UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**Rodrigo Martins Pagliares**

# UMA APLICAÇÃO PARA A MONITORAÇÃO DE ATALHOS MPOA CLIENT UTILIZANDO AGENTES MÓVEIS

**Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.**

**Prof. Dr. João Bosco Mangueira Sobral**

**Florianópolis, Março de 2002**

# UMA APLICAÇÃO PARA A MONITORAÇÃO DE ATALHOS MPOA CLIENT UTILIZANDO AGENTES MÓVEIS

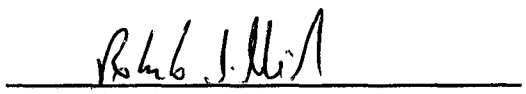
**Rodrigo Martins Pagliares**

**Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração (Sistemas de Computação) e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.**

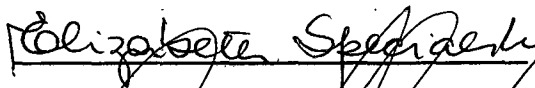
**Fernando Álvaro Ostuni Gauthier, Dr.**
**(Coordenador do curso)**

**Banca Examinadora:**

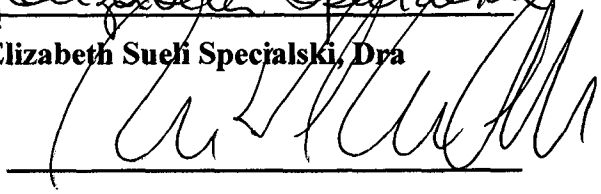
**João Bosco Mangueira Sobral, Dr.**



**Roberto Willrich, Dr.**



**Elizabeth Sueli Specialski, Dra**



**Mirela Sechi Moretti Annoni Notare, Dra.**

**"Para ser grande, sê inteiro: nada**
**teu exagera ou exclui.**
**Sê todo em cada coisa.**
**Põe quanto és no mínimo que fazes.**
**Assim em cada lago a lua toda brilha,**
**Porque alta vive."**

**Fernando Pessoa**

# AGRADECIMENTOS

**Muitas são as pessoas que me ajudaram no desenvolvimento deste trabalho, tanto direta como indiretamente.**

**Em especial, gostaria de agradecer à minha Super Mãe, por tudo que me proporcionou no decorrer de minha vida. Ao meu irmão, Guto, fica meu carinho especial...**

**Gostaria de agradecer aos amigos da República Casa da Vovó que estiveram ao meu lado a maior parte do ano. Aos amigos da República Pulgatório que me ensinaram o conceito de solidariedade e irmandade.**

**Ao meu orientador Prof. João Bosco Sobral que me guiou no decorrer do mestrado, ao Prof. Fernando Cruz, à minha amada amiga Mônica, Marcos, ao Sílvio que demonstraram grande paciência comigo no meu aprendizado, quando nem tudo caminhava conforme previsto.**

**Àqueles do NPD que em muito contribuíram para o desenvolvimento deste trabalho: em especial à Solange Sari, Edson Melo e Walter.**

**Aos membros da banca com enorme carinho e admiração.**

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE SIGLAS

| | |
|---|---|
| **ASN.1** | ***Abstract Sintaxe Notation One.*** |
| **AAL** | ***ATM Adaptation Layer.*** |
| **API** | ***Application Programming Interface.*** |
| **ARP** | ***Address Resolution Protocol.*** |
| **ASDK** | ***Aglets Software Development Kit.*** |
| **ATM** | ***Assynchronus Transfer Mode.*** |
| **ATP** | ***Agent Transfer Protocol.*** |
| **BUS** | ***Broadcast and Unknown Server.*** |
| **CLP** | ***Cell Loss Priority.*** |
| **CPI** | ***Common Part Indicator.*** |
| **IEEE** | ***Institute of Eletrical and Eletronics Engineers.*** |
| **IETF** | ***Internet Engineering Task Force.*** |
| **GUI** | ***Graphical User Interface.*** |
| **IP** | ***Internet Protocol.*** |
| **IPv4** | ***Protocolo da Internet versão 4.*** |
| **ITU-T** | ***International Telecommunications Union.*** |
| **JVM** | ***Java Virtual Machine.*** |
| **LAN** | ***Local Area Network.*** |
| **LANE** | ***LAN Emulation.*** |
| **LEC** | ***LAN Emulation Client.*** |
| **LECS** | ***LAN Emulation Configuration Server.*** |
| **LES** | ***LAN Emulation Server.*** |
| **MAC** | ***Medium Access Control.*** |
| **MASIF** | ***Mobile Agent System Interoperability Facility.*** |
| **MIB** | ***Management Information Base.*** |
| **MPOA** | ***Multi Protocol Over ATM.*** |
| **MPS** | ***MPOA Server.*** |
| **MPC** | ***MPOA Client.*** |

| | |
|---|---|
| **NBMA** | *Non Broadcast Multiple Access.* |
| **NHC** | *Next Hop Client* |
| **NHRP** | *Next Hop Resolution Protocol.* |
| **NHS** | *Next Hop Server.* |
| **NNI** | *Network to Network Interface.* |
| **OMG** | *Object Management Group* |
| **OSI** | *Open Systems Interconection* |
| **PDU** | *Protocol Data Unit* |
| **QoS** | *Qualidade de Serviço* |
| **RFC** | *Request For Comments* |
| **RMAV** | *Rede Metropolitana de Alta Velocidade.* |
| **RSVP** | *Resource Reservation Protocol.* |
| **SAR** | *Segmentation and Reassembling.* |
| **SNMP** | *Simple Network Management Protocol.* |
| **TCP** | *Transmission Control Protocol.* |
| **UNI** | *User to Network Interface.* |
| **URL** | *Uniform Resource Locator.* |
| **UU** | *User to User Indicator.* |
| **VC** | *Virtual Circuit.* |
| **VCI** | *Virtual Channel Identifier.* |
| **VPI** | *Virtual Path Identifier.* |

# RESUMO

**Este trabalho demonstra a viabilidade de se integrar a tecnologia dos Agentes Móveis à de Gerência de Redes, utilizando recursos como a ferramenta *Aglets* proposta pela IBM do Japão para agentes móveis e a API para gerência SNMP desenvolvida por Jonathan Sevy. Foi desenvolvida uma aplicação para a monitoração de atalhos MPOA Client utilizando agentes móveis. Para isso, foi monitorado o número de pacotes transmitidos sobre o atalho criado pelo *MPOA Client* entre duas subredes distintas. O *MPOA Client* é um dos componentes da arquitetura *MPOA (Multiprotocol over ATM)*, implementada em um comutador ATM. Com a criação deste atalho, é possível que dois computadores localizados em redes diferentes consigam se comunicar sem a necessidade de um roteador. Dessa forma observou-se a eliminação do problema de sobrecarga nos roteadores da rede diminuindo a latência de comunicação.**

# ABSTRACT

**This work demonstrates the viability to integrate the Mobile Agents and Network Management's technologies by using tools such IBM Aglets and the API SNMP developed by Jonathan Sevy. It was ilustrated an application for monitoring the MPOA Client's shortcut using mobile agents. To accomplish this task, it was monitored the number of packets trasmitted over the shortcut created by the MPOA Client, which is a component of MPOA architecture implemented in a ATM switch. With this shortcut, it is possible the communication between two computers residing in distincts networks without the use of a router. In this way, the router's overhead has been eliminated decreasing the communication latency.**

# 1. Introdução

As redes de computadores apresentam um crescimento significativo em termos de volume de tráfego de dados e favoreceram o surgimento de novos equipamentos bem como de novas aplicações. As aplicações multimídia destacam-se entre as novas aplicações demandando uma grande capacidade de processamento, de cálculo de rotas e de encaminhamento de pacotes nos roteadores. Como conseqüência da utilização dessa aplicação, uma maior demanda de largura de banda é exigida, e, assim, devem ser identificados problemas de sobrecarga, gargalo, aumento de latência e queda de desempenho nos roteadores da rede.

Propõe-se para resolver o problema de sobrecarga e queda de desempenho dos roteadores a junção das tecnologias IP (*Internet Protocol*) e ATM (*Assynchronus Transfer Mode*). Para tanto, faz-se necessário ter uma rede de serviços com roteamento IP e recursos de alta velocidade, fornecidos pela infra-estrutura ATM. Dessa forma considera-se possível obter os benefícios das duas tecnologias: a simplicidade de implementação do IP e a qualidade de serviço da tecnologia ATM.

O ATM Forum e o IETF (*Internet Engineering Task Force*) uniram forças e seus grupos de trabalho para disponibilizar uma nova tecnologia que integrasse o IP e o ATM. Dessa união, surgiu o MPOA, cuja especificação foi concluída em 1997.

Com o surgimento do MPOA tornou-se clara a necessidade de ferramentas para gerenciamento e monitoramento dos recursos por ele oferecido e, em especial, a criação de atalhos entre computadores de LANEs distintas que conseguem se comunicar sem o uso de roteadores.

No que diz respeito à questão de monitoramento, uma das tecnologias que mais vêm sendo estudada atualmente é a da utilização de agentes móveis como ferramenta de gerenciamento de redes de computadores.

Este trabalho utiliza essa tecnologia para o desenvolvimento de uma aplicação prática para o monitoramento do atalho criado pelo MPOA Client entre subredes distintas. Será analisado para isso o número de pacotes transmitidos sobre este atalho.

A motivação principal na escolha do tema de monitoramento do protocolo MPOA utilizando agentes móveis surgiu do fato de se querer avaliar a viabilidade da utilização da tecnologia de mobilidade através de agentes no monitoramento de redes de computadores.

Como motivação extra, pretende-se utilizar o trabalho aqui desenvolvido, na Rede Metropolitana de Alta Velocidade - RMAV-FLN, que faz parte do projeto RMAV em nível nacional.

Neste sentido, apresenta-se este trabalho, no contexto das redes ATM, utilizando MPOA (Multiprotocol Over ATM), como uma contribuição ao Projeto RMAV executado no âmbito da Universidade Federal de Santa Catarina. O projeto RMAV é uma parceria entre a RNP (Rede Nacional de Pesquisa) e o ProTeM-CC (Programa Temático Multi-institucional em Ciência da Computação) com apoio financeiro do CNPq e o Comitê Gestor da Internet no Brasil. O projeto RMAV-FLN tem como objetivos: colocar em operação a Rede Metropolitana de Alta Velocidade de Florianópolis; testar e disponibilizar experimentos de aplicações interativas na rede metropolitana; capacitar pessoal e instituições para operar e gerenciar redes de tecnologia ATM e implementar aplicações nesta, bem como, criar condições para cooperação e capacitação permanente das instituições envolvidas. A exemplo da RMAV-FLN as demais redes metropolitanas que compõem o backbone nacional de alta velocidade, utilizam a tecnologia IP sobre ATM (IP Clássico) e LAN emulada (LANE).

Optou-se por utilizar a tecnologia MPOA uma vez que a mesma propõe melhorar o desempenho da arquitetura LANE, acima mencionada, e atualmente configurada nas redes metropolitanas.

## 1.1 Histórico da Pesquisa

Algumas pesquisas sobre MPOA (Multiprotocol Over ATM) e gerência de redes de computadores utilizando agentes móveis foram consultadas, de forma a auxiliar o trabalho aqui desenvolvido. Dentre elas destacamos:

Em [Costa/99] é apresentada uma avaliação analítica do uso de agentes móveis em redes de computadores. O trabalho aborda uma dedução matemática para analisar o uso de agentes móveis e do protocolo SNMP, avaliando efeitos como a variação de latência e da banda passante, o tamanho inicial do agente e o tamanho de três diferentes tarefas de gerenciamento no tempo de resposta.

Em [Greenwood/97] é feita uma análise de desempenho de duas formas de *pooling* utilizando agentes móveis: uma distribuída com particionamento de uma rede em subdomínios e uma outra chamada "*Get and Stay*" para aquisição de dados para gerenciamento "*offline*".

As referências acima mencionadas serviram como ponto de partida para o desenvolvimento de uma aplicação, visando o monitoramento de redes de computadores utilizando agentes móveis.

Após ter sido feito o estudo sobre gerência de redes de computadores e mobilidade computacional, nossa atenção foi desviada no sentido de melhor compreender a tecnologia MPOA. Para isso, destacamos os seguintes trabalhos:

Venkataraman, da Universidade de New Hampshire [Natham/98], apresenta um *framework* para facilitar a automação de uma série de testes. Este trabalho serviu como forma de esclarecimento das nuanças proporcionadas pela tecnologia

MPOA. Através deste trabalho decidiu-se pelo monitoramento da criação de atalhos pelo MPOA Client.

Em [Siqueira/00] foi realizado um estudo-de-caso sobre interoperabilidade e gerência do MPOA através da utilização de diversos cenários. A principal contribuição desta pesquisa foi o fato de a mesma ter sido testada sobre os mesmos equipamentos utilizados no trabalho aqui apresentado, minimizando, assim, os possíveis problemas de criação das LANs Emuladas configuradas num roteador e de configuração de um *switch* com a solução MPOA Client integrada.

## 1.2 Objetivo

Este trabalho tem como objetivo o desenvolvimento de uma aplicação para o monitoramento do número total de pacotes transmitidos sobre o atalho criado pelo cliente MPOA (MPOA Client), empregando o paradigma de gerência de redes de computadores através de agentes móveis. Tal recurso a ser monitorado se encontra presente na MIB MPOA implementada em um comutador ATM localizado no NURCAD (Núcleo de Redes de Alta Velocidade e Computação de Alto Desempenho) da Universidade Federal de Santa Catarina - UFSC. Com esse monitoramento, torna-se possível ratificar a criação de um atalho entre dois computadores residentes em LANEs distintas, de forma que a comunicação entre eles se dá sem a utilização de roteadores.

Para implementação desta aplicação utilizou-se a plataforma de agentes móveis *Aglets* da IBM [AGLETS/99] que é fundamentada na linguagem de programação Java, juntamente com a API (*Application Programming Interface*) para gerenciamento de redes de computadores utilizando o protocolo SNMP. Tal API foi desenvolvida por Jonathan Sevy e pode ser encontrada em [JON/99]. Vale a pena lembrar que esta API possui código aberto e é disponível gratuitamente. O equipamento onde será feito o monitoramento é um comutador ATM (IBM 8371) com suporte ao MPOA Client.

## 1.3 Justificativa

Deve-se salientar que a transição LAN-ATM das aplicações atualmente existentes poderão em um estágio futuro acontecer com a utilização do MPOA. Pensando nisso, uma maneira prática de monitoramento do serviço MPOA será mostrada neste trabalho através do uso de agentes móveis já que um dos principais problemas encontrados no monitoramento de redes de computadores tradicional é o constante *pooling* que é feito dos gerentes em direção aos agentes estáticos acarretando um aumento do tráfego na rede. Com a utilização de agentes móveis é possível reduzir este tráfego extra através de delegação de autoridade dos gerentes em direção aos agentes.

## 1.4 Estrutura do Trabalho

No capítulo 2 descreve-se os principais aspectos da mobilidade computacional através de agentes bem como uma plataforma para execução de agentes móveis baseada na linguagem JAVA.

O capítulo 3 permite ao leitor um entendimento das tecnologias necessárias para a concretização desse trabalho. Serão mostrados neste capítulo conceitos sobre ATM, LANE, NHRP e MPOA.

Já o capítulo 4 mostra conceitos básicos de gerência de redes de computadores.

O capítulo 5 é devotado à descrição do cenário utilizado na realização deste trabalho.

Conclusões, resultados obtidos, dificuldades e perspectivas futuras estão contidos no capítulo 6.

O apêndice A exibe a MIB MPOA.

# 2 AGENTES MÓVEIS

Para se entender o modelo de gerência proposto, é necessário que se conheça as funcionalidades dos agentes móveis e seu modo de operação. Neste capítulo são apresentadas a definição e caracterização dos agentes, os benefícios que a mobilidade adiciona, e a forma de utilização de agentes.

## 2.1 Definição de Agentes

Várias definições são encontradas para conceituar um agente [MASIF/97]. No contexto deste trabalho assume-se que os agentes são entidades de software com um conjunto de operações em nome de um usuário ou de outro programa com algum grau de independência ou autonomia, e assim dispõem de algum conhecimento ou representação das metas de usuários.

Agentes móveis são utilizados em sistemas em rede com objetivo de facilitar o controle e a realização de tarefas distribuídas e coordenadas entre indivíduos e sistemas. O fundamento principal está na capacidade de transferir o seu programa, ou código de execução, de um ponto a outro da rede e reiniciar ou continuar sua execução neste novo ponto. Uma das formas mais primitivas disso são os Applets Java [Cornell/00] que se transferem do servidor para o navegador do usuário onde iniciam sua execução. As definições mais estritas de Agentes Móveis exigem que o próprio código de execução tenha autonomia para invocar a sua transferência para o novo local. Agentes móveis, também são chamados de objetos ou agentes autônomos, código móvel, agentes transportáveis, ou mesmo objetos móveis.

O Agente também pode armazenar informações e transferí-las junto com o seu código. Uma ação típica para um Agente Móvel é tranferir-se para algum ponto da rede, buscar alguma informação, armazená-la e depois transferir-se de novo carregando essa informação para o local onde foi criado e ativado.

Vários sistemas dão suporte a criação ou programação de agentes móveis, com funcionalidades dependendo do foco principal de aplicação.

## 2.2 Propriedades

As características e propriedades principais de agentes móveis ainda não foram completamente formalizadas. O esforço mais recente de padronização é a norma MASIF (*Mobile Agent System Interoperability Facilities*) [MASIF/97] proposta pelo OMG (*Object Managment Group*) que define um conjunto de interfaces básicas de um sistema de Agentes Móveis. Os conceitos e funcionalidades encontrados na literatura podem ser descritos pelas seguintes propriedades [LANGE/98]:

- Autonomia
- Mobilidade
- Reatividade
- Comunicação
- Aprendizado

### 2.2.1 Autonomia

Uma das principais diferenças entre um agente e um programa comum é capacidade do agente de seguir suas metas com autonomia, isto é, sem interações ou comandos do ambiente. Um agente não precisa ter cada um de seus passos aprovados pelo usuário, ou por outros agentes; ele é capaz de atuar por si só.

Para ter um comportamento autônomo, um agente deve ter controle sobre suas ações, estados internos e ter recursos e capacidades necessárias para executar suas tarefas, como por exemplo, a capacidade de percorrer a rede ou se comunicar com outros agentes.

### 2.2.2 Mobilidade

Esta é uma propriedade fundamental para um agente móvel; se constitui na transferência dos dados, código e estado de execução de um lugar para outro.

### 2.2.3 Interação

É a propriedade que determina que um agente deve reagir adequadamente, por influências ou informações de seu ambiente. Este ambiente pode consistir de outros agentes, usuários humanos, fontes de informações externas ou objetos físicos.

O agente deve ter sensores ou deve possuir seu próprio modelo interno do ambiente (do qual pode obter conclusões por si próprio), para ser capaz de reagir a mudanças no ambiente. O comportamento do agente depende do ambiente.

### 2.2.4 Comunicação

A capacidade de comunicação permite que o agente se comunique com seu ambiente. Uma linguagem de comunicação de agentes oferece protocolos padronizados para a troca de informações entre agentes.

