agentes.

O Gerente de Administração Concordia adicionalmente gerencia a segurança. Dentro dos aspectos de segurança temos:

- Gerência de relações de confiança entre Servidores Concordia;
- Administração de permissão de usuários: contas, grupos e acesso a serviços;
- Administração de chave de criptografia;
- Monitoramento de *logs* de segurança;
- Monitoramento de estatísticas de segurança.

6.7. Conclusão do Capítulo

Concordia oferece completo sistema para desenvolvimento e gerenciamento de redes com aplicações de AMs. O objetivo de seu projeto está centrado no provimento de suporte flexível à mobilidade, colaboração, persistência, transmissão confiável e segurança dos agentes.

O mecanismo de mobilidade do agente Concordia estende a funcionalidade encontrada em sistemas de agentes baseados em Java, oferecendo esquema flexível para invocação dinâmica de métodos dentro de aplicações comuns. O sistema Concordia oferece suporte para interação de agentes através da noção de colaboração, a qual permite ao agente interagir, modificando estados externos e internos do AM contatado.

Através do uso de *proxies* e persistência, Concordia provê robustez no ambiente onde aplicações podem ser recuperadas a partir de falhas no sistema ou na rede. Usando sistema de enfileiramento de mensagens transacional, Concordia provê transmissão confiável do AM na rede, mesmo onde conexões não são estáveis.

A infraestrutura do Concordia estende o padrão do mecanismo de segurança da linguagem Java, provendo sistema baseado em identidade onde direitos atribuídos aos agentes, são determinados a partir usuário que o lançou na rede. Concordia também protege o agente e a informação por ele carregada, quando armazenada em disco ou quando transmitida ao longo da conexão.

Capítulo 7. Integração de Agentes Móveis com SNMP

7.1. Introdução

A tecnologia de AMs é um paradigma emergente com forte aceitação em diversos campos de aplicação. O gerenciamento de redes é uma área extremamente indicada para utilização desta tecnologia que, devidamente provida, pode interoperar com soluções legadas já existentes.

7.2. A Integração

Hoje, SNMP é suportado pela maioria dos fabricantes de computadores, comutadores e demais dispositivos computacionais. Muitos sistemas possuem o agente SNMP já pré-configurado. Por isso, é interessante que novos sistemas de gerenciamento suportem interfaces com SNMP. –Agentes móveis têm forte aceitação em diversos campos de aplicação, incluindo o gerenciamento de redes. Embora com fortes diferenças conceituais, a integração de SNMP com AMs consiste de uma importante proposta de união entre sistemas de gerenciamento centralizados e descentralizados.

Para integrá-los, criamos interface SNMP nos AMs a partir de diferentes objetos Java, habilitando-os a execução de comandos do SNMP. Estes objetos, devem ser entendidos pelos agentes SNMPv2, dada a necessidade de tratar com tabelas conceituais existentes na MIB AToM, por nós empregada. Nossa interface SNMP-Java consiste de bibliotecas de classes modeladas a partir da estrutura de dados ASN.1 usada nos padrões originais do SNMP. Genérica, ela é importada de outro pacote SNMP-Java, o AdventNet [4]. Desta forma, nossa proposta se torna simples, não havendo necessidade de

implementação de novas estruturas de dados, deixando os mecanismos de comunicação para o sistema de agentes.

A principal idéia com uso do SNMP é manipular dados de gerenciamento no local. Assim, a troca de grande número de mensagens entre a estação de gerenciamento e agentes SNMP, ao longo da rede, é substituída pela migração de AM Concordia, habilitado ao SNMP, fazendo operações localmente. Com isso, a carga resultante da aplicação de gerenciamento de PVCs pode ser largamente reduzida.

O emprego da plataforma de AMs necessita de alguns recursos computacionais no dispositivo. Na possibilidade de não prover tais recursos, torna-se inadequado ou mesmo impossível manter a execução de tal plataforma. Exemplo desta situação ocorre em alguns dispositivos não aptos a executar JVM e, consequentemente, a instalar o servidor de AM. Neste caso, para gerenciar tal dispositivo, agente SNMPv2 deve estar instalado nele e seu gerenciamento feito por outro dispositivo responsável para esse fim. Dessa forma, SNMP é utilizado para realizar operações remotas aos dados de gerenciamento, empregando o modelo "cliente/servidor". Comutadores, computadores, enfim, qualquer dispositivo, embora não esteja habilitado a executar AMs, podem ser gerenciados segundo nossa proposta.

O principal objetivo deste trabalho é integrar duas tecnologias, Agentes Móveis e Sistemas Legados, como proposta de solução aos problemas relacionados na seção 1.2.

7.2.1. Solução para o Paradigma

AMs superam algumas limitações do paradigma "cliente/servidor". Com AMs, os problemas surgidos com o crescimento das redes são diminuídos consideravelmente porque o tempo de conexão é reduzido ao tempo necessário ao transporte do AM de um dispositivo a outro. Agentes clássicos existentes nos sistemas de redes tendem a ser

monolíticos. AMs não residem estaticamente em um único dispositivo e, portanto, podem ser criados sob demanda e destruídos quando não mais necessários. Eles tendem a ser menores do que os agentes clássicos de gerenciamento de redes, pois normalmente realizam simples tarefas. Em geral, AMs podem reduzir a carga no lado gerente, considerando-se que o processo de gerenciamento pode ser dividido em pequenas tarefas, delegadas a tais AMs.

7.2.2. Solução para o Gerenciamento de Redes

Gerenciamento de redes baseado em SNMP ou CMIP conduz a arquiteturas centralizadas. O uso do *polling* é desaconselhado para grandes redes por causa das desvantagens no desempenho, escalabilidade e eficiência. Gerenciamento de redes através de delegação e de AMs são tentativas de introduzir arquiteturas de gerenciamento descentralizadas.

Embora essas novas pesquisas pareçam resolver o problema de arquiteturas centralizadas, não podemos subestimar o poder das soluções de gerenciamento baseadas nos protocolos legados SNMP. Esse protocolo é aceito como padrão e, portanto, vastamente suportado por produtos comerciais. É interessante integrá-los com as novas propostas de gerenciamento.

A interoperabilidade com tecnologias de gerenciamento existentes, tais como SNMP e CMIP, é muito importante para o sucesso dos AMs na área de gerenciamento de redes. Para garantir o desenvolvimento de aplicações de gerenciamento para redes grandes e heterogêneas, AMs precisam prover interoperabilidade.

Primeiro, porque a tecnologia legada provê acesso a informações de gerenciamento e serviços. Com isso, convém que AMs incorporem em seus códigos os serviços locais de gerenciamento, no sentido de realizar tarefas inteligentes próximas aos dados gerenciados. O emprego de SNMP fica restrito apenas a coletar dados localmente,

ao invés de fazê-lo remotamente. Portanto, AMs não são propostos para substituir os protocolos clássicos usados para gerenciar redes heterogêneas. Ao contrário, AMs deverão complementá-los com seu poder de construção de programas, permitindo soluções eficientes à gerência da rede.

Segundo, aplicações de AMs para gerenciamento precisam coexistir com sistemas legados. Eles são bem aceitos para desenvolver novos serviços, específicos para solução de problemas em grandes ambientes de gerenciamento, ainda baseados nos paradigmas clássicos.

Podemos observar que soluções de aplicações de AMs apresentam vantagem em situações onde é apropriada a migração do código para o local dos dados, em vez de enviar os dados até a aplicação. Dividindo-se as funções do gerenciamento de redes em entidades computacionais móveis, autônomas e inteligentes, superamos problemas de escalabilidade, flexibilidade e desempenho provenientes da arquitetura estática:

- Escalabilidade cresce quando o gerenciamento não é feito apenas por uma estação, mas, delegada a AMs distribuídos;
- Melhor desempenho é alcançado com a funcionalidade do gerenciamento movida para próximo do elemento de rede, reduzindo com isso o tráfego;
- Detalhes de baixo nível de diferentes equipamentos podem ser escondidos pela interface do AM;
- Sistemas legados podem ser integrados através do uso de AM com finalidade de realizar interoperabilidade.

7.2.3. Soluções em Redes ATM Heterogêneas

Operações e configurações de PVC são tarefas principais do gerenciamento de redes ATM. Fabricantes de comutadores provêem gerenciamento proprietário e heterogêneo e, com isso, criam ambiente onde se torna difícil automatizar esse processo. O controle do

gerenciamento torna-se mais imprevisível com o contínuo aumento da rede em tamanho e complexidade.

Conforme vimos anteriormente, o sistema de gerenciamento de redes consiste de estação de gerenciamento, nó gerenciado, MIB e protocolo de redes. Devido à natureza de diferentes projetos, cada fabricante desenvolve sua MIB. ATM Forum e IETF desenvolveram padrões internacionais de MIBs tais como ILMI [30] e AToM (RFC 1695) [29], respectivamente.

Porém, a MIB AtoM (RFC 1695), por ser mais orientada à configuração de PVCs, foi empregada neste trabalho. Com visão genérica, RFC 1695 definiu uma MIB usada para gerenciar interfaces, equipamentos, redes e serviços ATM.

Integrando facilidades do SNMP com tecnologia de AMs para complementar a arquitetura "cliente/servidor", permitimos o gerenciamento PVCs em redes ATM heterogêneas. Portanto, uma maneira simples e genérica de realizar tais tarefas de gerenciamento é proposta, escondendo as peculiaridades de cada sistema do equipamento ATM.

7.3. Protótipo do Sistema de Gerenciamento de PVCs

Para analisar a solução proposta de AMs provendo funcionalidade de configuração de PVC, um protótipo de implementação Concordia foi desenvolvido, oferecendo visão genérica de três fases desse processo: configuração, liberação e reconfiguração de PVC em dispositivos de redes ATM [57].

7.3.1. Suposições

O sistema conta com certas suposições que resultam na simplificação do processo de desenvolvimento. Essas suposições são necessárias para isolar os assuntos centrais deste trabalho, procurando manter a aplicabilidade da solução proposta:

- A funcionalidade do processo está apenas relacionada à configuração de PVCs ponto a ponto;
- O parâmetro de QoS usado é Taxa de Pico da Célula (*Peak Cell Rate - PCR*), a qual é considerada máxima largura de banda alocada para conexão;
- O usuário tem conhecimento de todo ambiente (computadores e comutadores) ao longo do caminho da conexão, ou seja, a rota é predefinida e nenhuma decisão de roteamento deve ser feita;
- As configurações dos comutadores onde JVM não é suportada e, por isso, não permitem execução da plataforma de AMs, são feitas por outro dispositivo responsável por seu gerenciamento;
- Cópia de avaliação do Sistema Concordia foi empregada e, com isso, alguns aspectos de segurança não são considerados.

