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

REGINALDO HUGO SZEZUPIOR DOS SANTOS

MOBILIDADE EM GERÊNCIA DE REDES ATM

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, Abril 2002.

MOBILIDADE EM GERÊNCIA DE REDES ATM

REGINALDO HUGO SZEZUPIOR DOS SANTOS

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

Fernando Álvaro Ostuni Gauthier. Dr.
(Coordenador)

Banca Examinadora

João Bosco Mangueira Sobral. Dr.
(Orientador)

Aloysio de Castro Pinto Pedrosa, Dr.

Mirella Sechi Moretti Annoni Notare, Dra.

Paulo Sérgio da Silva Borges, Dr.

"A chave do sucesso pessoal e
profissional é você ser sempre você
mesmo e não o que os outros esperam"
(Al Nirenstein)

AGRADECIMENTOS

Agradeço a Deus por mais uma oportunidade na vida, de novos conhecimentos e novas descobertas.

A Universidade Federal de Santa Catarina, por trazer o primeiro curso de Mestrado em Ciência da Computação para o Estado de Mato Grosso, que era aguardado ansiosamente por todos nós. E a todos os professores que aqui estiveram e deram sua parcela de contribuição, em especial ao Coordenador Dr. João Bosco Mangueira Sobral que teve muito esforço e dedicação para com o curso.

A Diretoria do Centro de Processamento de Dados do Estado de Mato Grosso - CEPROMAT.

A todos que participaram direta e indiretamente deste trabalho.

RESUMO

A tecnologia ATM é a infra-estrutura de comunicação necessária para o transporte de informação, para o roteamento de tráfego de multimídia através da rede. O atributo principal da tecnologia de ATM é a sua capacidade de administração de redes, permitindo o gerenciamento da rede para o transporte de todos os tipos de tráfego, dados, voz e imagem.

As especificações relacionadas à gerência possibilitam a integração plena das redes baseadas na tecnologia ATM e as ferramentas de gerência facilitam o desenvolvimento da estrutura de gerenciamento.

Os agentes móveis apresentam-se como uma solução no gerenciamento de redes rápidas e cada vez mais heterogêneas, e os modelos utilizados hoje não são capazes de gerenciar tais características de forma satisfatória. Sua mobilidade e capacidade de colaboração permitem um gerenciamento distribuído, com tráfego de gerenciamento reduzido e capaz de respostas rápidas às alterações ocorridas nas redes.

Serão apresentadas as tecnologias utilizadas no desenvolvimento de um agente móvel para gerência do *backbone* ATM, a INFOVIA-MT, sendo as principais funções do *Aglet* as de coletar e tratar informações gerenciáveis. O desenvolvimento e os testes foram realizados no Centro de Processamento de Dados do Estado de Mato Grosso – CEPROMAT, onde estão disponíveis os equipamentos com tecnologia ATM que fazem parte da INFOVIA-MT.

ABSTRACT

ATM Technology is the infrastructure of necessary communication for the information carrier, for the traffic routing of multimedia through the network. The main attribute of the ATM technology is its capacity of managing networks, allowing the management of the network for the carrier all the types of traffic, data, voice and image.

The specifications related to the management make possible the full integration of the networks based on ATM technology and the management tools facilitate the development of management structure.

The mobile agents present themselves as a solution in the management of more heterogeneous fast networks and each time, and the models used today are not capable to manage such features in a satisfactory form. Its mobility and capacity of contribution allow a distributed management, with traffic of reduced and capable management of fast responses to the occured alterations in the networks.

We will present the technologies used in the development of a mobile agent for management of backbone ATM, the INFOVIA-MT, being the main functions of the Aglet to collect and to treat information you managed. The development and the tests had been carried through in the Data processing center of the State of Mato Grosso - CEPROMAT, where the equipment with technology ATM are available that they are part of the INFOVIA-MT.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

1. INTRODUÇÃO

A Mobilidade em Gerência de Redes ATM requer estudos em grandes áreas da tecnologia da informação, arquitetura de redes, gerência de redes e sistemas distribuídos, que são intensamente integradas, visando maximizar o desempenho e ao mesmo tempo minimizar os custos.

Sobre a tecnologia ATM além dos conceitos de arquitetura, enfatizando na prática, a maior rede ATM do Estado de Mato Grosso que visa integrar através de uma rede ATM de alta performance e confiabilidade, todos os órgãos do Governo do Estado de Mato Grosso, constituindo assim a INFOVIA-MT, a qual permite o tráfego simultâneo de voz, dados e imagens. Apresentadas as configurações dos principais serviços para a disponibilização da rede ATM.

É descrito o processo de gerência de redes ATM com o gerenciamento de configuração, falhas, desempenho, segurança e contabilização. Demonstrado também a utilização de um software utilizado na gerência da INFOVIA-MT. O monitoramento de redes ATM descrito com a apresentação também do produto que monitora as *MIB's* dos elementos ativos do *Backbone* ATM.

A tecnologia de Agentes é definida, onde citadas as características para a implementação dos agentes e apresentadas também as principais plataformas de agentes móveis, com utilização do *AGLETS* da IBM, onde são demonstrados os conceitos de Mobilidade dos agentes móveis através da interface gráfica do *TAHITI*.

Aplicabilidade dos agentes móveis em gerência de redes, enfatizando os resultados obtidos para gerência de INFOVIA-MT, com a implementação de um agente para coleta de dados da *MIB* ATM.

O objetivo principal é apresentar novos métodos, com a utilização dos agentes móveis, mais eficientes para o gerenciamento da INFOVIA-MT. Isto se traduzirá em

um melhor aproveitamento dos recursos existentes, facilitando inclusive a ampliação da rede de acordo com a aplicabilidade dos agentes móveis.

Os custos para a implementação de uma rede ATM são elevados, e como justificativa para este estudo, uma estrutura onde podemos aplicar na prática a utilização dos agentes móveis.

Além de todo o embasamento teórico sobre arquitetura de redes, gerência de rede e agentes, como justificativa, este trabalho proporcionará a você leitor a possibilidade de acompanhar na prática a configuração dos equipamentos de uma grande rede ATM, mostrará as ferramentas utilizadas para gerenciar esse *backbone*, disponibilizando trechos de um protótipo que facilitará o entendimento do método utilizado para busca de informações via SNMP nos equipamentos e logo após essa mesma busca, realizada com a utilização dos conceitos de mobilidade, movendo por servidores instanciados pelo TAHITI.

2. REDES ATM

As redes de computadores são estruturas tecnológicas indispensáveis no mundo atual. Várias são as tecnologias de rede de computadores, cada uma mais apropriada para determinado tipo de tráfego. As tecnologias tradicionais de redes não são adequadas para a transmissão de voz e imagem, uma vez que utilizam técnicas projetadas para a transmissão de dados e que não garantem a qualidade do serviço.

A tecnologia de redes ATM - *Assyncronous Transfer Mode* - foi desenvolvida especialmente para a transmissão de tráfego assíncrono, garantindo a *QoS – Quality of Service*.

2.1. Assyncronous Transfer Mode – Modo de Transferência Assíncrono

A combinação de características da comutação por circuitos e pacotes resultou na definição da tecnologia ATM, que é a resultante daquilo que melhor há nessas duas tecnologias. A diferença está no tamanho das células que trafegam sobre ATM, 53 *bytes*, que são de tamanho fixo, e por isso proporcionam um melhor desempenho em relação às células de tamanho variável, alcançando taxas da ordem de até centenas de megabits por segundo. Este fator provém de uma relação de compromisso entre a eficiência da transmissão, complexidade da rede e atraso. Podemos também ressaltar o fato da multiplexação, ou seja, vários canais lógicos que podem ser multiplexados sobre um único meio físico através de canais e caminhos virtuais, recurso este disponível na tecnologia ATM. Ela pode ser comparada a uma técnica de comutação de circuitos multitaxa, porém nela os canais virtuais possuem suas taxas dinamicamente definidas no momento da conexão, diferindo dos canais de taxa fixa. Outro aspecto importante da tecnologia que deve ser considerado é o tamanho das células.

O ATM é a tecnologia adotada como modo de transmissão na RDSI-FL, ela é capaz de integrar os mais diversos tipos de serviços utilizados na RDSI-FL, assim como é capaz de suportar várias outras aplicações para as quais não foi desenvolvido, como

interconexão de LAN's e WAN's ou interconexão de *mainframes* e supercomputadores. A transmissão ATM envolve o estabelecimento de conexões de caminhos e canais virtuais entre usuários. Como a largura de faixa desses canais é alocada dinamicamente, temos o ATM funcionando perfeitamente tanto em taxas de transmissão muito altas (155 Mbps, 622 Mbps e 2.5 Gbps) quanto em taxas relativamente baixas. O ATM pode ser considerado, uma integração das técnicas de comutação por pacotes e por circuitos. Utilizando células como unidade básica de transmissão, a transmissão por pacotes foi criada basicamente para dar suporte a serviços de taxas variáveis em tempo não-real, enquanto o ATM pode suportar serviços em tempo real e de taxa constante. Em uma transmissão ATM, a informação do usuário é mapeada nas células, que são assincronamente multiplexadas e transmitidas. Essas células são transmitidas através de canais virtuais estabelecidos na rede. Quando um canal virtual é estabelecido, é criado um identificador de canal virtual, que após a desconexão do canal é removido. A ordem das células dentro dos canais é mantida pela camada ATM e os mais diversos tipos de serviços possuem o mesmo tipo de célula, estando a diferença apenas na quantidade destas. Quanto aos serviços em tempo real, eles são garantidos devido à transmissão das células através dos canais virtuais [PRY95].

2.2. Modelo do Protocolo

Os protocolos RDSI-FL (Rede Digital de Serviços Integrados de Faixa Larga) definido pela recomendação I.321 do ITU-T, conforme a Fig. 1, é composto por três planos: plano do usuário, plano de controle e plano de gerenciamento. O plano do usuário é responsável pela transferência de informações do usuário e do controle associado a esta transferência, tais como: controle de fluxo e recuperação de erros. O plano de controle é responsável pelo controle da chamada e pelas funções de controle das conexões. Ele cuida de toda a sinalização referente ao estabelecimento, supervisão e liberação de chamadas e conexões. O plano de gerenciamento possui funções de dois tipos: gerenciamento de planos e gerenciamento de camadas [ITU93].

- Gerenciamento de planos é utilizada para coordenação dos planos do usuário, de controle e do próprio plano de gerenciamento, como um todo.
- Gerenciamento das camadas corresponde à sinalização referente aos parâmetros residentes nas suas entidades de protocolo. Trata dos fluxos de informação de operação e manutenção (*OAM - Operation and Maintenance*) específicas de cada camada.

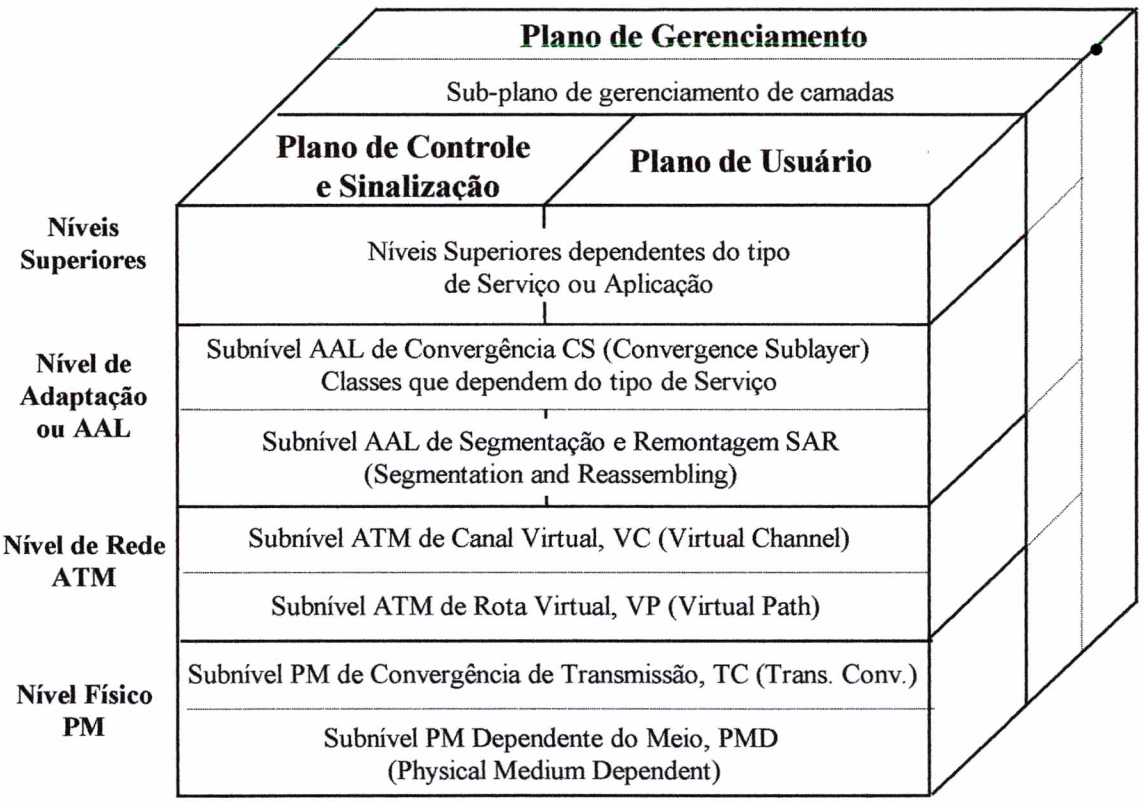
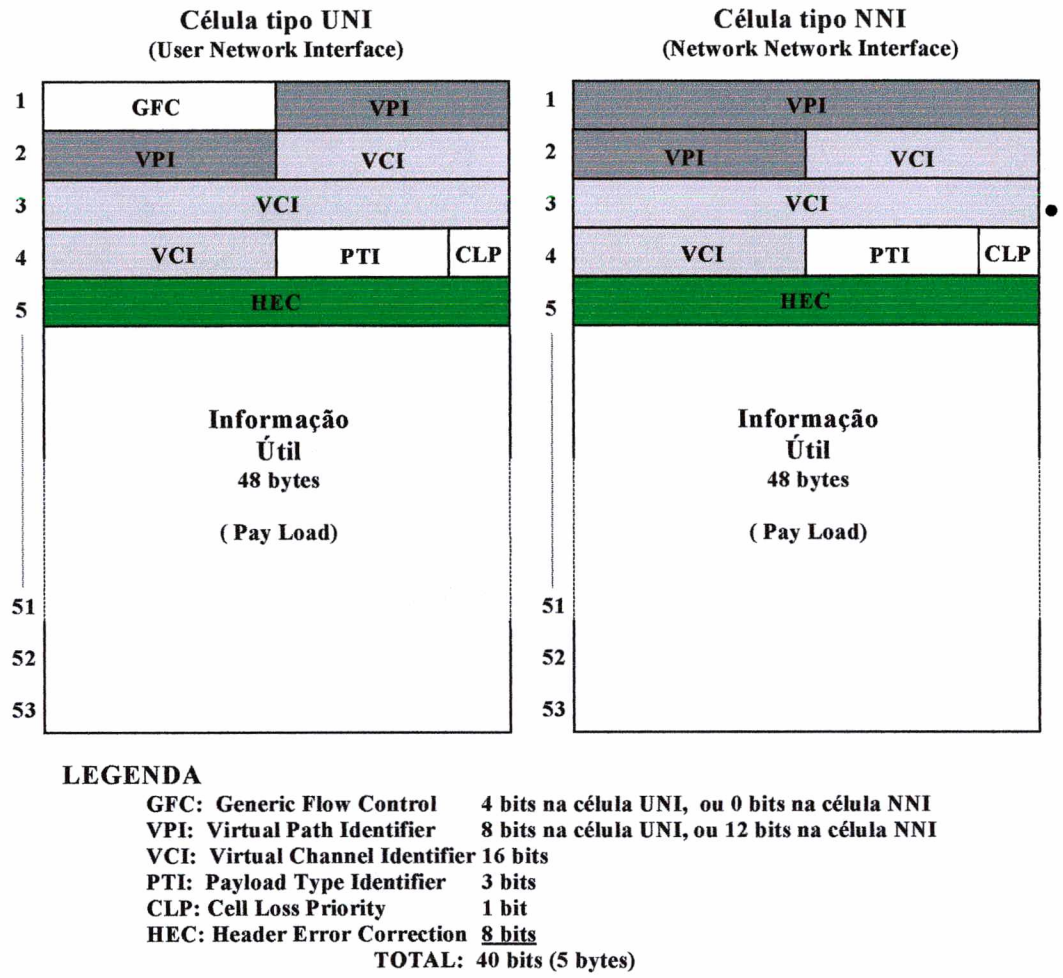


Figura 1 - Modelo de Referência de Protocolos da B-ISDN segundo a Rec. I.321

2.3. Células ATM

As células são pequenas e de tamanho fixo, o que implica em algumas vantagens, por exemplo, podem ser comutadas com muita eficiência, o que é muito importante devido às altas taxas de transmissão ATM, além disso, a utilização de células pequenas pode reduzir o tempo de espera em fila, de células prioritárias. Cada

célula possui 53 *bytes*, ou *octetos*, sendo utilizados 5 *bytes* para cabeçalho e os outros 48 *bytes* são de informações. A estrutura do cabeçalho difere na NNI (*Network-Network Interface*) e na UNI (*User-Network Interface*), conforme a Fig. 02 [TAN97].



LEGENDA

GFC:	**Generic Flow Control**	**4 bits na célula UNI, ou 0 bits na célula NNI**
VPI:	**Virtual Path Identifier**	**8 bits na célula UNI, ou 12 bits na célula NNI**
VCI:	**Virtual Channel Identifier**	**16 bits**
PTI:	**Payload Type Identifier**	**3 bits**
CLP:	**Cell Loss Priority**	**1 bit**
HEC:	**Header Error Correction**	**8 bits**
	TOTAL:	**40 bits (5 bytes)**

Figura 2 - Formato das células para a UNI e NNI

2.3.1. Generic Flow Control (GFC)

O campo de controle correspondente ao controle de fluxo genérico (GFC) está presente apenas nas células da UNI, não aparecendo nas células da NNI. Por isso, o GFC pode ser utilizado para controle de fluxo apenas ao nível da UNI local. O GFC pode ser utilizado como auxílio ao usuário no que se refere ao controle de tráfego para diferentes tipos de serviços.

2.3.2. Virtual Path (VPI) e Virtual Channel Identifiers (VCI)

Os campos identificadores de caminho virtual (VPI) e de conexão virtual (VCI) são utilizados em funções de roteamento das células através da rede. O VPI é constituído de 8 *bits* na célula utilizada na UNI e 12 *bits* na célula utilizada na NNI, permitindo assim um número maior de caminhos virtuais internamente à rede. O VCI possui 16 *bits* em ambas as células e é responsável pelo roteamento entre usuários finais.

2.3.3. Payload Type (PT)

O campo de informações (campo de dados) indica o tipo de informação presente no campo de informação, ou seja, indica se o campo de informação contém dados do usuário ou informações de gerenciamento. Qualquer nó congestionado, ao receber a célula, pode modificar seu cabeçalho de forma a indicar que a mesma passou por um nó em congestionamento.

2.3.4. Cell-loss Priority (CLP)

O campo indica a prioridade da célula. Em situações de congestionamento da rede, células que têm o *bit* CLP *setado* são consideradas de baixa prioridade, podendo ser descartadas de um dado buffer, caso chegue uma célula de alta prioridade (sem o *bit* CLP *setado*) e o *buffer* já estiver ocupado.

2.3.5. Header-error Control (HEC)

As células ATM possuem um campo de 8 *bits* em seu cabeçalho, reservados para controle de erros. O campo de controle de erro HEC é calculado à partir dos outros

32 *bits* do cabeçalho, de modo a permitir que o receptor verifique a integridade do mesmo e para a identificação do início da célula. Na maioria dos protocolos existentes que possuem campo de controle de erros, o dado que serve como base para o cálculo do código equivalente ao controle de erros é geralmente muito maior que o código resultante. Como no ATM são gerados 8 *bits* de controle de erro à partir de apenas 32 *bits*, o código de controle de erros pode ser usado não apenas para detecção de erros, como também para correção, pois há redundância suficiente no código para isso. O HEC não detecta erros nas informações contidas em uma célula, apenas no cabeçalho. As funções de detecção e correção de erro servem tanto para erros em *bits* isolados no cabeçalho quanto para rajadas de erros que também podem ocorrer (uma vez que as características de erros em uma transmissão por fibra óptica são uma mistura de erros isolados e rajadas de erros). A integridade do campo de informações é responsabilidade de camadas superiores (como por exemplo, a AAL) que pode solicitar, se for o caso, a retransmissão de alguma informação alterada. A decisão de não inclusão de controle de erro relativo ao campo de informação na célula se deve à necessidade de máxima simplificação do processamento nos nós intermediários (devido às altas taxas de transmissão). Por outro lado, é importante o controle do cabeçalho, pois além dele geralmente ser modificado a cada etapa da transmissão, pode ocorrer um erro na identificação da conexão, havendo assim a inserção de uma célula "errada" nesta conexão [DZI97].

2.3.6. Cabeçalhos Reservados

Algumas células existentes são chamadas reservadas, pois possuem valores pré-definidos para os bits do cabeçalho, com exceção do HEC, que deve ser calculado. Os valores pré-definidos do cabeçalho das células reservadas são para uso da camada física, isto é, células que não são passadas da camada física para a camada ATM. Uma célula ociosa é uma célula incluída para manter a taxa de transmissão das células ATM válidas, compatível com a capacidade do sistema. Ela não é utilizada quando não há nenhuma célula ATM.