Agentes cooperativos são usados quando um problema excede a capacidade de um agente individual. Os agentes existentes que já possuem uma solução podem ser usados por outros agentes.

A cooperação entre agentes permite soluções rápidas e melhores para tarefas complexas que excedam as capacidades de um único agente. Os agentes se beneficiam da cooperação porque suas tarefas são divididas entre outros agentes.

### 2.2.5 Aprendizado

É um processo onde o agente processa as informações obtidas ao longo de seu ciclo de vida, adaptando seu curso de ação em função das novas informações. Esta funcionalidade é extremamente interessante em aplicações de busca de informação, onde o agente pode, ao longo da migração, refinar seus parâmetros de busca através de referências nos documentos já recuperados. Pode-se distinguir 3 níveis de aprendizado:

- **Deliberativo – O agente deve poder realizar uma modelagem simbólica de regras de todo seu ambiente. O conjunto de regras irá guiar as suas decisões sobre suas ações.**

- **Incremental – O agente aprende apenas através da aquisição de novos conhecimentos, ou seja, incorporando novos algoritmos ou funções. Ele não tem a capacidade de identificar se eles são ou não mais eficientes.**

- **Adaptativo – Neste caso, o agente deve ser capaz de alterar o seu processo de decisão através da eficiência do seu resultado. Técnicas de programação evolutiva, Redes Neurais, Algoritmos Genéticos entre outras, podem contribuir para a implementação deste tipo de aprendizado.**

**A inteligência de um agente é formada por três componentes principais: a base de conhecimento interna, a capacidade de raciocínio baseada no conteúdo da base de conhecimento e a habilidade de aprender ou adaptar-se as mudanças no ambiente. A capacidade de raciocínio coloca o agente na posição de ser capaz de observar seu ambiente e tomar ações específicas quando ocorrem mudanças no ambiente. A habilidade de aprender, baseada em experiências anteriores e adaptar seu comportamento ao ambiente, é de extrema importância para o comportamento inteligente do agente.**

## 2.3 Arquitetura

**A arquitetura de um sistema de agentes móveis é constituída pela sua plataforma de operação e pelos agentes que rodam sobre ela. Os agentes são na verdade algoritmos, escritos em alguma linguagem, que são executados pela plataformas. Nas subseções seguintes serão descritas a plataforma de operação escolhida para efetivação desse trabalho bem como a linguagem de programação utilizada nessa plataforma para o desenvolvimento de agentes móveis.**

**Para construção ou programação de um agente é necessário definir uma linguagem com suporte necessário às funcionalidades básicas para agentes móveis.**

Elas são, em geral, baseadas em linguagens já existentes, como o JAVA, Tcl/Tk e XML. [LANGE/98].

A linguagem JAVA foi utilizada neste projeto tendo em vista que essa é a linguagem utilizada pela plataforma de agentes móveis escolhida para o desenvolvimento do sistema proposto.

## 2.4 A plataforma Aglets

A plataforma de operação de agentes móveis utilizada neste trabalho chama-se Aglets, que nada mais é que uma tecnologia de agentes móveis do laboratório de pesquisa da IBM do Japão. O pacote Aglets Software Development Kit (ASDK) [AGLETS/99] da IBM inclui os arquivos de classe Java para aglets, extensiva documentação, inúmeros exemplos, código fonte, e Tahiti, um servidor de aglets. Com um servidor Tahiti executando em dois diferentes hosts, é bastante fácil desenvolver aglets que trafegam de um host para outro, bem como comunicar com outros aglets através dos hosts. A API também inclui classes para implementação de funcionalidade do servidor em aplicações Java.

### 2.4.1 Introdução ao Aglet

Um aglet é simplesmente uma classe Java que herda as propriedades e comportamento da superclasse Aglet.

Todos os aglets herdam uma variável de estado para texto simples, e ambientes aglets podem escolher exibir este texto de alguma forma. A figura 2.1 é um exemplo de como estabelecer uma variável texto após criação.

```
Import com.ibm.aglet.*;
Import com.ibm.aglet.event.*;
Public class Dissertacao extends Aglet {
  Public void onCreation(Object init) {
    setText("Mobilidade Computacional.");
  }
}
```

**Figura 2.1: Exemplo de como exibir um texto na saída padrão.**

Quando um aglet Dissertação é criado dentro do Tahiti, ele exibe o aglet como um item em uma lista selecionável. O valor do texto do aglet preenche a área mais a direita do item. Ver figura 2.2:

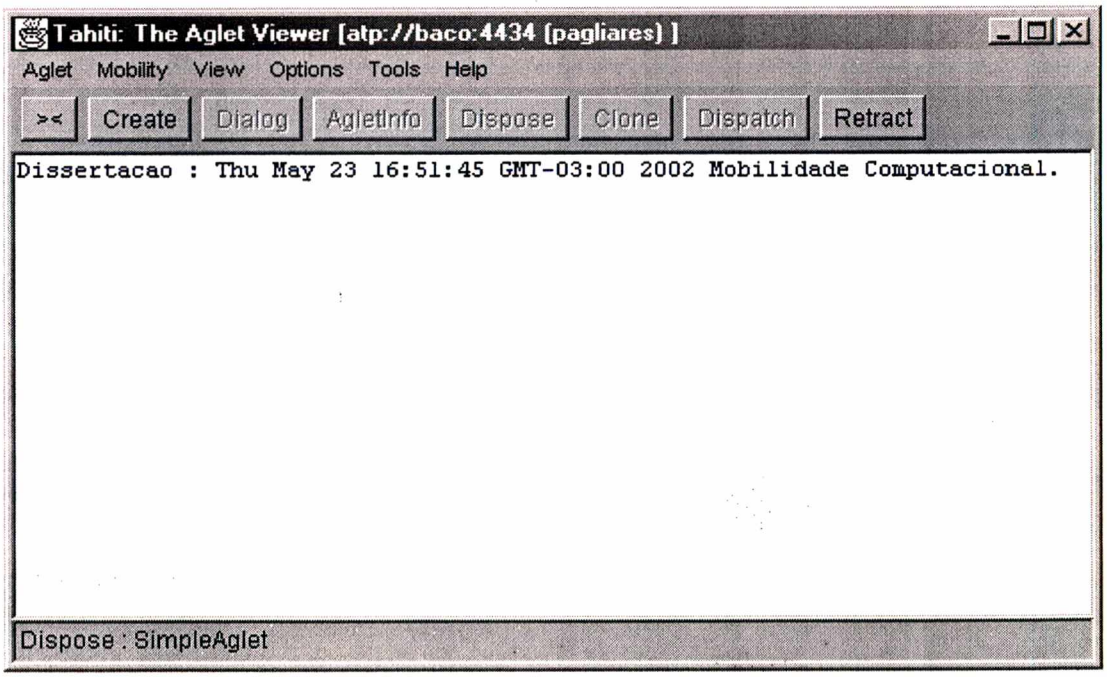



**Figura 2.2: Visualização do Aglet Dissertação dentro do servidor Tahiti.**

O servidor Tahiti fornece botões de comando para controlar o aglet selecionado. Por exemplo, ao selecionar o aglet Dissertação, podemos enviá-lo para uma localização remota, e subseqüentemente, trazê-lo de volta. O servidor Tahiti

usa diálogos para reunir informações pertinentes de operações tais como dispatch (enviar) e retract (recolher).

### 2.4.2 Aglet e seu proxy

Um *aglet* é na verdade gerenciado e referenciado através de seu *proxy* (o qual não é visto de dentro do Tahiti). O relacionamento entre aglets e um contexto é exibido na figura 2.3:

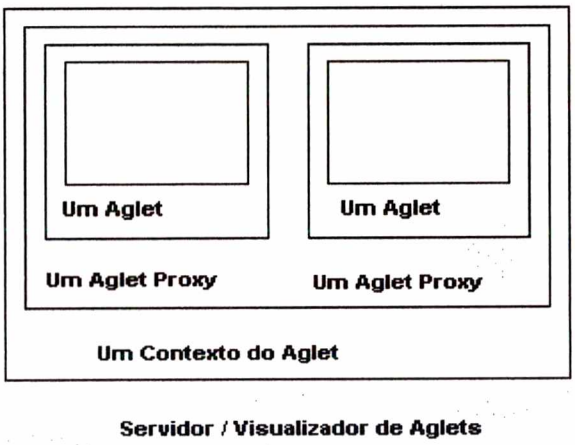



**Figura 2.3: O contexto de um Aglet.**

Em uma maneira programável, manipula-se um aglet através de seu proxy. Por exemplo, o seguinte segmento de código primeiramente usa getAgletContext() para conseguir uma referência a um objeto que implementa AgletContext:

```
...
AgletContext ac = getAgletContext();
AgletProxy proxy = ac.createAglet(null, "UmAglet", null);
...
```

O método createAglet() cria um aglet dentro de um contexto, retornando a referência para o proxy do aglet.

O proxy fornece transparência de localização para o aglet. Ou seja, um aglet pode ser manipulado via seu proxy, sem considerar se a sua localização atual é local ou remota. Por exemplo, o seguinte segmento de código cria um aglet, referencia o aglet via seu proxy para enviá-lo para um local remoto, e então o referencia via seu proxy para trazê-lo de volta:

```
...
URL destination = new URL("atp://localhost:5000/");
AgletContext ac = getAgletContext();
AgletProxy proxy = ac.createAglet(null, "UmAglet", null);
Proxy = proxy.dispatch(destination);
Proxy = ac.retractAglet(destination, proxy.getAgletID());
Proxy.dispose();
...
```

Outra vantagem da camada proxy é que ela permite que a estrutura aglet imponha checagem de segurança. Ou seja, o proxy emprega o gerenciador de segurança para verificar se uma invocação de método é válida ou não no contexto atual.

### 2.4.3 Operações do Aglet e seu ciclo de vida

Existem três operações fundamentais relacionadas com o ciclo de vida:

Create. Um aglet é criado dentro ou relativo a um aglet contexto; cada aglet é associado a um único identificador. Após criação, um aglet é inicializado e sua thread de execuções se inicia.

**Clone. Este processo produz um aglet distinto com um único identificador. Devido a limitações do Java, não é possível que o clone reflita o estado de execução do aglet clonado; portanto a thread de execução do clone se inicia novamente.**

**Dispose. Quando o trabalho de um aglet está feito, ele pode ser descartado. Este processo termina sua execução e remove o aglet do contexto. Quando nenhuma referência ao aglet sobrevive, o mesmo se torna à disposição para coleta de lixo (garbage collection).**

**Durante a existência do aglet, ele pode participar em várias operações fundamentais:**

**Dispatch. Um aglet é removido de seu contexto atual e retransmitido para outro, tipicamente remoto. No endereço destino, sua execução (seu método run()) se inicia.**

**Retract. Um aglet é removido do endereço remoto e re-introduzido para seu contexto de origem onde sua execução começa novamente.**

**Deactivate. A execução do aglet é paralizada e o seu estado é transferido para um meio de armazenagem secundário. O aglet não é removido do contexto; ele apenas "dorme" por um período específico.**

**Activate. Quando o período de sono do aglet expira, ele é ativado e sua execução começa novamente.**

### 2.4.4 O Servidor Tahiti

**O servidor Tahiti executa várias funções, mas talvez as operações mais comuns são a criação e carregamento de aglets, enviando-os para hosts remotos, e subseqüentemente os trazendo de volta [AGLETS/99]. Este processo é bastante simples com o Tahiti, como demonstrado nas próximas telas.**

**Primeiramente, o processo de criação de um aglet inclui várias operações:**

- **Carregar o arquivo classe (se ainda não estiver carregado)**

- **Instanciar o aglet**
- **Estabelecer o aglet no contexto**
- **Invocar onCreation()**
- **Invocar run()**

**Todas estas subtarefas acontecem quando o botão de comando "Create" é pressionado e o diálogo da figura 2.4 é respondido:**

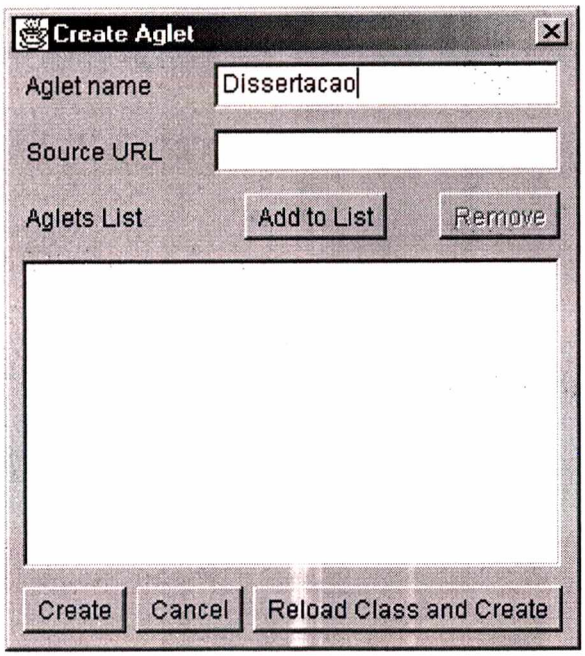



**Figura 2.4: Criação do Aglet no servidor Tahiti.**

**Após a criação, pode-se usar as funcionalidades que o servidor permite, como enviar o aglet a um host remoto. Essas funcionalidades estão exibidas na figura 2.5**

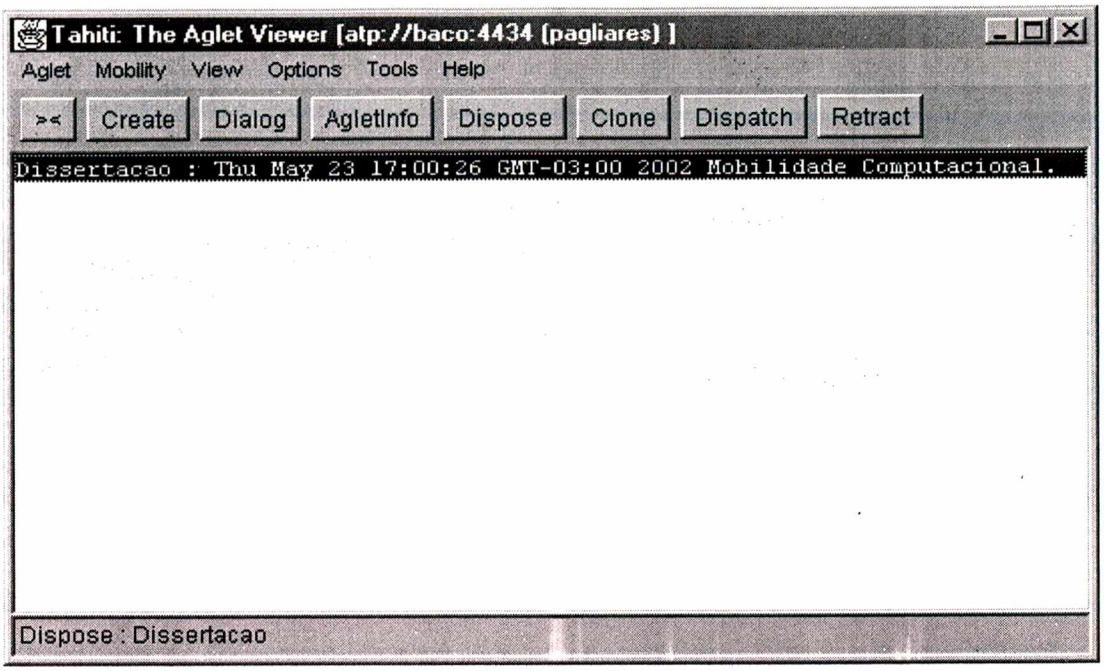



**Figura 2.5: Um Aglet pronto para ser enviado a um host remoto.**

**Para se enviar um aglet, neste caso, uma ïnstância "Dissertação", para um servidor remoto Tahiti (ou qualquer outro servidor projetado para aceitar e gerenciar aglets dentro de seu contexto aglet), utilizamos o botão de comando Dispatch que ativa um diálogo do qual você especifica a localização remota, tipicamente, com uma URL baseada no ATP (figura 2.6):**

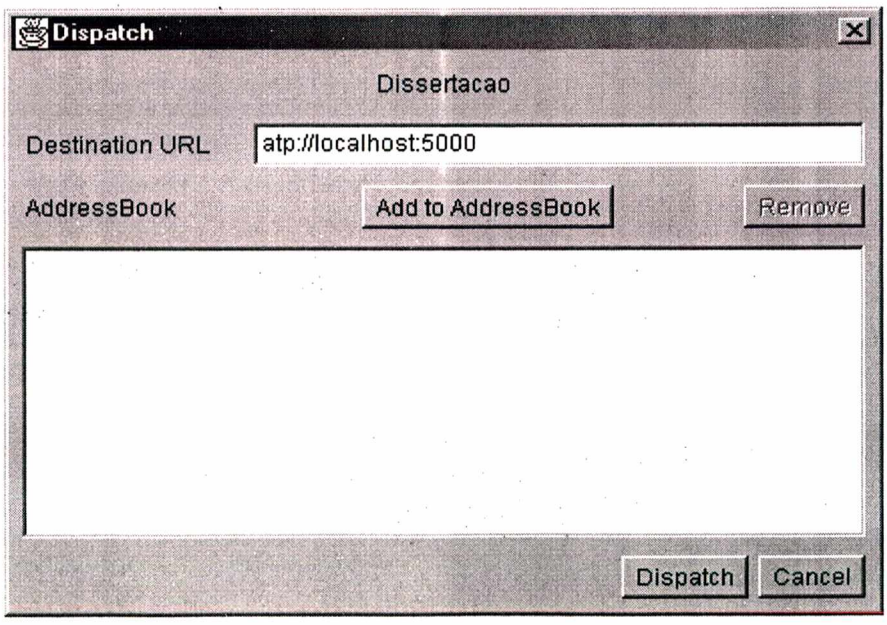



**Figura 2.6: Tela de envio de um Aglet.**

Após a especificação da URL, o aglet deve chegar na localização remota (figura 2.7):