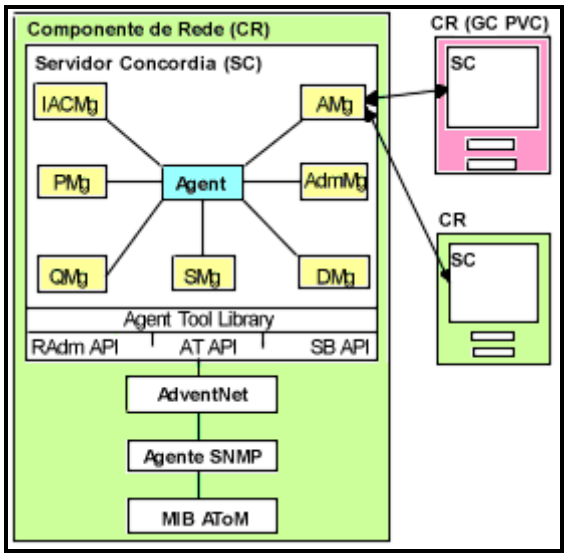
7.3.2. Requisitos

Os requisitos necessários são:

- Todos equipamentos devem ser gerenciáveis pelo SNMP através da MIB AToM;
- Versão 2 do agente SNMP deve ser utilizada nos equipamentos gerenciados, pois operação de configuração manipula tabelas conceituais do tipo *RowStatus*, não suportadas pela versão 1;
- Usuário tem uma interface *applet* lançando AMs para configuração do PVC;
- Conjunto de AMs para realizar configuração;
- Conjunto de AMs para realizar reconfiguração;
- Conjunto de AMs para realizar liberação.

7.3.3. Proposta da Arquitetura do Sistema

A arquitetura do sistema é constituída pelos componentes descritos abaixo. Todos os AMs dentro do protótipo são implementados em Concordia (Figura 7.1).



CR: Componente de Rede
IACMg: *Inter-Agent Commnunication Manager*
AMg: *Agent Manager*
AdmMg: *Administration Manager*
RAdm API: *Remote Admin API*
GC PVC: Gerente de Configuração PVC

PMg:*Persistence Manager*
SMg: *Security Manager*
QMg: *Queue Manager*
DMg: *Directory Manager*
AT API: *Agent Transport API*
SB API: *Service Bridge API*

Figura 7.1. Arquitetura de Implementação

O Sistema Concordia, já descrito anteriormente, deve estar presente em cada dispositivo, pois se trata do ambiente de execução dos AMs nas diversas plataformas. Caso o comutador não execute JVM, os componentes do sistema devem residir em outro CR executado num dispositivo separado, responsável pela gerência de seus recursos. O *Concordia Server* provê inteligência necessária para configuração da rede ATM. AMs são implementados para realizar diferentes funções de configuração de PVC usando as funcionalidades dos dispositivos ATM.

O componente Gerente de Configuração PVC (GC PVC), instalado no CR responsável pela gerência das tarefas de configurações PVC dos dispositivos, injeta AMs

na rede ATM. Ele especifica o conjunto de dispositivos ao longo do caminho do PVC, além dos valores VPIs, VCIs, largura de banda etc.

Para interação com agentes SNMP, é necessário incluir capacidades SNMP a AMs. Desta forma, eles se tornam habilitados a realizarem comandos GET e SET necessários ao estabelecimento do PVC. Isso é possível através da importação, pelo AM Concordia, de classes Java provenientes do AdventNet. Adventnet SNMP é o conjunto de bibliotecas de classe Java para desenvolver aplicações e *applets* de gerenciamento SNMP de redes. Adotamos o AdventNet v2c release 3.3 [4], pois suporta o JDK 1.1 e superiores. Todas as outras APIs e aplicações são projetadas para JDK 1.1, JDK 1.2 e máquinas virtuais mais recentes. O pacote pode ser usado para desenvolver aplicações para gerenciamento de agentes SNMPv1 e SNMPv2 e interagir com AMs. O acesso do AM Concordia à MIB AToM é possível pela importação de classes deste componente.

A MIB AToM contém objetos com atributos e valores, associados ao dispositivo ATM, definidos de acordo com SMIv2. Para o protótipo utilizado neste trabalho, objetos manipulados referem-se às tabelas conceituais necessárias para configuração de PVCs.

7.3.4. Configurações de PVCs

A MIB AToM tem seu foco principal no gerenciamento de PVCs e, com isso, na especificação de seus procedimentos de estabelecimento, liberação e reconfiguração. Dentro deste contexto, nossa proposta descreverá os passos necessários para realização de cada etapa acima citada [58].

Fator importante na utilização de AMs para configuração de PVC é prover uma maneira uniforme do operador de rede ATM realizar essa tarefa. Com ela, não é mais

necessário ter conhecimento de fundamentos dos sistemas particulares, pertinentes a cada dispositivo conectado na rede ATM heterogênea.

Exemplo de gerenciamento de PVC usando MIB AToM pode ser ilustrado na Figura 7.2, na configuração do PVC entre dois sistemas finais (computadores SF1 e SF2) e um sistema intermediário (comutador 8285 – SI1), envolvidos no projeto REMAV-FOR, pertencentes ao LARCES – UECE. Foram utilizados valores VPI/VCI, portas etc. da situação analisada em [59].

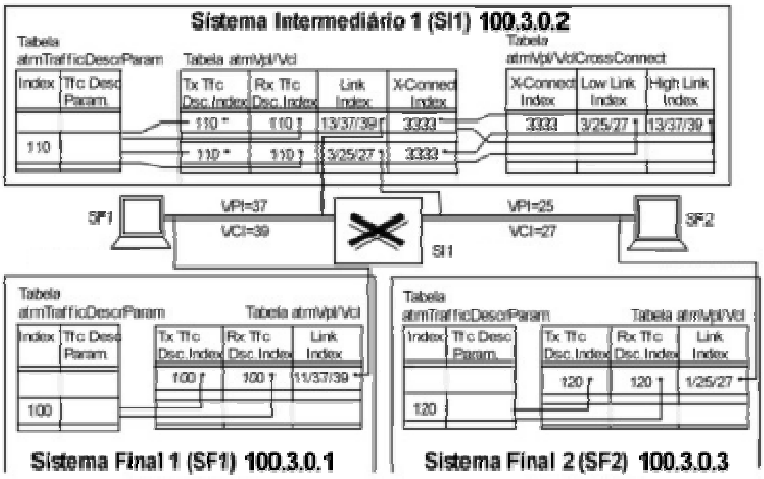


Figura 7.2. Ambiente de Teste no LARCES/UECE

Através do componente GC PVC, o operador inicia o processo dando entrada na largura de banda dos equipamentos que farão parte da conexão virtual. Conforme citado anteriormente, presume-se que a rota é previamente conhecida. Assim, um AM configurador do PVC é enviado para a rede (Figura 7.3). Inicialmente, com dados fornecidos pelo operador, o AM Concordia seleciona valor de VPI/VCI válido, através de algoritmo definido pelo desenvolvedor. Em seguida, reserva a largura de banda solicitada e encerra a configuração no primeiro computador (SF1).

Terminada a configuração, migra para o próximo dispositivo, comutador (SI1), onde, com o mesmo valor VPI/VCI do equipamento anterior (SF1) e com a largura de

banda requerida, faz a reserva na porta de entrada do equipamento corrente (SI1). Portanto, durante a migração, o valor VPI/VCI selecionado pelo AM no equipamento *n* é levado ao equipamento *n+1*. Ainda no equipamento corrente (SI1), o AM Concordia, utilizando o algoritmo definido pelo desenvolvedor, seleciona valor VPI/VCI válido para porta de saída. Concluída a seleção de VPI/VCI, o AM realiza o procedimento de cruzamento cruzado e conclui a configuração no comutador (SI1).

Existindo mais de um sistema intermediário, esses passos se repetem até que o AM chegue ao sistema final (SF2) e complete a tarefa de configuração do PVC. O procedimento de cruzamento de conexão só é necessário em sistemas intermediários. Percebe-se tratar, naturalmente, de um procedimento seqüencial, pois com finalidade de completar PVCs o AM finaliza previamente tarefa de configuração num dispositivo, antes de seguir para o próximo.

Quando ocorrem erros recuperáveis, a reconfiguração é realizada através de negociações entre AMs e dispositivos. Exemplo, valores VPI/VCI selecionados pelo equipamento anterior já em uso no equipamento corrente, ou largura de banda requisitada para o VL, não disponível. Para solucionar esses problemas, o AM pode retornar ao dispositivo anterior, executando decisões inteligentes.

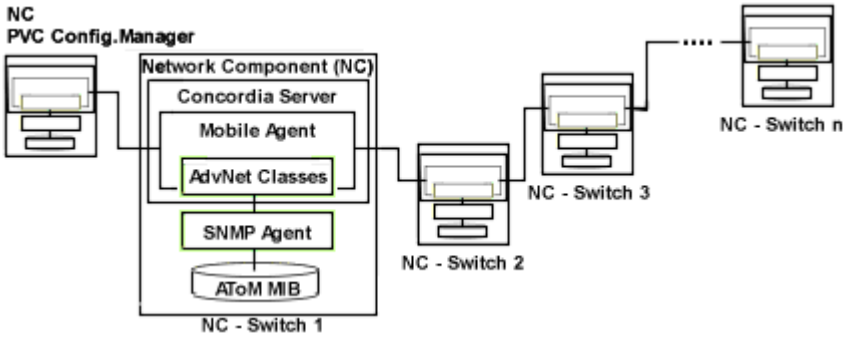


Figura 7.3. Metodologia de Configuração

A seguir apresentaremos detalhamento dos passos de configuração PVC da situação acima citada [59]. A Figura 7.2 deverá ser observada para melhor compreensão dos valores VPIs, VCIs etc.

7.3.4.1. Estabelecimento da VC

O processo de estabelecimento da VC consiste de três fases:

- Reserva do VL Apropriado;
- Caracterização do Tráfego no VL;
- Conexão cruzada do VL nos sistemas intermediários;

7.3.4.1.1. Reserva do VL Apropriado

Nesta fase, o AM Concordia cria uma entrada na tabela atmVclTable ativando a linha de status atmVclRowStatus para createAndWait. O índice da tabela atmVclTable, selecionado pelo AM, é constituído pelo ifIndex, atmVclVpi e atmVplVci. O GC PVC inicia reserva de VLs, ao longo da rota, enviando o AM responsável pela reserva do VL aos dispositivos ATM envolvidos. As Figuras 7.4a, 7.4b e 7.4c ilustram os passos da reserva de VL nos dispositivos da rede envolvidos no PVC em questão. Não ocorrendo nenhum erro, uma nova linha é criada e os valores VPI/VCI reservados naquela porta. Contadores de VPCs/VCCs (atmInterfaceVpcs/Vccs) são incrementados automaticamente na MIB AToM.