2.4. Camada Física (Physical Layer)

No modelo de referência dos protocolos da RDSI-FL a camada física é composta por duas subcamadas: *Physical Medium* (PM) e *Transmission Convergence* (TC).

Dependente do meio físico utilizado na transmissão a subcamada PM tem como função básica adequar os bits à linha de transmissão utilizada (incluindo conversões eletro-ópticas). Ela é responsável também, pela codificação das informações recebidas da subcamada TC, de modo a possibilitar a sincronização dos transmissores/receptores. A codificação também possibilita o delineamento das células, através de símbolos de violação de código.

A subcamada TC independe do meio físico utilizado para a transmissão, e é encarregada de fornecer serviços à camada ATM [PRY95]:

- A delimitação das células: para que seja possível a extração de uma célula específica do meio de um fluxo de *bits*, é necessário que as células estejam limitadas, de forma que haja a possibilidade de identificação do seu início e do início da célula subseqüente. Isto é feito à partir de uma correlação existente entre os quatros primeiros *octetos* da célula e o seu HEC (quinto *octeto*), ou seja, busca-se uma seqüência de 32 *bits* seguida de mais 8 *bits* correspondentes a um HEC válido. A Recomendação que trata da delimitação das células é a recomendação I.432;
- A geração do HEC: a subcamada TC é a responsável pelo cálculo e inserção do *Header Error Control* no cabeçalho da célula e por sua verificação na recepção. O HEC é composto por bits redundantes utilizados para detecção e correção de possíveis erros de transmissão. Quando possível, erros detectados são corrigidos; caso a correção seja impossível, as células são simplesmente descartadas;
- A desassociação da taxa de transmissão: para que o fluxo de células fique desassociado da taxa específica do sistema de transmissão utilizado, ocorre a inserção de células ociosas na transmissão. Essas células são introduzidas e descartadas na recepção pela subcamada TC; embaralhamento: para que sejam

evitadas as longas seqüências de *0's* ou de *1's*, a subcamada TC pode embaralhar a seqüência de *bits* do campo de informações da célula. Isto também diminui a probabilidade de uma seqüência de informações ser confundida com o cabeçalho, quando na recepção está ocorrendo o rastreamento do sincronismo pelo HEC. Note que o embaralhamento ocorre apenas entre os *bits* de informação da célula, sem afetar o cabeçalho.

Em relação à padronização da camada física a maior dúvida com certeza foi à definição da estrutura de transmissão utilizada. O ITU-T padronizou duas formas de transmissão para a UNI: a estrutura de transmissão baseada no SDH, que mantém a compatibilidade com a NNI, que é baseada no SONET; e uma estrutura baseada completamente em células (sem delimitação de quadros). Além desses dois padrões, ainda há a possibilidade da utilização das estruturas antigas já existentes baseadas no PDH (previstas principalmente pela ANSI e pelo ETSI - *European Telecommunications Standards Institute*) e de uma estrutura baseada na FDDI para utilização na UNI privativa (definida pelo Fórum ATM). É importante notar que a utilização de hierarquias como a PDH ou a SDH (baseadas no TDM síncrono) não quer dizer que o modo de transferência é síncrono, pois a alocação da capacidade do sinal básico pode ser feita de forma assíncrona (na definição do sinal básico de uma hierarquia qualquer não há restrição sobre a forma como a informação ocupará a capacidade do sinal definido) [MON94].

2.4.1. Estrutura de Transmissão

Na transmissão baseada em células, há um fluxo contínuo das células ATM sem a divisão em quadros. Deste modo, alguma forma de sincronismo diferente das implementadas para transmissão baseada em quadros torna-se necessária. Esse tipo de sincronismo é baseado no HEC do cabeçalho das células, conforme já foi citado anteriormente. Ao se iniciar uma transmissão, o receptor assume um estado de HUNT ("busca" ou "caça"), onde o HEC é calculado e comparado com o HEC da célula recebida. Caso haja coincidência entre os HEC's, o estado de PRESYNC é assumido.

Neste estado, o reconhecimento ocorre a nível de células. Se houver o reconhecimento de x células consecutivas, o receptor assume que o sincronismo foi atingido, o delineamento das células foi encontrado e ele passa para o estado SYNC. Caso em uma das x células houver um erro no cabeçalho, o receptor assume que o sincronismo não foi atingido e retorna ao estado de busca (HUNT). Se o receptor que já está no estado SYNC receber y células consecutivas com erro, ele considera que houve perda de sincronismo e também retorna ao estado de busca [TAN97].

O SDH é uma outra estrutura provinda do SONET, cuja estrutura básica é formada por um quadro de 810 *bytes*. Seu sinal básico é denominado STS-1 (*Synchronous Transport Signal - level 1*). O campo de informações é composto por um cabeçalho de caminho de 9 *bytes* e o resto do quadro contém as células ATM.

2.5. Camada ATM (ATM Layer)

Responsável pela formação do cabeçalho da célula, com exceção do campo de HEC. A camada ATM no sentido de transmissão utiliza a informação recebida do nível mais alto e do plano de gerenciamento de forma a criar o cabeçalho, o acrescenta ao campo de informação do usuário provindo da AAL, e os envia à camada física. No sentido de recepção, as células recebidas da camada física são desmontadas a fim de permitir a extração e o processamento do cabeçalho, e o campo de informação do usuário é enviado à AAL.

2.5.1. Conexão ATM

A conexão ATM é uma conexão transparente provida pela camada ATM. Ela é conectada fim-a-fim através do agrupamento de elementos de conexão e pode ser de dois tipos: *Virtual Channel* (VC) e *Virtual Path* (VP). Um VC é uma conexão lógica unidirecional entre dois nós utilizada para a transmissão de células ATM. O VP nada mais é do que uma combinação lógica de vários VC's. Para cada VC é associado um *Virtual Channel Identifier* (VCI) e a cada VP, um *Virtual Path Identifier* (VPI). Em

uma Virtual Path Connection (VPC) podem existir *links* VC's diferentes entre si que são diferenciados através do uso do VCI. Por outro lado, VC's pertencentes a diferentes VP's podem possuir o mesmo VCI. Desta maneira, um VC pode ser completamente identificado através de seus VCI e VPI. Quando ocorre a comutação de um VCC, o valor do VCI pode ser modificado. Também ocorrem mudanças no VPI quando um *link* VP termina em um comutador ou concentrador. Podemos ter uma idéia melhor dos conceitos de comutação envolvendo VC's e VP's na Fig. 3. O comutador *roteia* os VPC's que chegam aos seus respectivos VPC's de saída. Apesar de haver uma mudança nos canais, o VPI permanece o mesmo. As conexões podem ocorrer tanto à nível de VPC como à nível de VCC [PRY95].

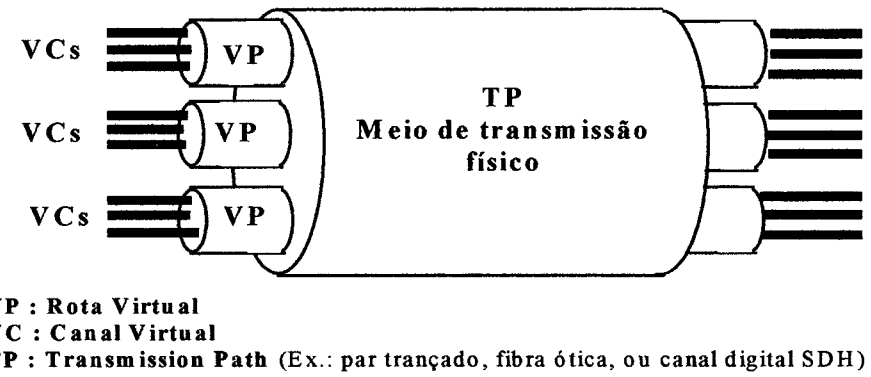


Figura 3 - Relacionamento entre VP e VC em conexões ATM

A utilização de VC's e VP's implicam em algumas vantagens, como por exemplo:

- A simplificação da arquitetura da rede: funções de transporte da rede podem ser agrupadas em VCC's, que por sua vez podem ser agrupados em VCP's.
- O aumento da segurança e da performance da rede: a rede passa a utilizar equipamentos mais agregados, portanto menos equipamentos.
- A redução no processamento e no tempo de conexão: mantendo um VPC conectado como "reserva" para futuras chamadas, as conexões de novos VCC's tornam-se muito rápida, através da execução de controle simples.

- A melhoria dos serviços da rede: o usuário pode, através dos VPC's, criar "sub-redes virtuais" ou grupos fechados de usuários.

Vejamos alguns exemplos de uso de um VCC:

- A ligação entre usuários: além de transportar dados entre os usuários, pode ser utilizado para transporte de sinalização de controle.
- A ligação entre usuário e elemento da rede: transporte de sinalização entre usuário e rede.
- A ligação entre elementos da rede: controle de tráfego e roteamento, através de uma rota exclusiva que pode ser definida.

Algumas características já definidas para um canal virtual são as seguintes [ROC95]:

- Qualidade do serviço: especificação de parâmetros como razão entre células transmitidas/perdidas e variação do atraso;
- Conexão de canal virtual semipermanente ou comutada: para este caso, canais podem ser dedicados. Há a necessidade de sinalização de controle de chamadas;
- Integridade da seqüência das células: a seqüência de transmissão das células em um VCC deve ser preservada;
- Negociação de tráfego e monitoração do uso do canal: para cada VCC, o tráfego através dele pode ser negociado entre usuário e rede. Este tráfego é monitorado pela rede de modo a não haver excesso (violação) do parâmetro negociado. O tipo de negociação que pode haver entre usuário e rede pode incluir taxa média, taxa de pico e duração do pico. A rede deve, ainda, controlar congestionamentos, simplesmente negando novas conexões VCC's ou descartando células, em casos mais extremos. Para o caso dos VPC's, as características citadas acima para os VCC's se aplicam igualmente, havendo ainda a inclusão de mais uma: restrição de uso dos VCC's em um VPC, ou seja, um VCC deve ser resguardado para o uso interno da rede (controle).

2.5.1.1. Virtual Channel Connection

Os VCC's consistem em agrupamentos de *links* VC utilizados para a conexão entre pontos de acesso de serviços ATM. Aqui, o termo *link* VC se refere a uma conexão virtual unidirecional capaz de transportar células ATM entre pontos onde o VCI é determinado e o ponto onde o VCI é modificado ou removido. O VCC pode ser determinado pelo equipamento de comutação e pode ser permanente ou semipermanente. A integridade da seqüência das células é garantida dentro de um mesmo VCC. No momento do *setup* do VCC, parâmetros de tráfego do usuário, como taxa de perda de células e atraso das células já estão definidos sob negociação entre o usuário e a rede, sendo estes parâmetros observados e controlados pela rede. Na UNI podem ser usados quatro métodos para a conexão e desconexão do VCC. Primeiro, o procedimento de sinalização pode ser ignorado, se o estabelecimento ou desestabelecimento da conexão ocorrer através de reserva. Este método se aplica a conexões permanentes ou semipermanentes. O segundo método faz uso dos procedimentos de metasinalização. Metasinalização é a sinalização responsável pelos procedimentos de estabelecimento de um VC de sinalização. Ela é transportada em um VCC permanente e possui valor de VPI e VCI fixos e pré-estabelecidos. Ou seja, um VC sinalizado é estabelecido ou removido através do uso de um VC metasinalizado [PRY95].

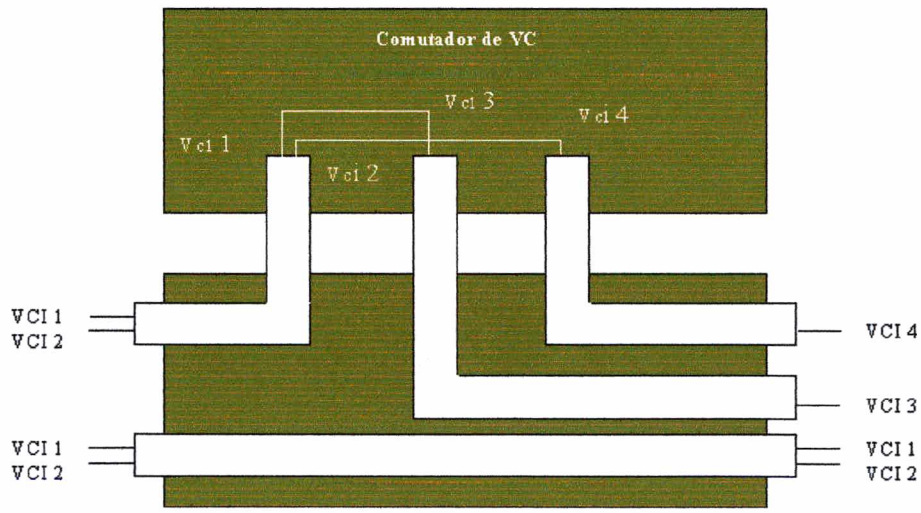


Figura 4 - Comutação dos VP's e VC's

O terceiro método utiliza o procedimento de sinalização usuário-rede. Isto implica no uso de um VCC sinalizado para estabelecer ou remover uma VCC para comunicação fim-a-fim. O quarto método faz uso do procedimento de sinalização usuário-usuário. Isto implica no uso de um VCC sinalizado para estabelecer ou remover um VCC interno a um VPC pré-estabelecido entre duas UNI. Quatro métodos também são possíveis para a atribuição de valores ao VCI na UNI: atribuição pela rede, pelo usuário, através de negociação entre usuário e rede e pelo uso de um método padronizado. Geralmente a atribuição do valor ao VCI não possui relação com o serviço provido pelo VC correspondente. Para fins de facilidade de inicialização e de mudanças, é desejável atribuir o mesmo valor ao VCI para algumas funções específicas. Uma vez que o cabeçalho das células é processado em equipamentos como comutadores ATM, *cross-connect* e concentradores, o processo de modificação do VCI e do VPI é realizado também no nível desses equipamentos. Por essa razão, quando um VCC é estabelecido ou removido internamente à rede ATM, o estabelecimento ou desconexão de um link VC pode ocorrer em mais de uma NNI. Neste caso, um *link* VC é estabelecido ou desconectado através de sinalização interna ou de procedimentos de sinalização inter-rede dos elementos ATM.

2.5.1.2. Virtual Path Connection

O VPC é um conjunto de *links* VP para a conexão de pontos, nos quais um VPI é atribuído a pontos, onde o VPI é modificado ou removido. Um *link* VP conecta pontos onde o VPI é criado ou removido. O VPC pode ser criado por um equipamento de comutação e pode ser permanente ou semipermanente. A seqüência das células é garantida em cada VCC pertencente a um mesmo VPC. No momento da conexão de um VPC, alguns parâmetros como tráfego, taxa de perda de células e variações de atraso são estabelecidas de acordo com a necessidade do usuário, e garante-se que a qualidade desses serviços seja mantida através da monitoração destes pela rede. Há duas maneiras de se estabelecer um VPC entre dois nós. Uma faz uso de reserva para conexão (não há utilização dos procedimentos de sinalização). Na outra maneira, os VPC's são atribuídos ou removidos de acordo com as necessidades de controle do usuário e da rede. Ainda,

como nos VCI, os VPI podem ser pré-definidos. Das funções da camada ATM, a mais importante, é considerada realmente, o roteamento dos VPI e VCI através da rede (conexão ATM). Porém, a camada ATM é responsável ainda por mais algumas funções, como por exemplo [CHA93]:

Generic Flow Control: a função do GFC é controlar o fluxo das várias conexões ATM. O GFC controla o acesso ao meio na UNI e controla o tráfego de modo a conter situações de início de congestionamento. Outras funções incluem: redução dos serviços de taxa constante e alocação justa da capacidade para serviços a taxa variável. O GFC é uma característica especial da camada ATM e é provida independentemente da camada física. Além disso, é aplicável a qualquer configuração de UNI (estrela, anel, barramento).

Cell Loss Priority: uma vez que os serviços a taxa variável podem ter variações muito grandes da taxa de *bits*, pode ocorrer um momento em que todos esses serviços estejam utilizando sua taxa máxima. Ocorre, neste momento, congestionamento na rede. De forma a tentar evitar esta situação, utiliza-se o CLP. O CLP é um campo onde é indicada a prioridade da célula no caso da necessidade de descarte. As células que têm seu *bit* CLP *setados* são descartadas primeiro. A função CLP deve ser provida em conjunto com a *QoS* determinada no momento da conexão VPC/VCC. Deve ser possível manter uma taxa mínima mesmo após a perda de células e o serviço pré-determinado deve ser mantido. Por causa disso, a rede deve determinar a taxa de bits das células de maior prioridade no momento do estabelecimento da conexão e esta taxa deve ser negociável, mesmo após o término da conexão. A rede deve monitorar constantemente o fluxo de células de modo que o número de células em uma conexão qualquer não exceda o valor pré-estabelecido; caso isso ocorra, até mesmo as células com prioridade mais alta podem ser descartadas pela rede.

Payload Type Indication: o campo *Payload Type* indica se a informação contida no campo do usuário da célula consiste em informação do usuário ou informação da rede e, adicionalmente, indica a ocorrência de uma situação de congestionamento. A informação do usuário consiste em informações, propriamente ditas, e informações

sobre a função de adaptação a serviços. Já a informação de rede inclui informações de OAM e de gerenciamento de recursos. Enquanto as células ATM para uso geral são criadas em um terminal de usuário e entra na rede através da UNI, as células utilizadas para transporte de informação da rede são criadas internamente à rede e atravessam a UNI.

2.6. Camada de Adaptação ATM (ATM Adaptation Layer)

A rede ATM oferece suporte a uma grande variedade de serviços. Como as características destes serviços são as mais diversas possíveis, é necessária uma adaptação das características específicas de cada serviço para que eles sejam transmitidos através da rede comum ATM. Essa adaptação é feita pela Camada de Adaptação ATM (AAL - ATM Adaptation Layer).

A camada AAL executa funções requeridas pelos planos de usuário, de controle e de gerenciamento, e suporta o mapeamento entre a camada ATM e a camada imediatamente superior. As funções executadas pela camada AAL dependem dos requisitos da camada superior. O AAL suporta múltiplos protocolos de modo a atender às necessidades específicas dos usuários do serviço AAL. Portanto, a camada AAL é dependente do serviço [ITU93a].

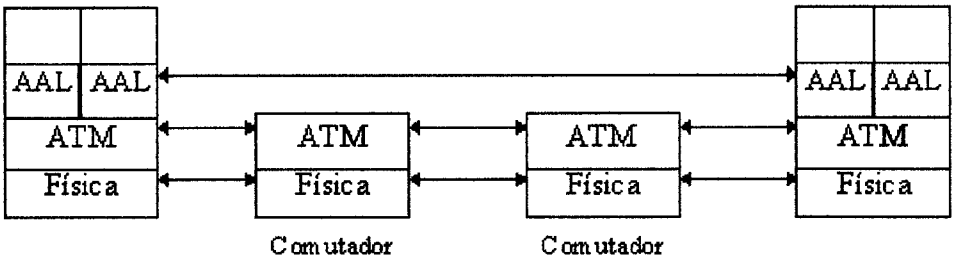


Figura 5 - Camada AAL

Para facilitar a definição das funções que a camada AAL deve suportar para cada tipo de serviços, eles são divididos em quatro classes: A, B, C e D, de acordo com a

Recomendação I.362 do ITU-T. O Fórum ATM ainda considera mais uma classe de serviços: a classe X [ITU93a].

Classe A: suas características básicas são: pequeno atraso máximo, variação de atraso desprezível, intervalo de tempo entre a transmissão de dois *bits/octetos* fixo e transmissão completa da cadeia de *bits/octetos*, isto é, nenhuma informação é perdida nem a ordem é alterada. Alguns exemplos de serviços pertencentes à classe A são: a emulação de circuitos para serviços assíncronos, como transmissão de voz e vídeo a taxas constantes (sem compressão ou compactação).

Classe B: as características básicas dos serviços da classe B são as mesmas dos serviços da classe A, porém a taxa de transmissão passa a ser variável. Por exemplo, transmissão de voz e vídeo a taxas variáveis (devido à compressão e compactação).

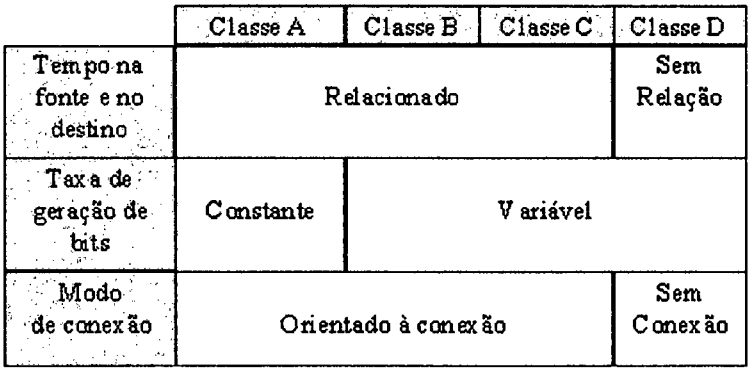


Figura 6 - Classes de Serviços

Classe C: os requisitos básicos em que os serviços desta classe devem se encaixar são: atraso máximo moderado, variação moderada do atraso, não há necessidade de sincronização entre *bits/octetos* ou quadros transmitidos e variação no comprimento da cadeia de dados transmitidos (porém, mantendo seu conteúdo e limitações).

Classe D: as características dos serviços pertencentes à classe D são as mesmas da classe C, porém os serviços da classe D não são orientados à conexão. Um exemplo é a interconexão de redes utilizando TCP/IP.

Classe X: a classe X define um serviço orientado à conexão ATM. A camada AAL, neste caso, não tem função.