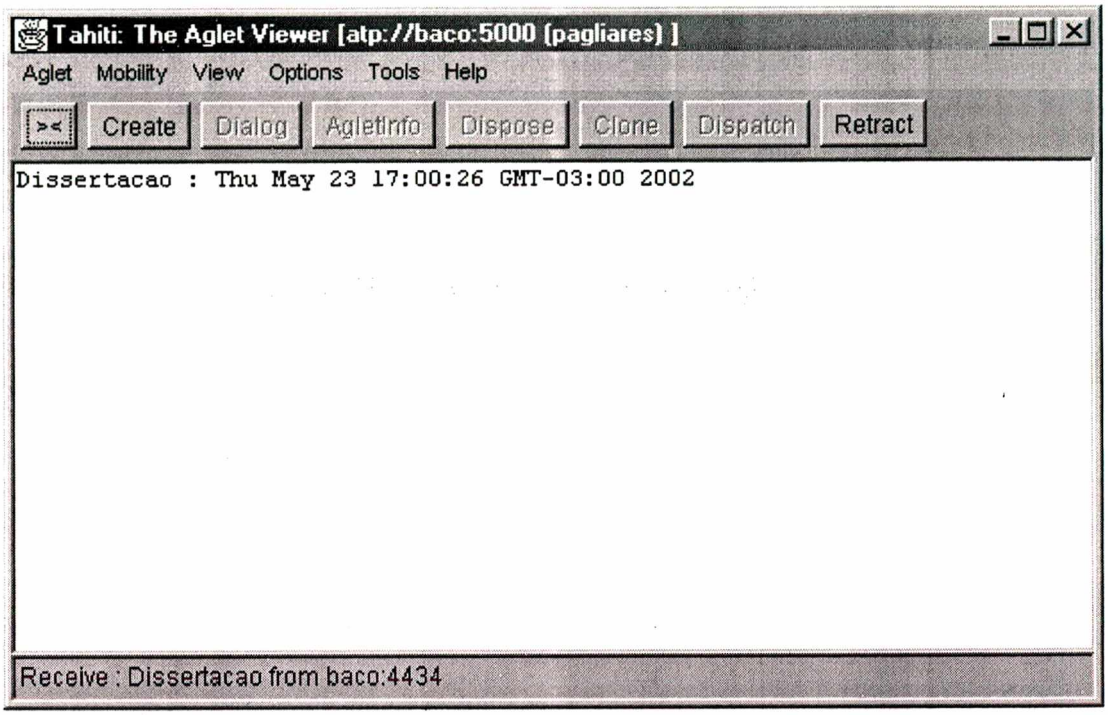



**Figura 2.7: Tela exibindo o Aglet na localização remota.**

**Para "Dissertação", os métodos onArrival() e run() executam as operações discutidas anteriormente, incluindo o estabelecimento do atributo texto do aglet, o qual o Tahiti monitora e exibe na área de exibição mais a direita do item.**

**O servidor Tahiti possui praticamente todas as funcionalidades necessárias a um agente móvel. Todavia, sua interface não reflete a necessidade deste projeto, tendo em vista que ela não é amigável no que se concerne ao trabalho aqui proposto e possui problemas de segurança já que a partir dela um usuário não autorizado passa a ter total controle sobre o aglet, podendo até alterar as opções de configuração, fazendo com que o mesmo não seja executado adequadamente. Pensando nisso, foi desenvolvida uma interface que implementa as funcionalidades principais do servidor Tahiti com intuito de tornar o sistema o mais amigável e seguro possível. Esta interface é exibida na figura 2.8:**

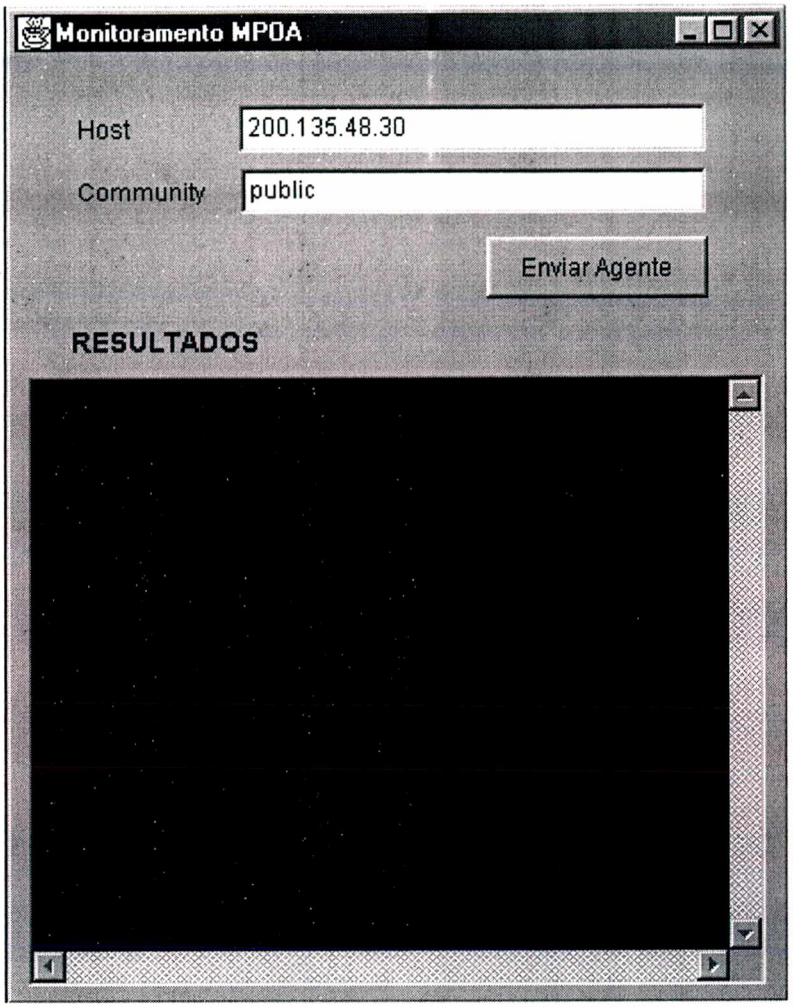



**Figura 2.8: A interface gráfica desenvolvida para o sistema.**

## 2.5 Resumo

**Este capítulo forneceu a tanto a parte técnica quanto a parte teórica sobre mobilidade computacional através de agentes móveis. A partir desse ponto, a atenção foi desviada para as tecnologias de redes de computadores que serviram de base para implementação da aplicaçào de monitormaento de criação de atalhos MPOA Client. Tais tecnologias são abordadas nos capítulos 3 e 4.**

# 3. A TECNOLOGIA ATM

ATM é uma tecnologia de comunicação, ou mais especificamente, de comutação rápida de pacotes, baseada em padrões abertos, que se propõe a servir de transporte comum para diversos tipos de tráfego, como dados, voz (áudio), imagem estática e vídeo. Neste capítulo é abordada a tecnologia ATM de forma suficiente para o entendimento deste trabalho.

## 3.1 Fundamentos

Redes ATM (*Asynchronous Transfer Mode*) são redes comutadas orientadas à conexão cujas taxas de transferência variam entre 25 a 622 Mbps, embora existam sistemas em testes operando na casa de Gbps. A taxa de transferência mais usual em redes ATM é de 155 Mbps.

O ATM trabalha nas camadas mais baixas do modelo OSI (de 1 a 3) [Torres/01] e, portanto, necessita de um protocolo trabalhando acima dele. Vários protocolos podem ser usados, entre eles o TCP/IP.

### 3.1.1 Funcionamento do ATM

No ATM cada canal virtual é identificado com um número de 24 bits. Podemos então ter até 16.777.216 canais em cada Switch ($2^{24}$). Em redes ATM um switch possui a mesma função de um roteador: definir a rota entre a origem e o destino.

A identificação do canal virtual possui dois campos: um de oito bits chamado VPI (Virtual Path Identifier) e outro de 16 bits chamado VCI (Virtual Channel Identifier). Normalmente a identificação de canal virtual é também chamada de VPI/VCI.

A divisão da identificação do canal virtual em dois campos facilita na hora dos switches da rede definirem o caminho da origem até o destino. Suponha a rede da figura 3.1. Todos os circuitos virtuais ligando o switch 1 ao switch 2 irão utilizar o mesmo número VPI, já que todos estes circuitos virtuais estão usando o mesmo caminho da origem até o destino. Todos os circuitos virtuais ligando o switch 1 ao switch 3 possuem um outro número VPI. Os números VCI identificam o circuito virtual dentro daquele caminho. Repare que circuitos virtuais em caminhos diferentes podem usar o mesmo número VCI.

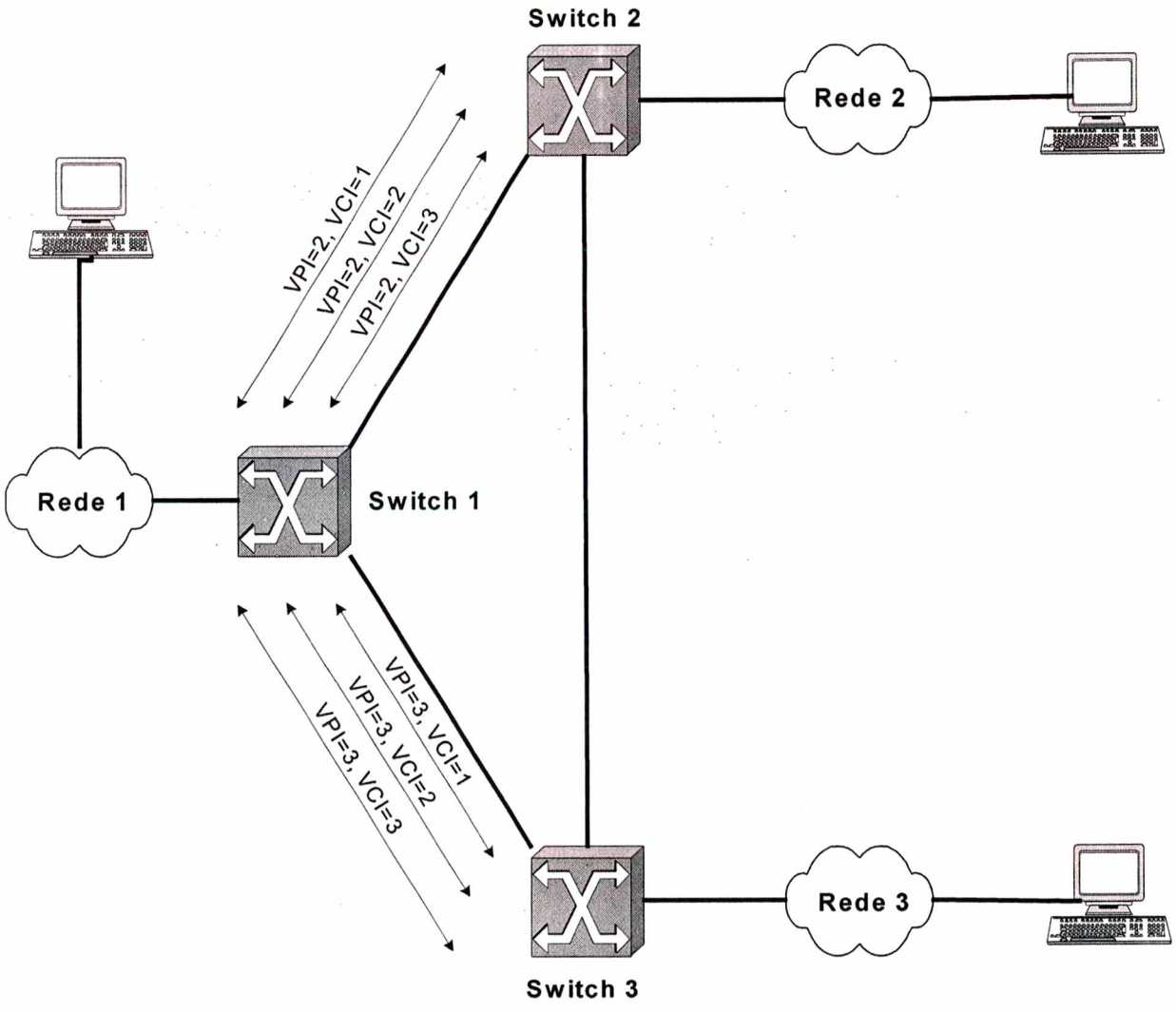



Figura 3.1: Exemplo de circuitos virtuais em uma rede ATM.

### 3.1.2 Transporte de células

Os pacotes ATM são chamados células. No ATM não existe a possibilidade de as células serem entregues fora de ordem. As células são enviadas em ordem e elas são recebidas na mesma ordem de envio.

Existem dois tipos de célula ATM: UNI (User to Network Interface) e NNI (Network to Network Interface) [CISCO850]. A célula do tipo UNI é usada no transporte de dados entre um computador e um switch, enquanto a célula do tipo NNI é usada no transporte de dados entre dois switches.

### 3.1.3 Estrutura da célula ATM

A estrutura da célula ATM é mostrada na figura 3.2. A diferença da célula UNI para a célula NNI está nos primeiros quatro bits do cabeçalho. Em células UNI, esse campo é usado para identificação de controle de fluxo. Em células NNI, esse campo é usado para endereçamento VPI, aumentando o endereçamento VPI em 4 bits, passando de oito par 12 bits.

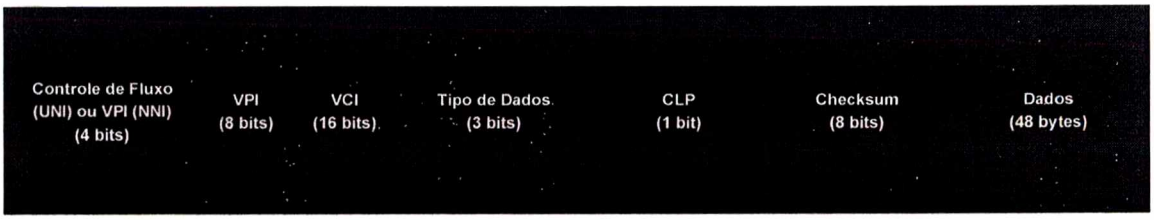



Figura 3.2: Estrutura de uma célula ATM.

Os campos existentes na célula ATM são os seguintes:

- Controle de fluxo: Este campo só existe em células UNI;
- VPI (Virtual Path Identification): Campo para endereçamento VPI;
- VCI (Virtual Channel Identification): Campo para endereçamento VCI;

- **Tipo de dados: Identifica o tipo de dados que a célula ATM está transportando. Na figura 3.3 mostramos os valores possíveis para esse campo;**

| Valor | Tipo de Dados |
|---|---|
| 000 | Dados sem congestionamento; não é a última célula. |
| 001 | Dados sem congestionamento; é a última célula do pacote. |
| 010 | Dados; há congestionamento; não é a última célula. |
| 011 | Dados; há congestionamento; é a última célula do pacote |
| 100 | Informações de controle entre dois switches. |
| 101 | Informações de controle entre o primeiro e o último switch do caminho |
| 110 | Informações sobre a taxa de transferência em serviços ABR |
| 111 | Reservado |

**Figura 3.3: Valores possíveis para o campo tipo de dados de uma célula ATM.**

- **CLP (Cell Loss Priority): Células que tenham esse bit ativado serão descartadas primeiro no caso de congestionamento da rede;**
- **Checksum: É o checksum dos dados do cabeçalho da célula. Este checksum não abrange, portanto, a área de dados.**

### 3.1.4 Camada de Adaptação

**Por causa do tamanho de 48 bytes das células ATM, existe uma camada intermediária entre os protocolos de alto nível a serem usados e a camada de transporte de células. Essa camada é chamada *Camada de Adaptação* e abreviada**

como AAL (ATM Adaptation Layer). O papel desta camada é pegar os dados recebidos dos protocolos de alto nível, encapsulá-los em um pacote específico dessa camada e dividir esses pacotes em blocos de 48 bytes a serem enviados para a camada inferior. Este processo é chamado de SAR (Segmentation and Reassembling).

Existem vários padrões utilizados nesta camada sendo que o mais utilizado é a especificação AAL5.

### 3.1.5 Estrutura do pacote AAL5

Ao contrário da maioria dos pacotes usados, que possuem as informações de controle em um cabeçalho, o pacote AAL5 possui as informações de controle ao final do pacote como pode ser visto na figura 3.4. Isto facilita o controle. Como já mencionado, nas células ATM, existe um campo que identifica o tipo de dado que está sendo transportado, indicando se a célula é ou não a última célula do pacote AAL5. Para ler os dados de controle do pacote AAL5, basta capturar a célula que possui os valores 001 ou 011 em seu campo tipo de dados e ler os últimos oito bytes desta célula.

| Área de Dados (até 65.535 bytes) | Pad (de 0 a 47 bytes) | UU (8 bits) | CPI (8 bits) | Comprimento (16 bits) | Checksum (32 bits) |
|---|---|---|---|---|---|

Figura 3.4: Estrutura do pacote AAL5.

A área de dados do pacote AAL5 é variável, podendo conter até 64 KB de dados, o que é mais do que suficiente para alojar pacotes recebidos dos protocolos de alto nível usados (TCP/IP, etc.).

Os campos encontrados no pacote AAL5 são os seguintes :

- **PAD: Como o pacote será dividido e transportado em células ATM que comportam 48 bytes cada, o comprimento total do pacote tem de ser um múltiplo exato de 48 bytes. Se isto não ocorrer, esse campo é usado. São adicionados zeros até que o pacote tenha um comprimento total múltiplo de 48 bytes;**
- **UU (User to User Indication): Esse campo atualmente não é usado, podendo ter qualquer valor;**
- **CPI (Common Part Indicator): Esse campo também não é usado, sendo preenchido com zeros;**
- **Comprimento: Indica o tamanho da área de dados, em bytes;**
- **Checksum: Checksum do pacote.**

### 3.1.6 Camada Física

**Redes ATM utilizam tipicamente fibras ópticas, embora exista a possibilidade de se usar cabo para trançado categoria 5 operando em 155 Mbps.**

## 3.2 LAN Emulation (LANE)

**LAN Emulation é um padrão definido pelo ATM Forum [LANE21/95] que possibilita a emulação de LANs (Ethernet/IEEE 802.3 ou IEEE 802.5 Token Ring) sobre uma rede ATM. Uma LAN emulada permite a comunicação de quadros de dados entre todos os seus usuários, bem como para uma LAN. Cada LAN emulada é independente uma da outra e os usuários não podem se comunicar além desses limites. Para que essa comunicação entre as LANs seja possível , devemos utilizar roteadores ou pontes (bridges).**

### 3.2.1 LANs Emuladas

**Em alguns ambientes existe a necessidade de se configurar domínios múltiplos, separados dentro de uma única rede. A partir desse requerimento surgiu a definição de LANs Emuladas – LAN Emulation (LANE) que forma um**

conjunto de dispositivos conectados ao ATM. Esse conjunto deve ser logicamente semelhante a um grupo de estações LAN conectados a um segmento Ethernet 802.3 ou a 802.5. Várias LANEs podem ser configuradas em uma rede ATM, sendo que os membros de uma LANE são independentes de onde o sistema final esteja fisicamente conectado.

### 3.2.2 Componentes de uma LANE

Os componentes de uma ELAN são apresentados como clientes e servidores de serviço, conforme a descrição abaixo:

Clientes:

- LAN Emulation Client (LEC) – É uma entidade em um sistema final, que faz encaminhamento dos dados, resolução de endereços e o registro de endereços MAC com o servidor LAN Emulation Server (LES). O LEC fornece uma interface padrão de LAN para os protocolos da camada superior em LANs tradicionais.

- Proxy LEC – Permite que estações LAN (tradicionais) participem de uma LANE.

Servidores:

- LAN Emulation Server (LES) – Implementa uma função de coordenação de controle para as LANEs. O LES torna possível o registro e a resolução de endereços MAC e ou descritores de rotas para endereços ATM. Toda vez que um LEC desejar se conectar a uma determinada LANE, ele irá contactar o LES para fazer a resolução do endereço MAC para o ATM. Há somente um LES por LANE.