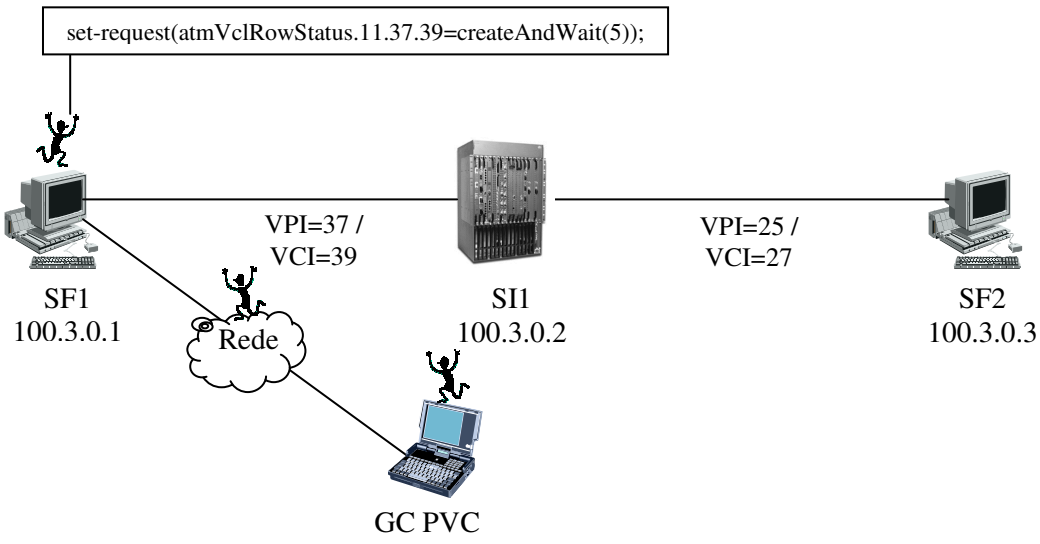


Figura 7.4a Reserva de VL no SF1

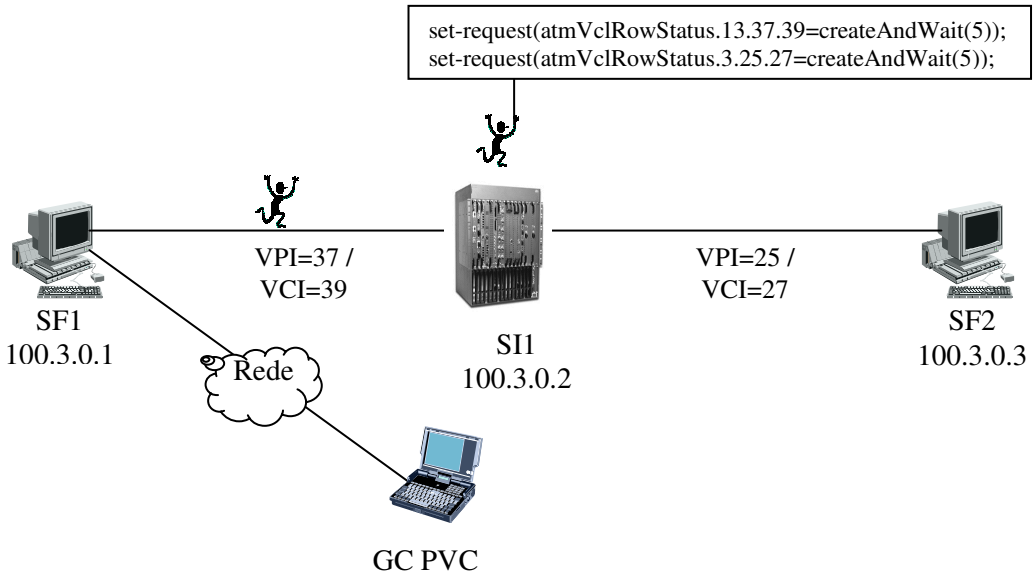


Figura 7.4b Reserva de VL no SI1

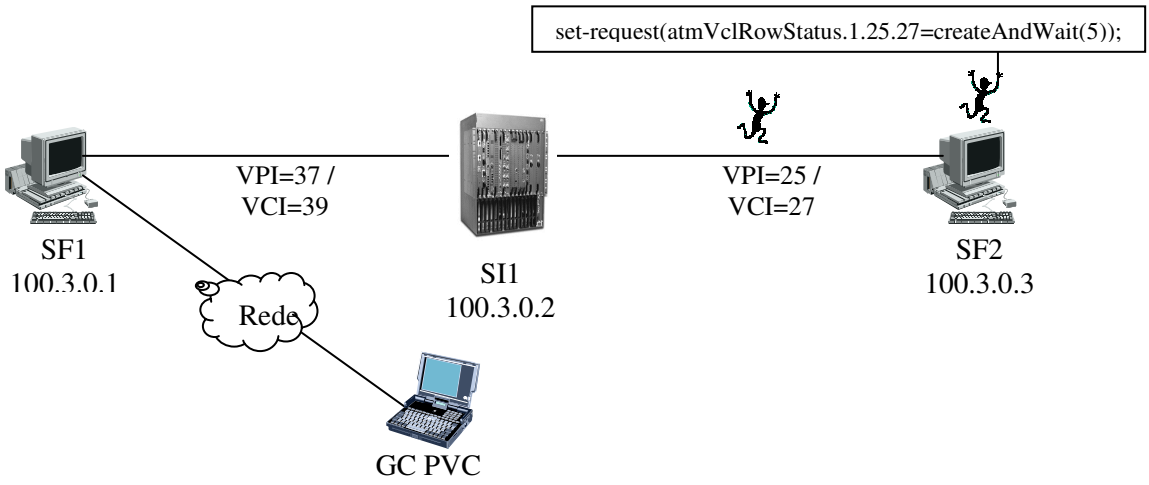


Figura 7.4c Reserva de VL no SF2

Exemplo do comando SET feito pelo AM Concordia no **SF1** é listado abaixo.

```
// Criação de Instância SnmpTarget
SnmpTarget target = new SnmpTarget();

// Seleção do Destino e OID
target.setTargetHost("100.3.0.1");
target.setObject(".1.3.6.1.2.1.37.1.7.1.13.11.37.39")

// Execução do comando Set SNMP para creatAndWait(5)
try {
            String result = target.snmpSet(5);
} catch (Exception e) {
            TrataErroatmVclTable(e);
}
```

Falhas ocorrem se:

- O agente SNMP suporta apenas operações na tabela conceitual;
- O valor **ifIndex** selecionado não existe ou não é interface ATM (**ifTable**);
- O máximo número de VCs suportados para interface foi atingido (**atmInterfaceMaxVpcs/VCCs**);
- Valores VPI/VCI selecionados estão indisponíveis para uso (**atmInterfaceMaxActiveVpi/VCiBits**);
- Valores selecionados VPI/VCI estão em uso ou reservados (linhas existentes na **atmVclTable**).

7.3.4.1.2. Caracterização do Tráfego no VL

A tabela atmVclTable caracteriza o tráfego transmissão/recepção, apontando com os índices atmVclTransmitTrafficDescrIndex e atmVclReceiveTrafficDescrIndex para respectiva entrada na tabela atmTrafficDescrParamTable. Múltiplos VLs na tabela atmVclTable podem apontar para o mesmo vetor da tabela atmTrafficDescrParamTable. Esta técnica permite a pré-definição de vetores de tráfego na tabela atmTrafficDescrParamTable. Além disso, o objeto da tabela possui modo de acesso ler-criar, possibilitando a especificação de vetores adicionais.

O vetor de tráfego consiste de uma coluna **atmTrafficDescrType** descrevendo o tipo de tráfego, cinco colunas com os parâmetros (**atmTrafficDescrParam1** até **atmTrafficDescrParam5**), uma coluna indicando a qualidade de serviço (**atmTrafficDescrQoSClass**), uma coluna com a linha de status (**atmTrafficDescrParamRowStatus**) e a coluna (**atmTrafficDescrParamIndex**) para indexar a tabela.

Assumindo a situação de [59] onde a VCC transporta 64-Kbps CES usando AAL1, utiliza-se atmNoClpNoScr como tipo de tráfego. Este tipo só especifica um parâmetro, o PCR. O resultado PCR é 171 células/s (64.000 bps/47 bytes por célula de 8 bits). A QoS requisitada para CES é Classe A, corresponde a 1.

Neste ponto, duas situações podem ocorrer: existir ou não linhas conceituais já definidas na tabela **atmTrafficDescrParamTable**, correspondentes ao tráfego a ser utilizado. Existindo, o AM Concordia, responsável pela caracterização do tráfego, a(s) seleciona da tabela **atmTrafficDescrParamTable** e, não existindo, ele a(s) cria.

Supondo que ainda não existam as linhas conceituais definidas, assumamos os valores 100, 110, 120 como valores das próximas linhas livres a serem criadas no SF1, SI1 e SF2, respectivamente. As Figuras 7.5a, 7.5b e 7.5c ilustram a seqüência de criação das linhas conceituais nos dispositivos da rede envolvidos no PVC em questão.