2.6.1. Tipos de AAL

Existem, atualmente, cinco tipos de AAL definidas [ITU93a]:

AAL 0: é a ausência de funções da camada AAL. Representa o processo que conecta o usuário da AAL diretamente ao serviço oferecido pela camada ATM. AAL 0 pode ser utilizada por equipamentos que querem fornecer seus próprios serviços, utilizando diretamente a tecnologia de transferência ATM.

AAL 1: a camada AAL 1 efetua os procedimentos necessários para satisfazer os requisitos dos serviços da classe A. Seus serviços oferecidos são: a transferência de unidades de dados com uma taxa constante de geração e a sua entrega ao destino na mesma taxa, transferência de informações de sincronismo entre origem e destino e indicação de perda de informações ou do recebimento de informações não recuperáveis pela AAL 1. Além dos serviços oferecidos, a AAL 1 pode executar as seguintes funções: segmentação e remontagem das informações, tratamento da variação do atraso das células, tratamento de células perdidas ou inseridas erroneamente, recuperação do relógio da fonte na recepção, monitoramento do campo de informações do usuário para detectar *bits* com erros possíveis de se corrigir, monitoramento e tratamento de *bits* com erro no PCI (*Protocol Control Information*).

AAL 2: o objetivo da AAL 2 é efetuar os procedimentos necessários para fornecer serviços da classe B. Seus serviços oferecidos são: transferência de unidades de dados a taxa variável (*VBR - Variable Bit Rate*), sincronismo entre origem e destino e

indicação de informações perdidas ou com erros não recuperáveis pela AAL 2. As mesmas funções oferecidas pela AAL 1 são fornecidas pela AAL 2.

AAL 3/4: efetua os procedimentos necessários para fornecer serviços das classes C e D. Como os procedimentos das AAL 3 e AAL 4 podem ser executados para ambas as classes de serviços, eles foram combinados durante o processo de definição das normas. Os serviços de transporte oferecidos pela AAL 3/4 podem ser de dois tipos: modo de mensagem (onde um quadro de informação é recebido do usuário e enviado a outro usuário na rede), e modo de fluxo (no qual um fluxo de quadros de informação é transportado pela rede a outro usuário). A AAL 3/4 também define dois tipos de operação: assegurada e não assegurada. Na operação assegurada, a AAL deverá efetuar a recuperação de erros fim-a-fim por retransmissão. Já na operação não assegurada, a recuperação de erros não é feita pela AAL, sendo deixado a cargo do usuário a decisão de receber ou não os quadros com erro. Esses modos (de mensagem e de fluxo) e operações (asseguradas e não asseguradas), são opções a serem implementadas de acordo com a definição de serviços específicos.

AAL 5: a AAL 5 foi elaborada para operar de forma mais simples e eficiente que a AAL 3/4 (embora não ofereça todas as funções da AAL 3/4). A AAL 5 também inclui os modos de mensagem ou de fluxo e as operações asseguradas ou não, não oferecendo a função de multiplexação de conexões.

<table>
<tr><td rowspan="2">AAL</td><td>CS</td><td>Convergência</td></tr>
<tr><td>SAR</td><td>Segmentação/reencapsulamento</td></tr>
<tr><td colspan="2">ATM</td><td>Controle de fluxo genérico,
geração/extração de cabeçalho,
translação VPI/VCI,
multiplexação/demultiplexação</td></tr>
<tr><td rowspan="2">Camada
Física</td><td>TC</td><td>Geração/verificação do HEC,
geração quadro de transmissão,
delineamento de células</td></tr>
<tr><td>PM</td><td>Temporização, meio físico</td></tr>
</table>

Figura 7 - Funções das Subcamadas

2.6.2. Estrutura da AAL

A camada AAL é dividida em duas subcamadas lógicas: subcamada de convergência (CS - *Convergence Sublayer*) e a subcamada de quebra e remontagem (SAR - *Segmentation and Reassembly Sublayer*). A SAR é responsável pela segmentação das informações das camadas superiores (PDU - *Protocol Data Units*), em um comprimento compatível com o campo de informações de uma célula ATM, para transmissão e remontagem dessas informações na recepção. A CS é uma subcamada que fornece as funções necessárias a aplicações específicas que utilizam a AAL. Conforme o tipo de serviço, a CS pode efetuar funções de multiplexação, detecção de perda de células e recuperação da relação temporal da informação original no destino [ITU96].

2.6.3. Recuperação do sinal de clock

A recuperação do sinal de *clock* no receptor é uma das principais funções da AAL para serviços em tempo real, devido à natureza estatística da rede ATM, as células não chegam ao destino periodicamente, nem mesmo no caso de serviços a taxa de bit constante. Se a rede ATM for baseada em uma rede assíncrona, ou seja, cada nó da rede possui diferentes referências de *clock*, a única informação disponível do *clock* é o *long-term average cell throughput*. Neste caso, o *clock* pode ser recuperado através de um PLL. Porém, se a rede for síncrona, ou seja, a referência de *clock* é única para toda a rede, então é possível sincronizar a transmissão e a taxa do serviço com o *clock* da rede. Neste caso, vários métodos podem ser utilizados para a recuperação do *clock*, como o *Synchronous Residual Time Stamp* (SRTS) e o *Time Stamp* (TS) [SID93].

2.7. Controle de Tráfego

A vantagem principal do ATM reside na eficiência da utilização dos recursos da rede e da sua flexibilidade para suportar os mais variados tipos de serviços. Porém, se por um lado temos isso como vantagem, por outro lado o problema do controle de

tráfego torna-se muito grave e deve ser resolvido. Os mecanismos de controle de tráfego largamente utilizados em redes de pacotes não possuem a eficiência necessária quando aplicados à RDSI-FL, pois as redes de baixa velocidade não levam em conta a Qualidade do Serviço (*QoS - Quality of Service*) requerida pelos serviços da RDSI-FL, (a QoS é definida na recomendação E.800 como sendo o efeito coletivo do desempenho do serviço e que determina o grau de satisfação do usuário deste serviço), nem a limitação da rede, devido à latência (as redes de pacote de baixa velocidade são limitadas pela largura de faixa). Outro problema, é que na RDSI-FL o controle de congestionamento deve servir para prevenção e não como solução do problema já ocorrido [ITU96].

2.7.1. Conceitos Básicos do Controle de Tráfego e do Gerenciamento de Recursos

Os objetivos principais do controle de tráfego e do gerenciamento de recursos na RDSI-FL são: a proteção da rede contra congestionamento, alcançar seus objetivos relativos à performance e otimização dos recursos da rede. Para se entender a estrutura do controle de tráfego na RDSI-FL, os procedimentos de estabelecimento de chamada e desconexão devem ser entendidos. Será dada a seguir uma visão simples desses procedimentos, com ênfase no controle de tráfego. Quando um usuário deseja estabelecer uma chamada, em primeiro lugar os parâmetros que representam as características estatísticas da fonte são passadas para a rede. À seguir, o CAC - *Call Admission Control* da rede decide onde a chamada pode ser aceita sem afetar a *QoS* das outras chamadas já estabelecidas. Se uma nova chamada está para ser aceita, um contrato de tráfego é feito com a fonte. A rede decide então o caminho a ser utilizado pela fonte para enviar suas mensagens ao destino. Uma vez estabelecida a chamada, a fonte pode enviar células pela rede na taxa especificada no contrato de tráfego. A rede monitora o tráfego para garantir que a fonte não esteja extrapolando a taxa negociada. Durante este processo, a rede pode alocar seus recursos de modo a separar os tráfegos de acordo com as características dos serviços (gerenciamento de recursos). O usuário ainda pode gerar diferentes prioridades de tráfego através do uso do *bit* CLP de modo

que um elemento da rede que se encontrar congestionado possa descartar células, caso seja necessário (controle de prioridade) [ITU96].

2.7.2. Parâmetros de Tráfego

A única informação necessária quando se estabelece uma chamada em uma rede síncrona, do ponto de vista do controle de tráfego, é a taxa máxima da fonte. Desta forma, o estabelecimento da chamada é simples, mas há um gasto excessivo de largura de banda. A fim de melhorar a eficiência em termos de largura de banda, o ATM utiliza o conceito de multiplexação estatística. Nas redes síncronas, a largura de banda necessária para a utilização de um serviço é a soma das suas taxas de pico. A multiplexação estatística implica em uma maior eficiência, visando o suporte a esses serviços com uma largura de banda menor. Isto pode ser conseguido se for conhecido um certo número de características estatísticas da fonte, tal que os mecanismos de controle de tráfego possam ser implementados de forma eficiente. Segundo o CCITT, um parâmetro de tráfego é uma especificação de um aspecto particular de tráfego de uma fonte. Um *"ATM descriptor"* é uma lista genérica dos parâmetros de tráfego que podem ser utilizadas para se obter as características de tráfego intrínseco de uma conexão ATM, enquanto um "*descriptor*" do tráfego da fonte é um conjunto de parâmetros de tráfego utilizados na negociação do contrato de tráfego entre o usuário e a rede para descrever as características de tráfego exigidas pela fonte. Um exemplo de "*descriptor*" de tráfego seria: taxa média de células, taxa máxima de células, comprimento de rajada médio. Esses parâmetros descrevem, individualmente, apenas alguns aspectos do tráfego da fonte, porém o conjunto descreve as características de tráfego da fonte como um todo. Caso várias fontes sejam multiplexadas, as características individuais se tornam menos significativas ainda, isto é, as características medidas na rede serão diferentes das características de tráfego originais. Assim, se os parâmetros forem significativos, os pontos de referência para sua definição devem estar o mais próximo possível da rede [ITU93].

2.7.3. Métodos de Controle de Tráfego

O *QoS* é degradado quando ocorre um congestionamento na rede, tendo como conseqüência um serviço pobre para o usuário. O principal objetivo do controle de tráfego é prevenir a ocorrência de congestionamento ou controlá-lo o mais rapidamente possível. Os métodos de controle de tráfego podem ser divididos em dois tipos principais. O primeiro método, *reactive control*, reage ao congestionamento após a sua ocorrência. Já o segundo método, chamado *preventive control*, visa prevenir a ocorrência do congestionamento. Nas redes de pacotes existentes tem sido usado o mecanismo de controle do tipo *reactive type*. Porém, na RDSI-FL, esse tipo de método de controle tende a se tornar ineficiente, esperando-se que os principais métodos de controle utilizados nas redes ATM sejam do tipo preventivo, e não reativo. Os mecanismos de controle de tráfego ainda podem ser divididos, com base nos níveis em que atuam, em mecanismos de controle a nível de célula [CHE97].

3. GERÊNCIA DE REDES ATM

O gerenciamento de redes ATM está atualmente definido em três áreas gerais: gerenciamento de interface, incluindo UNI, DXI e emulação de LANs, gerenciamento de camadas, e gerenciamento de rede como um todo, usada para o gerenciamento da rede ATM em si e de seus serviços, que são atualmente modelados em cinco categorias.

O gerenciamento de interface com a troca de informação à nível de interface é principalmente usado para configuração e alarmes de interfaces ATM, e provê a estrutura básica para dois dispositivos diferentes conectarem-se um ao outro. O gerenciamento de camada permite decidir sobre a continuidade ou reinicialização à nível de um segmento ou de circuito virtual fim-a-fim (ambos a nível de VC e VP). É principalmente usado para gerenciamento de circuitos fim-a-fim e permite a verificação de um circuito de usuário pela rede. O gerenciamento de rede global trata da configuração de uma rede ATM constituída de um ou mais *switches*, monitorando e controlando os dispositivos ATM na rede.

3.1. Estrutura de Gerenciamento de Rede

Um sistema de gerenciamento de rede consiste em quatro partes: estação de gerência (ou gerente), agente, MIB, e protocolo de gerência de rede.

A estação de gerência funciona como uma interface do gerente com o sistema de gerência de rede. Traduz os comandos de gerência em monitoramento real e controle dos elementos de rede. Vários serviços providos na estação de gerência incluem aplicações para prover análise de dados e recuperação de falhas, e um banco de dados para fornecer informação de gerência de rede, extraída dos bancos de dados de todos os elementos gerenciados na rede.

Cada nodo na rede (inclusive estações terminais) que participa da gerência contém uma entidade de gerência (NME - *Network Management Entity*), que é um software para executar tarefas relacionadas com a gerência de rede. Cada NME coleta

dados com recursos próprios e armazena relatórios estatísticos localmente. NMEs também respondem a comandos do administrador da rede (o gerente). Pelo menos um nodo na rede é designado para uma aplicação de gerência, que inclui uma interface para o gerente administrar a rede. NMEs são chamados de agentes. As aplicações de gerência de rede (NMA - *Network Management Applications*), sob o controle do operador, respondem a comandos de usuário passando as informações disponíveis em seu banco de dados. Também emitem comandos e trocam informação com as NMEs na rede. [MAR98].

O terceiro componente da estrutura de gerenciamento de rede é a MIB. Uma MIB é uma coleção de objetos, cada qual representa um aspecto particular de um agente gerenciado. Por exemplo, um gerente executa uma função para buscar o valor de um objeto particular ou para mudar as configurações de um recurso de rede modificando o valor do objeto correspondente.

Finalmente, a comunicação entre o gerente e os agentes ocorre usando um protocolo de gerência de rede. Os dois protocolos de gerência de rede padronizados são: o protocolo de gerência de rede simples, do IETF (SNMP - *Simple Network Management Protocol*) e o protocolo de informações comuns de gerência da ISO (CMIP - *Common Management Information Protocol*). Além de definir um protocolo específico, ambos, SNMP e CMIP, definem uma especificação de estrutura de banco de dados e um conjunto de objetos de dados (MIBs) [MAR98].

3.2. Áreas Funcionais do Gerenciamento de Redes

3.2.1. Gerenciamento de Configuração

O gerenciamento de configuração é o processo de achar e configurar dispositivos de rede. É um conjunto de atividades de curto e médio alcance para controlar os recursos de rede, avaliando desempenho, mantendo arquivos das dificuldades, avaliando

e negociando níveis de serviço, gerenciando e mudando níveis de segurança, e negociando custo e carga.

Os usuários desempenham um papel importante na gerência de configuração. As expectativas deles para com os serviços de rede e a demanda por banda é determinada quando a rede é configurada. É importante que estas expectativas sejam satisfeitas com um custo adequado. Se não, é necessário reconfigurar a rede. Em outras palavras, gerência de configuração consiste na obtenção de dados da rede e de seu uso, para administrar todos os recursos por meio de políticas de gerenciamento. Aplicado a uma rede de ATM, incluindo estações terminais, as funções de gerenciamento de configuração incluem:

- Criação e *deleção* de conexões ATM (VPCs e VCCs);
- Obtenção do estado de uma conexão;
- Determinação do número de conexões ativas em uma interface;
- Determinação do número máximo de conexões suportadas em uma interface;
- Determinação do número de conexões pré-configuradas em uma interface;
- Configuração do número de bits de VPI/VCI suportados;
- Configuração e determinação do estado da interface de informação de endereço.

3.2.2. Gerenciamento de Falhas

Gerenciamento de falhas é o processo de localizar problemas ou falhas na rede. É um conjunto de atividades necessárias para manter o nível de serviço de rede desejado. O funcionamento da rede é essencial para prover os serviços, porém condições anormais como erros excessivos e dificuldades para operar corretamente acontecem. Tais falhas ocasionais precisam ser detectadas depressa. Enquanto os componentes com falha estão sendo consertados ou substituídos para restabelecer a rede a seu estado inicial, a rede pode ser reconfigurada de forma que o impacto de componentes falhos seja minimizado. Quando uma falha acontecer é necessário, tão depressa quanto possível, que se:

- Determine o local do recurso falho e a natureza da falha;
- Isole e reconfigure o resto da rede;
- Restabeleça a rede a seu estado original.

Aplicando a uma rede ATM incluindo estações terminais, as funções da gerência de falhas incluem:

- Notificar a inabilidade para estabelecer conexões ATM;
- Notificar falhas em uma conexão ATM;
- Notificar falhas em conexões simultâneas múltiplas;
- Notificar falhas em um par UNI adjacente;
- suportar fluxos OAM de gerenciamento de falhas.

3.2.3. Gerenciamento de Desempenho

Gerenciamento de desempenho é medir o desempenho do hardware de rede, software e meios de transmissão. É uma investigação quantitativa dos recursos de rede para verificar que níveis de serviço são mantidos e apoiar outra administração de rede, que funciona na configuração e administração de falta. Na maioria dos casos, as aplicações exigem que os recursos de rede operem dentro dos seus limites de desempenho especificados. O gerenciamento de desempenho é usado para monitorar os níveis de desempenho de vários recursos da rede e permitir ajustes para melhorar o desempenho da rede. Esta função detecta a efetividade com que uma rede está entregando serviços a seus usuários. As funções do gerenciamento de desempenho são [MAR98]:

- Determinar se uma conexão ATM satisfaz ou não as exigências de *QoS*;
- Determinar o número de células que violam o contrato de tráfego;
- Suportar fluxos OAM de gerenciamento de desempenho;

- Suportar um conjunto de contadores apropriados para enviar e receber operações dos níveis VC e VP.

3.2.4. Gerenciamento de Segurança

Gerenciamento de segurança é o processo de controlar acesso à rede e proteção de objetos dentro da rede. Envolve a proteção da transferência de informação de um usuário autorizado de um local para outro. Os cinco principais serviços de segurança para o seu efetivo gerenciamento, são:

- Autenticação: Verificação da fonte de informação (ver se as informações vieram da fonte esperada);
- Controle de acesso: A habilidade para restringir ou controlar acesso a uma fonte ou recursos de rede. Esta função tenta assegurar que somente usuários autorizados têm acesso a arquivos correspondentes, partições de rede, e bancos de dados;
- Integridade: A verificação de que os dados recebidos realmente são os dados que foram enviados;
- Confidencialidade: A propriedade que os dados possuem, onde o receptor necessita saber uma senha de autorização para consultar a informação;
- Não-repúdio: A certificação de que o receptor ou o remetente realmente recebeu ou enviou os dados.

Conseqüentemente, as funções principais de um sistema de segurança incluem: proteção de armazenamento, proteção de transmissão e recebimento de dados, e controle de transmissão.

Todos estes podem ser cifrados, ou seja, sofrer uma alteração nos dados, de certo modo que os torne sem sentido a intrusos sem autorização.

3.2.5. Gerenciamento de Contabilização

O gerenciamento da contabilização encarrega-se de identificar o uso de recursos de rede pelos usuários e o custo que incorreu para o serviço. Também é usado para limitar a quantia de recursos alocada a usuários e informá-los dos custos incidentes sobre os recursos que usaram. Os vários processos e procedimentos associados com o gerenciamento da contabilização incluem:

- Identificar os componentes de custo, inclusive serviços, pessoal, e *overhead*;
- Estabelecer políticas de cobrança, que envolvem a determinação de como será cobrado o usuário e em que bases;
- Cálcular o consumo à partir da definição de procedimentos para tal;
- Processar a fatura de cobrança, que envolve computação dos consumos e sua transferência aos usuários da rede.

Aplicado a uma rede ATM incluindo estações terminais, as funções de gerenciamento de contabilização podem incluir:

- Registrar o QoS da conexão;
- Registrar a largura de banda da conexão;
- Registrar a duração da conexão;
- Registrar o número de células transmitidas e recebidas com sucesso;
- Registrar o número de células recebidas com erro;
- Registrar o número células recebidas e que violaram o contrato de tráfego.

A determinação do custo de comunicação é uma tarefa complexa. Existem vários fatores diretos, bem como indiretos, que compõem o custo de um serviço. Hardware, software, pessoal, e sobrecarga (*overhead*) são usados para prover todos os diferentes serviços de rede que contribuem com o custo. Porém, a contribuição de cada fator para um serviço específico não pode ser determinado com precisão.

3.3. Interface de Gerenciamento ATM

A interface de gerenciamento ATM é principalmente usada para configuração e obtenção de informações de estado das interfaces ATM. As interfaces atualmente definidas pelo ATM Forum, são: UNI, DXI, B-ICI, e LAN *emulation* UNI (LUNI). As definições de MIB para algumas destas interfaces são discutidas nas próximas três seções.

3.3.1. Interface UNI – User-to-Network Interface

Tanto público como privado, os dois lados de uma UNI trocam informações de configuração e de estado, usando um protocolo de gerência de rede. A informação é então disponibilizada para uma estação de gerência de rede pelos agentes, acessíveis remotamente, que residem nos terminais ATM. Há duas grandes organizações que definem as MIBs UNI ATM: o ATM Forum e o IETF. As perspectivas que estas duas organizações têm no gerenciamento de uma UNI são diferentes.

O ILMI (*Interim Local Management Interface*) é especificado pelo ATM Forum, e é usado para configuração de uma UNI específica, não provê funções de segurança, falha, ou de gerenciamento da contabilização. A estrutura do ATM Forum não requer um agente em nenhuma extremidade da UNI. Em susbstituição, o ATM Forum definiu uma UME (*UNI Management Entity*), que usa SNMP em cima de AAL 5. O ILMI suporta a troca de informações de gerência entre UMEs, ligadas a parâmetros da camada ATM e da camada física. A comunicação entre UMEs adjacente, de forma que uma UME possa ter acesso a informações associadas aos objetos da MIB da UME adjacente, é baseada em comandos SNMP, que são transportados em cima de AAL 5 e não usam UDP ou IP (distinto do SNMP original) [MAR98].

A solução de IETF exige que os agentes IP residam em um dos lados da interface administrada. O IETF não faz qualquer discussão explícita sobre o uso do protocolo ATM, nem define qualquer interação entre o nodos ATM adjacente. O

conjunto de objetos gerenciados pela UNI são chamados de atributos da UNI ILMI, que são organizados em uma estrutura padrão de MIB.