- **Broadcast and Unknown Server (BUS) – É um servidor de multicast utilizado para fluxo de tráfego de difusão (broadcast e multicast) aos clientes em uma determinada LANE. Cada LEC está associado a somente um BUS por LANE.**

- **LAN Emulation Configuration Server (LECS) – Possui uma tabela de dados de LECs e das LANEs às quais os LECs fazem parte. Os LECs fazem consultas ao LECS e esse responde com o identificador de LANE apropriado, ou seja, o endereço ATM do LES que serve a LANE apropriada. Um LECS por domínio administrativo serve a todas as LANEs existentes nesse domínio.**

### 3.2.3 Conexões

**Existem duas classes de conexões utilizadas em LANs Emuladas. As conexões de controle implementam as várias porções de controle da LANE. As conexões de dados transportam o tráfego de dados das LANs emuladas. Esta separação de controle e dados simplificou alguns dos aspectos de conceituação do projeto.**

**Existe uma conexão transiente entre o LEC e o LECS durante a fase de inicialização de conexão à LAN Emulada. Esta conexão não precisa permanecer durante todo o ciclo de vida da LAN Emulada. É apenas uma troca inicial de informação de configuração.**

**O circuito criado neste tipo de conexão com o LAN Emulation Server é usado para executar o registro/resolução de endereço do protocolo. Nesta fase, que é exibida pela figura 3.5, não existe conexão com o BUS.**

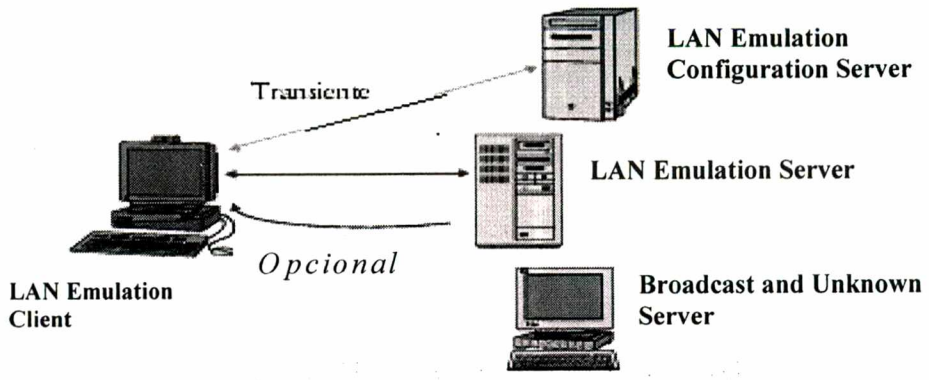



**Figura 3.5: Conexões de controle**

**O objetivo principal de um LEC é estabelecer uma conexão direta na qual os dados podem ser conduzidos. Circuitos virtuais, VCs, são criados entre os pares de endereço ATM fonte/destino. O tráfego para vários endereços MAC pode ser conduzido sobre o mesmo VC.**

**Além disso, o LEC criará uma conexão *multicast* com o BUS. Todo tráfego *Multicast/Broadcast* é desconhecido e é transmitido sobre esta conexão. O BUS cria uma conexão *multicast* para cada LEC, encaminhando o tráfego para cada LEC como exibido na figura 3.6:**

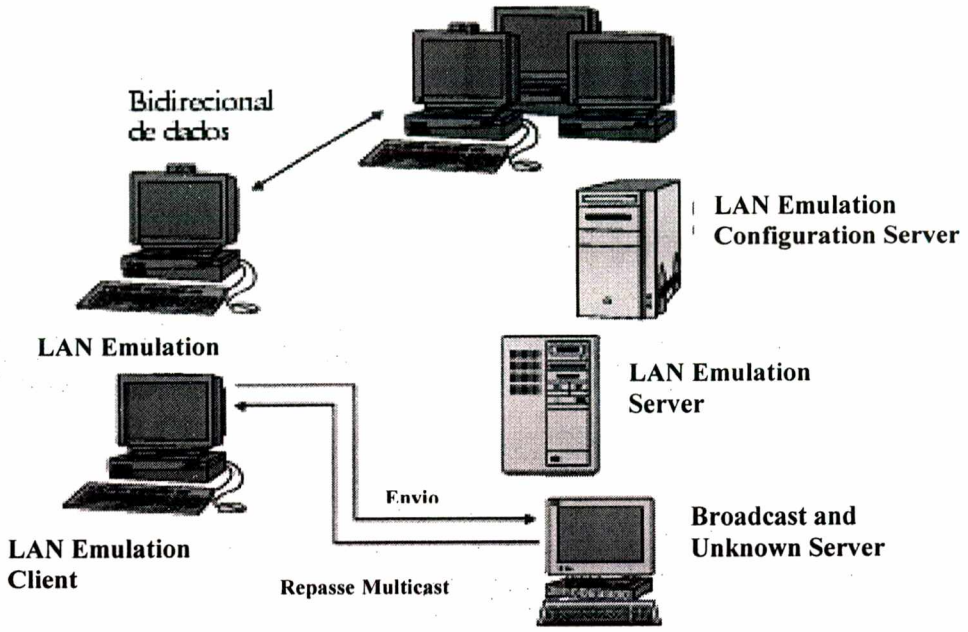



**Figura 3.6 - Conexões de dados**

## 3.2.4 Operação do LAN Emulation

**O processo de operação de uma LANE pode ser subdividido em quatro áreas.São definidas as áreas de Inicialização e Configuração, Registro e Conexão e Transferência de Dados. Cada área pode ser subdividida em uma fase de estabelecimento de circuito e uma fase de troca de informação.**

**Inicialização e Configuração:**

**Envolve a identificação do LEC com o LECS através do envio de seus endereços ATM/MAC e tipo de LAN desejada. Em resposta à solicitação de registro, o LECS retorna o endereço ATM do LES responsável pela rede lógica, juntamente com o tipo de LAN Emulada (Ethernet ou Token Ring).**

**Registro e Conexão:**

**Quando o LEC souber o endereço do LES ele pode desfazer a conexão com o LECS, e, então, estabelecer uma conexão de controle com o LES. O LES identifica um único número para o LEC e registra seus endereços MAC e ATM. O LES devolve para o LEC uma conexão unidirecional distribuída que pode ser usada**

pelo LEC para fazer requisições LE-ARP para resolver um endereço ATM em um endereço MAC.

Se o LES souber o mapeamento ATM-MAC ele pode escolher passar esse endereço diretamente pela conexão bidirecional com o LEC, caso contrário ele repassa a requisição pela conexão unidirecional distribuída para solicitar a resposta do LEC que conhece o endereço MAC requisitado. Para completar a conexão, o LEC transmite uma mensagem, via BUS, para todos hosts na LAN Emulada.

Transferência de dados:

Enquanto o LEC espera a resposta de resolução ARP do LES, o LEC envia pacotes para o BUS. O BUS entrega estes pacotes a todos os LECs. Se uma resolução ARP é recebida, o LEC estabelece uma conexão bidirecional de dados com o nodo destino, e usa esta conexão para transferir dados. Antes disto, o LEC deve usar o procedimento *flush* para verificar se todos os pacotes enviados anteriormente para o BUS foram entregues no destino. O LEC armazena localmente o mapeamento de endereços MAC-ATM que conheceu através de requisições ARP.

## 3.3 NHRP - Next Hop Resolution Protocol

Este protocolo desenvolvido pelo IETF possibilita o estabelecimento de uma conexão de circuito virtual comutado (SVC) através de uma subrede IP em um domínio ATM [NHRP2332/98].

O NHRP é composto de um servidor, denominado NHS (Next Hop Server), e pelo cliente NHC (Next Hop Resolution Protocol Client). O NHS geralmente está localizado junto com o roteador ou dispositivo de borda. O NHS é uma entidade que realiza o serviço NHRP dentro de uma nuvem Non Broadcast Multiple Access (NBMA) que conforme a plataforma do IP sobre ATM [IP-ATM1932/96] é uma subrede com um conjunto arbitrário de estações e roteadores que não possuem suporte nativo conveniente às facilidades de transmissões sem conexão a múltiplos

destinos, como nas subredes com capacidade broadcast e multicast. O NHC é uma entidade que inicia um NHRP Request de vários tipos e ordem para obter acesso aos serviços do NHRP [NHRP2332/98].

### 3.3.1 O modelo NHRP

A figura 3.7 exibe o modelo NHRP, bem como a operação do mesmo:

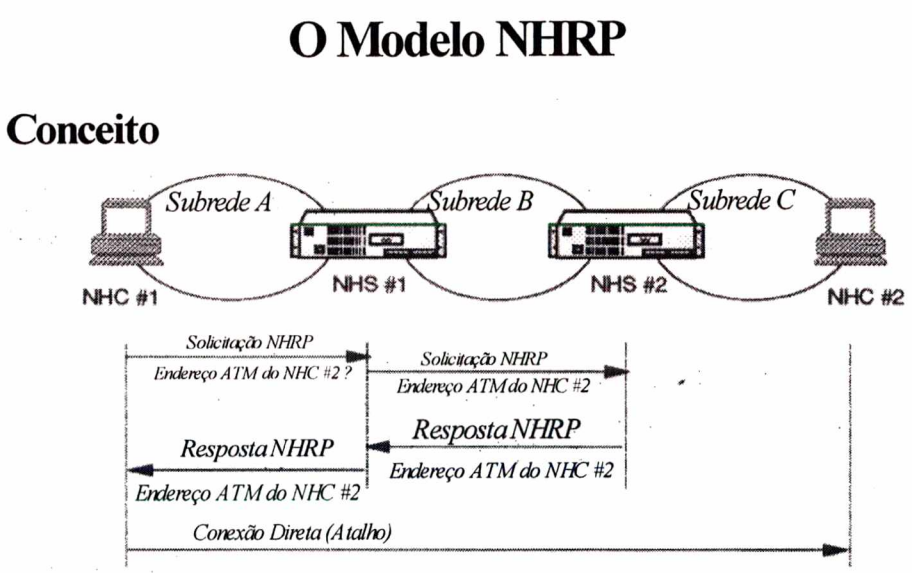



**Figura 3.7 – O modelo NHRP**

NHC-1 registra o seu endereço IP e NBMA com um ou mais servidores NHS. Quando este NHC-1 deseja estabelecer um atalho com o NHC-2, ele envia uma requisição ao NHS-1. A requisição inclui o endereço IP do destino; o NHS-1 que normalmente é um roteador, verifica se ele é o servidor do endereço de protocolo IP do NHC2. Caso seja, o servidor responde a requisição, enviando o endereço NBMA do destino e assim o NHC-1 pode estabelecer uma conexão de circuito virtual direto - Direct Virtual Channel Connection (VCC) com o NHC-2. Se o NHS-1 não puder atender a requisição do NHC-1 , o NHS-1 encaminha a requisição para o próximo roteador (next hop). Este encaminhamento continuará até que a requisição seja atendida ou finalizada quando o NHC destino não é alcançado.

## 3.4 MPOA (Multiprotocolo Sobre ATM)

MPOA é um protocolo desenvolvido pelo ATM Forum que provê interconexão de uma tríade de protocolos. De um lado existem os protocolos tradicionais, tais como IP Clássico Sobre ATM e LANE, que permitem *hosts* descobrirem uns aos outros em uma subrede lógica local e formarem a base das comunicações intra-LAN. Por outro lado, existem protocolos desenvolvidos pelo IETF que permitem *hosts* estabelecerem caminhos de comunicação direta através de uma rede ATM ultrapassando os limites de uma subrede sem a necessidade de um roteador. A parte final da tríade é fornecida por um conjunto de protocolos de serviços integrados, tais como RSVP e seus protocolos associados (RTP/RTCP/RTSP), que fornecem uma maneira de especificar QOS e ajudam na realização do monitoramento de performance da rede. A maior diferença entre MPOA e seus predecessores tais como LANE, é que MPOA suporta Next-Hop Resolution Protocol (NHRP) nos servidores multiprotocolo.

Para os administradores de rede, MPOA pode ser visto como um conjunto de protocolos, que se baseia em tecnologias existentes, e que é responsável pela resolução de problemas de estabelecimento de conexão entre pares de *hosts* que ultrapassam domínios administrativos. Isto possibilita capacitar aplicações a fazerem uso da capacidade da rede para fornecer QOS. Dentre as vantagens do MPOA, destacam-se:

- Os dispositivos de borda (edge devices) podem estabelecer conexão direta entre eles sem a necessidade de um roteador.
- Existe uma reduzida/restrita quantidade de tráfego broadcast.
- Existe flexibilidade na seleção do tamanho máximo da unidade de transferência para otimizar performance.
- O desenvolvimento de múltiplas LANs virtuais pode ser feito sobre uma única rede ATM.

### 3.4.1 LANs Virtuais

Um dos principais objetivos do MPOA é aumentar a capacidade dos administradores de rede no sentido de construir uma única rede ATM em um domínio empresarial, enquanto que, ao mesmo tempo, fornecer a capacidade de subdividir a rede em limites administrativos (LANs virtuais). Quando uma rede possui o conhecimento de se subdividir em múltiplos segmentos via servidores de endereço, a rede se torna conhecedora dos dispositivos conectados. Por esta razão, a rede é capaz de dinamicamente suportar movimento de hosts na rede e mudanças adicionais de configuração, detectando duplicação de endereços e adicionando endereços não-autorizados.

Ao se criar uma LAN virtual, a possibilidade de espionagem na rede pode ser reduzida. Conseqüentemente, a segurança pode ser muito maior que nos meios compartilhados convencionais. E de maneira mais importante, a possibilidade de se estabelecer informação de configuração dentro de um servidor central permite que a rede de um campus se torne mais dinâmica e mais gerenciável. Uma grande porcentagem de custos associados com movimentos/mudanças podem ser eliminados porque o servidor MPOA automaticamente colocará *hosts* em sua subrede virtual.

LANs virtuais podem dividir a rede em um grupo de hosts e podem restringir o acesso que esses grupos possuem com os servidores. Dessa maneira, a LAN virtual age como um firewall que fornece segurança adicional. Sem a utilização de LANs virtuais, administradores de redes precisam estabelecer filtros e listas de acesso nos roteadores ou nos servidores.

Segmentar a rede empresarial ATM em múltiplas LANs virtuais é também desejado para um aumento na escalabilidade. Dividir a rede é útil de maneira que os domínios broadcast e multicast são limitados a um tamanho apropriado. Portanto, a carga total da rede para este tráfego fica localizada.

### 3.4.2 Comunicação entre subredes utilizando MPOA

No modelo LANE do ATM Forum, uma LAN virtual é equivalente a uma LAN emulada. A comunicação entre LANs emuladas requer um roteador que é membro de duas ou mais LANs emuladas. O servico MPOA descreve como construir redes usando pontes LANE que sabem como se comunicar além dos limites de uma subrede. Estas pontes utilizam o Next-Hop Resolution Protocol (NHRP) para determinar a localização correta do dispositivo de egresso na rede ATM.

### 3.4.3 Introdução à arquitetura do MPOA

No modelo Multiprotocolo Sobre ATM, uma LAN virtual é similar à uma subrede virtual ou uma rede virtual. Entretanto, conexões entre LANs virtuais não necessitam de um roteador [MPOA114/99] . Com MPOA, os *hosts* são capazes de se comunicarem diretamente uns com os outros, mesmo no caso em que o caminho seja entre duas subredes lógicas IP. No modelo MPOA, a determinação do caminho que os dados podem tomar é baseada nas decisões tomadas pelo dispositivo de borda. Um circuito virtual pode ser criado para interconectar duas diferentes subredes. Este tipo de conexão é chamada de caminho mais curto (shortcut path).

Com MPOA, uma ponte LANE pode fornecer o serviço de comunicação com os servidores MPOA que fornecem serviços LES/BUS. Isto permite que usuários que implementaram produtos LANE possam reutilizá-los em uma rede MPOA com pouca ou nenhuma modificação. Quando um *upgrade* é feito na ponte LANE, ela pode monitorar o tráfego que sai da subrede e descobrir o endereço ATM do destino.

O modelo MPOA opera confiando nos servidores de roteamento multiprotocolo para descobrir a localização do dispositivo de saída da rede ATM mais próxima do destino. Quando a localização é encontrada, a rede ATM pode estabelecer uma chamada direta entre *hosts*, aproveitando-se da baixa latência da rede ATM e eliminando o auto-retardo de roteamento de pacotes IP.

### 3.4.4 Operação do MPOA

Uma rede MPOA consiste de vários componentes que podem ser subdivididos em roteadores, servidores, dispositivo de borda para conexão, servidores LANE, servidores next-hop, e hosts ATM. No modelo MPOA a malha ATM é considerada como uma rede física capaz de suportar muitas LANs virtuais. O protocolo atual documentado na especificação MPOA descreve quais componentes são usados para se construir a rede, juntamente com o fluxo de informações entre estes componentes. O modelo MPOA apenas define componentes lógicos, fluxo de mensagens, e comportamento sugerido, não proporcionando um produto.

No primeiro lançamento da especificação MPOA, os componentes principais de sua arquitetura são:

- Dispositivos de borda para conexão, que são usados para fisicamente anexar redes tradicionais ao sistema MPOA. Estes são similares às pontes LANE.
- Servidores de roteamento que possuem informações adquiridas pela execução de protocolos de roteamento e distribuição de estados entre eles. Estes são componentes de rede que suportam funções do MPS juntamente com LANE e NHRP. Um servidor MPOA mantém conhecimento de protocolos de camada 3 e topologias das áreas servidas.

Na figura 3.8 abaixo, são descritos os componentes do MPOA bem como o fluxo de informação entre estes.

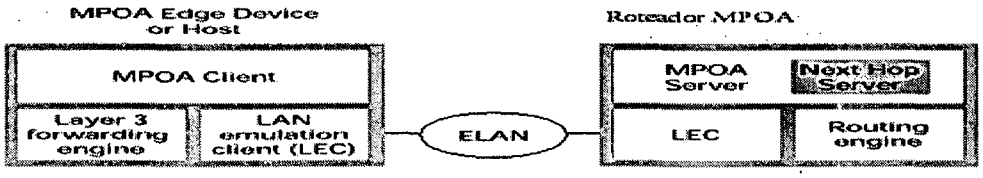



Figura 3.8: Conjunto de componentes lógicos e fluxos de informações entre componentes.

**Fluxo de informação**

**Um sistema MPOA utiliza vários fluxos de informações. O fluxo de informações define como os componentes trocam informações do estado MPOA e resolvem endereço- destino. Os componentes lógicos do MPOA podem ser divididos em clientes e servidores de forma que a implementação do protocolo segue quatro passos distintos, como descritos a seguir:**

1. **Configuração - É usada para recuperar informações de configuração e registro na rede.**

2. **Descobrimento – É a fase onde os dispositivos MPOA aprendem seus relacionamentos topológicos.**

3. **Fluxo de resolução de endereços – São usados após configuração para consultar servidores MPOA e então informar os servidores a respeito de mudanças de estado no lado-cliente.**

4. **Transferência de dados - É a transmissão de informação entre *hosts* utilizando um sistema MPOA.**

**Configuração**

**Antes que o MPC (Cliente MPOA) e MPS (Servidor MPOA) possam utilizar o sistema MPOA, eles devem se registrar e se configurar. O processo de configuração é executado, ou manualmente pelos administradores de rede, ou o MPC/MPS pode fazer uso do LECS (Servidor de Configuração da LAN Emulada). O servidor de configuração sabe quais clientes e servidores estão associados dentro de cada rede virtual.**

**Descobrimento**

O termo descobrimento quando diz respeito ao MPC e descreve a capacidade de se determinar a localização do NHS. Assim que a fase de descobrimento chega ao fim, os componentes MPOA dentro de um domínio podem passar mensagens NHRP/LANE, e permitir o MPC se comunicar entre eles ultrapassando os limites da subrede.