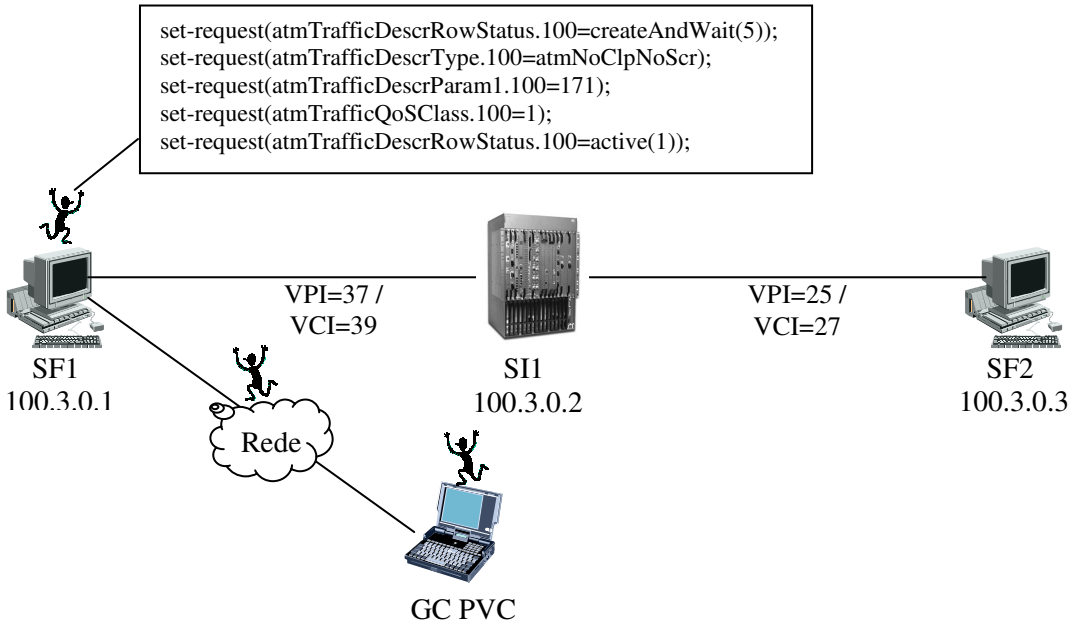


Figura 7.5a Criação da Linha Conceitual SF1

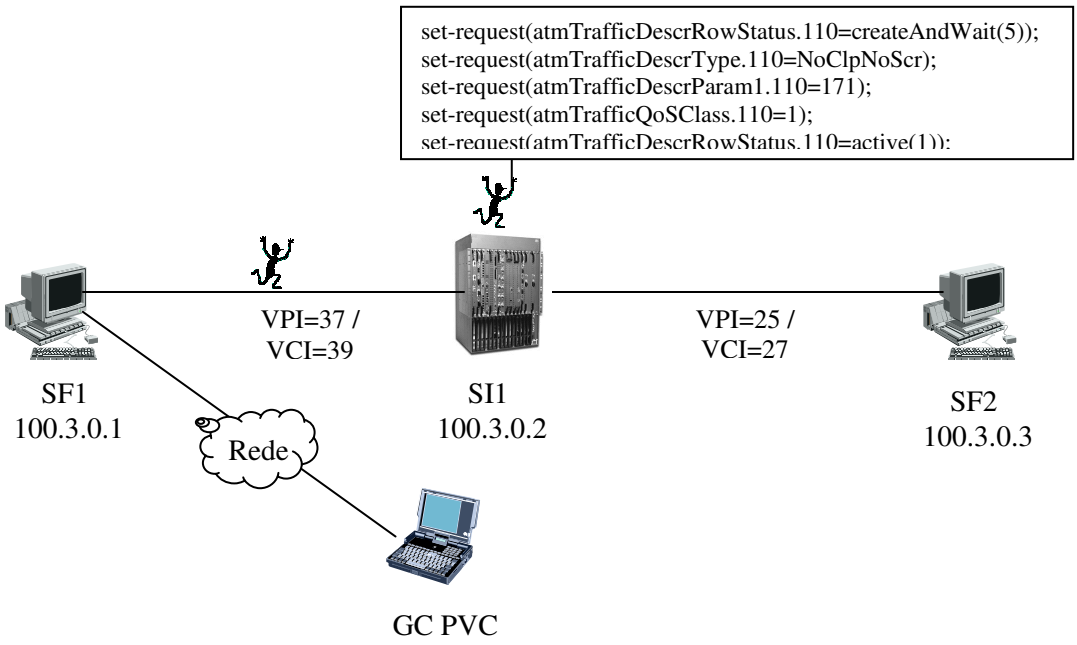


Figura 7.5b Criação da Linha Conceitual em SI1

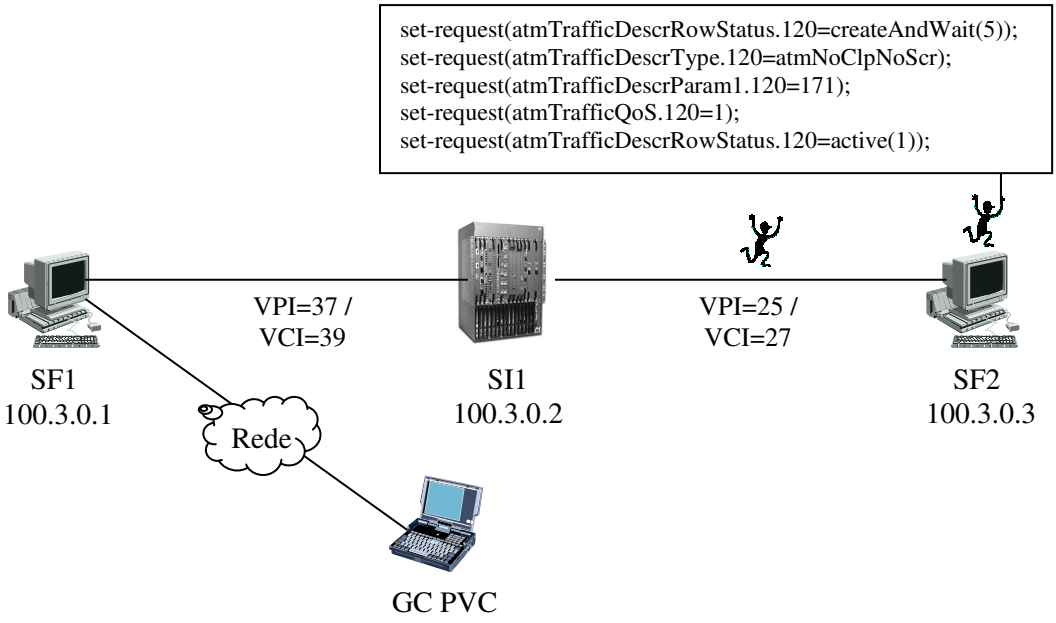


Figura 7.5c Criação da Linha Conceitual em SF2

Exemplo do Comando Set pelo AM Concordia no **SF1** é listado abaixo.

```
// Criação de Instância SnmpTarget
SnmpTarget target0 = new SnmpTarget();
SnmpTarget target1 = new SnmpTarget();
SnmpTarget target2 = new SnmpTarget();
SnmpTarget target3 = new SnmpTarget();
SnmpTarget target4 = new SnmpTarget();


// Seleção do Destino e OID
target0.setTargetHost("100.3.0.1");
target0.setObject(".1.3.6.1.2.1.37.1.5.1.9.100")
target1.setTargetHost("100.3.0.1");
target1.setObject(".1.3.6.1.2.1.37.1.5.1.2.100")
target2.setTargetHost("100.3.0.1");
target2.setObject(".1.3.6.1.2.1.37.1.5.1.3.100")
target3.setTargetHost("100.3.0.1");
target3.setObject(".1.3.6.1.2.1.37.1.5.1.8.100")
target4.setTargetHost("100.3.0.1");
target4.setObject(".1.3.6.1.2.1.37.1.5.1.9.100")

// Execução do comando Set SNMP para creatAndWait(5)
try {
      String result = target0.snmpSet(5);
} catch (Exception e) {
      TrataErroatmTrafficDescrParamTable(e);
}


// Execução do comando Set SNMP com parâmetros desejados
try {
      String result1 = target1.snmpSet("atmNoClpNoScr");
      String result2 = target2.snmpSet(171);
      String result3 = target3.snmpSet(1);
      String result4 = target4.snmpSet(1);

} catch (Exception e) {
      TrataErroatmTrafficDecrParamTable(e);
}
```

Estas ações falham se:

- O agente SNMP executado no dispositivo não suporta acesso ler-criar nessa tabela, porque somente conjunto prefixado de caracterização de tráfego for suportado;
- A linha especificada está ativa.
- Parâmetros são mutuamente inconsistentes;
- O agente SNMP não suporta valores requisitados.

Criadas as linhas conceituais, o AM caracteriza os parâmetros de tráfego dos VLs associados à VC através do índice de tráfego de transmissão/recepção (**atmVclTransmitTrafficDescrIndex** e **atmVclReceiveTrafficDescrIndex/**) na tabela de VL (**atmVclTable**), para as linhas da tabela de descrição de parâmetros de tráfego (**atmTrafficDescrParamTable**) (Figuras 7.6 a, 7.6b e 7.6c).

VLs são ativados alterando-se a linha de status **atmVclRowStatus** para **active** (Tabela 4.5). Não ocorrendo erros, a reserva de recursos é concluída, satisfazendo os valores dos parâmetros de tráfego e da classe de QoS do VL.