O grupo de estatística ATM é opcional, enquanto todos os outros são obrigatórios. A tabela contém informações sobre o estado da configuração da interface física, como endereços e tipo de transmissão (por exemplo, DS-3, OC-3), o tipo de mídia física, como por exemplo, UTP-3 ou fibra, seu estado operacional (por exemplo, para cima, para baixo, desligado, quebrado, on-line) e dados específicos, que variam com o tipo de mídia usada. O grupo da camada ATM especifica o número máximo de VPCs e VCCs suportados pela UNI. Além disso, incluem informações sobre o número de rotas configuradas, bits de indicação de rota, e tipo de porta (privada ou pública). O grupo VP contém um indicador de rota, seu estado operacional, e a classe de *QoS* suportada em cada direção. Há um grupo semelhante de informações armazenadas para circuitos virtuais. O grupo de prefixos de rede dá o prefixo associado ao endereço ATM e a tabela de endereço, que inclui a lista de endereços associados com a UNI [MAR98].

Finalmente, o grupo opcional de atributos estatísticos da camada ATM inclui: informação sobre a contagem do número de (excluindo inativo) células recebidas pela UNI, o número de células excluídas em função do resultado do processamento do HEC, e células com cabeçalhos inválidos.

3.3.2. Data Exchange Interface

A entidade DXI-LMI (*Data Exchange Interface – Local Management Entity*) opera junto com a interface DXI e define o protocolo para trocar dados pela DXI que inclui DXI, AAL, e UNI com gerenciamento de informações específico. A definição da entidade LMI assume uma relação de um para um, entre a interface DXI e a interface UNI. Foi projetada para suportar uma estação de gerência rodando SNMP e/ou um *switch* rodando ILMI.

A DXI é a interface entre o DTE (um roteador, por exemplo) e o DCE (uma unidade servidora de dados, por exemplo). Os comandos SNMP são emitidos somente no DTE e o DCE somente responde. O DCE pode originar somente mensagens de exceção (*trap messages*), mas pode responder a comandos do DCE. A MIB da entidade DXI LMI é composta por dois tipos de objetos, com funções de configuração e funções de gerenciamento de desempenho: o grupo de configuração DXI e o grupo DFA (DXI *frame address*). O grupo de configuração, inclui: informações de configuração (similar a MIB ILMI correspondente), incluindo o modo de operação e identificação da interface. O grupo DFA inclui: informações sobre o tipo de AAL suportado por um determinado DFA, entre outras [MAR98].

3.3.3. LAN Emulation

O LANE permite rodar aplicações, desenvolvidas para redes legadas, sobre o ATM sem qualquer mudança nas aplicações. A estrutura do LANE é formada por quatro componentes: clientes LANE, servidores LANE, barramento, e servidores de configuração.

O gerenciamento de redes do LANE trata somente das funções: gerenciamento de configuração, desempenho e falhas. O gerenciamento de configuração LANE trata de várias tarefas, que incluem: identificação de todos os clientes LANE, atualmente configurados em um dispositivo gerenciado; criação e destruição dos clientes LANE; controlar a permissão de acesso dos clientes às LANs emuladas; verificação e alteração de vários parâmetros do sistema; e identificação da configuração, controle, e VCCs multicast [TAF95].

Há vários fatores que tornam mais difícil a observação de LANs emuladas, que a observação de LANs legadas. O tráfego é esparramado sobre muitos circuitos virtuais, que são abertos e fechados freqüentemente, em vez de estar concentrado em um segmento de rede físico. O desempenho de cada circuito virtual pode ser afetado por fatores fora do controle dos equipamentos da LAN emulada (por exemplo,

congestionamento do *switch* e mecanismos de controle de congestionamento dos *switches*). Dadas estas dificuldades, as estações de gerenciamento monitoram a quantidade de tráfego que vai para um equipamento específico, recrutando a ajuda aos clientes LANE que coletam estas informações, colecionando e agregando estatísticas de desempenho sobre circuitos virtuais, colecionando estatísticas de desempenho de portas ATM, e escutando a comunicação entre equipamentos LANE.

Há vários níveis de gerenciamento de desempenho que incluem o monitoramento de tráfego de LUNI (LAN *Emulation* UNI), gerenciamento de desempenho de VCs individuais dentro de uma LAN emulada, e gerenciamento de desempenho de redes de ATM sobre a qual roda uma LAN emulada.

Finalmente, o gerenciamento de falhas preocupa-se com a prevenção, detecção, e correção de problemas em uma LAN emulada, que são causadas por falhas em elementos de rede.

A MIB de um cliente LANE é organizada em grupos, cada um correspondendo a uma tabela:

- Grupo de interfaces, que inclui LANS emuladas, estatísticas, e grupos de servidores de conexões;
- Grupo de endereços ATM;
- Grupo com registro de destinos, que inclui endereços MAC e grupos de descritores de rota;
- Grupo LANE ARP *cache*, que inclui traduções de endereços MAC e descritores de rota.

4. MONITORAMENTO DE REDES ATM

A *Remote Monitoring MIB Extensions for ATM Networks* é uma proposta do ATM FORUM, para implementar os benefícios do RMON em redes ATM. Esta proposta visa disponibilizar informações estatísticas sobre tráfego e funcionalidades da rede ATM. A proposta é baseada no RMON-1 e RMON-2 do IETF e sugere que com poucas mudanças efetuadas na MIB do RMON-2, é possível se ter estatísticas dos *frames* que são transportados pelo *AAL5* em uma rede ATM.

Aplicações baseadas em RMON podem prover sistemas administrativos com valiosos dados sobre a utilização da rede e seu comportamento. RMON é tradicionalmente disponibilizado como uma ou mais aplicações NMS, gerenciando múltiplos agentes RMON, cada qual monitorando um ou mais segmentos da rede. RMON para redes ATM necessita um modelo diferente de formação, bem como muitas outras novas características da MIB.

4.1. Monitoração do ATM

Administradores de rede necessitam acessar as estatísticas de tráfego de utilização, a fim de controlar eficazmente as suas redes. A MIB RMON está pronta para prover um conjunto completo de informações estatísticas e funções de gerenciamento para redes *Ethernet* e *Token Ring*, sendo que é altamente desejável estender este conjunto de funcionalidades para as redes ATM.

Adaptar o RMON para redes ATM, requer a modificação dos padrões em três áreas [MAR98]:

- Framework da Aplicação de Monitoração: um conjunto de exigências e uma focalização racional da aplicação e projeto da MIB.

- MIB de monitoração : um conjunto de funções de monitoração adequadas para a formação distribuída em várias plataformas, incluindo implementações nativas aos equipamentos e implementações externas.
- Identificadores de Protocolo RMON-2 para ATM : um conjunto de macros de encapsulamento do protocolo devem ser adicionadas à especificação dos identificadores do protocolo RMON-2, para suportar a decodificação dos pacotes dos níveis superiores para encapsulações específicas do ATM. Todas as análises do tráfego da rede ATM que são baseadas em *frames* podem ser suportadas diretamente pela RMON-2, sem objetos MIB adicionais.

As MIBs RMON correntes analisam o tráfego baseado em *frames*. Com algumas adições menores a MIB RMON-2, *frames* conduzidos nas células AAL 5 podem ser contados. Além disso, para a análise dos *frames*, uma MIB RMON a qual provê informações de tráfego baseada em células é bastante desejável.

Em uma perspectiva de desenvolvimento da aplicação, uma MIB de monitoração é requerida com os seguintes atributos [MAR98]:

- Manter a estrutura e a arquitetura RMON para desenvolvimento da aplicação e experiências existentes do usuário.
- Prover mecanismos de configuração flexíveis, adaptáveis a natureza orientada a conexão do ATM.
- Prover estatísticas básicas para cada seqüência de células monitoradas, para cada host ATM identificado, e na conversação entre dois hosts ATM identificados pelos agentes.
- Prover dados estatísticos agregados e compactados ao NMS, reduzindo a carga e o tempo da transferência dos dados.
- Ocultar detalhes de baixo nível, como informação específica do VPI/VCI, o qual não pertence a nenhum recurso gerenciavel pelo NMS. Este nível de detalhe não fornece informações de gerenciamento suficientemente úteis para justificar o gasto com armazenamento e indexação de uma enorme quantidade de dados nos

agentes, e então transferindo os dados ao NMS (outras MIBs existem para coletar estatistísticas por conexão).

- Permitir implementações eficientes para pesquisas nativas e stand-alone. Conceder monitoração 'cost-effective' a links de alta velocidade, através da optimização da configuração e da amostragem estatística do tráfego baseado em frames.

4.2. Aplicabilidade do RMON para Redes ATM

A MIB RMON provê várias funções de gerenciamento que podem diretamente e indiretamente ser aplicadas para redes ATM:

- Estatísticas detalhadas da camada LLC para segmentos Ethernet (EtherStats group). RMON ATM mantém um grupo "*status*", para simplesmente fornecer um total às tabelas associadas ao host e às tabelas matrizes.
- Sondagem remota de detalhadas estatísticas da camada LLC para segmentos Ethernet (history, etherHistory). RMON ATM utiliza a coleção de aplicações history genérica, encontrada na MIB RMON-2.
- Estatísticas básicas por host e por conversação, para todos os endereços MAC válidos descobertos em cada segmento monitorado (grupos host e matrix). RMON ATM contém as funcionalidades host e matrix baseada nas versões encontradas na RMON-2, a qual contém algumas atualizações da RMON-1.
- Relatórios de estatísticas TopN por host para cada endereço MAC válido, descoberto em cada segmento monitorado (grupo hostTopN). RMON ATM não implementa uma função hostTopN.
- Ponto simples de monitoração, que registra eventos de *logging* ou de notificações para qualquer instância da MIB. RMON ATM utiliza alarmes e grupos de eventos do RMON-1 sem modificações. Estes grupos do RMON-1 devem der implementados nas pesquisas do RMON ATM se um mecanismo simples é necessário.

Algumas estatísticas oferecidas são:

- Número de células enviadas e recebidas com sucesso pelo *host* e por interface;
- Número de pedidos de conexões com sucesso;
- Número de conexões efetuadas com sucesso;
- Totais de conexões por interface.

4.3. Requisitos Funcionais da MIB RMON ATM

Aplicar RMON para redes ATM requerirá algumas novas mudanças e novas funcionalidades. Problemas especiais, tais como: altas velocidades, questões entre "células versus frames", e a natureza orientada a conexão do ATM, necessitam soluções especiais da MIB a fim de implementar RMON para redes ATM.

O SNMP foi concebido para ser usado com *frames* que trafegam em redes de baixas velocidades. O problema referente a análise dos *frames,* reside no fato do overhead que será imposto por esta análise.

O agente RMON ATM pode ser implementado nos seguintes equipamentos:

- Em cada porta do switch ATM;
- Em um hardware *stand-alone,* que deverá ficar entre dois switches.

4.4. Definições dos Grupos RMON ATM

A MIB está baseada em parte na MIB RMON-2. Contém quatro grupos [TAF97]:

- ***portSelect*** - Este grupo contém definições de portas, que serão usadas pelos grupos *Stats, Host e Matrix.*

- ***atmStats*** - Contém informações sobre totais de tráfego, associado a uma ou mais interfaces. Este grupo é composto por duas tabelas: *atmStatsControlTable* e a *atmStatsTable*. À seguir, será descrito o conjunto de objetos que compõe uma linha da tabela *atmStatsTable*:
    - ✓ *CreateTime* : informa a quanto tempo esta entrada da tabela foi criada.
    - ✓ *Cells* : número total de células sem erro, detectadas de todas as conexões virtuais nesta porta.
    - ✓ *CellsRollovers* : número de vezes que o conteúdo do objeto *Cells*, retornou a zero, ou seja, chegou a seu limite e foi reinicializado.
    - ✓ *HCCells* : é a capacidade máxima do objeto *Cells*.
    - ✓ *NumCallAttempts* : informa o número de tentativas para estabelecer conexões de SVC detectadas feitas, associadas a uma ou mais portas do equipamento gerenciado.
    - ✓ *NumCalls* : informa a quantidade de *calls* estabelecidos com sucesso, detectados para a porta do equipamento gerenciado.
    - ✓ *ConnTime* : número total de segundos de todos os NumCallsAttempts.
- ***atmHost*** - Contém contadores sobre o tráfego enviado por cada endereço ATM pelo agente RMON ATM. Os seus objetos estão descritos abaixo :
    - ✓ *HostAddress* : endereço ATM de 20 bytes, referentes a um equipamento.
    - ✓ *SClass* : informa o tipo do QoS para esta linha da tabela. Os valores podem ser 1 para *QoS* do tipo CBR e VBR e 2 para QoS do tipo ABR e UBR.
    - ✓ *CreateTime* : informa a quanto tempo esta entrada da tabela para o *host* ATM está ativa.
    - ✓ *InCells/OutCells* : número de células sem erros recebidas e enviadas das conexões virtuais, em que o *host* referenciado por esta linha da tabela é identificado como origem e destino.
    - ✓ *InCellsRollover/OutCellsRollover* : informa o número de vezes que os objetos *InCells* e *OutCells* retornaram seus contadores a zero.

- ✓ *InHCCells/OutHCCells* : informa a capacidade máxima dos objetos *InCells* e *OutCells.*
- ✓ *InNumCallAttempts/OutNumCallAttempts* : informa o número de tentativas chamadas nas conexões ATM, associadas com o *host* referenciado por esta linha.
- ✓ *InNumCalls/OutNumCalls* : número de chamadas estabelecidas com sucesso, detectadas em conexões ATM, associadas com esta linha da tabela, sendo para *InNumCall* o *host* é a origem da chamada e para o *OutNumCalls* o *host* é o destino da chamada.
- ✓ *InConnTime/OutConnTime* : informa o tempo total em segundos de todas as *In* e *Out CallsAttempts.*

- ***atmMatrix*** - Tem informações sobre o total de tráfego enviado entre pares de *hosts* ATM, descobertos pelo agente RMON. Este grupo é formado pelas tabelas *atmMatrixControlTable, atmMatrixSDTable, atmMatrixDSTable, atmMatrixTopNControlTable e atmMatrixTopNTable.* As tabelas *atmMatrixSDTable* e *atmMatrixDSTable,* contém os mesmos objetos, sendo diferentes apenas na ordem da apresentação. Uma representa dados estatísticos sobre o tráfego enviado, de um *host* X para Y e a outra de um *host* Y para X. Os objetos destas tabelas são descritos à seguir :
  - ✓ *Srcaddress* : é o endereço ATM do *host* de origem.
  - ✓ *Dstaddress* : informa o endereço ATM do *host* de destino.
  - ✓ *CreateTime* : informa em que hora esta linha da tabela foi criada.
  - ✓ *SClass* : idem ao objeto *SClass* do grupo *hosts.*
  - ✓ *Cells* : número de células sem erro, transmitidas e recebidas pelos *hosts,* identificados pelo *srcaddress e dstaddress.*
  - ✓ *CellsRollovers* : indica o número de vezes que o objeto Cells retornou seu contador a zero.
  - ✓ *NumCallAttempts* : número de tentativas para estabelecer conexões de SVC detectadas, feitas pelo equipamento, indicado pelo endereço *Srcaddress,* com destino o equipamento de endereço *Dstaddress.*

- ✓ *NumCalls* : número de pedidos de conexões de SVC detectadas com sucesso, feitas pelo equipamento indicado pelo endereço *Srcaddress* com destino o equipamento de endereço *Dstaddress.*
- ✓ *ConnTime* : indica o tempo em segundos de todos os *NumCallAttempts* efetuados.

- A tabela ***atmMatrixTopNTable*** contém objetos referentes a pares de *hosts,* que mais geraram tráfego no período analisado pelo agente. Esta tabela contém quatro objetos, descritos abaixo :
  - ✓ *TopNSrcAddress* : contém o endereço ATM do equipamento responsável pela origem do tráfego.
  - ✓ *TopNDstAddress* : contém o endereço ATM do equipamento de destino.
  - ✓ *TopNRate* : contém o valor referente ao intervalo de monitoramento de um dos objetos, à seguir : *Cells, NumCallattempts, NumCalls, ConnTime.*
  - ✓ *TopNReverseRate* : contém o valor referente ao inverso do objeto anterior. Exemplo : se o *TopNRate* indica o total de células transmitidas do equipamento *SrcAddress* para *DstAddress,* o *TopNReverseRate* indica o total de células transmitidas de *DstAddress* para *SrcAddress.*

5. AGENTES MÓVEIS

Agentes móveis são módulos de software capaz de interagir com um dispositivo de rede e recolher informações importantes para o gerenciamento do mesmo. Um gerente pode, então, solicitar aos agentes tais informações de gerenciamento, através de um protocolo de gerência.

Para cada dispositivo gerenciado, deve obrigatoriamente, existir um agente que reporte um conjunto de informações do dispositivo. Este agente está, então associado ao dispositivo que captura informações.

5.1. Definição de Agentes

No gerenciamento devemos definir alguns princípios básicos, que representam a base de estudos que englobam os agentes móveis, o sistema de gerenciamento é basicamente composto por:

- Recurso representado pelos próprios equipamentos ativos na rede.
- Programas gerente e agente, agindo como chefe e subordinado em uma determinada operação especifica.
- Operação de gerenciamento, são as respostas básicas para se ter informações mínimas de uma atividade que se deseja ter controle.
- Eventos representando as mudanças de comportamento de um determinado objeto.
- Notificação, indica as mensagens de representações, que alguma ação ocorreu proveniente de um determinado evento.
- Objeto gerenciado é o equipamento propriamente dito, que se deseja gerenciar, base de informações de gerenciamento, representa os resultados colhidos dos diversos agentes gerenciados, de forma que os dados fiquem armazenados em algo semelhante a um banco de

informações para serem analisados e através desta análise encontrar formas mais versáteis de se controlar uma estrutura de rede.

Agentes móveis são módulos de software capazes de interagir com um dispositivo de rede e recolher informações importantes para o gerenciamento do mesmo. Um gerente pode, então solicitar aos agentes tais informações de gerenciamento, através de um protocolo de gerência.

Cada agente móvel possui um conjunto de modelos que definem seu comportamento. Existe um modelo para o *ciclo de vida* do agente, um modelo *computacional*, um modelo de *segurança* e um modelo de *comunicação* [BIE98]. O ciclo de vida de um agente, define como o mesmo se comportará durante sua existência, em que circunstâncias haverá clonagem do agente, quando uma instância deve ser excluída, etc... O modelo de comunicação não define um protocolo, como o SNMP, por exemplo, mas sim a forma como os agentes se comunicam entre si e com o ambiente. Além dos modelos citados, o modelo mais importante de um agente móvel é o de *locomoção,* que define como um agente move-se na rede. De acordo com este modelo, um agente é capaz de decidir, por exemplo, qual o próximo nodo de uma rede a ser visitado. Se isso já ocorreu que outra ação de locomoção deve ser tomada.

Os modelos que definem o agente devem interagir entre si para formar o comportamento do agente móvel. Por exemplo, se todos os nodos de uma rede foram visitados, será que o agente deve ser excluído? Uma relação entre o modelo de locomoção e do ciclo de vida do agente responde esta questão. Qual o próximo nodo a ser visitado, tendo em vista que o agente deve procurar por dispositivos com mau funcionamento? Uma relação entre o modelo de locomoção e o modelo computacional resolve esta outra questão [COS99].

As capacidades de aprender e cooperar dos agentes móveis, citadas anteriormente, tem impacto direto no comportamento dos mesmos. Isso significa que um agente é capaz de analisar os dados coletados e com isso tomar alguma decisão, mas principalmente registrar essa ação internamente e poder assim agilizar futuras

operações. Isto é, aprender. Além da interação com os dispositivos de rede, um agente deve interagir com outros agentes, para que as tarefas possam ser executadas cooperativamente. Esta interação permite ainda que um agente possa aprender, não apenas com as suas ações, mas também através do conhecimento das ações de outros agentes. Isto é: colaboração.

Sempre que um problema de performance for detectado, o agente do primeiro grupo que detectou o problema poderá informar esta situação a um agente do segundo grupo. O segundo grupo de agentes é então utilizado sempre que um problema for detectado. Os agentes do primeiro grupo informam o problema existente. Ações de reparo são tomadas de acordo com a análise dos dados de performance. Se necessário, agentes especializados podem locomover-se até um dispositivo problemático e alterar os parâmetros de funcionamento necessário. A falha pode ocorrer, entretanto por uma determinada combinação de parâmetros de dispositivos diferentes. Os agentes responsáveis pelo reparo devem reconhecer esta situação e proceder com as alterações necessárias para a solução do problema.

Aparentemente, os agentes móveis possuem uma complexidade grande, por realizarem tarefas complexas, como as citadas anteriormente. Entretanto, isso não necessariamente é verdade. Cada agente de um sistema pode ser muito simples, executando um conjunto de tarefas bem limitado. Por outro lado, o comportamento resultante da interação entre estes agentes simples é capaz de solucionar problemas complexos de uma rede. Isto é, um agente móvel sozinho é muito simples, mas o comportamento resultante de um grupo destes agentes é complexo. Isso facilita a tarefa de criação e manutenção dos agentes por parte do usuário, sem que se perca com isso capacidade de resolução dos problemas.