### Resolução de endereço

Uma vez que o host tenha se configurado e registrado com a rede, ele pode começar a se comunicar com outros hosts. Para se estabelecer o mapeamento de endereços de camada 2 para camada 3 deve ser resolvido via o MPS. Em uma comunicação entre subredes, servidores multiprotocolos, no decorrer do caminho, irão gerar e processar mensagens NHRP. Cada mensagem NHRP é passada salto a salto (hop by hop), até que ela alcance o MPS de saída. No MPS de saída da rede, uma mensagem é passada do servidor para o MPC para determinar os parâmetros ATM para realizar o encapsulamento como descrito na RFC 1483. O MPC servindo o host anexado responde com informação de endereçamento para o MPS, que por sua vez gera uma resposta NHRP seguindo o caminho de resolução tomado pela mensagem.

### Transferência de dados

Quando um MPC receber uma resposta para sua consulta NHRP, ele pode estabelecer um circuito virtual direto e começar a transferência de dados do usuário. Se o MPC destino é capaz de suportar a solicitação de qualidade de serviço (QOS) do MPC fonte, ele pode sinalizar este fato ao fonte retornando uma extensão NHRP. QOS é deixado como uma opção para os dispositivos-fim e deve ser tratado via um protocolo chamado RSVP [MPOA129/99].

# 4. GERENCIAMENTO DE REDES DE COMPUTADORES

Este capítulo contém a conceituação básica sobre a área de Gerência de Redes e destaca a estrutura e a identificação da informação de gerenciamento.

## 4.1 Características Gerais

A gerência de redes de computadores permite a monitoração, o controle e a coordenação do uso dos recursos físicos e lógicos no ambiente das redes gerando informações gráficas ou numéricas em tempo real sobre as alterações na topologia e no tráfego da rede. O gerenciamento de rede auxilia na manutenção e na identificação de problemas da rede para melhor poder atender aos usuários e empresas a terem um acesso consistente e de confiança aos recursos da rede.

A comunicação entre os elementos de rede para o propósito de gerência ou monitoramento é feita através do protocolo SNMP que define a forma com que as estações de gerenciamento executam as aplicações que monitoram e controlam os elementos de rede.

Os elementos de rede são equipamentos tais como hospedeiros, gateways, servidores de terminais, e similares, que possuem agentes de gerenciamento, responsáveis pela execução das funções de gerenciamento de rede, requisitadas pelas estações de gerenciamento.

## 4.2 Estrutura e Identificação da Informação de Gerenciamento

Um dos componentes conceituais de importância nos sistemas de gerenciamento é a forma com que as informações sobre os elementos básicos que se quer monitorar, gerenciar ou administrar, são armazenados. Estas informações precisam estar disponíveis de forma padronizada, de forma que qualquer aplicação de gerenciamento possa resgatá-las e torná-las úteis de alguma forma.

**Objetos gerenciados são acessados via informação virtual armazenada, denominada Base de Informação Gerencial (*Management Information Base*) ou MIB. Objetos de uma MIB são especificados usando a Notação Sintática Abstrata (*Abstract Syntax Notation One - ASN.1*).**

**Cada tipo de objeto (denominado *Object Type*) tem um nome, uma sintaxe e uma codificação. O nome é representado unicamente como um identificador de objeto (*Object Identifier*). Um Identificador de objeto é um nome administrativo determinado.**

**A sintaxe para um tipo de objeto define uma estrutura abstrata de dados correspondente para este tipo de objeto. Por exemplo, a estrutura de um dado tipo de objeto pode ser um Inteiro (INTEGER) ou uma String de Octetos (OCTET STRING) ou ainda qualquer construção ASN1 a ser avaliada para uso na definição da sintaxe de um tipo de objeto.**

**Uma codificação de um tipo de objeto é simplesmente definida como uma instância daquele tipo de objeto e esta é representada usando a sintaxe deste tipo de objeto. Deve-se salientar que o termo objeto não possui o sentido utilizado em um sistema orientado a objetos. Os objetos aqui discutidos são variáveis que possuem apenas estados e não dispõem de métodos além da leitura e escrita de seus valores**

**O identificador do objeto, que é usado para identificar os nomes dos objetos gerenciáveis, é uma seqüência de inteiros que atravessa uma árvore global. Esta árvore consiste de uma raiz conectada a um número de nós rotulados lado a lado. Cada nó pode, deste modo, ter seus próprios filhos os quais são rotulados. Neste caso, pode-se assumir este nó como uma sub-árvore. Este processo pode continuar de um nível arbitrário ao nível mais profundo na árvore. A figura 4.1mostra a árvore de registro utilizada para nomeação de objetos definidos na MIB.**

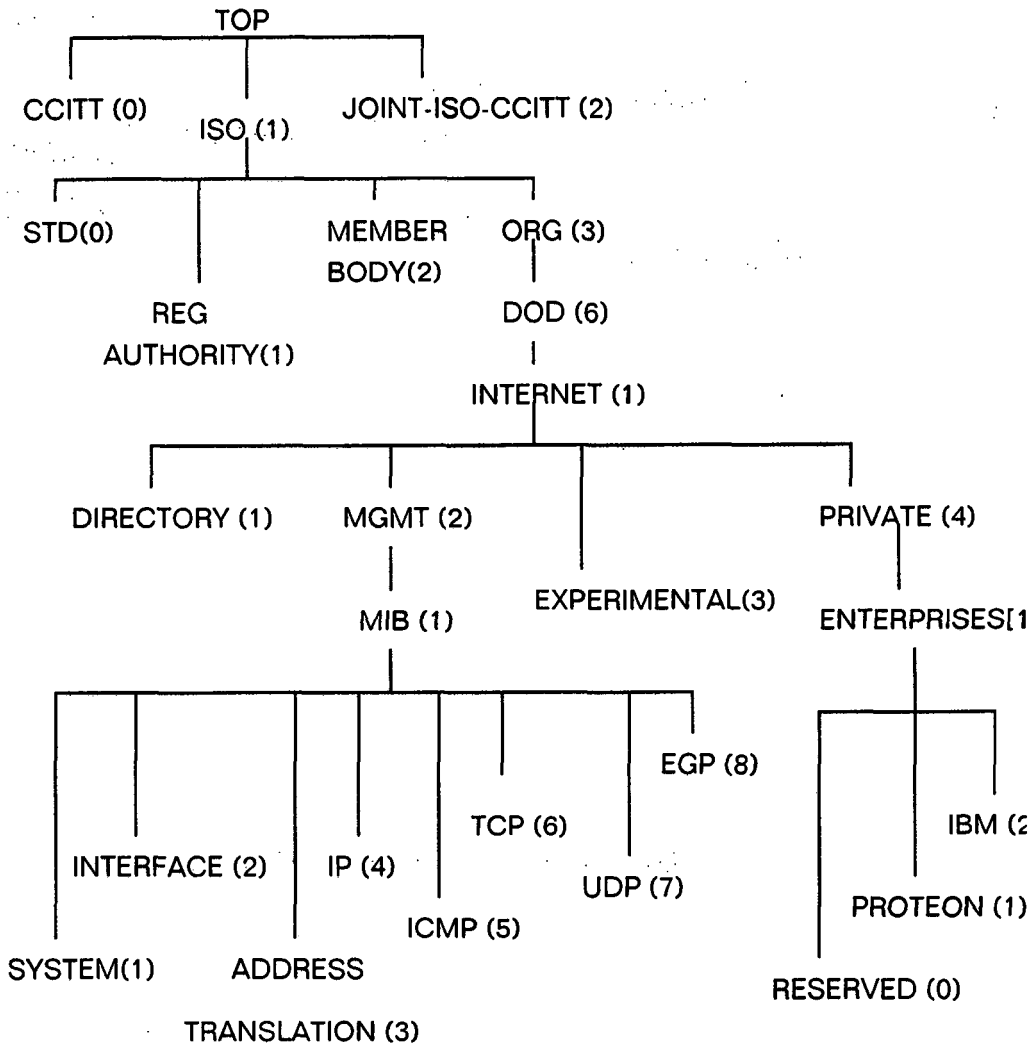



**Figura 4.1: Árvore de registro de tipos de objetos [Specialski/97].**

**Abaixo do nó iso(1), a ISO designou uma sub-árvore para ser usada por outras organizações (inter)nacionais, denominado org(3). Destes nós filhos, dois foram designados para o *National Institutes of Standards and Technology* dos EUA. Uma destas sub-árvores foi transferida pelo NIST para o departamento de defesa dos EUA, denominado dod(6).**

**O DoD não mostrou como ele gerencia sua própria sub-árvore de Identificadores de Objetos. Desta forma, assume-se que o DoD aloca um nó para a comunidade Internet, para ser administrada pela *Internet Activities Board (IAB)* como se segue:**

**internet OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) IAB(1) }**

**Com isso, a sub-árvore da Internet, de Identificadores de Objetos começa com o prefixo: 1.3.6.1.**

## 4.3 Variáveis utilizadas no trabalho

**Para se atingir o objetivo de monitoração do atalho criado pelo *MPOA Client,* deciciu-se pela monitoração das variáveis mpcIngressCacheTotalPackets, cujo identificador é 1.3.6.1.4.1.353.5.8.1.1.2.10.0 e sysDescr pertencente ao grupo System da MIB II cujo identificador 1.3.6.1.2.1.1.1. Esta última possui apenas caráter informativo do dispositivo a ser monitorado, não tendo relação com a criação de atalhos.**

# 5. MONITORANDO A CRIAÇÃO DE ATALHOS ATRAVÉS DE AGENTE MÓVEL

A motivação para a proposta de uma ferramenta para monitoramento MPOA, reside no fato de esta ser uma tecnologia nova, complexa, com padrões de gerenciamento ainda em desenvolvimento, mas que vem sendo implantada gradativamente em muitas LANs.

As propostas para gerenciamento MPOA, normalmente apresentam-se diluídas em soluções de gerenciamento de redes ATM, não sendo voltadas exclusivamente a este propósito. Ainda assim, algumas destas soluções se encontram em fase inicial, e muitas ainda em estágio de desenvolvimento. O mais comum é encontrar soluções proprietárias baseadas em MIBs proprietárias, específicas para um único hardware, o do próprio fabricante.

Conforme já mencionado, os padrões existentes para gerenciamento de redes ATM se baseiam, ainda fortemente no protocolo SNMP, que se tornou um padrão de fato e de direito para as redes atuais. A ferramenta aqui apresentada também se utiliza o SNMP como protocolo de gerenciamento.

A utilização de agentes móveis, implementados utilizando o modelo de *Aglets*, para a concepção desta ferramenta visa analisar, na prática, a viabilidade de combinação entre a tecnologia de agentes móveis e métodos tradicionais de gerenciamento, através do SNMP.

A ferramenta que está sendo proposta não pretende ser uma solução completa para o gerenciamento de ambientes MPOA, e sim, servir como

aplicação na prática dos conhecimentos adquiridos ao longo deste estudo.

O principal objetivo desta ferramenta é contribuir na tarefa de gerenciamento de redes com backbone ATM utilizando MPOA, tanto de forma prática como também servindo como uma plataforma de aprendizado para este novo paradigma tecnológico. A ferramenta deverá ainda contribuir de forma prática para facilitar o acesso e visualização dos objetos que residem nas MIBs ATM, em especial a MIB MPOA, além de oferecer uma redução no custo de aquisição destes. O aspecto de aprendizado, citado no parágrafo anterior, diz respeito ao fato de que esta ferramenta poderá servir também como uma base para futuras pesquisas e experimentos sobre gerenciamento de redes ATM/MPOA utilizando a tecnologia de agentes móveis.

Este capítulo apresenta o processo de desenvolvimento desta ferramenta, discutindo os principais aspectos de sua implementação.

## 5.1 Considerações sobre a implementação

A implementação de agentes móveis segue o modelo de componentes de *software*, com a capacidade adicional de mobilidade. Assim, na implementação desta ferramenta, foram levados em conta aspectos de orientação a objetos, tornando o código mais claro e reutilizável.

Como as plataformas de gerenciamento atuais não suportam a utilização nativa de agentes móveis, tornou-se necessária a criação de estruturas auxiliares que proporcionaram a integração dos agentes móveis com estas plataformas. De um modo geral, estas novas estruturas se encontram na forma de bibliotecas de classes, sendo normalmente implementadas usando a linguagem de programação Java, principalmente por sua portabilidade.

Dentre estas, foi escolhida para este trabalho a *Aglets* da IBM [AGLETS/99]. A infra-estrutura *Aglets* fornece um conjunto de classes em Java que podem ser herdadas para a construção de agentes móveis, neste caso referenciado como um *aglet.*

Para a codificação, foi escolhida a linguagem de programação Java. Isto deve-se basicamente a dois motivos: permite aplicar a maior parte dos conceitos de orientação a objetos; é compatível com a infra-estrutura de agente móvel escolhida para este trabalho – Aglets da IBM.

Na construção e manipulação do agente móvel, deve-se considerar:

1. um agente móvel é um módulo de *software* que apresenta pelo menos uma classe de objetos;

2. as interfaces do agente móvel devem ser exportadas e conhecidas. Isso permitirá a interação com a base de gerenciamento e a comunicação com outros agentes móveis que trabalharão colaborativamente.

3. um agente móvel é capaz de criar cópias de si mesmo quando julgar necessário, para dinamizar uma tarefa. Estas cópias podem ainda ser instanciadas em máquinas remotas à máquina do agente criador.

## 5.2 Arquitetura da solução

A arquitetura definida para a ferramenta é baseada no modelo Mestre/Escravo com a utilização de um *proxy*. Foi desenvolvido um *aglet* mestre, que inclui a interface gráfica de usuário (GUI) e o *AgletProxy* que

cria um *aglet* escravo e o envia para uma máquina remota de onde o mesmo realiza requisições SNMP a um elemento alvo pré-determinado a fim de verificar a criação do atalho MPOA. As informações coletadas pelo *aglet* escravo são então enviadas ao *proxy* que, por sua vez, os remete à aplicação mestre que os apresenta através da GUI, como mostrado na figura 5.1.

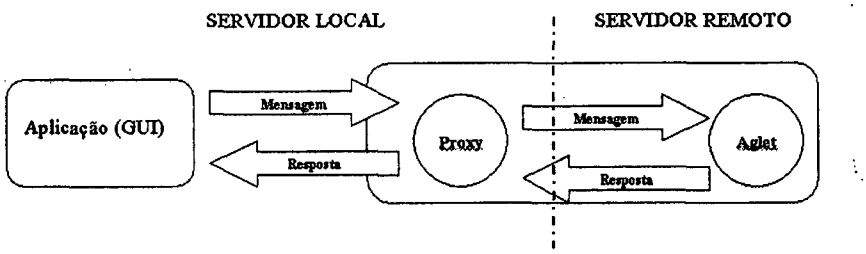



Figura 5.1: Troca de mensagens entre mestre e escravo.

A ferramenta foi implementada a partir de 3 classes Java:

- *mpoaMaster*: esta classe implementa o *aglet* mestre. É a classe principal e responsável por instanciar a GUI e o *AgletProxy*; A partir desta classe, é possível controlar o *proxy*, enviando o agente móvel para o elemento a ser gerenciado, por exemplo.

- *mpoaSlave*: esta classe implementa o agente móvel que deve ser enviado através da rede. Contém ainda as funções de gerenciamento, ou seja, nesta classe estão os métodos de chamada a API SNMP;

- *Window*: esta classe implementa a interface gráfica, através da qual pode-se interagir com a aplicação.

## 5.3 Funcionamento

Ao ser ativado, o *aglet* mestre instancia a interface gráfica de usuário

(figura 5.2), e cria o *AgletProxy*, que será utilizado para controlar o *aglet* escravo a ser criado.

Para a criação do *aglet* escravo é necessário informar os seguintes dados, como mostrado na figura 5.2:

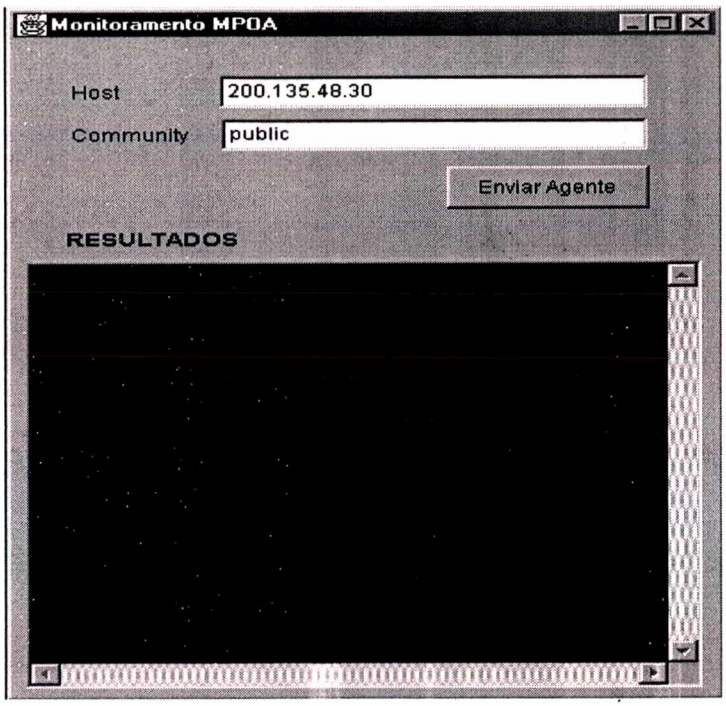



**Figura 5.2: Interface gráfica da aplicação.**

- *Serv.Aglet*: identifica o servidor para o qual o *aglet* escravo deve ser enviado. Neste servidor deverá também estar executando o servidor *Tahiti*, para que o *aglet* possa ser recebido;
- Porta: identifica a porta sobre a qual o servidor *Tahiti* remoto está rodando;
- *Community*: identifica o valor do parâmetro *community* (a senha) a ser utilizado nas requisições SNMP que serão realizadas pelo *aglet* escravo;
- Alvo SNMP: identifica o endereço IP do elemento a ser

gerenciado pelo *aglet* escravo.

Após terem sido informados todos os dados, ao pressionar o botão "Enviar Agente", o *aglet* mestre cria o *aglet* escravo e o envia para o "*Serv Aglet*" informado.

Uma vez que o *aglet* escravo chega ao seu destino, tenta-se estabelecer conexão com o agente instalado no elemento a ser gerenciado. Após a conexão ter sido estabelecida, são enviados diversas mensagens SNMP requisitando valores de determinados dados da MIB. Após ter recebido todos os dados do agente SNMP, o *aglet* escravo envia-os ao *AgletProxy*, que, por sua vez, os repassa para o *aglet* mestre que os lista através da interface gráfica.

Após ter executado sua função, o *aglet* escravo retorna ao seu contexto original e em seguida é destruído.

## 5.4 Objetos coletados das MIBs

Os objetos a serem coletados das MIBs são definidos a partir do conjunto de objetos disponíveis nas MIBs padronizadas RFC1213-MIB e MPOA-MIB, residentes nos equipamentos de rede ATM com suporte ao MPOA e que possuam um agente SNMP residente.