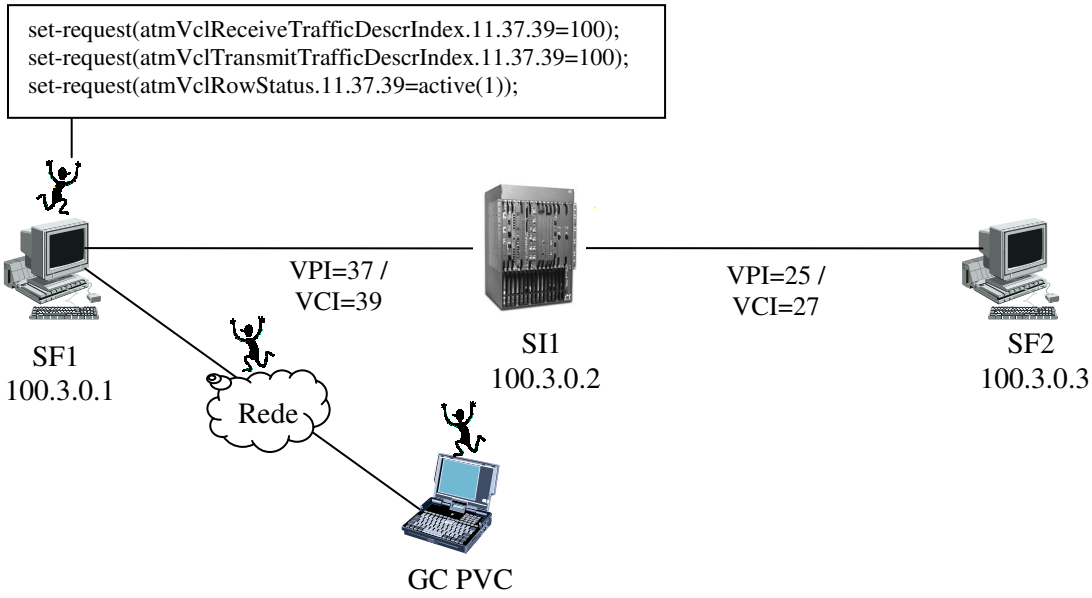


Figura 7.6a Caracterização do Tráfego no SF1

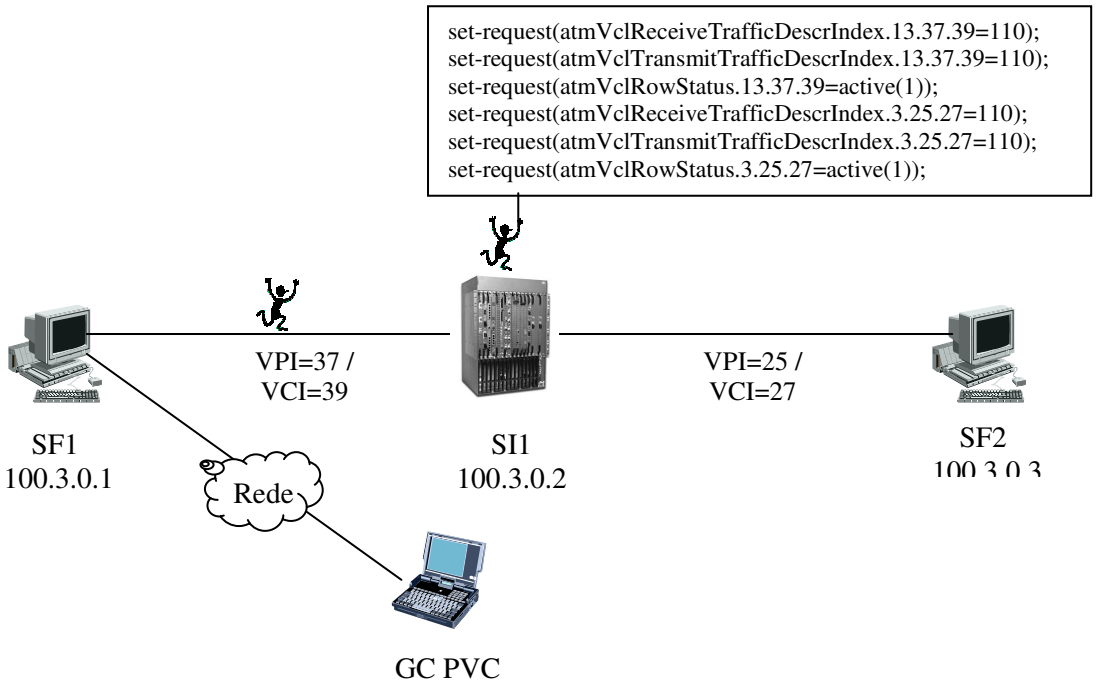


Figura 7.6b Caracterização do Tráfego em SI1

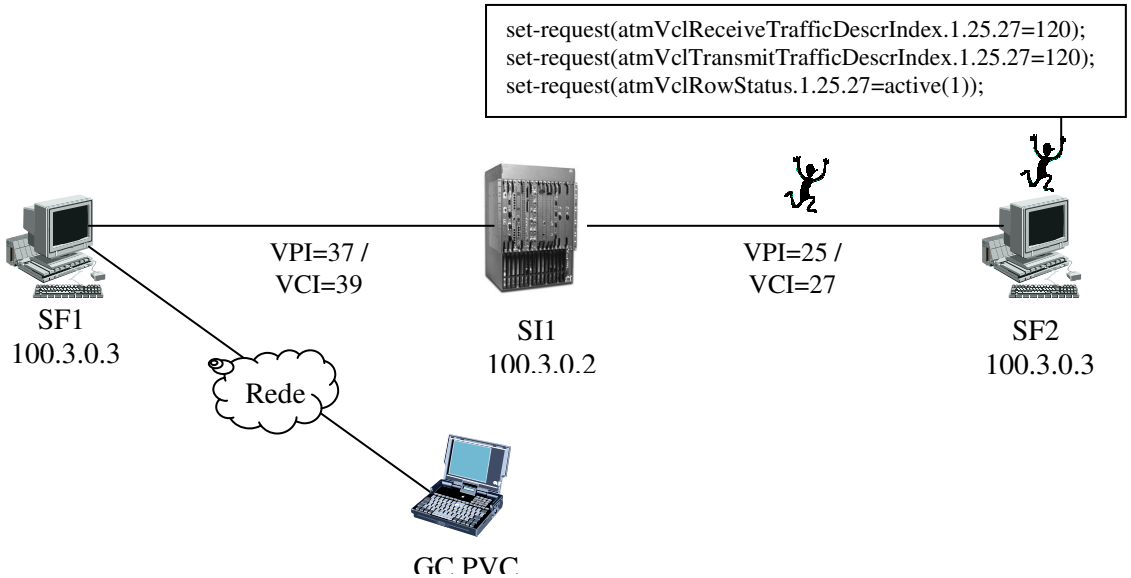


Figura 7.6c Caracterização do Tráfego em SF2

Exemplo do Comando Set pelo AM Concordia no **SF1** é listado abaixo.

```
// Criação de Instância SnmpTarget
SnmpTarget target1 = new SnmpTarget();
SnmpTarget target2 = new SnmpTarget();
SnmpTarget target3 = new SnmpTarget();


// Seleção do Destino e OID
target1.setTargetHost("100.3.0.1");
target1.setObject(".1.3.6.1.2.1.37.1.7.1.6.11.37.39")
target2.setTargetHost("100.3.0.1");
target2.setObject(".1.3.6.1.2.1.37.1.5.1.7.11.37.39")
target3.setTargetHost("100.3.0.1");
target3.setObject(".1.3.6.1.2.1.37.1.7.1.13.11.37.39");


// Execução do comando Set SNMP com parâmetros desejados
try {
      String result1 = target1.snmpSet(100);
      String result2 = target2.snmpSet(100);

} catch (Exception e) {
      TrataErroatmTrafficDecrParamTable(e);
}


// Execução do comando Set SNMP ativando VLs
try {
       String result3 = target3.snmpSet(1);

} catch (Exception e) {
       TrataErroatmTrafficDecrParamTable(e);
}
```

Esta ação falha num VL particular se:

- Não há recursos suficientes disponíveis.

7.3.4.1.3. Cruzamento de conexão do VL nos sistemas intermediários

No sistema intermediário (SI1), a tabela atmVcCrossConnectTable deve ser usada para cruzar conexões de VLs. A tabela atmVclTable possui coluna identificadora com esse propósito, atmVclCrossConnectIdentifier. Diferentes linhas na tabela atmVclTable com mesmo identificador realizam cruzamento de conexão.

Antes de criar nova linha na tabela **atmVcCrossConnectTable**, um único índice deve ser obtido com **atmVp/VcCrossConnectIndexNext**. O processo de cruzamento de conexão do VL consiste dos seguintes passos:

- Criar uma linha na tabela **atmVcCrossConnectTable**;
- Obter o valor do índice de cruzamento de conexão nas linhas da tabela **atmVclTable;**
- Ativar a linha na tabela **atmVcCrossConnectTable** e
- Ativar o tráfego.

Finalmente, o tráfego nos computadores é ativado atribuindo-se o valor **up** às linhas atmVclAdminStatus de suas respectivas tabelas atmVclTable. As Figuras 7.7a, 7.7b e 7.7c ilustram o cruzamento de conexão no SI1 e a ativação do tráfego nos dispositivos da rede envolvidos no PVC em questão, feitos pelo AM.

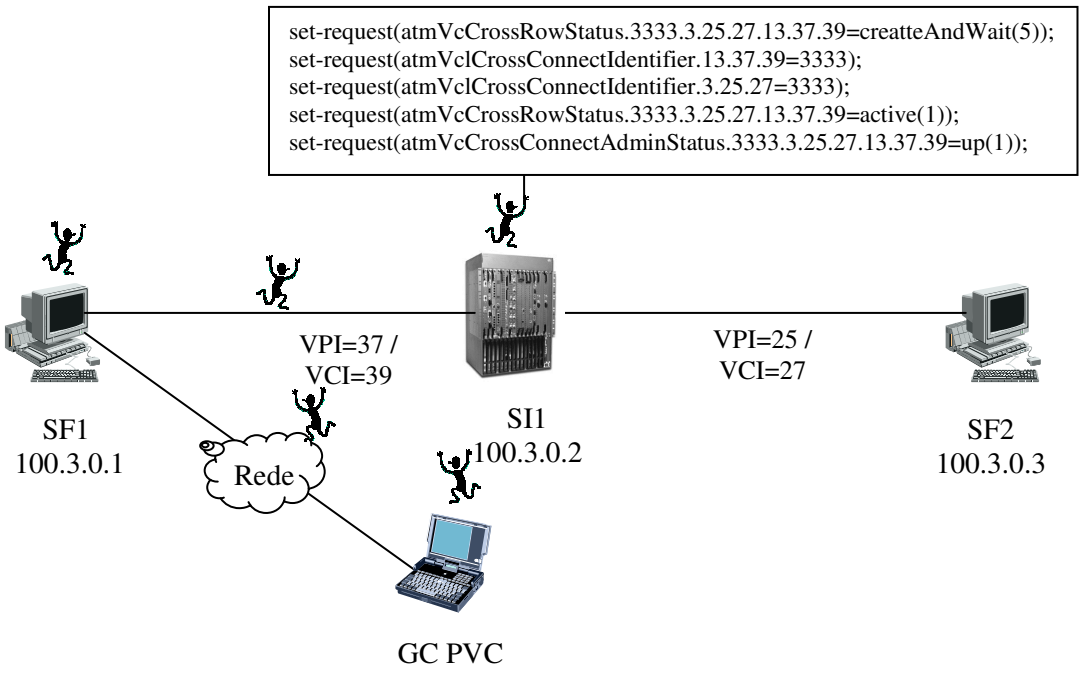


Figura 7.7a *Cross-Connect* no SI1 e Ativação do Tráfego

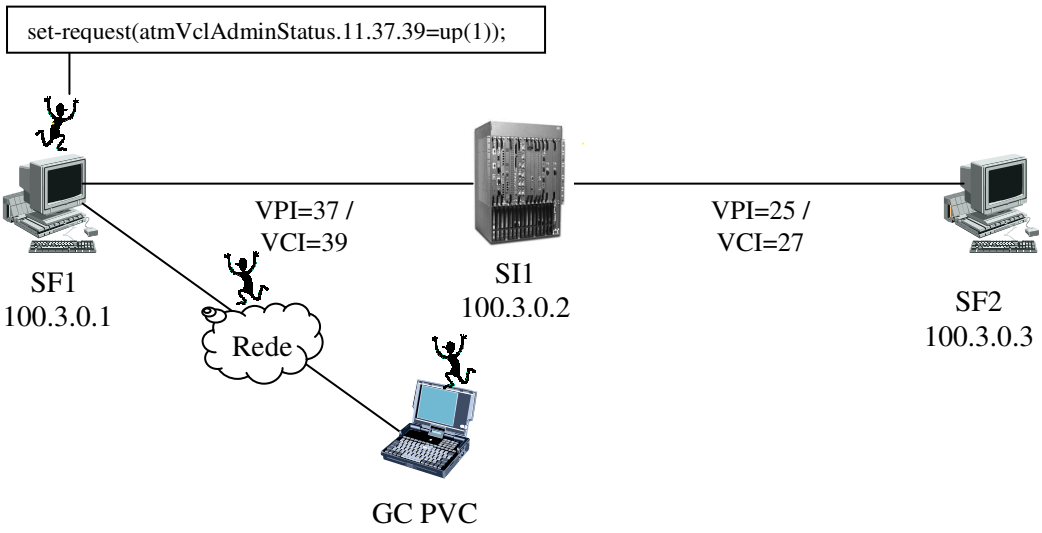


Figura 7.7b Ativação do Tráfego no SF1

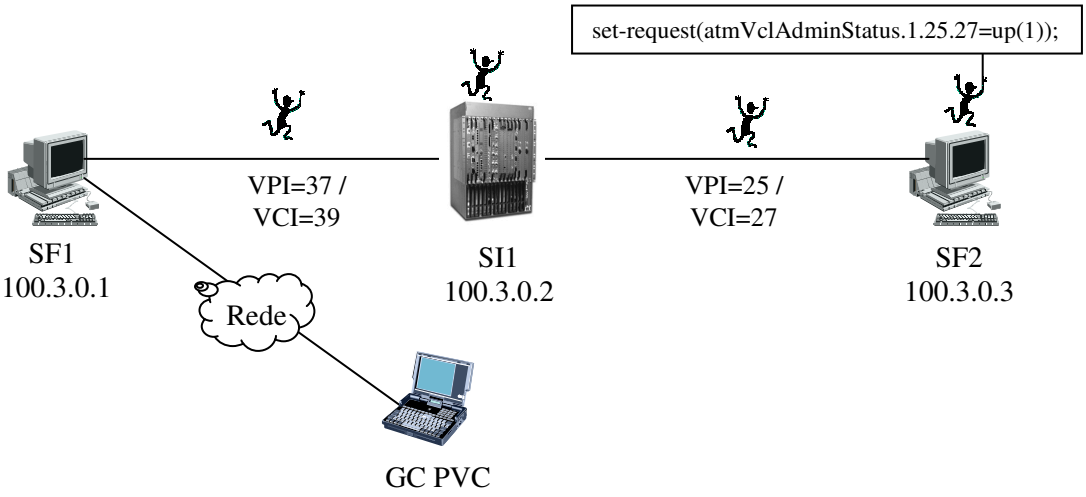


Figura 7.7c Ativação do Tráfego no SF2

Os passos acima podem ser suprimidos, se atribuído o valor **createAndGo** à linha de status **atmVclRowStatus**. Porém, não é possível obter a análise detalhada de erros. O processo passo a passo (**createAndWait**), portanto, é mais recomendado.