Por fim uma capacidade importante dos agentes móveis é sua comunicação com uma base de gerenciamento. A base de gerenciamento é um computador principal que substitui agora a estação de gerenciamento. Esta base é ponto inicial do gerenciamento e é ela a responsável pela colocação inicial de agentes móveis em uma rede. Como os agentes são autônomos, após criados estes têm vida própria e são capazes de executar

tarefas apenas com suas próprias ferramentas. Tarefas mais simples, sequer precisam ser reportadas à base de gerenciamento. Por exemplo, quando um agente móvel detecta uma interface que se encontra desativada, este pode ativar a interface problemática sem avisar à base. Porém, existem outras tarefas que precisam ser reportadas ao gerente da rede, que por sua vez interage através da base de gerenciamento. Para estas tarefas, os agentes móveis devem ser capazes de se comunicar com à base, enviando resultados de suas tarefas, relatórios de atividades e *logs* de execuções. Com estas informações, a base de gerenciamento pode informar ao usuário o estado atual da rede gerenciada.

Resumindo, um agente móvel deve possuir as seguintes características, para executar tarefas de gerenciamento de uma rede:

- Ser autônomos;
- Possuir um tempo de vida determinado;
- Executar tarefas simples;
- Cooperar entre si;
- Possuir mobilidade;
- Possuir uma inteligência coletiva;
- Comunicar-se com à base de gerenciamento.

Os agentes convencionais de gerenciamento exportam uma visão do dispositivo gerenciado. Esta visão permite ao gerente da rede observar aspectos do dispositivo e alterar determinados parâmetros. Tal visão do dispositivo é implementada através de uma MIB (*Management Information Base*). Uma MIB é uma base de dados conceitual que informa ao gerente que parâmetros podem ser observados; ao agente que parâmetros devem ser exportados.

Nos padrões que definem o que é uma MIB, existem os conceitos de objetos e instâncias. Se compararmos estes com a orientação a objetos atuais, teremos uma relação interessante. Cada MIB define um conjunto de classes de objetos. Cada agente exporta objetos destas classes. Se uma classe de objetos fizer parte de uma tabela da MIB, então o agente pode exportar mais de um objeto para aquela classe. Se a classe

não fizer parte de uma tabela, então existirá apenas uma instância da classe (objeto) no agente.

Todas as classes da MIB possuem apenas dois métodos: *get(getnext)* e *set*. O métodos *get(getnext)* permite a obtenção do valor associado ao objeto. Já o método *set* permite a modificação de tal valor. Nem todos os objetos implementam estes dois métodos. Questões de segurança podem exigir que determinados objetos possuam apenas o método *get(getnext)*, enquanto restrições mais severas não permitem nem mesmo um *get*. Questões de segurança não envolvem apenas a disponibilização dos métodos, mas também a autenticação do gerente.

Um mapeamento destas classes para um esquema *Corba* ou *Java*, por exemplo, mostra-se muito simples. Entretanto, o número de classes a serem criadas será extremamente grande, consumindo muitos recursos na implementação de agentes móveis. Uma outra abordagem mais apropriada de mapeamento foi criada. Como as classes em uma MIB estão organizadas de forma hierárquica, no lugar de se associar uma classe da MIB a uma classe de agente móvel, pode-se associar um conjunto de classes da MIB a uma classe de um agente. Desta forma, uma classe no agente possuiria vários atributos, que corresponderiam às classes da MIB. A alteração desses atributos, na nova classe, é conseguida através de *get(getnext)* e *set*. Logo, existe uma maneira onde uma MIB pode ser adequadamente mapeada, para classes de objetos em um agente móvel.

5.2. Implementação dos Agentes

A implementação de agentes móveis segue o modelo de componentes de software, com a capacidade adicional de mobilidade. Logo, na implementação, obrigatoriamente deve-se levar em conta aspectos de orientação a objetos.

Um agente móvel é um módulo de software que apresenta pelo menos uma classe de objetos. Esta classe deve implementar pelo menos um serviço (método), e

deve haver pelo menos uma instância desta classe. Opcionalmente tal instância pode não existir, se considerarmos que o método exportado é um método de classe.

Outra característica importante é de que 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.

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.

A maior parte dos ambientes que suportam agentes móveis, são resultados de pesquisas no gerenciamento de rede através de *Corba*. Os agentes são criados através das estruturas *Corba,* interagem com plataformas de gerenciamento padrão, também através de *Corba*. Na interação com os dispositivos gerenciados o protocolo SNMP, é o mais extensivamente utilizado. Existem momentos onde o protocolo utilizado é proprietário do fabricante do dispositivo. Nestes casos, geralmente a comunicação é via portas seriais.

Outras pesquisas investigam a utilização de agentes móveis Java. A interação com a plataforma de gerenciamento requer algumas conversões entre Java e a API disponibilizada pelas plataformas. A interação entre Java e os dispositivos é facilitada também pelo SNMP, já que as últimas versões de Java já suportam tal protocolo [GOT97].

5.3. Plataformas de Agentes Móveis

A linguagem que melhor se adapta para este tipo de aplicação, é o JAVA, devido este possuir os requisitos propostos logo acima. Sob a plataforma JAVA podemos observar algumas plataformas internas, que dão suporte aos agentes móveis, onde podemos citar como principais as seguintes:

- *Aglets (IBM);*
- *Concordia (Mitsubishi);*
- *Voyager (ObjectSpace).*

O *Concordia* da *Mitsubishi,* apresenta mobilidade via definição de objetos da classe *Itinerary*, definindo o que deve ser executado em cada destino. Comunicação através de, definição de Eventos em um Gerente de Eventos, Mecanismo de *Colaboração* [CON00].

O *Voyager* da *ObjectSpace,* apresenta um mecanismo de mobilidade mais simples. Itinerários podem ser *Vector* (não há uma classe específica). Qualquer objeto pode se tornar um agente (uso de *Agent.of()*); objetos podem migrar dele para *applets*; a comunicação entre agentes é feita via chamada a métodos de objetos ou pode ser; chamadas a métodos do modo usual, não importando a localização do objeto; suporta CORBA/IIOP. [VOY00]

O Aglets da IBM será detalhado no próximo capítulo.

6. AGLETS

Dando uma ênfase maior na estrutura baseada em *aglets,* podemos dizer que este representa uma sigla significando *Applet,* móveis do inglês (agile applet), que representa um pequeno programa criado como se fosse um *applet,* mas com a novidade de ter características de um agente móvel, sendo as principais características deste: a capacidade de passagem de objeto, podendo interagir localmente com outros *aglets,* tomada de decisão de qual máquina seria melhor para se conectar e processar suas tarefas, a possibilidade de atuar de forma concorrente com outros *aglets,* localizados em outras máquinas.

6.1. A Plataforma

A forma mais didática de experimentar este tipo de agente móvel, é utilizar o servidor desenvolvido para esta finalidade, denominado TAHITI [TAH00], que faz uso de interface gráfica realizando monitoramento, controle e execução dos *aglets*, tendo como dificuldade o fato da plataforma ser proprietária, restringindo assim o desenvolvedor a este meio.

A comunicação dos *aglets* se faz com o uso de um protocolo chamado ATP (*Agent Transfer Protocol*), um protocolo independente de plataforma, atuando no nível de aplicação sob as normas do HTTP, define-se quatro métodos padrões [OSH98]:

- ***Dispatch*:** solicita a uma máquina que reconstrua um agente e ponha em operação.
- ***Retract:*** Solicita que o agente enviado volte e entre em operação.
- ***Fetch:*** pede ao receptor recuperar e enviar algo como uma classe.
- ***Message*:** enviar uma notificação a um agente.

Para se fazer uso deste protocolo, somente as classes que herdaram a classe Aglet podem se mover pela rede.

Os Aglets possuem métodos próprios, que lhes dão esta capacidade de se locomoverem, sendo eles [OSH98]:

- ***Creation*:** inicializa o aglet.
- ***Cloning*:** Cria uma cópia no mesmo contexto.
- ***Dispatching*:**determina que ele seja enviado de uma máquina a outra.
- ***Retraction*:** chama de alguma máquina, remove-o e executa na máquina que o chamou.
- ***Deactivation*:** remove o aglet temporariamente, podendo reativá-lo posteriormente.
- ***Disposal*:** ira terminar a execução e remover o aglet.
- ***Messaging*:** manipula as mensagens enviadas e recebidas entre os aglets.

Para se fazer uso deste protocolo, somente as classes que herdaram a classe Aglet podem se mover pela rede.

O fato mais importante para se definir uma estrutura de agente está no contexto da comunicação, nos primeiros projetos envolvendo *Applet* usando JAVA buscava-se a nível experimental, a criação básica de um gerente que agiria como um servidor onde diversos clientes se conectavam a este e estabeleciam uma comunicação, usando uma aplicação não tão complexa como *aglets* podemos exemplificar este fato e após estar bem enfatizado a sua forma de trabalho, teremos como base, entender a forma de ação de um agente móvel autônomo, em uma estrutura de rede de tamanho indeterminado, citaremos logo abaixo um exemplo, com fonte em JAVA que trabalha como uma *applet*.

O servidor será inicialmente instanciado em uma máquina que contenha a máquina virtual JAVA à partir deste momento, ele irá aguardar chamadas provenientes de clientes, que pelo apelido do *Host* irá localizar o servidor e se identificará com um nome de usuário, atuando com um canal e estabelecendo contato de um servidor com vários clientes, onde o servidor apenas escutará os clientes, neste caso não retornando dados agindo como um elemento de escuta da rede, o cliente neste nosso exemplo

enviará dados, que são digitados e serão ecoados na rede em forma de bytes onde ao chegar no servidor será novamente convertida em string. A estrutura de desenvolvimento desta aplicação explora a classe de rede disponibilizada pela *Sun*. Pode-se também realizar esta operação usando *socket* [OSH98].

7. IMPLEMENTAÇÃO

A Mobilidade em gerência de redes ATM requer estudos em grandes áreas da tecnologia da informação, arquitetura de redes, gerência de redes e sistemas distribuídos, que são intensamente integradas. Neste Capítulo de Implementação, são apresentadas todas as atividades realizadas, a instalação e configuração do *backbone* ATM, todo o processo de gerência e monitoração, as ferramentas utilizadas e finalizando com a implementação do agente e os resultados obtidos. A descrição desse processo baseado nos conceitos teóricos descritos nos capítulos anteriores.

7.1. Ambientes de Desenvolvimento

A configuração de um pequeno *backbone* ATM para o desenvolvimento, teste e implementação, utilizando como principal elemento ativo um *Switch Core* ATM 200BX da Marconi, e outros ativos com módulos de *uplinks* ATM, conforme a Fig. 8.

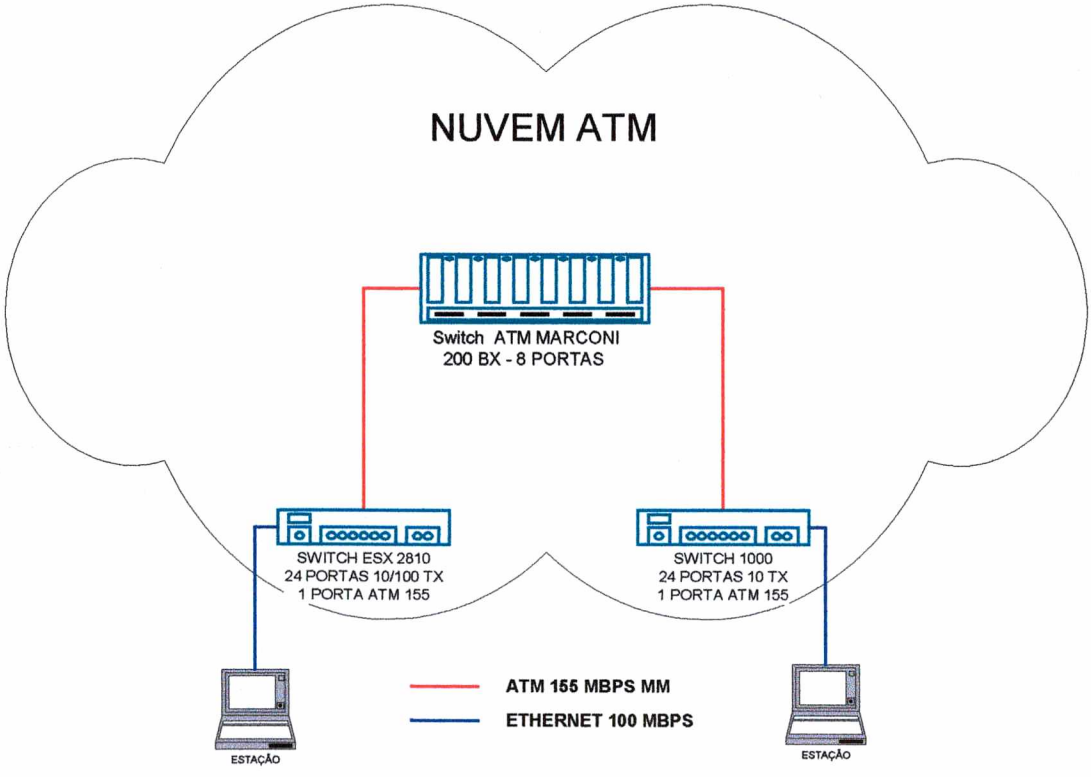


FIGURA 8 – Ambiente de Estudos e Desenvolvimento

Após o desenvolvimento do agente no ambiente de teste, a implementção em produção, utilizando na prática a maior rede ATM do Estado de Mato Grosso, que integra, através de uma rede ATM de alta performance e confiabilidade, todos os órgãos do Governo do Estado de Mato Grosso, constituindo assim a INFOVIA-MT, a qual permite o tráfego simultâneo de voz, dados e imagens.

Através do Centro de Processamento de Dados dos Estado de Mato Grosso – CEPROMAT, com a participação na elaboração e implantação do projeto INFOVIA-MT, que padronizou o protocolo ATM do *backbone* óptico, com a possibilidade de aprofundamento na tecnologia ATM. O *Core* do *Backbone* é descrito na Fig. 9.

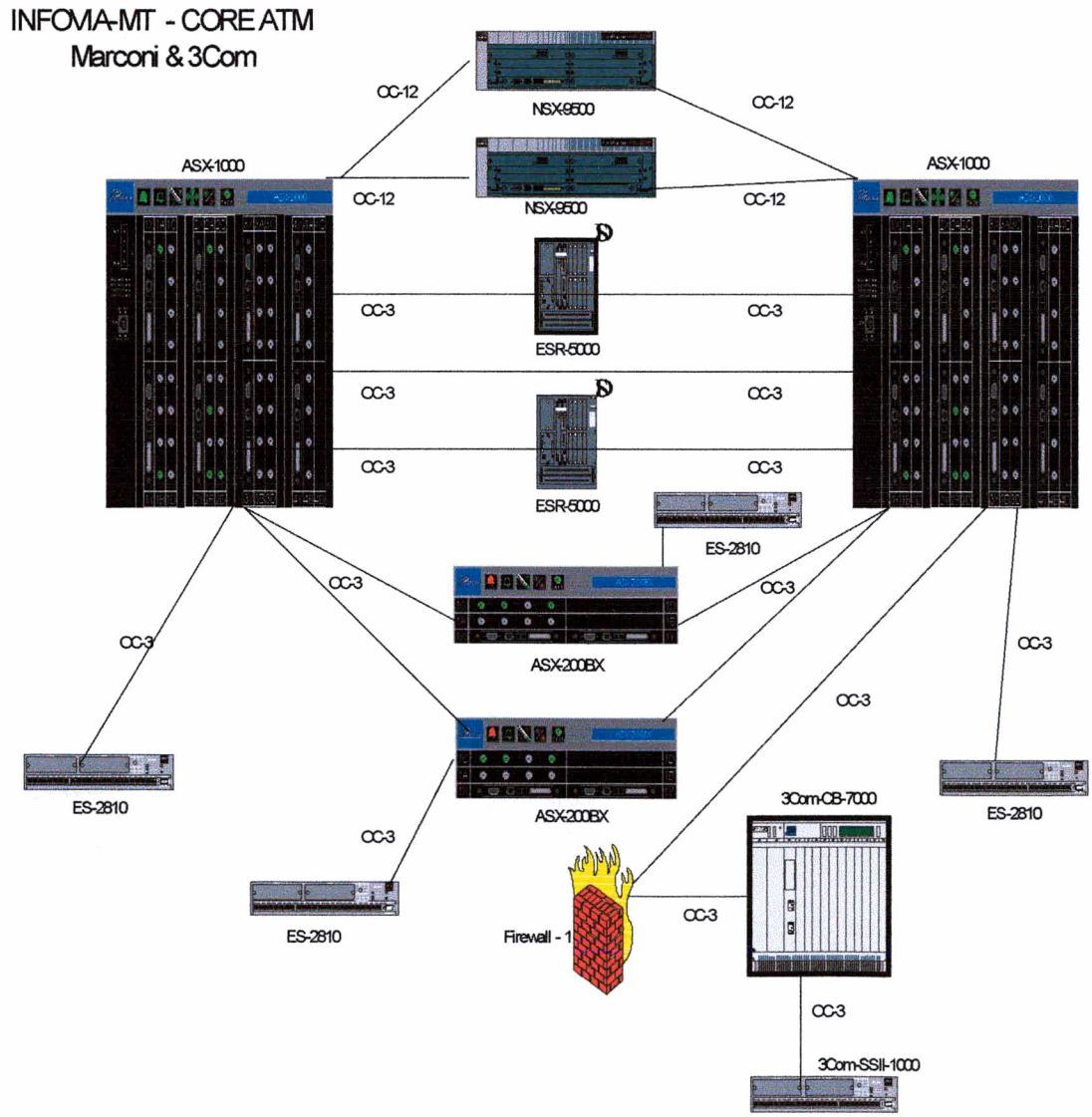


FIGURA 9 – Backbone ATM da INFOVIA-MT

7.2. Configuração do Backbone INFOVIA-MT

O *Backbone* é composto por diversos *switches* e roteadores que utilizam a tecnologia ATM, onde se destacam os *Switches Enterprise* modelo ASX1000 e AS200BX, os *Switches Routers* modelo ESR5000 e NSX9500, todos da Marconi e ainda o *Switch Enterprise* CoreBuilder 7000 da 3COM. Também faz parte da solução *Switches Workgroup* da ES2810 da Marconi e *Switch* 1000 da marca 3COM.

A INFOVIA-MT adotou o LANE, que permite rodar aplicações, desenvolvidas para redes legadas, sobre o ATM sem qualquer mudança nas aplicações. A estrutura do LANE é formada por quatro componentes: clientes LANE (LEC), servidores LANE (LES), barramento (BUS), e servidores de configuração (LECS). O serviço de LES do *Backbone* é realizado pelo *Switch Enterprise* ASX1000. Abaixo passamos a descrever os procedimentos para a configuração dos serviços de LECS, LES, BUS e LEC.

```
elan01.Address: c51111111111111111111111111111111111111
elan01.Accept:  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
elan02.Address: c51222222222222222222222222222222222222
elan02.Accept:  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
elan03.Address: c51333333333333333333333333333333333333
elan03.Accept:  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
elan04.Address: c51444444444444444444444444444444444444
elan04.Accept:  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
elan05.Address: c51555555555555555555555555555555555555
elan05.Accept:  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
elan06.Address: c51666666666666666666666666666666666666
elan06.Accept:  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
elan07.Address: c51777777777777777777777777777777777777
elan07.Accept:  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Match.Ordering: elan01,elan02,elan03,elan04,elan05,elan06,elan07
```

QUADRO 1 – Script de Criação do LECS

Definindo o LECS no ASX1000 através do *script*, conforme o quadro 1, usa-se o seguinte formato:

- ***elanname.address***: *Switch ID* (NSAP);
- ***elanname.accept*:** String de identificação do *switch;*
- ***match.ordering***: elanname1, elanname2, ..., elannamen.

Onde é especificado como:

- ***elanname*** – nome da ELAN;
- ***address*** – *switch ID* (NSAP), endereço do *switch* (no exemplo qualquer *switch* do CORE ATM responderá pela ELAN definida);
- ***accept*** – string de *accept* da ELAN (no exemplo string xxxxxxxxxxxx para todos os *switches* do CORE ATM);
- ***match.ordering*** – lista das ELAN's definidas na seqüência de entrada, para a sua construção.

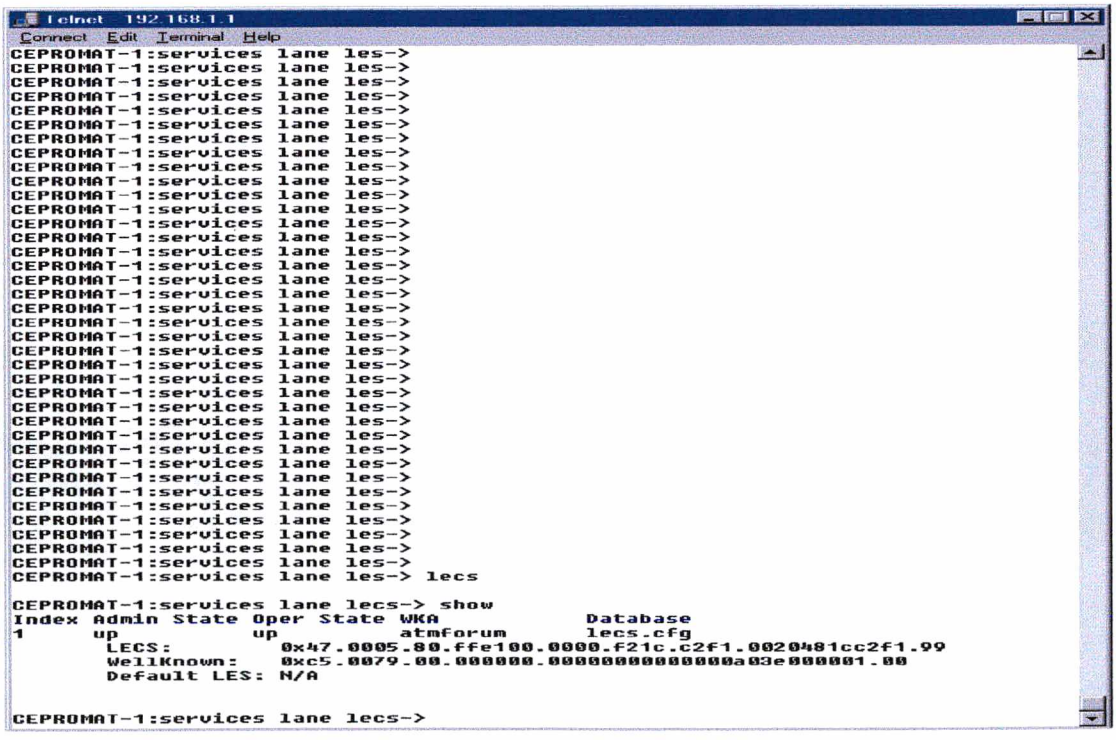


FIGURA 10 – Visualizando a configuração do serviço **LECS**

Na Fig. 10 mostra através do comando *show* a visualização do LAN *Emulation Configuration Server* (LECS), e através do comando *new*, os parâmetros de configuração do servidor, conforme a Fig 11.