A coleta das informações dos objetos das MIBs deve ser efetuada pelo agente móvel através da utilização de uma API SNMP.

Existem várias APIs, inclusive de domínio público, que permitem a consulta à MIBs de agentes SNMP, principalmente escritas em Java. Estas ferramentas contêm um conjunto de métodos, no caso do Java, que oferecem várias funções para acessar os objetos nas MIBs. Para a implementação da ferramenta proposta foi utilizada uma API SNMP de código aberto e

implementada em Java, desenvolvida por Jonathan Sevy [JON/99]. Esta escolha se deu principalmente por sua natureza aberta, sendo possível alterar o código da API em função das necessidades da ferramenta.

## 5.5 Objetos a serem analisados

Para o funcionamento da ferramenta proposta, é necessário que seja definido um conjunto inicial de objetos a serem analisados. Este conjunto forma um perfil, que define um conjunto mínimo de objetos que devem ser analisados em uma rede ATM/MPOA com base nas MIBs disponíveis. Os objetos que compõem este perfil foram definidos com base em MIBs especificadas por organizações responsáveis pelas padronizações para gerenciamento de redes ATM. As MIBs que foram analisadas são: MIB II definida na RFC1213 e a MIB MPOA definida pelo ATM Forum. A escolha dos objetos para a ferramenta se ateve às MIBs padronizadas, a fim de evitar que a ferramenta fique vinculada às características de alguma MIB de fabricante, tornando-se pouco flexível.

Atualmente existem inúmeras MIBs proprietárias para gerenciar equipamentos ATM, porém estas MIBs muitas vezes não tem documentação suficiente e se destinam a atender somente as necessidades do próprio fabricante do equipamento. Com isso, estas foram desconsideradas neste trabalho.

Os objetos escolhidos permitem monitorar aspectos relacionados à criação de atalhos entre LANEs. São estes os objetos mpcIngressCacheTxTotalPackets (1.3.6.1.4.1.353.5.8.1.1.2.10.0) e sysDescr

(1.3.6.1.2.1.1.1.0).

Cabe ressaltar que o conjunto reduzido de objetos deve-se à natureza primordialmente educativa da ferramenta. Além disso, parte do interesse deste trabalho é voltado para a aplicabilidade do paradigma de agentes móveis no gerenciamento de redes e seu desempenho.

## 5.6 Considerações sobre a implementação

Como visto na seção 5.2, a ferramenta foi implementada através de 3 classes Java: mpoaMaster, mpoaSlave e Window.

A classe mpoaMaster é, dentro da arquitetura de execução definida, a classe principal. É nesta classe que as demais classes são instanciadas.

Esta classe é implementada como um aglet, apartir de herança. Ao ser instanciada, por ser uma aglet, é executado o método *OnCreation*, onde se dá também a instanciação da classe Window que representa a interface gráfica com o usuário (GUI), como mostrado no trecho de codigo abaixo.

```
public void onCreation(Object init) {
  System.out.println("Iniciando master...");
  window = new Window(this);
  window.setSize(window.getPreferredSize().width+10,
                 window.getPreferredSize().height+50);
  window.setVisible(true);

  try {
     name = (String)
getAgletContext().getProperty("aglets.user.name",

"Unknown");
     } catch (Exception ex) {
     }
    }
```

**Figura 5.3: Método *OnCreation* implementado na classe mpoaMaster.**

Como mostrado na seção 5.3, apartir da GUI o usuário informa o endereço do host destino e o valor para a *community* SNMP. Após ter-se informado estes dados, o usuário pode "Enviar o Agente" pressionando o botão correspondente. Neste momento a classe Window dispara o método *dispatchSlave* implementado na classe mpoaMaster, através do código `master.dispatchSlave("atp://"+host.getText())`. Pode-se observar que há uma concatenação do substring "atp://" com o endereço do host destino informado pelo usuário. Isto é necessário pois o aglet será enviado utilizando o protocolo ATP (*Agent Transfer Protocol*).

O método *dispatchSlave* recebe como parâmetro o endereço destino, cria um novo aglet apartir da classe mpoaSlave e o envia ao destino indicado, como mostrado na figura 5.4 .

```
public void dispatchSlave(String dest) {
  try {
    // Criando o contexto para a criação do novo aglet
    AgletContext context = getAgletContext();

    // Criando um novo aglet a partir de um elemento proxy
//que servirá
    // para controlar e manter a comunicação com o aglet
criado.
    AgletProxy proxy = context.createAglet(null,
                        "mpoaManager.mpoaSlave",
getProxy());

    // Criando uma URL para o destino indicado
    URL url = new URL(dest);

    // Enviando o aglet para a URL criada
    proxy.dispatch(url);


  } catch (InvalidAgletException ex) {
    ex.printStackTrace();
  } catch (Exception ex) {
    ex.printStackTrace();
  }
}
```

Figura 5.4: Método *dispatchSlave* implementado na classe mpoaMaster.

O método leVariaveisDaMIB presente na classe mpoaSlave, é o responsável pela leitura da variável da mib. Segue O trecho principal deste método:

```
itemID = "1.3.6.1.4.1.353.5.8.1.1.2.10.0";
                    newVars = comInterface.getMIBEntry(itemID);
                    pair =
(SNMPSequence)(newVars.getSNMPObjectAt(0));
                    snmpValue = pair.getSNMPObjectAt(1);
                    String mpcIngressCacheTxTotalPackets =
snmpValue.toString();
                    System.out.println("mpcIngressCacheTxTotalPackets =
"+mpcIngressCacheTxTotalPackets);
sendResults("mpcIngressCacheTxTotalPackets =
"+mpcIngressCacheTxTotalPackets);
```

## 5.7 Estrutura da rede ATM

A estrutura principal do ambiente de teste utiliza equipamentos ATM que conectam as redes: POP-SC ( Ponto de Presença da Rede Nacional de Pesquisa em Santa Catarina – RNP), POP-UFSC (Ponto de Presenca da Rede Catarinense na UFSC), redeUFSC, Cluster da UFSC e da RMAV-FLN (Rede Metropolitana de Alta Velocidade de Florianópolis). As conexões são de interface com 155Mbs, conforme é mostrado na figura 5.1.

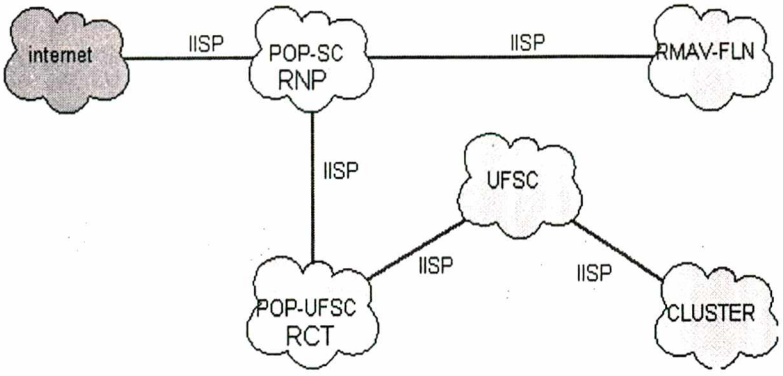



**Figura 5.6: Backbone central ATM.**

**A rede do POP-SC tem a funcao de prover servico de acesso ao backbone da RNP, enquanto que o POP-UFSC prove acesso ao backbone da Rede de Ciencia e Tecnologia de Santa Catarina (RCT-SC). A rede UFSC possui um backbone ATM próprio, permitindo acesso aos diversos departamentos técnicos e administrativos da UFSC. Paralelo ao backbone da rede UFSC, está a rede Cluster que tem por objetivo avaliar novas tecnologias associadas às aplicacões de multimidia e processamento paralelo. Com este objetivo a RMAV-FLN, conecta, além da UFSC outras três instituicões: Universidade do Estado de Santa Catarina (UDESC), Centro Integrado de Metereologia e Recursos Hidricos de Santa Catarina / Empresa de Pesquisa Agropecuária e Extensao Rural de Santa Catarina (CLIMERH/EPAGRI) e a Empresa de Telecomunicacoes de Santa Catarina (TELESC).**

## 5.8 Ambiente de testes

**Na figura 5.7 é descrito o ambiente utilizado neste trabalho; são considerados um roteador IBM 8210 MPS, um comutador ATM IBM 8371, duas LANs emuladas, denominadas LANE 48 e 49. Dois microcomputadores são utilizados para testes, sendo cada um deles conectado a uma LANE distinta. O roteador 8210 estabelece as LANEs,**

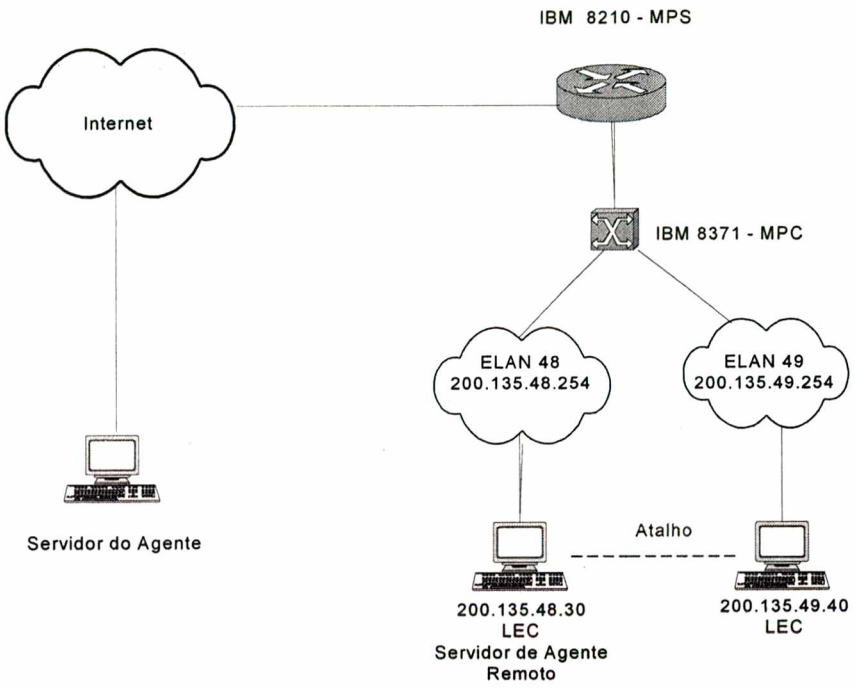



**Figura 5.7 – O Ambiente para o Desenvolvimento e Testes.**

**enquanto o comutador 8371 é responsável pela criação de atalho entre os dois microcomputadores. Neste contexto, evita-se que o tráfego gerado entre os dois microcomputadores necessite passar pelo roteador. No comutador 8371 está definida uma parte da MIB MPOA denominada MIB MPOA Client. Nesta MIB encontra-se a variável denominada *mpcIngressCacheTxTotalPackets*, que contém o número total de pacotes enviados considerada para monitoramento da criação do atalho. Este ambiente de desenvolvimento foi estabelecido no NURCAD/UFSC.**

**Uma vez criado o atalho, necessita-se da geração de tráfego de informação entre os dois microcomputadores. Este tráfego foi gerado através da ferramenta Qcheck [Qcheck/00], que possibilita, inclusive, a geração do tráfego, remotamente. Após a geração do tráfego, pode-se executar o aplicativo desenvolvido neste trabalho, ou seja, o agente móvel, o qual pode ser disparado a partir de uma máquina remota em direção ao ambiente como descrito na figura 5.7.**

Uma vez que esse agente móvel seja disparado para o monitoramento da criação do atalho, ele viaja através da rede e se estabelece na máquina que executa o Servidor de Agente Remoto (200.135.48.30). A execução do agente móvel corresponde à comunicação com o agente estacionário contido no comutador IBM 8371, o qual corresponde a uma consulta (pooling) cujo resultado é exibido na interface de monitoramento MPOA, desenvolvida conforme a figura 5.8.

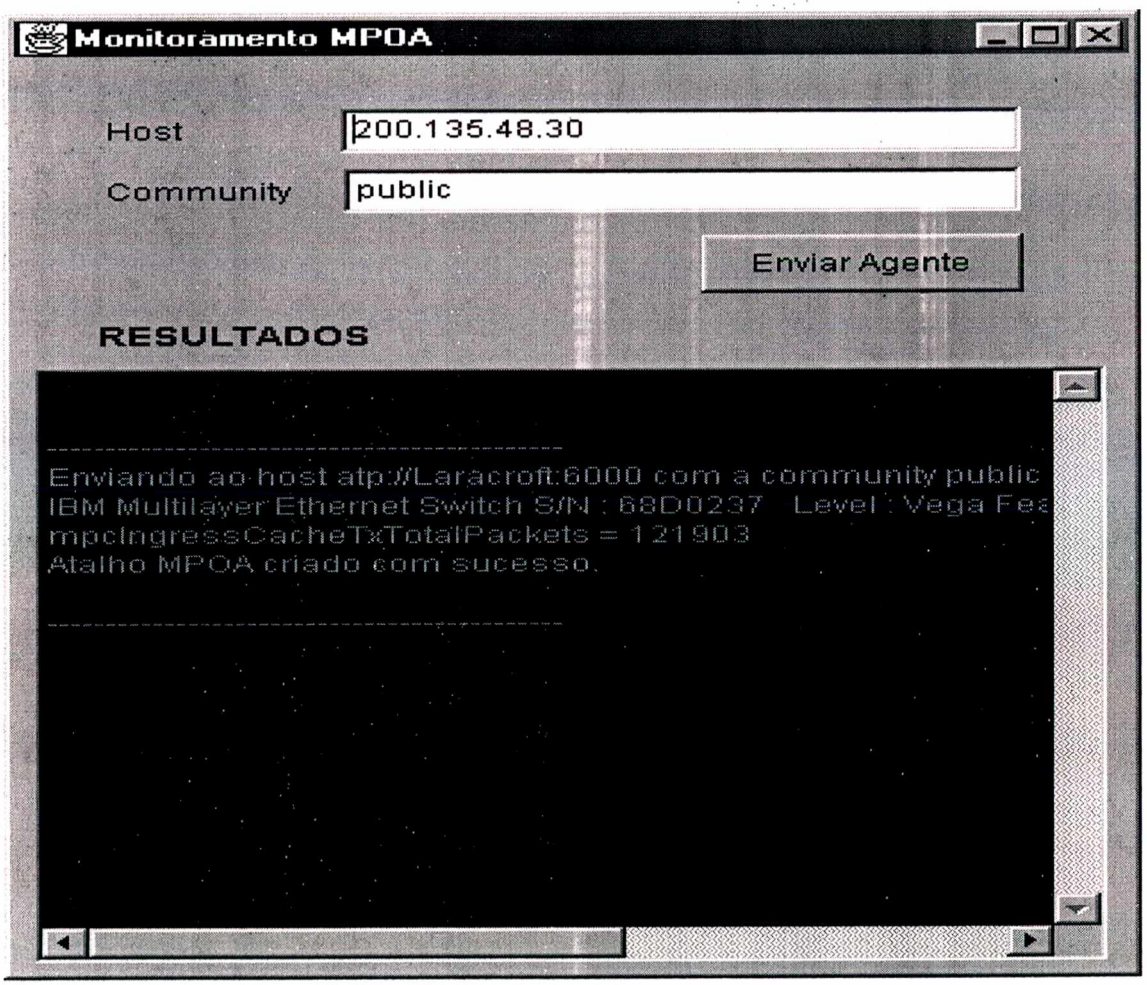



Figura 5.8 – Interface de Monitoramento.

Esta interface de monitoramento MPOA é usada para disparar o agente móvel. Pode-se notar que é também apresentado o resultado do monitoramento, representado pelo valor da variável

*mpcIngressCacheTxTotalPackets* que indica a criação bem sucedida do atalho entre esses dois microcomputadores. Esta variável é definida no Grupo do Total de Pacotes do Ingress, constituinte da MIB MPOA Client. No anexo B é mostrada a MIB MPOA Client de uma forma mais completa.

## 5.9 Extensões à Aplicação

De forma a delimitar o problema a ser pesquisado, o código da aplicação deste trabalho foi desenvolvido com intuito de monitorar a criação de atalhos entre dois computadores de LANEs distintas. Entretanto, algumas outras funcionalidades podem ser desenvolvidas tendo como base o código desenvolvido. A seguir, ilustra-se alguns exemplos que demonstram como se implementar outras funcionalidades utilizando o código já desenvolvido:

1– Qual o tempo de criação do atalho?

2– Quantos quadros passam pelo atalho em um segundo?