7.3.4.2. Liberação da Conexão Virtual

A liberação da VC consiste de duas fases:

- Liberação da conexão cruzada no sistema intermediário;
- Liberação de todos VLs.

7.3.4.2.1. Liberação da Conexão Cruzada no Sistema Intermediário

Para remover um VL, todos os cruzamentos de conexão e VLs associados devem ser liberados, atribuíndo-se destroy à linha de status atmVclRowStatus da tabela atmVclTable (Figura 7.8). Isso liberará o valor 3333 para uso futuro pelo atmVcCrossConnectIndexNext, e os valores atmVclCrossConnectIdentifier serão removidos do VL associado. Como resultado, recursos do conexão cruzada será liberada.

É recomendado liberar o cruzamento de conexão antes de destruir os VLs individualmente, porque liberar primeiro o VL pode, em muitas implementações, ser interpretado como uma requisição para troca de configuração.

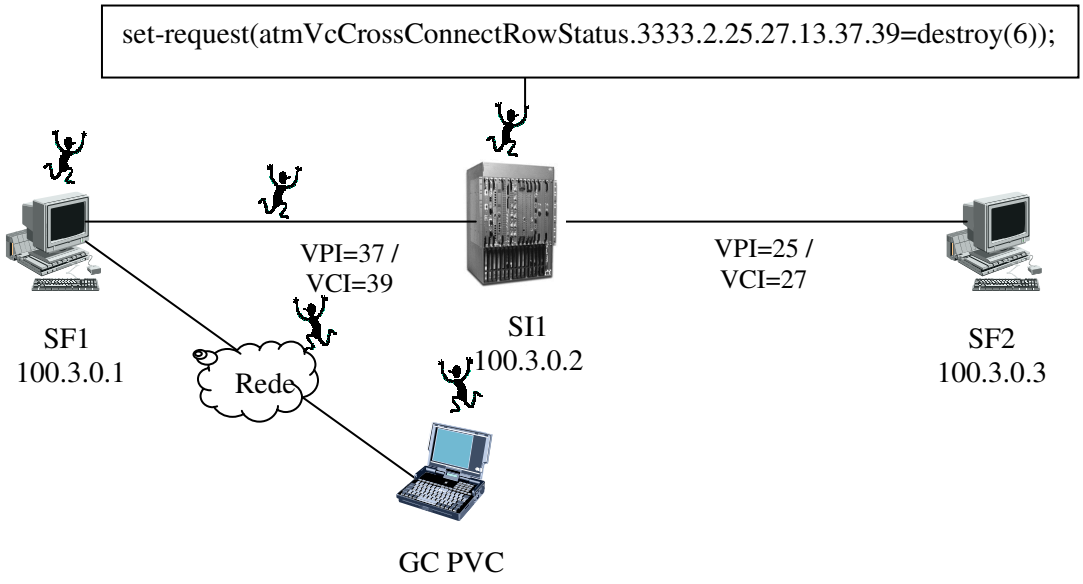


Figura 7.8 Liberação da Conexão Cruzada no SI1

Exemplo do Comando Set pelo AM Concordia no **SI1** é listado abaixo.

```
// Criação de Instância SnmpTarget
SnmpTarget target = new SnmpTarget();

// Seleção do Destino e OID
target.setTargetHost("100.3.0.2");
target.setObject(".1.3.6.1.2.1.37.1.7.1.11.1.13.3333.3.25.27.13.37.39");

// Execução do comando Set SNMP destruindo atmVcCrossConnectRowStatus
try {
     String result = target.snmpSet(6);

} catch (Exception e) {
     TrataErroatmVcCrossConnectTable(e);
}
```

7.3.4.2.2. Liberação dos VLs e Descritores de Tráfego

Para reformular os VLs associados com a VC, cada linha atmVclRowStatus da tabela atmVclTable dos respectivos dispositivos deve ser destruída. Sob essas ações, AMs libertarão os recursos associados ao VL e decrementarão atmInterfaceVccs.

Para liberar os valores dos parâmetros dos descritores de tráfego associados com a diretiva de transmissão/recepção dos VLs, as linhas da tabela **atmDescrParamParamTable** apontadas para os VLs que devem ser liberbadas. A liberação de VLs e dos descritores de tráfego feita pelo AM Concordia é ilustrada nas Figuras 7.9 a, 7.9b e 7.9c.

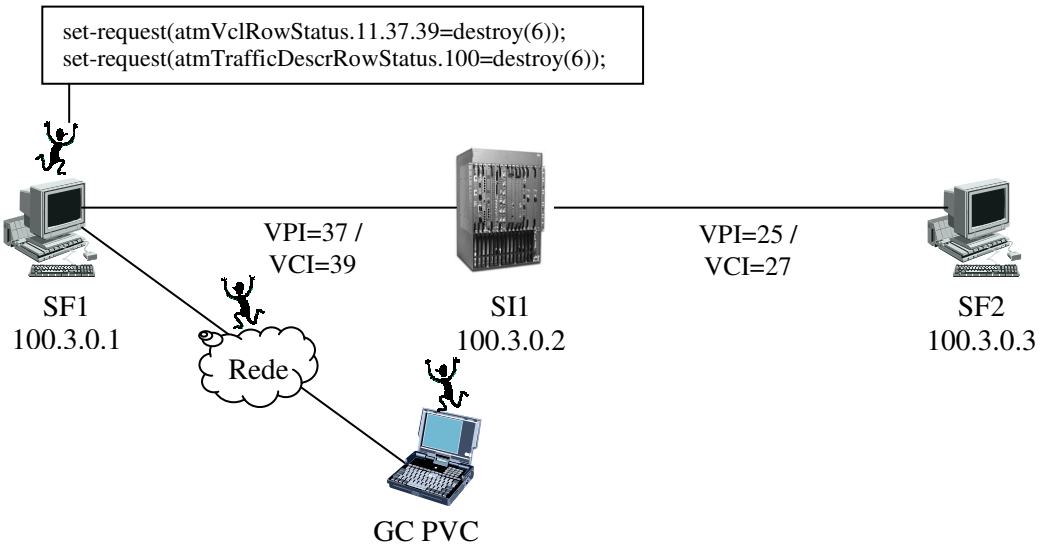


Figura 7.9a Liberação do VL e Descritores de Tráfego no SF1

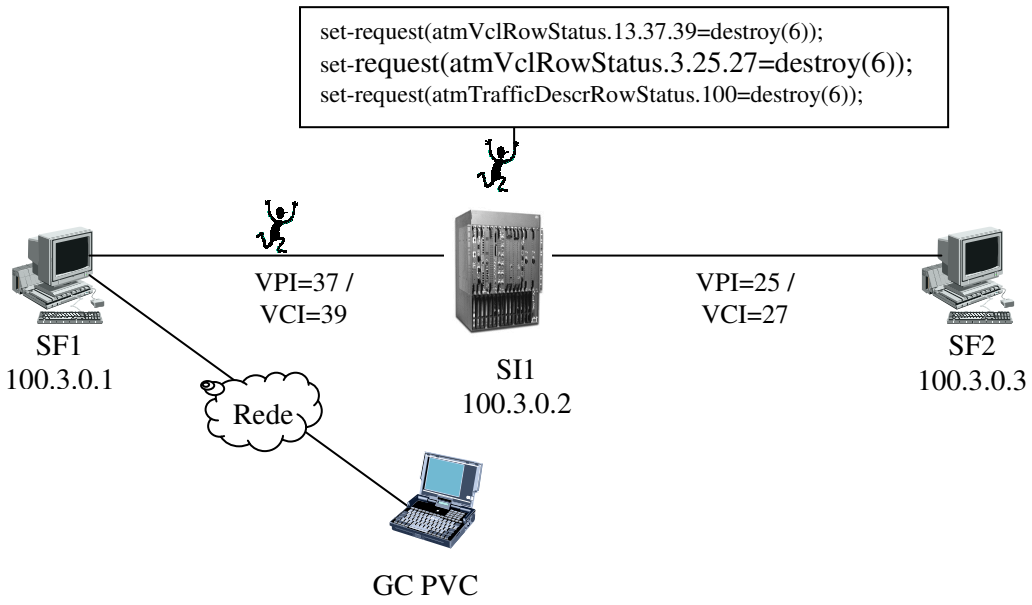


Figura 7.9b Liberação do VL e Descritores de Tráfego no SI1

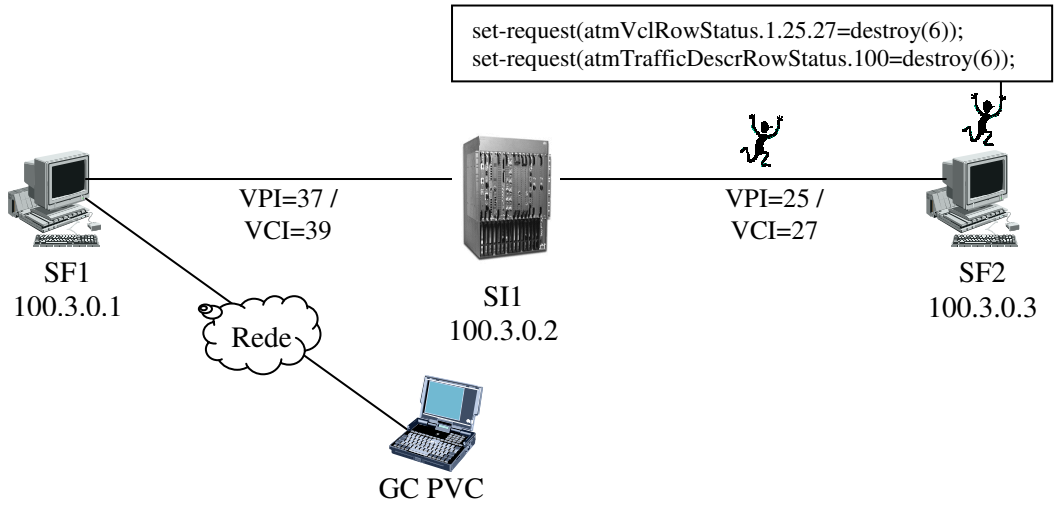


Figura 7.9c Liberação do VL e Descritores de Tráfego no SF2

7.3.4.3. Reconfiguração da Conexão Virtual

As principais aplicações de reconfiguração consistem das trocas de:

- Valores dos parâmetros de tráfego e/ou QoS;
- Troca de Topologia.