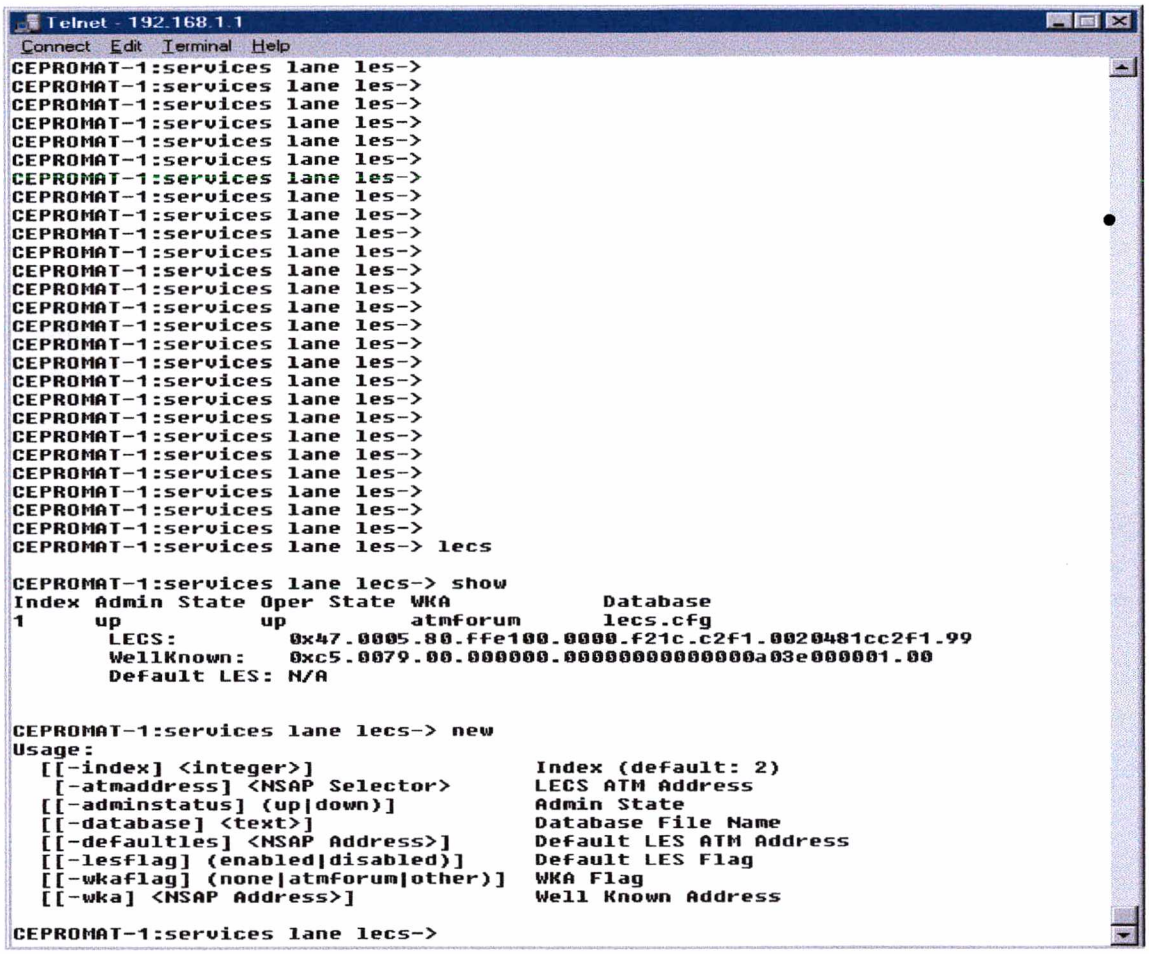


FIGURA 11 – Parâmetro do comando **NEW** para definição de um novo **LECS**

Definindo o LES e o BUS no *switch* ASX1000 através do comando **new** *index selectorbyte name.* A Fig. 12 mostra todos os parâmetros, onde:

- *index*: é o número identificador do LES (índice)
- *selectorbyte*: é o endereço ATM do LES
- *name*: é o nome da ELAN.

Quando se define o serviço de LES, o BUS é automaticamente criado usando o *Switch ID* (NSAP), de onde o serviço fora criado.

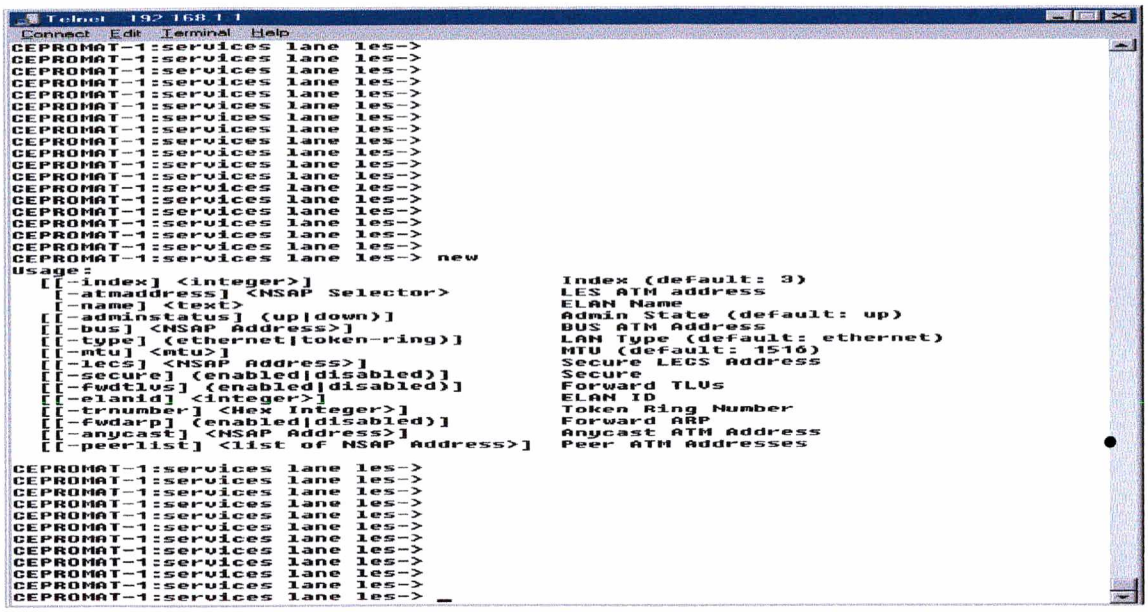


FIGURA 12 – Parâmetro do comando **NEW** para definição de um novo **LES**

Podemos visualizar na Fig. 13, através do comando *show*, a configuração do LES e BUS no ASX1000.

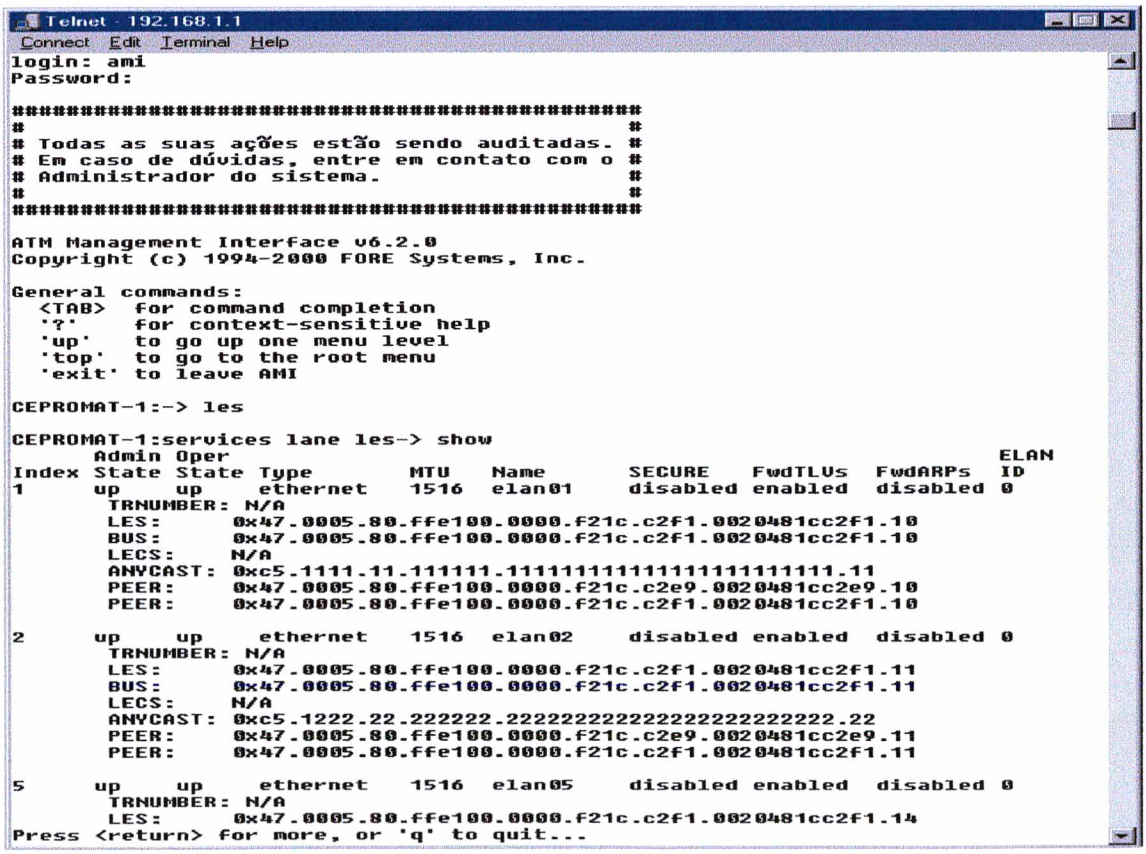


FIGURA 13 – Visualizando configuração do **LES**

Definindo uma ELAN no switch ASX1000 através do comando: **new** *index selectorbyte elanname status anycast* ***99999999*** *peerlist* "***999999999***", onde:

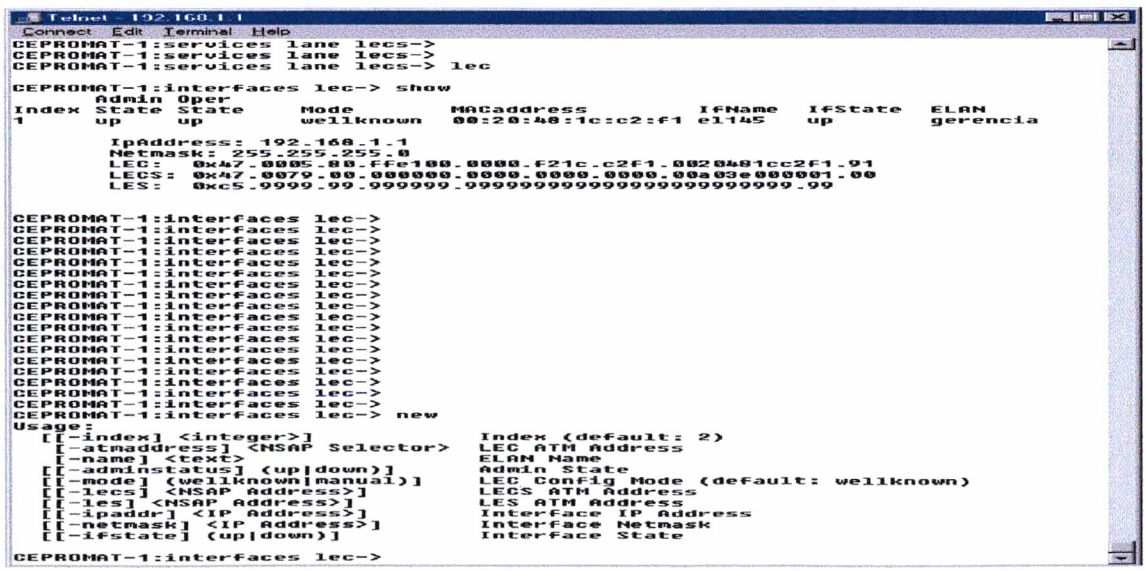
- ***index***: é o número identificador da ELAN (índice);
- ***selectorbyte***: é o endereço ATM do LES;
- ***elanname***: é o nome da ELAN definida;
- ***status***: *up / down* – status da ELAN definida;
- ***anycast***: *Switch* ID que responde pela ELAN criada, no exemplo qualquer *switch* do CORE responderá;
- ***peerlist***: *Switches* ID's para configuração do serviço de contingência (DLE).

Exemplo do comando, conforme o Quadro 2. Mostra todos os parâmetros de configuração de um novo LEC na Fig 14.

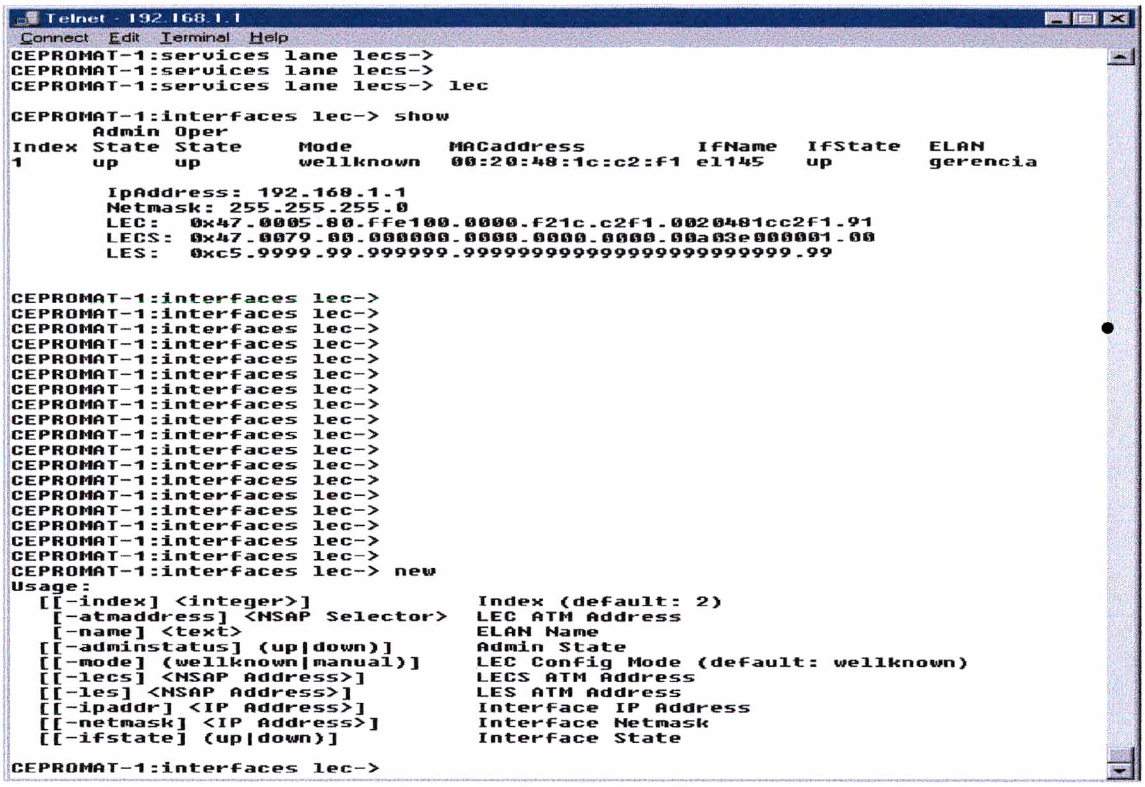
new 54 0x32 elan54 up -anycast c56888888888888888888888888888888888888888 -peerlist "47000580ffe1000000f21cc0300020481cc03032 47000580ffe1000000f21cbfbc0020481cbfbc32"

new 36 0x36 elan17 up -anycast c527777777777777777777777777777777777777 -peerlist "47000580ffe1000000f21cc0300020481cc03036 47000580ffe1000000f21cbfbc0020481cbfbc36"

QUADRO 2 – Exemplo de Comando para Criação de ELAN


Telnet 192.168.1.1
Connect Edit Terminal Help
CEPROMAT-1:services lane lecs->
CEPROMAT-1:services lane lecs->
CEPROMAT-1:services lane lecs-> lec
CEPROMAT-1:interfaces lec-> show
Index Admin State Oper State Mode MACaddress IfName IfState ELAN
1 up up wellknown 00:20:48:1c:c2:f1 e1145 up gerencia
IpAddress: 192.168.1.1
Netmask: 255.255.255.0
LEC: 0x47.0005.80.ffe100.0000.f21c.c2f1.0020481cc2f1.91
LECS: 0x47.0079.00.000000.0000.0000.0000.00a03e000001.00
LES: 0xc5.9999.99.999999.999999999999999999999999.99
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec-> new
Usage:
[[-index] <integer>] Index (default: 2)
[-atmaddress] <NSAP Selector> LEC ATM Address
[-name] <text> ELAN Name
[[-adminstatus] (up|down)] Admin State
[[-mode] (wellknown|manual)] LEC Config Mode (default: wellknown)
[[-lecs] <NSAP Address>] LECS ATM Address
[[-les] <NSAP Address>] LES ATM Address
[[-ipaddr] <IP Address>] Interface IP Address
[[-netmask] <IP Address>] Interface Netmask
[[-ifstate] (up|down)] Interface State
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->

A visualização do serviço LEC no *Switch* ASX1000 é mostrada na Fig 15 .



```
Telnet - 192.168.1.1
Connect  Edit  Terminal  Help
CEPROMAT-1:services lane lecs->
CEPROMAT-1:services lane lecs->
CEPROMAT-1:services lane lecs-> lec

CEPROMAT-1:interfaces lec-> show
      Admin Oper
Index State State     Mode          MACaddress          IfName   IfState   ELAN
1     up    up        wellknown     00:20:48:1c:c2:f1   el145    up        gerencia

       IpAddress: 192.168.1.1
       Netmask: 255.255.255.0
       LEC:   0x47.0005.80.ffe100.0000.f21c.c2f1.0020481cc2f1.91
       LECS:  0x47.0079.00.000000.0000.0000.0000.00a03e000001.00
       LES:   0xc5.9999.99.999999.999999999999999999999999.99


CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec->
CEPROMAT-1:interfaces lec-> new
Usage:
   [[-index] <integer>]                  Index (default: 2)
    [-atmaddress] <NSAP Selector>        LEC ATM Address
    [-name] <text>                       ELAN Name
   [[-adminstatus] (up|down)]            Admin State
   [[-mode] (wellknown|manual)]          LEC Config Mode (default: wellknown)
   [[-lecs] <NSAP Address>]              LECS ATM Address
   [[-les] <NSAP Address>]               LES ATM Address
   [[-ipaddr] <IP Address>]              Interface IP Address
   [[-netmask] <IP Address>]             Interface Netmask
   [[-ifstate] (up|down)]                Interface State

CEPROMAT-1:interfaces lec->
```

FIGURA 15 – Visualizando configuração do **LEC**

Finalizando as principais configurações do *backbone* INFOVIA-MT, na Tabela 1 mostra as Identificações dos *Switches,* e os NSAP que foram configurados.

Switch	Switch ID (NSAP)
CEPROMAT-1	47000580ffe1000000f21cc2f10020481cc2f1
CEPROMAT-2	47000580ffe1000000f21cc0300020481cc030
CEPROMAT-3	47000580ffe1000000f21cc2a20020481cc2a2
CEPROMAT-4	47000580ffe1000000f21cbf380020481cbf38
CEPROMAT-5	47000580ffe1000000f21cc2e90020481cc2e9
CEPROMAT-6	47000580ffe1000000f21cbfbc0020481cbfbc
CEPROMAT-7	47000580ffe1000000f21cc0410020481cc041
CEPROMAT-8	47000580ffe1000000f21cc0550020481cc055

TABELA 1 – NSAP dos Switches

7.3. Ferramentas de Gerência

No processo de gerência de rede, na estrutura de gerenciamento, suas áreas funcionais e na interface de gerência ATM, são utilizados geralmente *web browsers* e consoles para o gerenciamento dos equipamentos centrais e ferramentas de gerenciamento proprietárias, para os elementos ativos de borda.

O software *Fore Stack View*, foi usado para os elementos ativos de borda fabricado pela Marconi. Com este produto podemos gerenciar todos os recursos do equipamento, e temos como exemplo o *Switch* de borda ES-2810, com 24 (vinte e quatro) portas *Fast Ethernet* e 01 (um) *uplink* Atm OC-3 de 155 Mbps do tipo multimodo, conforme a Fig. 16.