Tanto a informação do tempo de criação de atalho quanto a quantidade de quadros podem ser obtidas pelas variáveis mpcSetupFrameTime e mpcSCSetupFrameCount respectivamente. Ambas variáveis eestão presentes na MIB MPOA Client. Segue o código para responder estas perguntas, lembrando que O método leVariaveisDaMIB está presente na classe MpoaSlave.

```
public void leVariaveisDaMIB() {

    // *** SNMP
      // Definições necessárias para a manipulação da API SNMP

      // Interface para comunicação. Utiliza um socket.
      SNMPv1CommunicationInterface comInterface = null;
      String community;           // SNMP community do agente SNMP
      InetAddress hostAddress; // IP do elemento a ser gerenciado
      int version;              // versão do protocolo SNMP

      String itemID;                  // OID
      SNMPVarBindList newVars; // Variable Bindings
      SNMPSequence pair;          // SNMP Sequence
      SNMPInteger snmpSetValue, snmpIntegerValue; // Tipos de retorno
      SNMPObject snmpValue;    // Valor para cada OID

      // *** SNMP

      boolean conexaoOK = true; // flag

      try
      {
            community = "public";
            version = 0;   // SNMPv1
            hostAddress = InetAddress.getByName("200.135.4.6");
```

```
comInterface = new SNMPv1CommunicationInterface(version, hostAddress,
community);
            comInterface.setSocketTimeout(5000);
            System.out.println("Conexao estabelecida com sucesso. ");

      }
      catch(Exception e)
      {
            System.out.println(e+"Exception durante o estabelecimento da
conexão: ");
            conexaoOK = false;
      }

      if (conexaoOK) {
            try {
                  // obtendo o valor do objeto sysDescr da MIB SNMP
                  // Obs: esta lógica para a leitura de objetos na MIB
                  //     se repete para qualquer outro OID.

// Identificando o OID desejado
                  itemID = "1.3.6.I.2.1.1.1.0";
                  // Ativando o método correspondente ao SNMP GET para
este OID
                  // e recebendo a SNMP Response PDU
                  newVars = comInterface.getMIBEntry(itemID);
                  // Lendo a primeira sequencia do campo Variable
Bindings
                  pair = (SNMPSequence)(newVars.getSNMPObjectAt(0));

// Obtendo o valor apartir da sequencia
                  snmpValue = pair.getSNMPObjectAt(1);
                  // Transformando este valor para String
                  String sysDescr = snmpValue.toString();

                  System.out.println("sysDescr = "+sysDescr);

                  sendResults(sysDescr);

itemID = "1.3.6.1.4.1.353.5.8.1.1.2.10.0";
                  newVars = comInterface.getMIBEntry(itemID);
                  pair = (SNMPSequence)(newVars.getSNMPObjectAt(0));
                  snmpValue = pair.getSNMPObjectAt(1);
                  String mpcIngressCacheTxTotalPackets =
snmpValue.toString();
```

```
/*
                    System.out.println("mpcIngressCacheTxTotalPackets =
"+mpcIngressCacheTxTotalPackets);

                    sendResults("mpcIngressCacheTxTotalPackets =
"+mpcIngressCacheTxTotalPackets);

                    itemID = "1.3.6.1.4.1.353.5.8.1.1.2.2.1.5";
                    newVars = comInterface.getMIBEntry(itemID);
                    pair = (SNMPSequence)(newVars.getSNMPObjectAt(0));
                    snmpValue = pair.getSNMPObjectAt(1);
                    String mpcSCSetupFrameCount = snmpValue.toString();
                    System.out.println("mpcSCSetupFrameCount =
"+mpcSCSetupFrameCount);

                    sendResults("mpcSCSetupFrameCount =
"+mpcSCSetupFrameCount);

                    itemID = "1.3.6.1.4.1.353.5.8.1.1.2.2.1.6";
                    newVars = comInterface.getMIBEntry(itemID);
                    pair = (SNMPSequence)(newVars.getSNMPObjectAt(0));
                    snmpValue = pair.getSNMPObjectAt(1);
                    String mpcSetupFrameTime = snmpValue.toString();
                    System.out.println("mpcSetupFrameTime =
"+mpcSetupFrameTime);

                    sendResults("mpcSetupFrameTime = "+
mpcSetupFrameTime);
*/
                    sendResults("Atalho MPOA criado com sucesso.");

                    comInterface.closeConnection();

            } catch(Exception e) {
                    System.out.println(e.getMessage()+"Erro na leitura da MIB");
            }
    }
}
```

**Repare que a única alteração no código a ser feita para monitorar uma variável qualquer da MIB é a inclusão do seu ID no método leVariaveisDaMIB; o que torna a implementação bastante flexível. Isso pode ser constatado pela linha em vermelho no código acima.**

3 – Qual a latência do Agente móvel?

```
public void dispatchSlave(String dest) {
  try {
    // Criando o contexto para a criação do novo aglet
    AgletContext context = getAgletContext();

    // Criando um novo aglet a partir de um elemento proxy
//que servirá
    // para controlar e manter a comunicação com o aglet
criado.
    AgletProxy proxy = context.createAglet(null,
                        "mpoaManager.mpoaSlave",
getProxy());

    // Criando uma URL para o destino indicado
    URL url = new URL(dest);

    // Enviando o aglet para a URL criada
    proxy.dispatch(url);

    // Neste momento, é guardada a data em que o aglet foi
enviado
    // para posteriormente ser utilizado no cálculo de
latência do
    // agente
    whenGone = new Date();

  } catch (InvalidAgletException ex) {
    ex.printStackTrace();
  } catch (Exception ex) {
    ex.printStackTrace();
  }
}
```

A fim de mensurar a latência do agente, pode-se observar que neste método é guardado o momento em que o agente foi enviado. Este dado é usado mais tarde, no momento em que o aglet retorna. Para identificar este momento, foi implementado uma troca de mensagem entre o aglet escravo (da classe mpoaSlave) e o aglet principal (da classe mpoaMaster). Assim, após ter realizado sua tarefa, o aglet escravo retorna para o host origem e

envia uma mensagem para o aglet mestre indicando seu retorno. O trecho de código abaixo mostra como este mecanismo foi implementado.

```
MENSAGEM ENVIADA PELO AGLET ESCRAVO:

masterProxy.sendMessage(new Message("backing"));

DO LADO DO AGLET MESTRE:

public boolean handleMessage(Message msg) {
  if (msg.sameKind("backing")) {
  // A mensagem "backing" indica que o aglet enviado
(mpoaSlave) já
  // retornou. Assim, procede-se com o cálculo da latência
do agente,
  // dada pelo tempo em milisegundos que durou toda a
viagem (ida,
  // processamento e volta)

  whenBack = new Date();
  long t1 = whenGone.getTime();
  long t2 = whenBack.getTime();
  long difference = t2 - t1;
  window.showResult("Tempo decorrido(milliseconds) da
viagem: " +  difference);
  } else {
  ...
```

Este cálculo de latência, apesar de ser bastante primário permite que sejam realizados testes no sentido de comparar o desempenho desta solução que combina agentes móveis (aglet) e SNMP com a solução tradicional de modelo SNMP centralizado, por exemplo.

# 6 CONSIDERAÇÕES FINAIS

Os objetivos da pesquisa foram alcançados com sucesso. Com a possibilidade de utilização da Rede Metropolitana de Alta Velocidade – RMAV-FLN, foi permitido avaliar as caracteristicas da arquitetura MPOA em uma rede de alta velocidade. A implementação de um aplicativo para monitoramento SNMP utilizando agentes móveis permitiu a configuração de que um atalho pode ser criado entre duas subredes distintas permitindo assim que a comunicação entre os hosts seja feita sem a utilização de um roteador. Dessa forma observou-se a eliminação do problema de sobrecarga nos roteadores da rede diminuindo a latência de comunicação.

A utilização de agentes móveis deu um grau a mais de complexidade no aplicativo desenvolvido. Inferiu-se que a tecnologia de mobilidade pode ser usada sem problema algum como ferramenta de gerenciamento de rede. A grande ressalva com relação à tecnologia é que os sistemas operacionais não oferecem suporte nativo a agentes móveis. Uma camada de software deve ser instalada sobre o sistema servindo como servidor de agentes. Para que a tecnologia consiga um aumento significativo de utilização, acredito que este suporte a nível de sistema operacional deve ser desenvolvido ou então a desenvolvimento de novos chips nos equipamentos de redes com suporte a agentes móveis.

## 6.1 Trabalhos Futuros

Este trabalho abriu um leque de alternativas com relação a pesquisas futuras; entre elas destacamos:

- Estudo comparativo entre gerência de redes de computador centralizada e distribuída;
- Investigar a viabilidade de utilização de agentes cooperativos no monitoramento de redes de computadores

- **Fazer experimentos sobre a questão de segurança com relação a agentes móveis.**
- **Investigar o uso de agentes móveis em redes de longa distância.**

**Com relação ao MPOA destaco:**

- **Uma avaliação analítica das tecnologias MPOA e MPLS (Multiprotocol Label Switching) no que se diz respeito a redes locais de longa distância.**

# ANEXO 1 – DESCRIÇÃO DA MIB MPOA

**MPOA MIB Versão 1.1**

**Este módulo define as Bases de Informações de Gerenciamento (Management Information Base – MIB) para gerenciamento de clientes e servidores MPOA, atualizando a versão 1.0.**

**Convenções Textuais:**

| | |
|---|---|
| LecIndex | Identifica o LEC para qual as entradas contém informações de gerenciamento |
| AtmConfigAddr | Endereço ATM da entidade |
| InternetworkAddrType | Internework Layer Address Types |
| InternetworkAddr | O valor de um endereço em nível internetwork |
| MpcIndex | Um valor único para cada MPC o qual o agente SNMP gerencia. É recomendado associar os valores contiguamente a partir de 1. Os valores devem permanecer constantes, mesmo se o MPS ou o agente SNMP forem reinicializados |

**Grupos do módulo MIB MPOA**

***Grupo Comum MPOA***

***Grupo de Tipo do Dispositivo***

**deviceTypeTable – Esta tabela representa o mapeamento do endereco Lane Data ATM para dispositivo MAC capacitado. A chave primária é o endereço Lane Data ATM e o Lec Index do LEC associado com o endereço MAC.**

| | |
|---|---|
| deviceTypeIndex | As entradas na deviceTypeMpsMacAddressTable com esse índice e cujo valor no deviceTypeType é mps ou mpsAndMps são considerados para ser endereço MAC do MPS |
| deviceTypeLecIndex | LecIndex do LEC que suporta esse endereço ATM de dados |
| deviceTypeRemoteLecAtmAddre | O endereço ATM aprendido via LE ARP |

| ss | |
|---|---|
| deviceTypeType | indica o tipo do dispositivo{ 1 – nomMpoa<br>2 – mps<br>3 – mpc<br>4 – mpsAndMpc } |
| deviceTypeMpsAtmAddres | associado ao endereço do MPS, zero quando mpc ou não MPOA |
| deviceTypeMpcAtmAddres | associado ao endereço do MPC, zero quando mps ou não MPOA |

*Grupo Device Type Mps Mpc*

deviceTypeMpsMacAtmAddressTable

| deviceTypeMpsMacAddress | endereço MAC do MPS contido no TVL que é identificado pelo deviceTypeIndex na deviceTypeTable |
|---|---|

**Grupo do Cliente MPOA**

*Grupo de Configuração*

**mpcNextIndex – O valor apropriado para novas entrada na mpcConfigTable (0 indica que novas entradas não são permitidas)**
**mpcConfigTable – contem informações de todos os MPCs que este agente gerencia**

| mpcIndex | O índice do mpc |
|---|---|
| mpcRowStatus | Este objeto permite criação e destruição de MPCs |
| mpcConfigMode | Especifica qual modo de configuração será usado quando o MPC (re-)inicializar:<br>1 – automatico ( via LECS)<br>2 – manual (via mpcConfigTable, por ex.) |
| mpcCtrlAtmAddr | O endereço ATM que será usado nas comunicações com o MPS (não pode mudar) |
| mpcSCSetupFrameCount | contador dos frames no atalho |
| mpcSCSetupFrameTime | contador do tempo no atalho |
| mpcInitialRetryTime | Tempo de timeout para uma nova tentativa nas MPOA Resolution Request (novas tentativas com um timeout de 'retry time'*numero de frames, ou até mpcRetryTimeMaximum) |
| mpcRetryTimeMaximum | Valor máximo acumulado de tempo de timeout |

| | |
|---|---|
| mpcHoldDownTime | Tempo mínimo de espera para reiniciar uma tentativa sem sucesso de resolução |

*Grupo de configuração atual*

MpcActualTable – tabela de leitura onde estão as informações sobre os valores atuais usados pelos MPCs, que podem estar diferentes dos configurados

| | |
|---|---|
| mpcActualState | Indica o estado atual do MPC : 1 – desconhecido<br>2 – inicializando<br>3 – ligado<br>4 – desligado |
| mpcDiscontinuityTime | O tempo entre o início do sistema até uma falha em um ou mais contador(es) |
| mpcActualConfigMode | Indica como o MPC será configurado na próxima inicialização :<br>1 – automático (via LECS)<br>2 – manual (via mpcConfigTable, por ex.) |
| mpcActualSCSetupFrameCount | Número de frames |
| mpcActualSCSetupFrameTime | Tempo (este parâmetro junto com o mpcActualSCSetupFrameCount configura a tàxa para o disparo do atalho) |
| mpcActualInitialRetryTime | Tempo para uma nova tentativa de estabelecer o atalho |
| mpcActualRetryTimeMaximum | Quantidade máxima de tempo (cumulativo) para novas tentativas |
| mpcActualHoldTime | Tempo mínimo para espera antes de desistir de uma tentativa de atalho |

*Grupo de endereços ATM de dados*

MpcDataAtmAddressTable – esta tabela mostra todos os endereços de dados associados aos MPCs

| | |
|---|---|
| MpcDataAtmAddress | O endereço de dados do MPC |
| MpcDataAtmAddressRowStatus | Este objeto permite a criação e a destruição de MPCs |

*Grupo de estatísticas*

mpcStatisticsTable – tabela de leitura onde estão os dados estatísticos dos MPCs. Todos os contadores a seguir podem ser descontinuados se o sistema de gerenciamento ou o MPC for reinicializado.

| | |
|---|---|
| MpcStatTxMpoaResolveRequests | O número de requisições MPOA enviadas |
| MpcStatRxMpoaResolveReplyAcks | O número de Respostas MPOA recebidas indicando sucesso no atalho |
| MpcStatRxMpoaResolveReplyInsufECResources | O número de respostas MPOA recebidas indicando erro por insuficiência de recursos para aceitar uma nova entrada na tabela de cache do egress |
| mpcStatRxMpoaResolveReplyInsufSCResources | O número de respostas MPOA recebidas indicando erro por insuficiência de recursos do egress para aceitar um novo atalho |
| mpcStatRxMpoaResolveReplyInsufEitherResources | O número de respostas MPOA recebidas indicando erro por insuficiência de recursos para aceitar uma nova entrada na tabela de cache do egress ou um novo atalho |
| mpcStatRxMpoaResolveReplyUnsupportedInetPort | O número de respostas MPOA recebidas indicando erro devido ao protocolo em nível de rede não ser suportado |
| mpcStatRxMpoaResolveReplyUnsupportedMacEncaps | O número de respostas MPOA recebidas indicando erro devido ao encapsulamento MAC proposto não ser suportado |
| MpcStatRxMpoaResolveReplyUnspecifiedOther | O número de respostas MPOA recebidas indicando erro por outros motivos |
| mpcStatRxMpoaImpRequests | O número de Requisições de entradas de cache recebidas |
| MpcStatTxMpoaImpReplyAcks | O número de respostas de entradas de cache enviadas indicando sucesso |
| mpcStatTxMpoaImpReplyInsufECResources | O número de respostas de entradas de cache recebidas por este MPS indicando erro devido à falta de recursos para uma nova entrada na tabela de cache do egress |
| mpcStatTxMpoaImpReplyInsufSCResources | O número de respostas de entradas de cache recebidas por este MPS indicando erro devido à falta de recursos para um novo atalho |
| mpcStatTxMpoaImpReplyInsufEitherResources | O número de respostas de entradas de cache recebidas por este MPS indicando erro devido à falta de recursos para uma nova entrada na tabela de cache do egress ou um novo atalho |
| mpcStatTxMpoaImpReplyUnsupportedInetProt | O número de respostas de entradas de cache recebidas por este MPS indicando erro devido ao protocolo em nível de rede não ser suportado |
| mpcStatTxMpoaImpReplyUnsupportedMacEncaps | O número de respostas de entradas de cache recebidas por este MPS indicando erro devido ao encapsulamento MAC proposto pelo ingress não ser suportado |
| mpcStatTxMpoaImpReplyUnspecifiedOther | O número de respostas de entradas de cache recebidas por este MPS não contadas em nenhum outro contador |
| mpcStatTxMpoaEgressCachePurgeRequests | O número de requisições de atualização de alguma entrada na tabela cache do egress transmitidas |

| | |
|---|---|
| mpcStatRxMpoaEgressCachePurgeReplies | O número de respostas de atualização de alguma entrada na tabela cache do egress recebidas |
| mpcStatRxMpoaKeepAlives | O número de mensagens keep alive recebidas |
| mpcStatRxMpoaTriggers | O número de disparos MPOA |
| mpcStatRxMpoaDataPlanePurges | O número de requisições de atualização dados do egress transmitidas |
| mpcStatTxMpoaDataPlanePurges | O número de respostas de atualização dados do egress recebidas |
| mpcStatRxNhrpPurgeRequests | O número de pedidos de atualização NHRP recebidos |
| mpcStatTxNhrpPurgeReplies | O número de respostas de atualização NHRP enviados |
| MpcStatRxErrUnrecognizedExtensions | O número de pacotes recebidos com erro devido à extensões desconhecidas |
| mpcStatRxErrLoopDetecteds | O número de pacotes recebidos com erro devido à detecção de loop |
| mpcStatRxErrProtoAddrUnreachables | O número de pacotes recebidos com erro devido ao endereço de protocolo inalcançável |
| mpcStatRxErrProtoErrors | O número de pacotes recebidos com erro devido à erros de protocolo |
| mpcStatRxErrSduSizeExceededs | O número de pacotes recebidos com erro devido ao tamanho da SDU (Service Data Unit) ter excedido o permitido |
| mpcStatRxErrInvalidExtensions | O número de pacotes recebidos com erro devido à extensões inválidas |
| mpcStatRxErrInvalidReplies | O número de pacotes recebidos com erro devido à erros na resposta |
| mpcStatRxErrAuthenticationFailures | O número de pacotes recebidos com erro devido à erros na autenticação |
| mpcStatRxErrHopCountExceeded s | O número de pacotes recebidos com erro devido ao número de nodos ser maior que o permitido |

***Grupo de protocolos suportados***

**MpcProtocolTable – cada entrada nesta indica um protocolo para o qual o MPC irá monitorar o tráfego. Se a configuração foi obtida a partir do LECS, e o protocolo está habilitado, então os valores recebido do LECS irão refletir nesta tabela.**

| | |
|---|---|
| mpcFlowDetectProtocol | O protocolo na qual a monitoração do tráfego será executada |
| mpcLESCValue | Este objeto indica se a entrada corrente é obrigada a restaurar suas informações a partir do LECS ou não.<br>1 – verdadeiro<br>2 – falso |
| MpcProtocolRowStatus | Este objeto permite a criação e a destruição de MPCs |

*Grupo de mapeamento LEC -> MPC*

**mpcMappingTable – a tabela de mapeamento do índice do LEC para o índice do MPC(lecIndex->mpcIndex)**

| | |
|---|---|
| **mpcMappingRowStatus** | **Este objeto é usado pelo agente ou gerente para criar, apagar ou modificar uma entrada nesta tabela** |
| **mpcMappingIndex** | **O mpcIndex do MPC que está executando a monitoração do tráfego para o LEC representado pelo lecIndex** |

*Grupo de informações sobre o MPS*

**mpcMpsTable – esta tabela contem as informações do MPS que os MPCs conhecem o endereço**

| | |
|---|---|
| **MpcMpsIndex** | **O índice do MPS** |
| **MpcMpsAtmAddr** | **O endereço de controle ATM do MPS** |

*Grupo de endereços MAC do MPS*

**mpcMpsMultipleMacAddressTable – esta tabela contem informações sobre todos os endereços MAC do MPS que os MPCs conhecem (descobertas durante o monitoração do tráfego).**

| | |
|---|---|
| **MpcFlowDetectLecIndex** | **O lecIndex que representa o LEC associado** |
| **MpcMpsMacAddressIndex** | **Este valor é usado para diferenciar endereços MAC do mesmo MPS usados pelo MPC durante a monitoração do tráfego. Este valor deve ser único dentro do escopo desta tabela** |
| **MpcMpsFlowDetectMacAddress** | **O endereço MAC do MPS usado durante a monitoração do tráfego** |