7.3.4.3.1 Troca de Valores dos Parâmetros de Tráfego e/ou QoS

Não é exigida capacidade adicional do agente SNMP. O AM derruba a VC corrente, define novas VLs com parâmetros desejados e cria nova VC seguindo as regras descritas acima. Esta operação é mais simples quando novo conjunto para VLs é definido inteiramente. Caso o gerente deseje manter valores VPIs/VCIs, deve seguir os seguintes passos [21]:

- Desativar o tráfego do VL nos sistemas finais assinalando **atmVclAdminStatus** para **down**;
- Liberar a conexão cruzada no sistema intermediário assinalando **atmVcCrossConnectRowStatus** para **destroy**;
- Desativar o tráfego do VL no sistema intermediário assinalando **atmVclAdminStatus** para **down**;
- Colocar VLs nos sistemas finais e intermediários fora de serviço assinalando **atmVclRowStatus** para **notInService**.

Então, como antes, configure os VLs, conexões cruzadas e ative o tráfego, implementando os seguintes passos [21]:

- Escolha ou crie nova linha de parâmetros de tráfego;
- Associe VLs com novas linhas de parâmetro de tráfego;
- Ative VLs nos sistemas finais e intermediários assinalando **atmVclRowStatus** para **active**;
- Ative o tráfego do VL no sistema intermediário assinalando o **atmVcCrossConnectAdminStatus** para **up** e
- Ative o tráfego dos VLs dos sistemas finais assinalando **atmVclRowStatus** para **up**.

7.3.4.3.2. Troca de Topologia

Na troca de topologia, ao contrário da reconfiguração acima, são requisitados capacidades adicionais do agente SNMP, incluindo suporte a hardware/software [22].

7.4. Conclusão do Capítulo

Algumas conclusões podem ser tiradas a partir da integração destas duas tecnologias. Inicialmente, confirma-se a necessidade de descentralização para evitar alto tráfego na rede durante o monitoramento. O mais importante aspecto para nós é demonstrar que sistemas de AMs podem, de maneira fácil e efetiva, ser empregados em aplicações tradicionais como gerenciamento de redes. AMs autônomos podem ser explorados desde simples tarefas de configuração até complexas decisões a serem tomadas, no sentido de garantir perfeito gerenciamento de redes.

Envio de AMs à rede pode resolver problemas de escalabilidade de gerenciamento, reduzindo a carga da rede e distribuindo o processamento dos dados. Embora AMs tenham sido aqui utilizados para o gerenciamento de PVCs, suas estratégias e resultados podem ser aplicados para outros sistemas de gerenciamento que suportam mobilidade. Nossa proposta integra, de forma eficiente, sistemas descentralizados de AMs com técnicas bem conhecidas, testadas e difundidas do SNMP.

Conforme visto na seção 1.3, a partir de diversos estudos dedicados a este assunto, percebe-se que há alto grau de interesse na aplicação do paradigma de AM para suportar tarefas de gerenciamento de redes.

Capítulo 8. Conclusão

Como vimos, o termo agente está sendo largamente utilizado, atualmente. Agentes trabalham em ambientes específicos, nas mais diversas máquinas conectadas na rede. Suas tarefas contam com um conjunto de *scripts* para entrar na rede, postar e manter mensagens e gerenciar as comunicações necessárias entre os diversos *sites*.

Além de aspectos técnicos recaindo sobre o desenvolvimento de sistemas de agentes móveis, grande número de assuntos éticos e legais devem ser explorados pela comunidade de computação. A maioria das discussões destes aspectos tem sido em cima do âmbito legal e filosófico cujos membros são normalmente contrários às comunidades técnicas. Os tópicos mais relevantes são [39]:

- Autenticação. O mais importante aspecto do uso de agentes, no ambiente aberto de redes. Verificar a veracidade de qualquer encontro, com a autenticação, é um fator importante do mundo real e, logo, se tornará o mais importante componente da comunidade dos agentes.
- Segredo. A necessidade de manter a propriedade da informação, garantindo o sucesso contínuo das operações empresariais, irá aumentar, dada a habilidade do agente realizar operações independentes da plataforma do servidor. Embora o uso de esquema que gerencie a autoridade e a permissão de realizar somente tarefas específicas seja uma resposta à necessidade de segurança, o desconforto, junto com a sofisticação dos agentes, está condenado a progredir.
- Privacidade. A privacidade é muitas vezes mais importante para os indivíduos, do que para as companhias e governos, que freqüentemente procuram expandir suas influências, apesar de manter a propriedade da informação que aumenta sua margem competitiva. A privacidade de um indivíduo irá correr os mesmos riscos que os segredos das empresas; de

qualquer forma, indivíduos não podem ser tão cautelosos e protetores de suas privacidades.

- Responsabilidade. Idealmente, agentes devem ser cidadãos responsáveis pelas suas ações, ao entrarem num computador, no encontro com outros agentes, e no uso da informação, para entregar os melhores resultados ao seu proprietário. Aspectos relacionados com as operações dos agentes recaem, no mundo dos desenvolvedores, como uma função do processo de projeto responsável.

Pode-se concluir que, embora o sucesso do desenvolvimento de agentes funcionais tenha se movido, além do estágio embrionário, pelo menos três aspectos específicos fazem deles tecnologia atraente: mobilidade, adaptabilidade e colaboração. Usando esses três elementos, desenvolvedores podem construir agentes sofisticados e autônomos. Com o acréscimo de ambiente pessoal de preferências e comportamentos de um componente e suas capacidades, um agente se torna sistema dinâmico e esperto, vez que aprende e troca seu comportamento, de acordo com as preferências do usuário, com o qual interage.

A mobilidade permite aos agentes se moverem, para ajudar pessoas a resolverem seus problemas. Não limitados a um único caminho ou restrito a uma única ação, agentes podem ser enviados, para explorar e estabelecer novos contatos. Parece claro que, no mundo dos negócios, uma operação não será eficiente, se cada indivíduo permanecer fixo no seu escritório pessoal, sem fazer visitas a outros.

A adaptabilidade permite ao agente descobrir novas soluções, quando trata de um problema desconhecido. Para operar efetivamente, agentes devem possuir habilidade mínima de se adaptar a novos ambientes.

A colaboração conduz a uma sinergia criativa, dentro do mundo dos negócios, tornando o todo maior do que a soma de suas partes. Da mesma forma, agentes, colaborando, podem compartilhar informações e juntá-las para solucionar problemas. Atualmente, grande número de desenvolvedores estão explorando esse tópico, independentemente do trabalho dedicado aos simples agentes de Internet.

A arquitetura AMI, desenvolvida no LARCES/UECE, apresentará plataforma própria de execução e desenvolvimento de AMs. Embora baseado nela, para o nível de pesquisa deste trabalho, foi necessário apenas emprego de plataforma de AMs já existente. Escolhemos o sistema Concordia que apresenta ambiente especializado para o uso eficiente desses AMs. Os programas utilizam os serviços do Concordia, para se moverem na rede e acessarem os serviços nela disponíveis. Exemplos comuns utilizam GUIs, banco de dados e outros agentes. Administradores controlam serviços disponíveis para cada agente e usuário, e completo gerenciamento é fornecido. Com o Concordia, nova classe fácil de escrever e fácil de executar está disponibilizada.

Concordia entrega um rico conjunto de características, suportando a implementação de agentes, para uma vasta área de usuários finais. Agentes, implementados com Concordia, são objetos móveis. Podem trafegar para diferentes locais e dispositivos da rede, para realizar tarefas nos diversos ambientes. A mobilidade do Concordia o coloca ao lado de objetos distribuídos e *applets* Java.

Concordia esconde a complexidade da mobilidade do programa de aplicação, para programadores, desenvolvendo aplicação, fundamentada em agentes similar ao desenvolvimento de programas estáticos e não móveis. Agentes mantêm seu estado interno, enquanto viajam pela rede, então são capazes de resumir a execução, na chegada ao novo local. Todo o trabalho de transporte do agente é tratado de forma transparente sem a intervenção do programador.

As potentes capacidades do Concordia ajudam desenvolvedores, a entregar completo acesso ao sistema de informação, a qualquer hora ou local. Usuários desse sistema podem ganhar acesso a informação, sem levar em consideração sua localização, mobilidade ou conexão de rede. Dando cobertura aos desenvolvedores das complexidades da comunicação de rede, Concordia reduz o tempo de desenvolvimento.

Esse trabalho apresentou proposta de integração da tecnologia de AMs com sistemas legados de gerenciamento de redes. Através dessa integração, superamos deficiências de escalabilidade e flexibilidade de redes decorrentes do paradigma "cliente servidor", apresentamos solução a problemas de gerenciamento centralizado de redes e finalmente criamos modo genérico de estabelecer PVCs em redes ATM heterogêneas. Através do ambiente proposto, o Gerente de Configuração PVC tem visão comum de todos dispositivos pertencentes à rede. O operador não precisa se preocupar com os sistemas de cada comutador e está apto a delegar a responsabilidade da configuração ao AM. Com uso da MIB AToM, temos visão genérica de todos os dispositivos envolvidos no processo de PVC, independentemente de seu fabricante. Seu acesso é possível através da utilização de classes Java do AdventNet, que permitiu interoperabilidade entre AMs e agentes SNMP. Nossa proposta automatizou a tarefa de configuração de PVC sem necessidade de intervenções de operadores nas decisões.

A metodologia apresentada é seqüencial devido às características naturais do procedimento de configuração de PVC. Estudos no envio de AMs de forma paralela estão sendo realizados, uma vez que, intuitivamente, com o aumento de nós na rede, o tempo gasto para configurá-los tende a ser menor do que na metodologia serial.

Os resultados aqui obtidos oferecem ótima indicação de que operações com AMs integrados a sistemas legados, tais como SNMP, têm significante impacto no desempenho das aplicações de gerenciamento de redes.

Ao final, gostaríamos de salientar que o resultado do esforço no desenvolvimento desta pesquisa gerou publicações nacionais e internacionais [57] [58] [60] [61]. Desdobramentos futuros virão onde continuaremos com o estudo do gerenciamento dinâmico de banda passante para aplicações executas em redes ATM e trabalharemos para o lançamento de um livro sobre agentes móveis de co-autoria com os professores Joaquim Celestino Júnior e Ana Luíza Bessa de Paula Diniz.