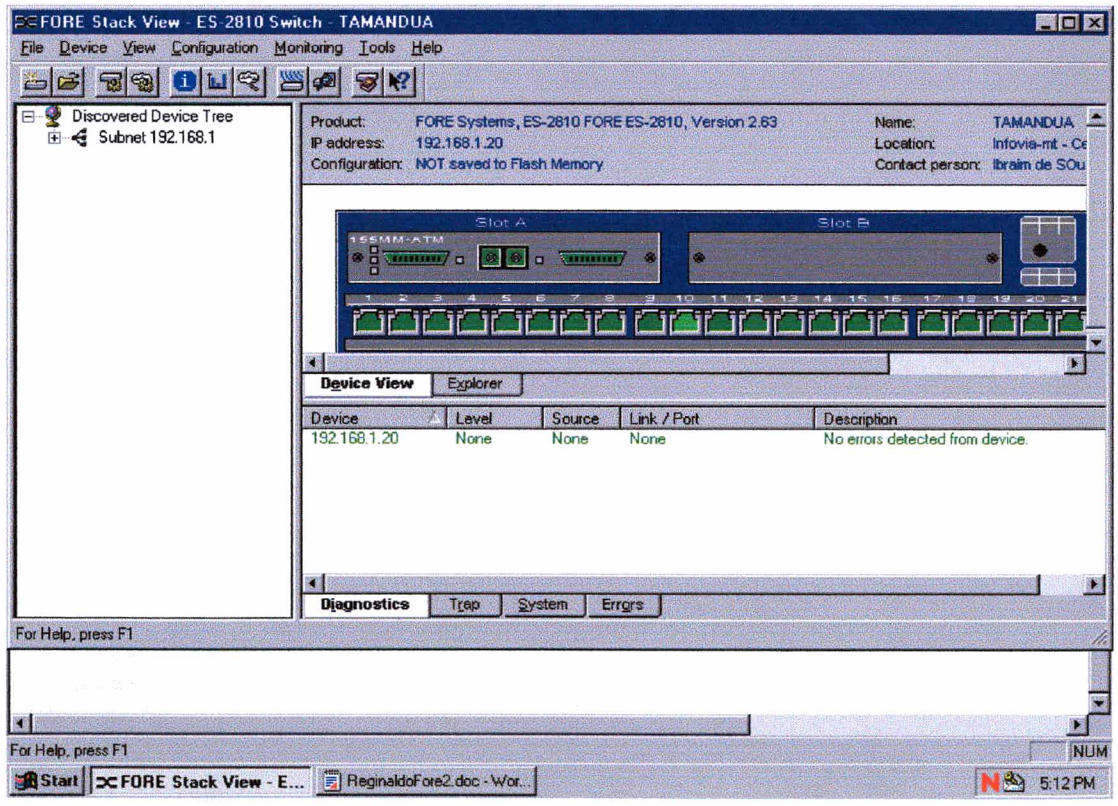


FIGURA 16 – Visualizando o ES2810 com o Fore Stack View

O Produto possui uma interface gráfica de fácil manuseio e compreensão, onde podemos selecionar o chassi do equipamento, observe a Fig. 17, para obter assim as configurações gerais:

- *Description;*
- *Contact Person;*
- *Name;*
- *Location;*
- *System up-time;*
- *Ip Address;*

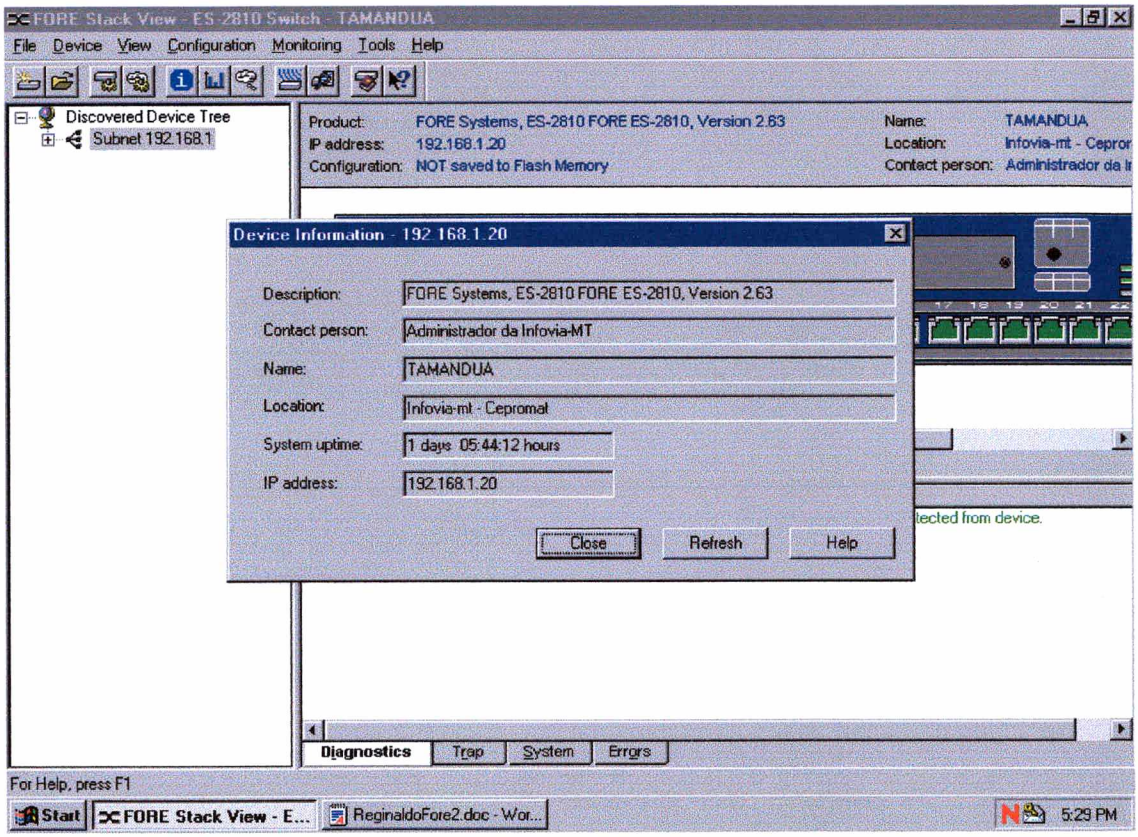


FIGURA 17 – Características Gerais do ES-2810

Além das características gerais do ES-2810, podemos selecionar recursos gerenciáveis das portas *fast ethernet* ou do módulo ATM, selecionando diretamente na porta para visualizar as informações pertinentes aquela porta ou módulo específico, observe a Fig. 18, que podem ser desde a VLAN que está associada, até os detalhes da porta, como tráfego de pacotes e estatísticas de RMON, e com isso podemos, por exemplo desabilitar uma porta do *switch* que esteja com grande taxa de erros ou perdas de *frames*.

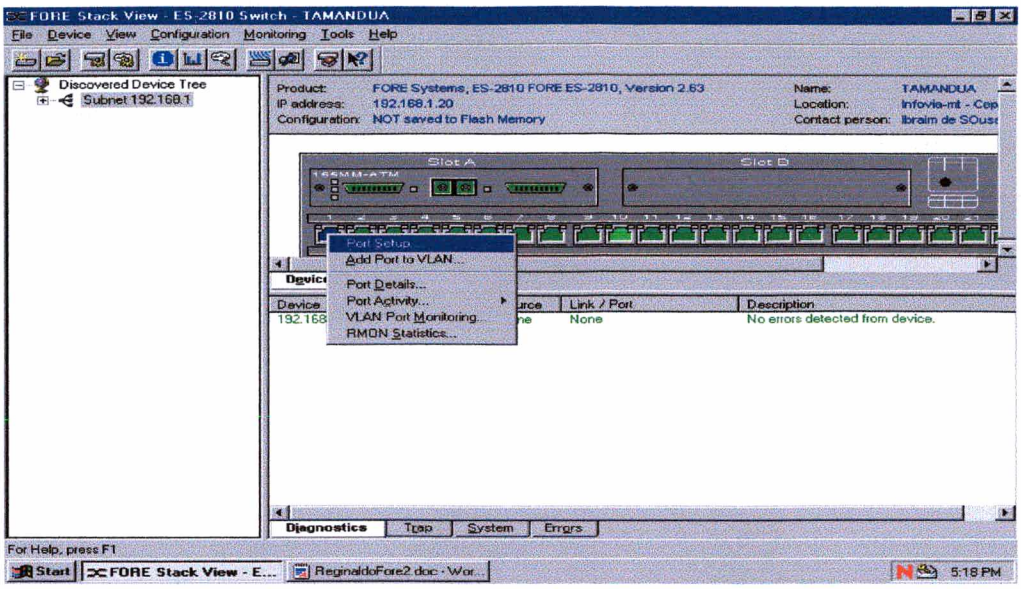


FIGURA 18 – Selecionando recurso de uma *porta fast ethernet* específica

Para os equipamentos de borda da 3COM, temos o software de gerência *Transcend,* ou assim como os demais elementos ativos, podemos acessar os recursos através da console de gerenciamento, onde é permitida a total configuração e monitoração, observe a Fig. 19, neste caso visualizando o tráfego de pacotes recebidos nas portas.

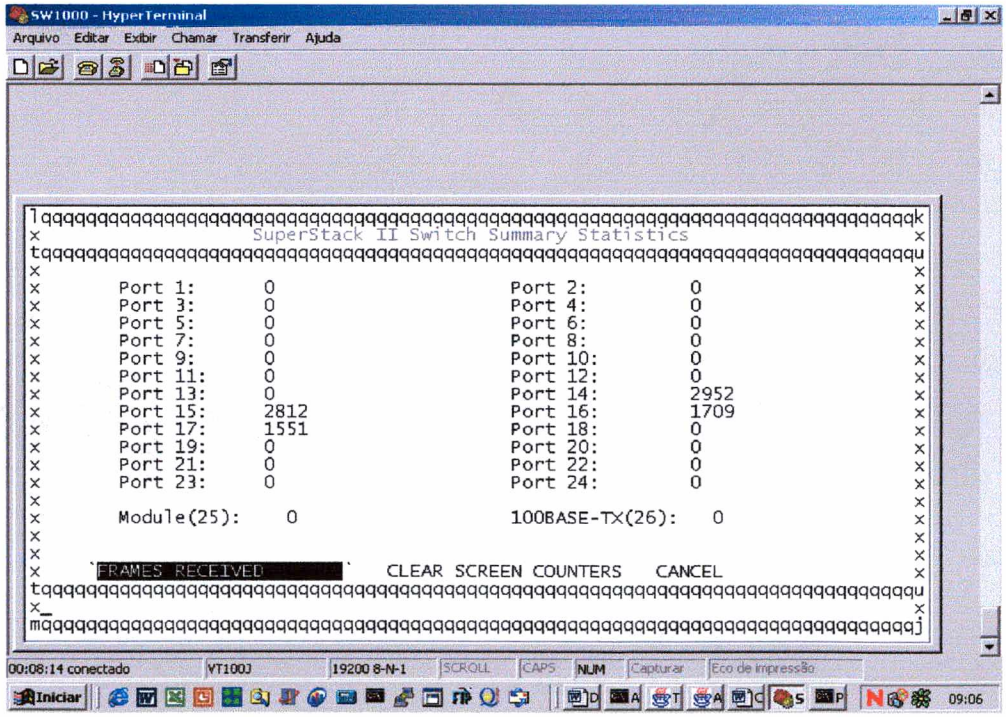


FIGURA 19 – Configuração do *Switch* 1000 da 3Com via Console de Gerenciamento

A interface de gerenciamento *web* está disponível na maioria dos ativos das novas tecnologias de mercado, esse recurso não é utilizado só na configuração, mas principalmente em todo o monitoramento do equipamento, pois veio para substituir os cansativos e antigos comandos de console via *prompt*, por uma interface de gerência amigável e utilizável, assim diminuindo o tempo gasto na configuração e aumentando a possibilidade de identificação de possíveis falhas. A Fig. 20, mostra a interface de gerência do NSX 95000 via *web browser*, onde os as cores dos *displays* nas portas representam o status da mesma, e também as opções de configurações de chassis, meio físico, softwares, políticas do *switch*, estatísticas e monitoramento. Temos nesse exemplo todas as informações das interfaces IP de roteamento *Layer 3*, onde são configurados os endereços IP's, sub-máscara para cada interface do *Siwtch Router* NSX9500.

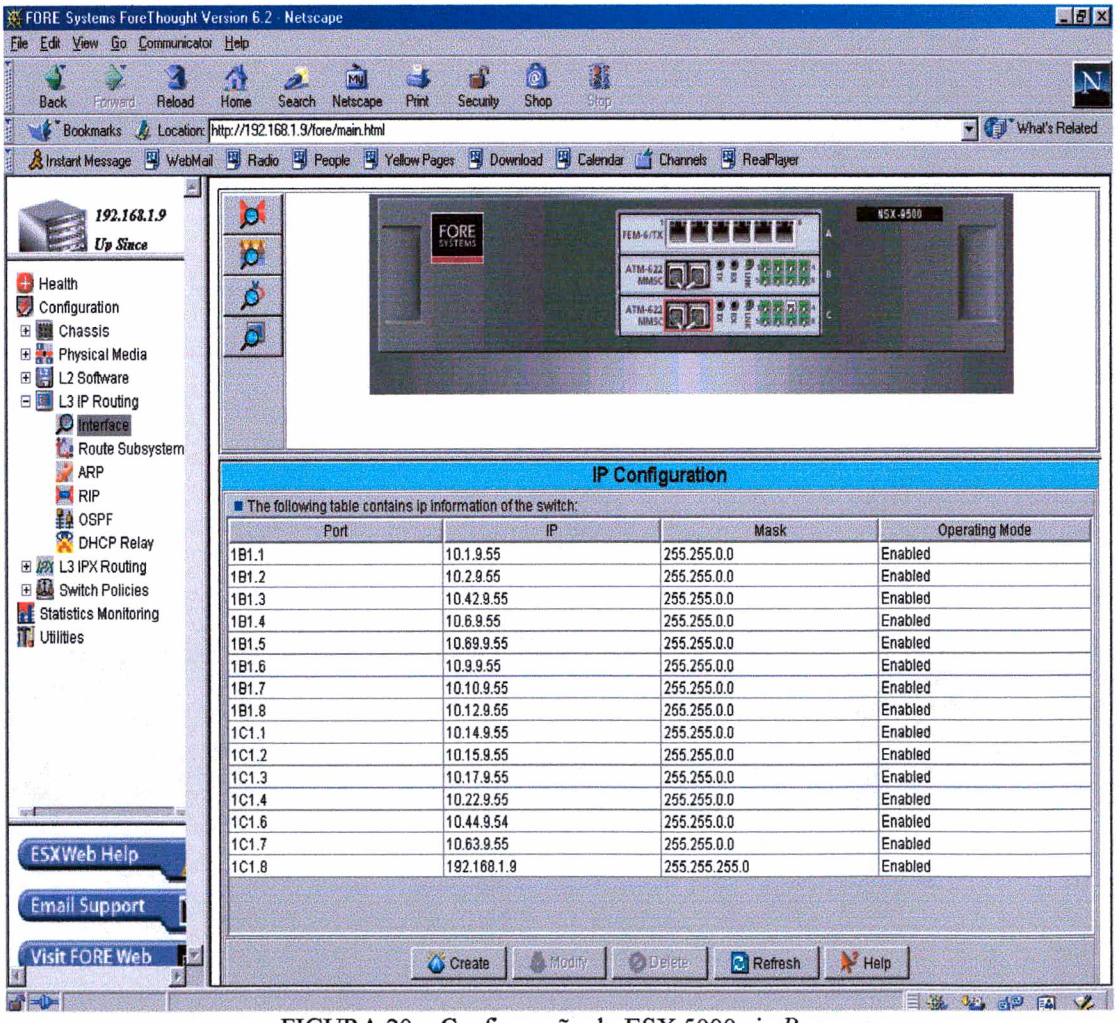


FIGURA 20 – Configuração do ESX 5000 via *Browser*

A Fig. 21 mostra a visualização do monitoramento das estatísticas do módulo ATM do *switch router* ESR 5000 via *web browser*, por exemplo, o número de células, (quinhentos e cinqüenta milhões, trezentos e oitenta e cinco mil, duzentas e vinte três células), do módulo ATM no instante da captura da informação.

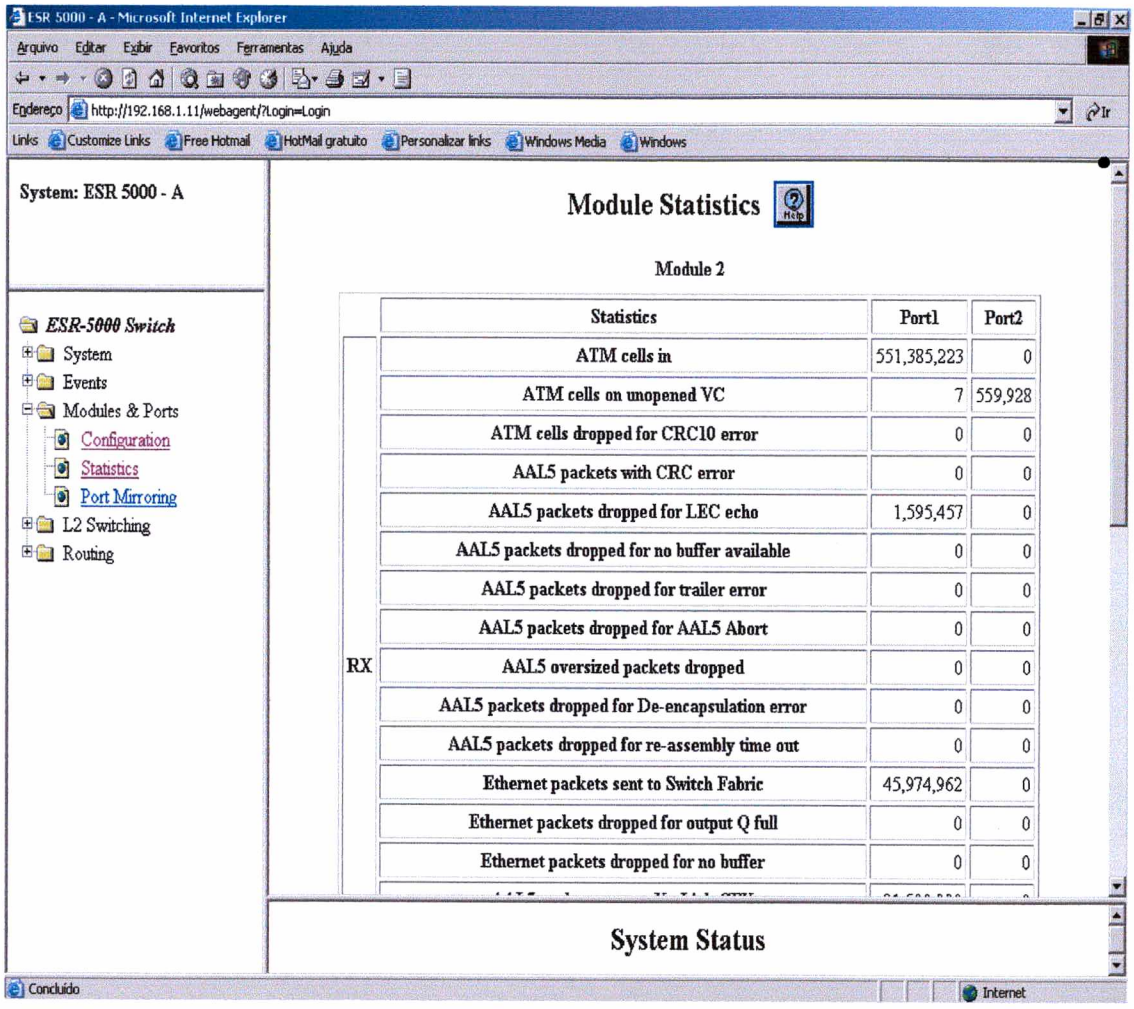


FIGURA 21 – Monitoramento do ESX 5000 via *Browser*

Todas as ferramentas de configuração e monitoração de um determinado equipamento, seja ela via *browser*, software proprietário ou console de gerenciamento, são necessárias para que o administrador possa garantir o funcionamento da rede com sucesso. Nesse trabalho foram utilizadas as ferramentas de gerência acima citadas, para verificar e garantir que o software proposto na implementação possa trazer dados reais do *backbone* ATM e para futuras comparações dos resultados por ambos obtidos.

7.4. A Implementação das MIBs

O monitoramento de redes ATM, sendo de performance, de protocolos e as MIBs para gerenciamento de redes ATM, foram visualizados através das ferramentas de gerenciamento.

O protocolo SNMP (descrito nos RFCs 1155, 1157, 1212, 1213) foi projetado como uma resposta aos problemas de comunicação entre diversos tipos de redes. A idéia básica por trás do SNMP era oferecer uma maneira facilmente implementável, e com baixo overhead para o gerenciamento de roteadores, servidores, *workstation* e outros recursos de redes heterogêneas.

A base de informação gerencial (MIB - *Management Information Base*) é o nome conceitual para a informação de gerenciamento, incluindo os objetos gerenciados e seus atributos, operações e notificações. Pode-se também considerar as informações para a configuração do sistema como também pertencentes à MIB. O *AdventNet MibBrowser,* é para visualizar as bases de MIBs dos elementos ativos, Fig. 22. Foram definidos quais os campos utilizar no gerenciamento. O *AdventNet* foi desenvolvido em Java e traz consigo as classes de API para implementação dos GETs e SETs.