*Grupo do total de pacotes do Ingress*

**mpcIngressCacheTxTotalPackets – número total de pacotes enviados por atalhos MPOA**

*Grupo do total de octetos do Ingress*

**mpcIngressCacheTxTotalOctets – número total de octetos enviados por atalhos MPOA**

*Grupo da cache do Ingress*

**mpcIngressCacheTable – Esta tabela contem informações de cache para o MPC ingress**

| | |
|---|---|
| MpcIngressCacheDestInetworkAddrType | O tipo do endereço (de rede) do destino |
| MpcIngressCacheDestAddr | O endereço (de rede) do destino |
| MpcIngressCachePrefixLen | Define uma classe equivalente de endereços que identifiquem-se quanto a posição dos bits indicando o tamanho do prefixo do endereço de destino |
| MpcIngressCacheScrAtmAddr | O endereço ATM do fonte para a requisição MPOA |
| MpcIngressCacheDestAtmAddr | O endereço ATM de destino recebido por uma resposta MPOA |
| MpcIngressCacheEntryState | Indica o estado desta cache :<br>1 – inexistente : ainda não disponível<br>2 – inativo : existente porém ainda não ativo<br>3 – ativo : existente e ativo<br>4 – negativo : existente porém negativo (por causa de um NAK ou retry) |
| MpcIngressCacheEgressCacheTagValid | Se o valor deste objeto for verdadeiro, então o Tag de cache do Egress está presente e o valor desta Tag esta em mpcIngressCacheEgressCacheTag.<br>1 : verdadeiro<br>2 : falso |
| MpcIngressCacheEgressCacheTag | Se uma Tag de cache do Egress está presente, este objeto contem o valor desta tag. Observar o valor da mpcIngressCacheEgressTagValid |
| MpcIngressCacheLastNhrpCieCode | O último código CIE (NHRP Client Information Element). Este valor é valido durante o tempo especificado em Hold Down Time |
| MpcIngressCacheSigErrCode | O código de erro ou sucesso da última requisição |
| MpcIngressCacheRetries | O número atual de vezes que este MPC emitiu um pedido da definição desde que recebeu uma resposta válida |
| MpcIngressCacheTimeUntilNextResolutionRequest | O tempo que o MPC deve esperar antes de enviar a próxima requisição |
| MpcIngressCacheHoldingTime | O tempo que esta entrada será valida |
| MpcIngressCacheServiceCategory | As categorias de serviço suportadas para este atalho |

*Grupo do total de pacotes Egress*

**mpcEgressCacheRxTotalPackets – número total de pacotes recebidos por atalhos MPOA**

*Grupo do total de octetos do Egress*

**mpcEgressCacheRxTotalOctets – número total de octetos recebidos por atalhos MPOA**

*Grupo da cache do Egress*

**MpcEgressCacheTable – informações de cache do MPC Egress**

| | |
|---|---|
| MpcEgressCacheId | Identificador providenciado pelo MPS na Cache Imposition Request |
| MpcEgressCacheDestInetworkAddrType | O tipo do endereço (de rede) nesta entrada |
| MpcEgressCacheIDestAddr | O endereço (de rede) do destino para qual esta entrada é definida |
| MpcEgressCachePrefixLen | Define uma classe equivalente de endereços que identifiquem-se quanto a posição dos bits indicando o tamanho do prefixo do endereço de destino |
| MpcEgressCacheEntryState | Indica o estado desta cache :<br>1 – inexistente : ainda não disponível<br>2 – inativo : existente porém ainda não ativo<br>3 – ativo : existente e ativo<br>4 – negativo : existente porém negativo (por causa de um NAK ou retry) |
| MpcEgressCacheEgressCacheTagValid | Se o valor deste objeto for verdadeiro, então o Tag de cache do Egress está presente e o valor desta Tag esta em mpcIngressCacheEgressCacheTag.<br>1 : verdadeiro<br>2 : falso |
| MpcEgressCacheEgressCacheTag | Se uma Tag de cache do Egress está presente, este objeto contem o valor desta tag. Observar o valor da mpcIngressCacheEgressTagValid |
| MpcEgressCacheHoldTime | O tempo restante em que esta entrada será valida |
| MpcEgressCacheDataLinkHeader | O cabeçalho com a qual o cliente egress reconstrói o pacote original |
| MpcEgressCacheIngressMpcDataAtmAddr | O endereço ATM de dados do MPC ingress que emitiu a requisição MPOA |
| MpcEgressCacheLecIndex | Este valor é o lecIndex do LEC associado com este tráfego. Pode ser usado para obter o nome da LANE ou outros parâmetros LANE |
| MpcEgressCacheServiceCategory | as categorias de serviço suportadas. Representa um OU dos bits:<br>1 – se rt-VBR é suportado<br>2 – se nrt-VBR é suportado<br>4 – se ABR é suportado<br>8 – se CBR é suportado |

| | |
|---|---|
| | 0 – indica que UBR é suportado |

**Grupo do Servidor MPOA**

**mpsNextIndex – um valor apropriado para ser usado como indice para novas entradas na tabela de configuracao do MPS. O valor 0 indica que não há possibilidade de novas entradas**

***Grupo de Configuração***

**mpsConfigTable – representa as informacoes de configuracao para todos os MPSs que este agente gerencia**

| | |
|---|---|
| MpsIndex | Identifica uma única entrada na tabela de configuração |
| MpsRowStatus | Indica o estado da entrada |
| MpsConfigMode | Indica o modo de configuração |
| MpsCrtlAtmAddr | Indica o endereço de controle do MPS |
| MpsKeepAliveTime | Indica o tempo máximo entre os envios de mensagens keep alive-> MPS-p1 |
| MpsKeepAliveLifeTime | Indica o intervalo de tempo onde a mensagem de keep alive pode ser considerado válido-> MPS-p2 |
| MpsInitialRetryTime | O valor de tempo em segundos para uma nova tentativa -> MPS-p4 |
| MpsRetryTimeMaximum | O valor de tempo máximo (cumulativo) para novas tentativas-> M PS-p5 |
| MpsGiveupTime | O valor de tempo em segundos a serem aguardados antes de desistir de uma tentativa de resolução -> MPS-p6 |
| MpsDefaultHoldingTime | Para uso em respostas NHRP-> MPS-p7 |

***Grupo das configurações atuais***

**mpsActualTable – Esta tabela contem identificação, estado e informações operacionais atualizadas sobre os MPSs que este agente gerencia, uma vez que esses valores podem ser diferentes dos iniciais.**

| | |
|---|---|
| MpsActualState | Indica o estado atual do MPS : 1 – desconhecido<br>2 – inicializando<br>3 – ligado<br>4 – desligado |
| MpsDiscontinuityTime | O tempo decorrido entre a inicialização do sistema e a mais recente ocasião em que qualquer um ou mais contadores apresentaram alguma descontinuidade |

| | |
|---|---|
| MpsActualConfigMode | Indica como o MPS será configurado na próxima inicialização : 1 – automático (via LECS)<br>2 – manual (via mpcConfigTable, por ex.) |
| MpsActualKeepAliveTime | Indica o tempo máximo entre os envios de mensagens keep alive |
| MpsActualKeepAliveLifeTime | Indica o intervalo de tempo onde a mensagem de keep alive pode ser considerado válido |
| MpsActualInitialRetryTime | Tempo para uma nova tentativa de estabelecer o atalho |
| MpsActualRetryTimeMaximum | Quantidade máxima de tempo (cumulativo) para novas tentativas |
| MpsActualGiveupTime | Tempo mínimo para espera antes de desistir de uma tentativa de atalho |
| MpsActualDefaultHoldingTime | O tempo atual de Holding Time usado nas respostas NHRP |

*Grupo de Estatísticas*

mpsStatisticsTable – esta tabela representa as informações estatísticas para os MPSs gerenciados (por este agente). Cada linha da tabela representa as estatísticas de um MPS

| | |
|---|---|
| MpsStatRxMpoaResolveRequests | O número de Requisições MPOA recebidas por este MPS, as quais serão traduzidas para Requisições NHRP |
| MpsStatRxMpoaResolveReplyAcks | O número de Respostas MPOA transmitidas indicando sucesso no atalho |
| MpsStatRxMpoaResolveReplyInsufECResources | O número de respostas MPOA transmitidas indicando erro por insuficiência de recursos para aceitar uma nova entrada na tabela de cache do egress |
| MpsStatRxMpoaResolveReplyInsufSCResources | O número de respostas MPOA transmitidas indicando erro por insuficiência de recursos para aceitar um novo atalho |
| MpsStatRxMpoaResolveReplyInsufEitherResources | O número de respostas MPOA transmitidas indicando erro por insuficiência de recursos para aceitar uma nova entrada na tabela de cache do egress ou um novo atalho |
| MpsStatRxMpoaResolveReplyUnsupportedInetProt | O número de respostas MPOA transmitidas indicando erro devido ao protocolo em nível de rede não ser suportado |
| MpsStatRxMpoaResolveReplyUnsupportedMacEncaps | O número de respostas MPOA transmitidas indicando erro devido ao encapsulamento MAC proposto não ser suportado |
| MpsStatRxMpoaResolveReplyUnspecifiedOther | O número de respostas MPOA transmitidas indicando erro por outros motivos |

| | |
|---|---|
| MpsStatRxMpoaResolveReplyOther | O número de respostas MPOA transmitidas não contadas em nenhum contador acima, incluindo erros NHRP |
| MpsStatGiveupTimeExpireds | O número de vezes que o tempo mínimo para aguardar uma resolução expirou |
| MpsStatTxMpoaImpRequests | O número de Requisições de entradas de cache transmitidas por este MPS |
| MpsStatRxMpoaImpReplyAcks | O número de respostas de entradas de cache recebidas por este MPS indicando sucesso |
| MpsStatRxMpoaImpReplyInsufECResources | O número de respostas de entradas de cache recebidas por este MPS indicando erro devido à falta de recursos para uma nova entrada na tabela de cache do egress |
| MpsStatRxMpoaImpReplyInsufSCResources | O número de respostas de entradas de cache recebidas por este MPS indicando erro devido à falta de recursos para um novo atalho |
| MpsStatRxMpoaImpReplyInsufEitherResources | O número de respostas de entradas de cache recebidas por este MPS indicando erro devido à falta de recursos para uma nova entrada na tabela de cache do egress ou um novo atalho |
| MpsStatRxImpReplyUnsupportedInetProt | O número de respostas de entradas de cache recebidas por este MPS indicando erro devido ao protocolo em nível de rede não ser suportado |
| MpsStatRxImpReplyUnsopportedMacEncaps | O número de respostas de entradas de cache recebidas por este MPS indicando erro devido ao encapsulamento MAC proposto pelo ingress não ser suportado |
| MpsStatRxImpReplyUnspecifiedOther | O número de respostas de entradas de cache recebidas por este MPS não contadas em nenhum outro contador |
| MpsStatRxMpoaEgressCachePurgeRequest | O número de requisições de limpeza de alguma entrada na tabela cache do egress |
| MpsStatTxMpoaEgressCachePurgeReplies | O número de respostas de limpeza de alguma entrada na tabela cache do egress |
| MpsStatTxMpoaTriggers | O número de disparos MPOA transmitidos |
| MpsStatTxNhrpResolveRequests | O número de requisições MPOA traduzidas para requisições NHRP e enviadas ao NHS |
| MpsStatRxNhrpResolveReplies | O número de respostas NHRP recebidas pelo MPS ingress |
| MpsStatRxNhrpResolveRequests | O número de requisições NHRP recebidas pelo MPS egress do NHS |
| MpsStatTxNhrpResolveReplies | O número de respostas NHRP transmitidas pelo MPS egress |

*Grupo de Protocolos Suportados*

**MpsProtocolTable – cada linha indica um protocolo para o qual o MPS irá executar resoluções MPOA**

| | |
|---|---|
| MpsInternetworkLayerProtocol | Um protocolo suportado pelo servidor |
| MpsLECSValue | Se a entrada atual é configurada via LECS ou não |
| MpsProtocolRowStatus | Permite aos gerentes da rede habilitarem resoluções para o 'mpsInternetworkLayerProtocol' |

*Grupo de mapeamento LEC -> MPS*

**MpsMappingTable – uma tabela de mapeamento entre o índice dos LECs para os índices dos MPSs correspondentes**

| | |
|---|---|
| MpsMappingRowStatus | Permite criação, habilitar/desabilitar esta linha |
| MpsMappingIndex | Este valor esta associado ao LEC. Corresponde ao mpsIndex |

*Grupo de informações sobre os MPCs do MPS*

**MpsMpcTable – está tabela contem informações apenas de leitura sobre os MPCs que estes MPSs conhecem**

| | |
|---|---|
| MpsMpcIndex | Índice local para o MPC representado nesta entrada |
| MpsMpcCrtlAtmAddr | O endereço de controle ATM do MPC |

*Grupo da cache do Ingress*

**MpsIngressCacheTable – esta tabela guarda todas informações da cache do ingress dos MPSs que este agente gerencia**

| | |
|---|---|
| MpsIngressCacheDestInternewworkAddrType | O tipo do endereço do destinatário |
| MpsIngressCacheDestAddr | O endereço do MPS ingress destinatário |
| MpsIngressCachePrefixLen | O tamanho do prefixo do mpsIngressCacheDestAddr |
| MpsIngressCacheEntryState | O estado desta cache do MPS ingress. Os valores possiveis são : 1 – inexistente : ainda não disponível<br>2 – inativo : existente porém ainda não ativo<br>3 – ativo : existente e ativo<br>4 – negativo : existente porém negativo |
| MpsIngressCacheSrcInternetwo | O tipo do endereço do fonte |

| | |
|---|---|
| rkAddrType | |
| MpsIngressCacheSrcAddr | O endereço do MPS ingress fonte |
| MpsIngressCacheResolveAtmAddr | O endereço de controle do MPC ingress |
| MpsIngressCacheHoldTime | O intervalo de tempo onde esses valores são válidos |
| MpsIngressCacheMpoaRequestId | O ID da requisição contido na requisição MPOA do MPC ingress |
| MpsIngressCacheNhrpRequestId | O ID da requisição para os quais o MPS gera para identificas a requisição NHRP |
| MpsIngressCacheServiceCategory | As categorias de serviço suportadas por este atalho. |

*Grupo de (Imposições de) Cache do MPS Egress*

MpsEgressCacheTable – esta tabela contem informações relativas à tabela de cache do MPS egress

| | |
|---|---|
| MpsEgressCacheId | Identifica uma única entrada na cache |
| MpsEgressCacheDestInternetworkAddrType | O tipo do protocolo do endereço do destinatário |
| MpsEgressCacheDestAddr | O endereço do MPS destinatário |
| MpsEgressCachePrefixLen | O tamanho do prefixo do destinátario |
| MpsEgressCacheHoldTime | O intervalo de tempo em que estes valores permanecerão válidos |
| MpsEgressCacheEntryState | O estado desta cache do MPS ingress. Os valores possiveis são : 1 – inexistente : ainda não disponível<br>2 – inativo : existente porém ainda não ativo<br>3 – ativo : existente e ativo<br>4 – negativo : existente porém negativo |
| MpsEgressCacheDataLinkHeader | O cabeçalho DLL |
| MpsEgressCacheElanId | O id da elan para qual esta imposição de cache esta sendo enviada |
| MpsEgressCacheSourceClientAtmAddr | O endereço ATM do NHC ingress usado na imposição de cache original |
| MpsEgressCacheNhrpRequestID | O ID da requisição da requisição NHRP original |
| MpsEgressCacheMpoaRequesID | O novo ID da requisição gerado pela imposição de cache |
| MpsEgressCacheServiceCategory | As categorias de serviços suportadas para este atalho |
| MpsEgressCacheNextHopInternetworkAddrType | O tipo do protocolo de endereço do próximo nodo |
| MpsEgressCacheNextHopAddr | O endereço do próximo nodo |

## REFERÊNCIAS BIBLIOGRÁFICAS


[AGLETS/99] *Aglets Plataform*. Desembro, 1999. Site http://www.trl.ibm.co.jp/aglets/

[CISCO850] Guide to *ATM Tecnology for the Catalyst 8540 MSR*. Chapter 2: ATM Signaling and Addressing.

[CORNELL/00] CORNELL, G. *Core Java*, 1999, Sun Microsystems.

[COSTA/99] COSTA, Taís Freire da Silva. *Avaliação Analítica do Uso de Agentes Móveis na Gerência de Redes*. Dissertação de mestrado do CPGCC-UFSC. Florianópolis, outubro, 1999.

[Downes/00] DOWNES, Kevin. FORD, Merille et al. *Internetworking: manual detecnologias: uma referência essencial para todos os profissionais.*

[FRANCES/96 FRANCESCHI, Analúcia Schiaffino Morales de. *Aplicação de Desempenho para Validar a Gerência Pró-ativa de Redes*, Dissetação de Mestrado em Ciências da Computação – Departamento de Informática e Estatística – Universidade Federal de Santa Catarina – UFSC, Brasil, 1996.

[Greenwood/97] GREENWOOD, D. *Complimentary Polling Modesfor Network Performance Management Employing Employing Mobile Agentes*.

[IP-ATM1932/96] Cole, R. at all. *Ip Over ATM – A Framework Document*. IETF: RCF 1932, April/1996.

[JON/99] SEVY, Jonathan. *API Java SNMP*. http://edge.mcs.drexel.edu/GICL/people?sevy/

[LANE21/95] ALTMAN, Asher, BULLARD, Carter, FINKELSTEIN, Louis et al. *The* ATM *Forum Technical Committee: LAN Emulation OverATM, Version 1.0*, af-LANE-0021.000, January. 1995. ftp://ftp.ATMforum.com/pul/approved-specs/af-lane-0021.000.mib

[LANGE/98] Lange, D., and Oshima, M. *Deploying Java Mobile Agents with Aglets*. Addison – Wesley Longman, 1998

[MASIF/97] Object Management Group. *Mobile Agent System Interoperability Facilities Specification*, orbos/97-10-05, 1997, ftp://ftp.omg.org/pub/docs/orbos/97-10-05.pdf

[MPOA114/99] Petri, B. *Multi-Protocol Over ATM version 1.1* – af-mpoa-0114.000. ATM *Forum* Technical Committe. May/1999.

[Natham/98] Natham, V.R. *Monitoring Multi-Protocol over ATM (MPOA) Transactions for Application to Interoperability Testing*. Tese submitida a Universidade de New Hampshire para o grau de 'Master of Science in Electrical Engineering Set/1998. http://www.iol.unh.edu/training/atm/thesis (11/11/00)

[NHRP2332/98] Luciani, J. Et. All. *NBMA Next Hop Resolution Protocol (NHRP)*. RFC2332 – Internet Society, 1998.

[Q.2931/95] ITU-T Recommendation Q.2931 (1995). *B-ISDN – Digital Subscriber Signalling System No.2 (DSS 2) – User-Network Interface (UNI) Layer 3 Specification For Basic Call/Connection Control* – Telecommunication Standartization Sector of ITU – Series Q: B-ISDN Application Protocol for Access Signalling.

[QCHECK/00] Ferramenta para geração de tráfego entre duas subredes distintas. www.netiq.com/qcheck/

[SCHMIDT/98] Schmidt, A. & Minoli, D. *Multiprotocol over ATM. Building State-of-the Art ATM Intranets Utilizing RSVP NHRP, LANE Flow Switching, and WWW Tecnology.* Manning Publication, 1998.

[SIQUEIRA/00] Siqueira, Walter Ferreira. *Multi-Protocolos sobre ATM, interoperabilidade e gerência um estudo de caso*. Dissertação de Mestrado em Ciências da Computação – Departamento de Informatica e Estatística – Universidade Federal de Santa Catarina – UFSC, Brasil, 2000.

[SOARES/95] SOARES, Luiz Fernando G. *Redes de Computadores: das*

*LANs, MANs e WANs às redes ATM*. Rio de Janeiro: Editora Campus, Segunda edição, 1995.

[SPECIALSKI/97] Specialski, E. Apostila de Gerência de Redes de Computadores e de Telecomunicações. Localizado em http://notes.ufsc.br/aplic/beth.nsf

[TORRES/01] Torres, G. Redes de Computadores. Capítulo 7: ATM