Referências

[1] STALLINGS, W. "*SNMP, SNMPv2, SNMPv3 and RMON1 and 2*", Addison Wesley, Third Edition, 1998.

[2] CCITT Recommendation X.711, "Common Management Information Protocol (CMIP) Specification," 1991.

[3] Concordia System Page URL=http://www.concordiaagents.com, 2001.

[4] AdventNet v2c Release 3.3 URL= http://www.adventnet.com, 2001.

[5] SAHAI, A., MORIN, C., "*Enabling a Mobile Network Manager (MNM) Through Mobile Agents*", *In Proceedings of Mobile Agents, Second International Workshop MA98*, Stuttgart, Alemanha (1998).

[6] NICKLISH, J., et.al., "*INCA: an Agent-based Network Control Architecture*", *In Proceedings of IATA'98 (2nd International Workshop on Intelligent Agents for Telecommunication Applications)*, Paris, France (1998).

[7] ZAPF M.,"*Design Paradigm in Agent-based Systems*", *In Proceedings of Distributed Aplications and Interoparable Systems – DAIS'97*, Cottbus, Alemanha, Setembro de 1997.

[8] ZAPF M., HERMANN K., and GEIHS K., "*Decentralized SNMP Management with Mobile Agents.*" In Proceedings of IM '99, Boston, May 1999.

[9] CHEIKHROUHOU, M., CONTI, P., LABETOULLE, J.,"*Flexible Software Agents for Automatic Provision of PVCs in ATM Networks*", Novembro de 1998.

[10] CHEIKHROUHOU, M., CONTI, P., LABETOULLE, J., MARCUS, K. "*Intelligent Agents for Network Management: Fault Detection Experiment*", Julho de 1998.

[11] Perpetuum Mobile Procura Project, Carleton University, URL=http://www.sce.carleton.ca/netmanage/perpetum.shtml, 2000.

[12] SUSILO, G., et.al. "*Infrastructure for Advanced Network Management based on Mobile Code*" in Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'98), pp. 322-333, February 1998.

[13] PAGUREK B., et.al. "*Configuration Management in Heterogeneous ATM Environments Using Mobile Agents*" In Proceedings of the Second International Workshop on Intelligent Agents in Telecommunications Applications (IATA'98), Paris, July 4th – 7th, 1998.

[14] PAGUREK B., et.al. "*Integration of Mobile Agents with SNMP: Why and How*" In EEE/IFI Network Operations and Management Symposium, (NOMS 2000), Honolulu, 2000.

[15] BOYER, J., PAGUREK, B. WHITE, T., "*Methodologies for PVC Configuration in ATM Heterogeneous Environment using Intelligent Mobile Agents*" In Proceedings of the First International Workshop on Mobile Agents for Telecommunication Applications (MATA'99), Ottawa, Canada, October 1999.

[16] PAGUREK B., et.al. “Network Configuration Management in Heterogeneous ATM *Environments*” In: Proceedings of the 3rd International Workshop on Agents in Telecommunications Applications IATA'98, Agent-World'98, Paris, France, July 1998.

[17] YANG, C., PHAN, S., “PMS: a PVC Management System for ATM Networks” In Proceedings of IEEE International Conference on Networking (ICN’01), Colman, France, July 9^{th} – 13^{th}, 2001.

[18] YANG, C., PHAN, S., “Fault Identification and Prevention for PVC Management in *ATM Networks*” In Proceedings of IEEE ATM 2000, Heidelberg, Germany, June 26^{th} – 29^{th}, 2000.

[19] SILVA, L. et.al., “JAMES: A Platform of Mobile Agents for the Management of *Telecommunication Networks*” In Proceedings of the Third International Workshop on Intelligent Agents for Telecommunication Applications (IATA’99), Stockholm (1999).

[20] BLACK, U., “*ATM, Volume II: Signaling in Broadband Networks*”, Prentice-Hall, 1998.

[21] PAN, H., "SNMP-Based ATM Network Management," Artech House, September 1998.

[22] SOARES, L. et.al., “Redes de Computadores: das LANs, MANs, e WANs às Redes ATM”, Editora Campus, 1999.

[23] Hazeldyne,A., Bigham,J., ***“A Heterogeneous Multi-Agent Architecture for ATM Virtual Path Network Resource Configuration,”*** *in Proceedings of Intelligent Agents for Telecommunications Applications 1998*, Springer Verlag, (1998)

[24] ITU-T Recommendation M.3010, “Principles for a Telecommunications Management *Network,*” March 1993.

[25] ITU-T Recommendation X.700, “Management Framework for Open System Interconnection (OSI) for CCITT Applications,” September 1992.

[26] ITU-T Recommendation M.3400, “*TMN Management Functions,*” October 1992.

[27] Recomendação ITU-T X.700, “*Management Framework for Open Systems Interconnection (OSI) for CCITT Applications,*” Setembro de 1999.

[28] SPRENKELS, R., *Management of ATM Networks*, M.Sc.Thesis, Enschede, Holanda, Junho de 1996.

[29] AHMED, M., e TESINK, T., Internet Engineering Task Force, RFC 1695, “*Definitions of Managed Objects for ATM Management, Version 8.0 using SMIv2*” August 1992.

[30] ATM Forum,”*Integrated Local Management Interface (ILMI) Specification Version 4.o,*”af-ilmi-0065000, Setembro de 1996.

[31] BROWN, T., and TESINK, K., Internet Engineering Task Force, RFC 1595, “*Definitions of Managed Objects for the SONET/DSH Interface Type,*” March 1994.

[32] CASE, J. et.al., Internet Engineering Task Force, RFC 1442, “Structure of Management Information for Version 2 of the Simple Network Management Protocol,” March 1993.

[33] Internet Engineering Task Force, draft-ietf-atommib-atm2-04.txt, "*Definitions of Supplemental Managed Objects for ATM Management,*" November 1995.

[34] MCCLOGHRIE, K., ROSE, M., Internet Engineering Task Force RFC 1213 <STD 17>, "*Management Information Base for Network Management of TCP/IP-based Internets: MIB-II´*, Março de 1991.

[35] ANSI <T1.105>, "*Digital Hierarchy – Optical Interface Rates and Format Specifications,*" 1998.

[36] ANSI <T1.102>, "*Telecommunications – Digital Hierarchy – Elictrical Interfaces,*" 1993.

[37] Recomendação ITU-T G.707, "*Synchronous Digital Hierarchy Bit Rates,*" Março de 1993.

[38] DONOJUE E.; ELMORE, B. - Architecting enterprise-level - ColdFusion applications part 1-one step closer to emulating, In ColdDusion Developers Journal.com URL=http://www.syscon.com/coldfusion/archives/0105/donahueelmore/index.html, 1999.

[39] COCKAYNE, W.R.; ZYDA, M. - *Mobile agents*. USA, Manning Publications Co., 1998.

[40] Mobile Agente System Interoperability Facilities Specification – MASIF, OMG TC Document orbos/97-10-05, November 1997.

[41] Foundation for Intelligent Physical Agents – FIPA, URL=http://drogo.cselt.stet.it/fipa

[42] OMG MASIF and CORBA, Integrating Mobile Agent Technology and CORBA, URL=http://www.det.ua.pt/Projects/difference/work/D7/d7chap4.html

[43] OMG: Common Facilities rfp3. Request for Proposal OMG TC Document 95-11-3, Object Management Group, Framingham, MA, November 1995.

[44] Mobile Agent Computing – A White Paper URL=http://www.concordiaagents.com/MobileAgentsWhitePaper.html, 2000.

[45] WONG, D. et.al, "*Concordia: An Infrastructure for Collaborating Mobile Agents,*" *In Proceedings of the First International Workshop on Mobile Agents 97 (MA'97)*, Berlin, Abril de 1997.

[46] WALSH, T. et. al., "*Security and Reliability in Concordia*," *In the Proceedings of the 31st Annual Hawaii International Conference on System Sciences* ***(HICSS31),*** Kona, Hawaii on January 6-9, 1998.

[47] "*Remote method invocation for java",* Javasoft Corporation, URL= http://chatsubo.javasoft.com/current/rmi/index.html, 2000.

[48] SELTZER M., "*Transaction support in a log-structured file system*", In *Proceedings of the Ninth International Conference on Data Engineering*, February, 1993.

[49] SELTZER M. et. al., "*A Log-structured file system for unix*," In *Proceedings of the 1993 Winter Usenix Conference*.

[50] *Encina rqs programmer's guide*, Transarc Corporation, Pittsburgh, Pennsylvania, 1994.

[51] MQSeries: Message Queuing Interface Technical Reference, IBM Corporation, Armonk, New York, 1994.

[52] CONDICT M.et. al., "Towards a world-wide civilization of objects", In Seventh ACM SIGOPS European Workshop, 1997.

[53] GRAY, R.S.,"*Agent tcl: a flexible and secure mobile-agent system*," In Proceedings of the Fourth Annual Tcl/Tk Workshop (TCL96), Monterey, California, 1996.

[54] CARDOSO, A., CELESTINO JR., J. "Concordia: Ambiente para Desenvolvimento de Agentes Móveis," Monografia apresentada na UECE, Fortaleza, CE, 2000.

[55] WHITE, J., "*Telescript Technology: Mobile Agents,*" General Magic White Paper, 1996.

[56] SHAPIRO, M., "*Structure and Encapsulation in Distributed Systems: The Proxy Principles,*" In *Proceedings of the 6th International Conference on Distributed Systems (ICDCS)*, IEEE Computer Science, 1986.

[57] CARDOSO, A., CELESTINO JR, J., "*A Proposal of PVCs Configuration Methodology Employing Mobile Agents*" In Proceedings of the Third International Workshop on Mobile Agents for Telecommunication Applications (MATA'01), Springer Verlag Pub., Montreal, Canada, August 14th – 16th, 2001.

[58] CARDOSO, A., CELESTINO JR, J., "*Configuração de PVCs em Redes ATM Heterogêneas*" In III WRNP2 Workshop RNP2, Florianópolis-SC, Brasil, 21 e 22 de maio de 2001.

[59] TESINK, T., BRUNNER, T., "*(Re)Configuration of ATM Virtual Connections with SNMP,*" *The Simple Times*, Volume 3, Number 2, August 1994.

[60] CARDOSO, A., CELESTINO JR, J., CELESTINO, R., "*Integrating Mobile Agents with Legacy Systems to Manage ATM Heterogeneous Networks*" In Proceedings of the Gestion de Reseau et de Service (GRES'01), Marrakech, Maroc, December 17th – 21th , 2001.

[61] CARDOSO, A., CELESTINO JR, J., CELESTINO, R., "*Management of Heterogeneous ATM Networks Based on Integration of Mobile Agents with Legacy Systems* " In Proceedings of the Network Operations and Management Symposium (NOMS 2002), Florence, Italy, April 15th – 19th, 2002.