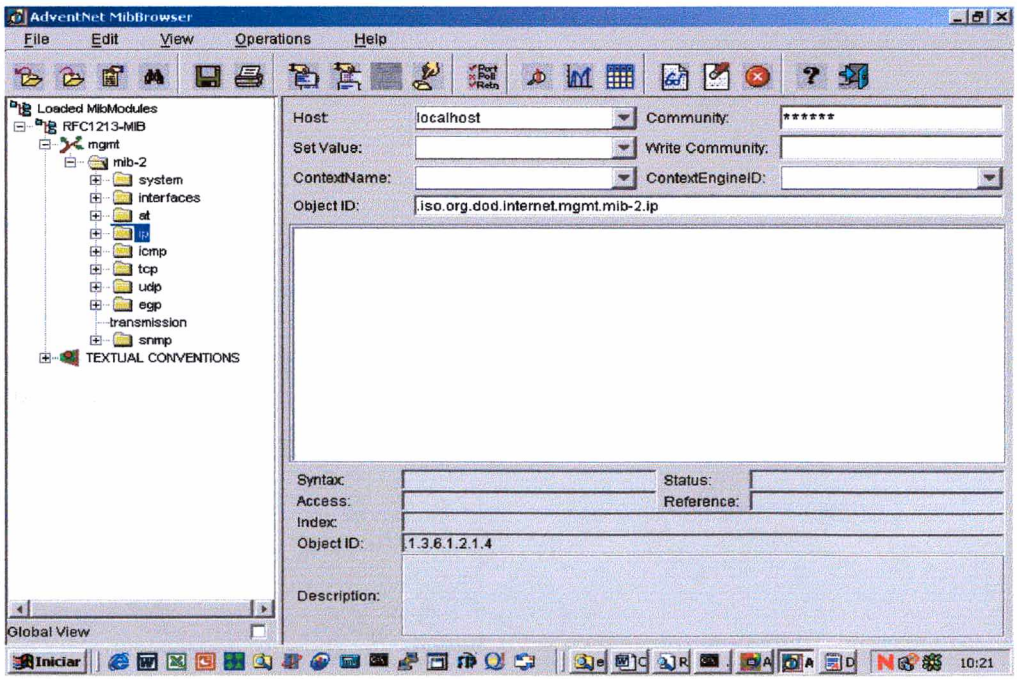


FIGURA 22 – Visualização da MIB do ES-2810 utilizando MibBrowser

A MIB é estruturalmente definida e normatizada em forma de árvore, onde cada nó da árvore representa uma informação gerenciável do equipamento. A identificação dessas informações é feita através do *Objet ID*, que representa a posição do nó na árvore. Na Fig. 23, temos a demonstração do *Object ID* ***.iso.org.dod.internet.mgmt.mib-2.ip.ipInReceives (.1.3.6.1.2.1.4.3),*** onde campo (nível) do endereço representa uma informação, neste caso os níveis ***.iso.org.dod.internet.mgmt (.1.3.6.1.2),*** representam as informações gerenciamento (*mgmt*) para RFC 1213 da MIB do *switch* ES-2810, o próximo nível é *.iso.org.dod.internet.mgmt.**mib-2** (.1.3.6.1.2.**1**),* que representa a *mib-2* do gerenciamento, logo após temos o nível *.iso.org.dod.internet.mgmt.mib-2.**ip** (.1.3.6.1.2.1.**4**),* que representa as informações sobre o protocolo *IP* dentro da *mib-2*, o último nível é *.iso.org.dod.internet.mgmt.mib-2.ip.**ipInReceives** (.1.3.6.1.2.1.4.**3**),* que representa a informação gerenciável dentro do protocolo *IP* da *mib-2*, sendo neste caso o **número total de datagramas recebidos pelas interfaces, incluindo os recebidos com error.** Além da visualização da árvore de gerenciamento e do *ObJect ID*, temos outras propriedades que podem ser visualizadas com a ferramenta: ***Syntax*** – Tipo da informação gerenciável; ***Access*** – Tipo de acesso (*Read-Only* ou *Read-Write*); ***Status*** – Status da informação; ***Description*** – Descrição sobre as informações.

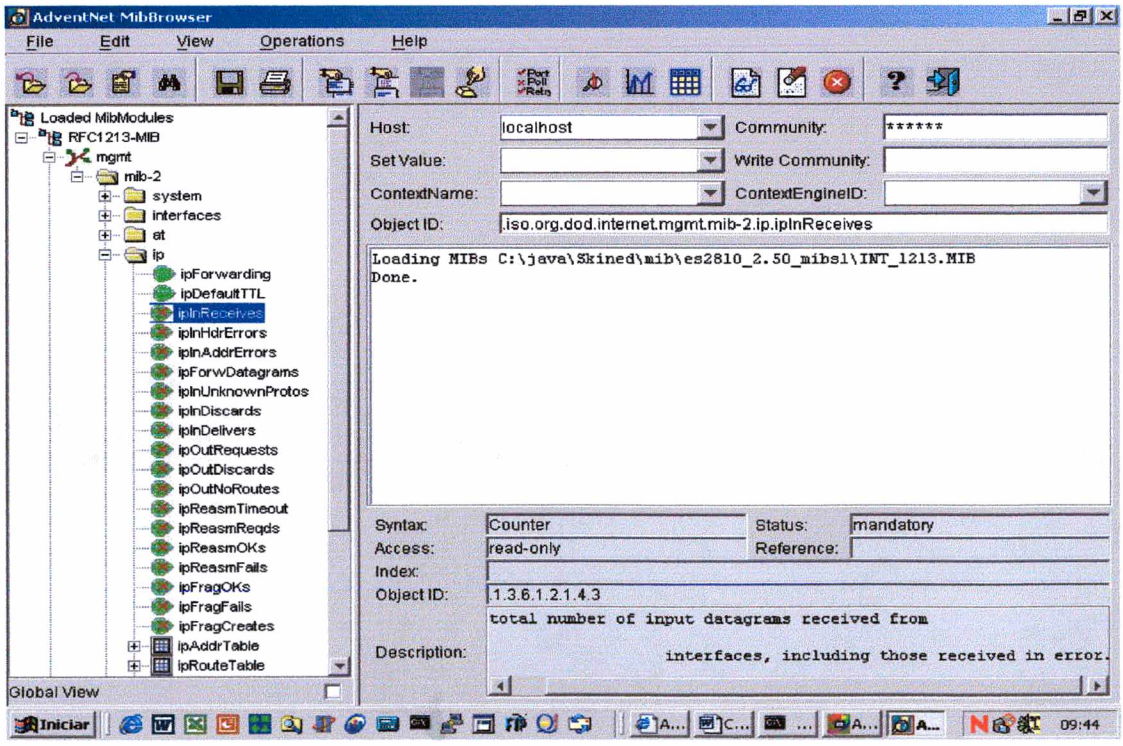


FIGURA 23 – Visualização do campo (.1.3.6.1.2.1.4.3) utilizando *MibBrowser*

Outro exemplo que podemos visualizar é *.iso.org.dod.internet.mgmt.mib-2.ip.ipRouteTable.ipRouteEntry.ipRouteDest (.1.3.6.1.2.1.4.21.1.1)*, que representa a rota *default*, que está localizada na árvore de gerenciamento dentro da tabela de roteamento *IP* para *mib-2*, Fig. 24.

À partir destas informações de gerenciamento é que foram definidos quais os campos da MIB dos elementos ativos que iremos utilizar para a construção do agente móvel proposto, que deverá coletar informações gerenciáveis do *backbone* ATM e também dos elementos ativos de borda, podendo até estabelecer alguns parâmetros para configurações do equipamento e até mesmo podendo chegar ao nível de bloqueio de portas ou ao nível de células com o desligamento de um PVC ATM.

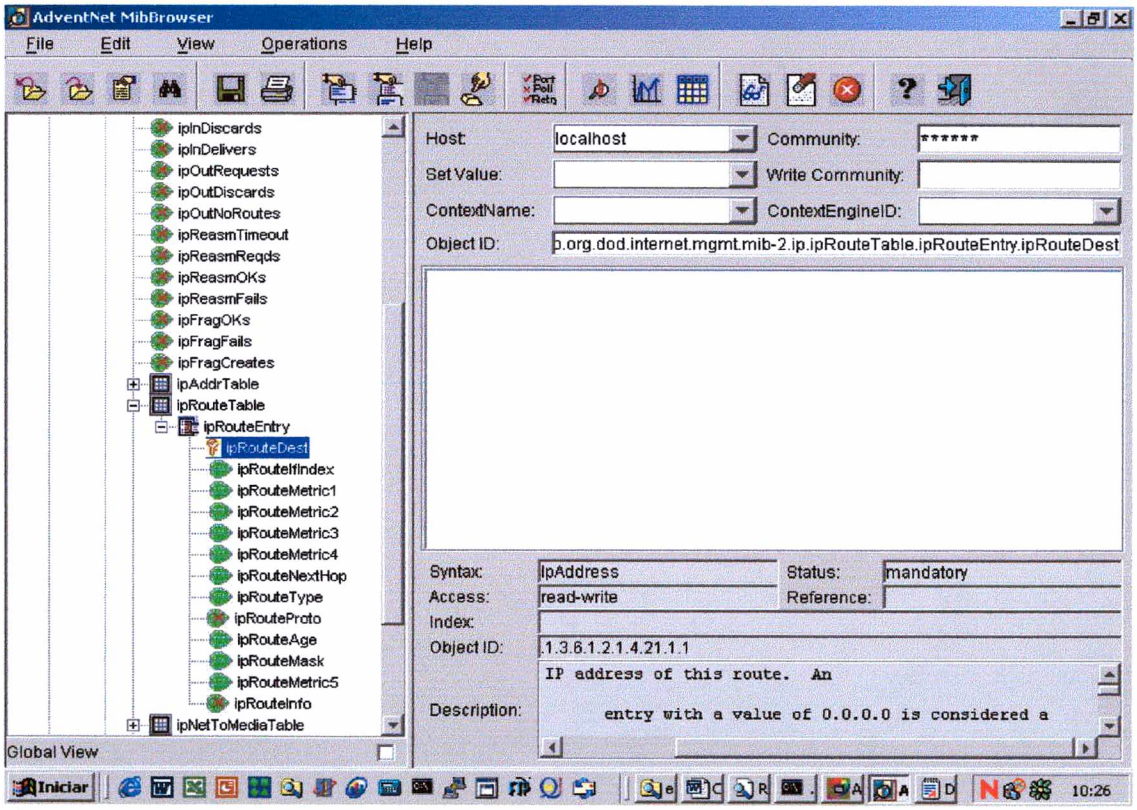


FIGURA 24 – Visualização do campo (.1.3.6.1.2.1.4.21.1.1) utilizando *MibBrowser*

O primeiro passo para a implementação foi fazer com que a estação de gerência possa obter informações do *switch*, para isso foi criado um protótipo, desenvolvido em *java* utilizando o JDK (*Java Development Kit*) versão 1.1.8, que é necessária para a plataforma AGLETS da IBM para mobilidade de agentes.

As classes API de gerência de rede do *AdventNetSNMPv3* é um pacote de desenvolvimento *java* que permite que se trabalhe diretamente com o protocolo SNMP. O código do Quadro 3, especifica as classes importadas para o desenvolvimento em *java*.

```
import com.adventnet.snmp.beans.*;
import com.adventnet.snmp.mibs.*;
import com.adventnet.snmp.snmp2.*;
import com.adventnet.snmp.snmp2.usm.*;
```

QUADRO 3 – Classes importadas do *AdventNet*

Inicialmente não havendo mobilidade, mas somente com a busca de informações na MIB do *switch*. Primeiramente, deve-se abrir uma sessão SNMP relacionando o equipamento remoto que está sendo monitorado, notifica-se com uma mensagem de erro, caso haja a ocorrência de alguma exceção. Constrói-se uma estrutura PDU (*Protocol Data Unit*) que é determinado com o comando *GET_REQ_MSG* que será executado na MIB.

Foram informados os valores do endereço IP do elemento ativo que será gerenciado e o identificador do objeto (*Objet ID*), campo da base de gerenciamento do *switch*, para a obtenção dos resultados de gerenciamento, através da sessão SNMP aberta, alguns métodos e propriedades utilizados das classses do *AdventNet* são:

- ***pdu.getCommunity*** – Comunidade de gerenciamento SNMP;
- ***pdu.getRemoteHost*** – Endereço IP do *host* remoto;
- ***session.getRemotePort*** – Porta de conexão do *host* remoto;
- ***session.getLocalPort*** – Porta de conexão do *host* local;
- ***session.getPacketBufferSize*** – Tamnaho do *buffer*.

O trecho de código em *java* de estabelecimento de uma sessão SNMP para a coleta das informações:

```
private void APIMib(String[] marg){
SnmpAPI api = new SnmpAPI();
SnmpPDU pdu = new SnmpPDU();
SnmpVar var = null;
SnmpOID oid = new SnmpOID(marg[1]);
api.start();
api.setDebug(true);
SnmpSession session = new SnmpSession(api);
        try     {       session.open();
        result.append("ABRINDO SESSION\n\n"); }
        catch(SnmpException e){ result.append("Erro abrindo o socket: "+e+'\n');
        System.exit(0); }
session.setPeername(marg[0]);
        try     {       var = session.get(oid);
result.append("RECEBENDO REQUISICAO\n\n");}
        catch(SnmpException e){result.append("Erro enviando requisição SNMP:
"+e+'\n');
        System.exit(0); }
pdu.setCommand(api.GET_REQ_MSG);
        pdu.addNull(oid);
        try     {       pdu = session.syncSend(pdu);}
                catch(SnmpException e){
                result.append("Erro recebendo requisição SNMP sincronizada:
"+e+'\n');      }
        try{    address = InetAddress.getLocalHost(); }
                catch(UnknownHostException w){ }
session.close();
api.close();
}
```

QUADRO 4 – Trecho de código em *java* para estabelecimento de uma sessão SNMP

A Fig. 25, mostra a execução em um *switch* utilizando os parâmetros, IP do *switch* e o *Object ID* da variável *counter* no formulário, e logo após selecionar o botão **Enviar Dados**, obtemos as informações, demonstrada em [DAN00].

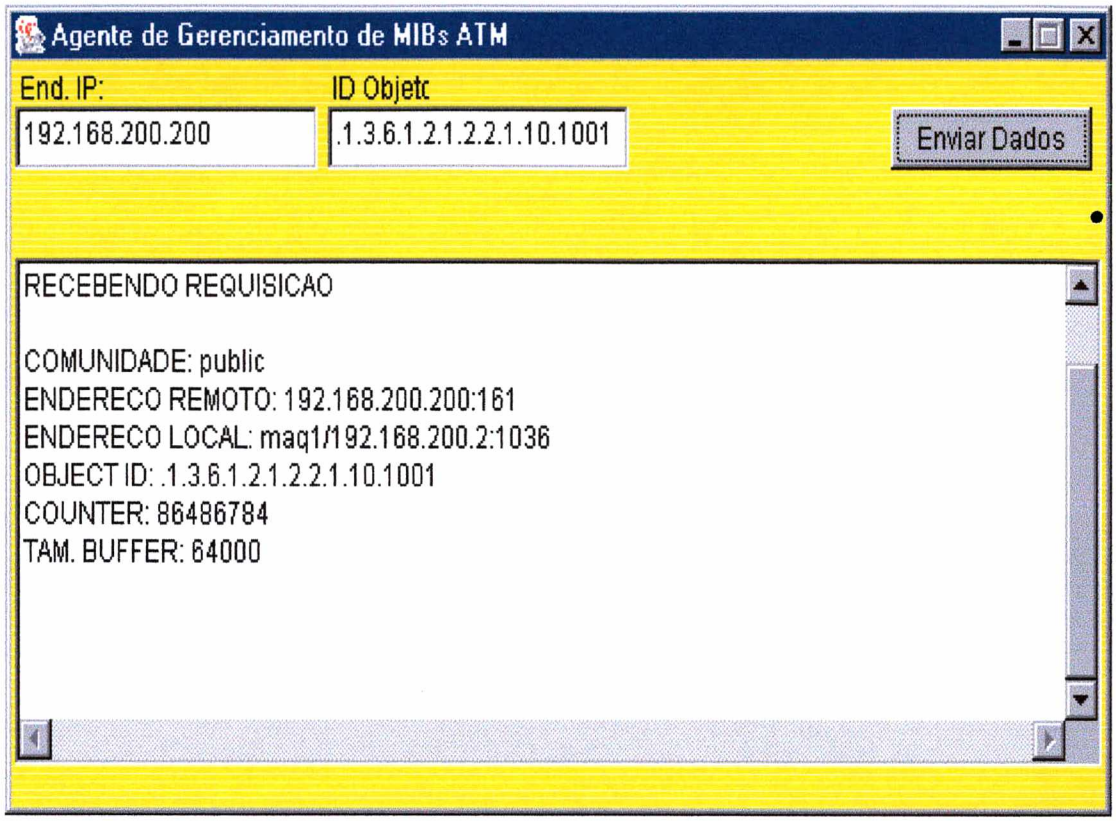


FIGURA 25 – Interface para Gerencia de Redes

Esse protótipo não possui a característica de mobilidade, foi implementado somente para teste e identificação das classes do *AdventNetSNMP*, e foi muito utilizado como ferramenta para a descoberta de quais seriam as informações gerenciais do *switch* monitoradas.

7.5. A Implementação do AGLET

O conceito de mobilidade é demonstrado através do TAHITI, Fig. 26, um servidor que utiliza uma interface gráfica para monitorar e controlar a execução dos agentes, conforme descrito em [DAN00]. Nessa interface são manipulados os métodos

(*creation, cloning, dispatching, retraction, deactivation, activation, disposal* e *messaging*) ou podemos também implementar internamente através das API's.

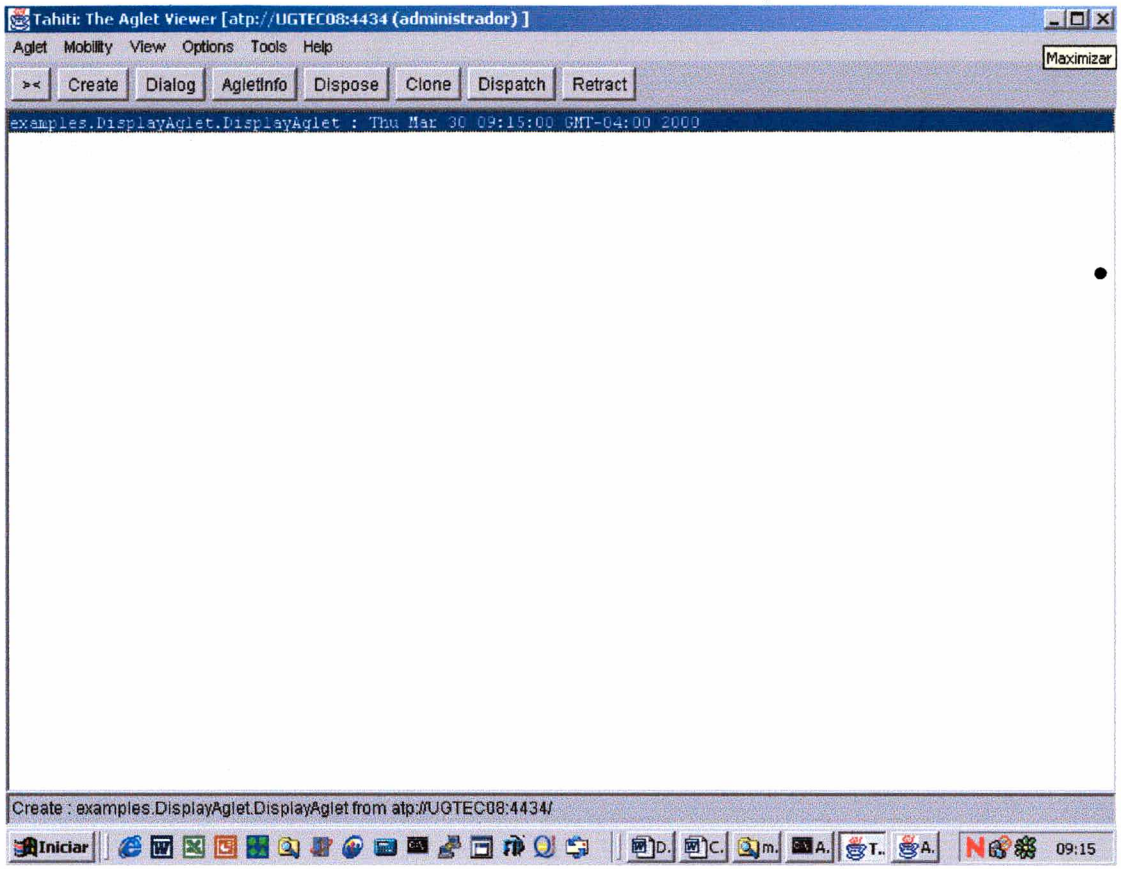


FIGURA 26 – Ambiente do TAHITI

Utilizando os AGLETS da IBM, para construir um agente móvel, são importadas algumas classes para a implementação, código do Quadro 4, que representa a importação para o desenvolvimento em *Java.*

```
import com.ibm.aglet.*;
import com.ibm.aglet.event.*;
import com.ibm.aglet.util.*;
```

QUADRO 5 – Classes importadas do AGLETS

Utilizando a mesma interface prototipada inicialmente para coletar as informações do *switch*, agora incluindo o conceito de mobilidade, temos um agente

coletor SNMP móvel, que é instanciado automaticamente em cada servidor que recebe o agente móvel. O código do Quadro 5, representa a criação automática da interface do agente coletor.

```
public class DisplayAglet extends Aglet{
        public void OnCreate(){
                Frame f = new frm_APIMib();
                f.show();       }
        public void run() { }
}
```

QUADRO 6 – Criação do AGLET

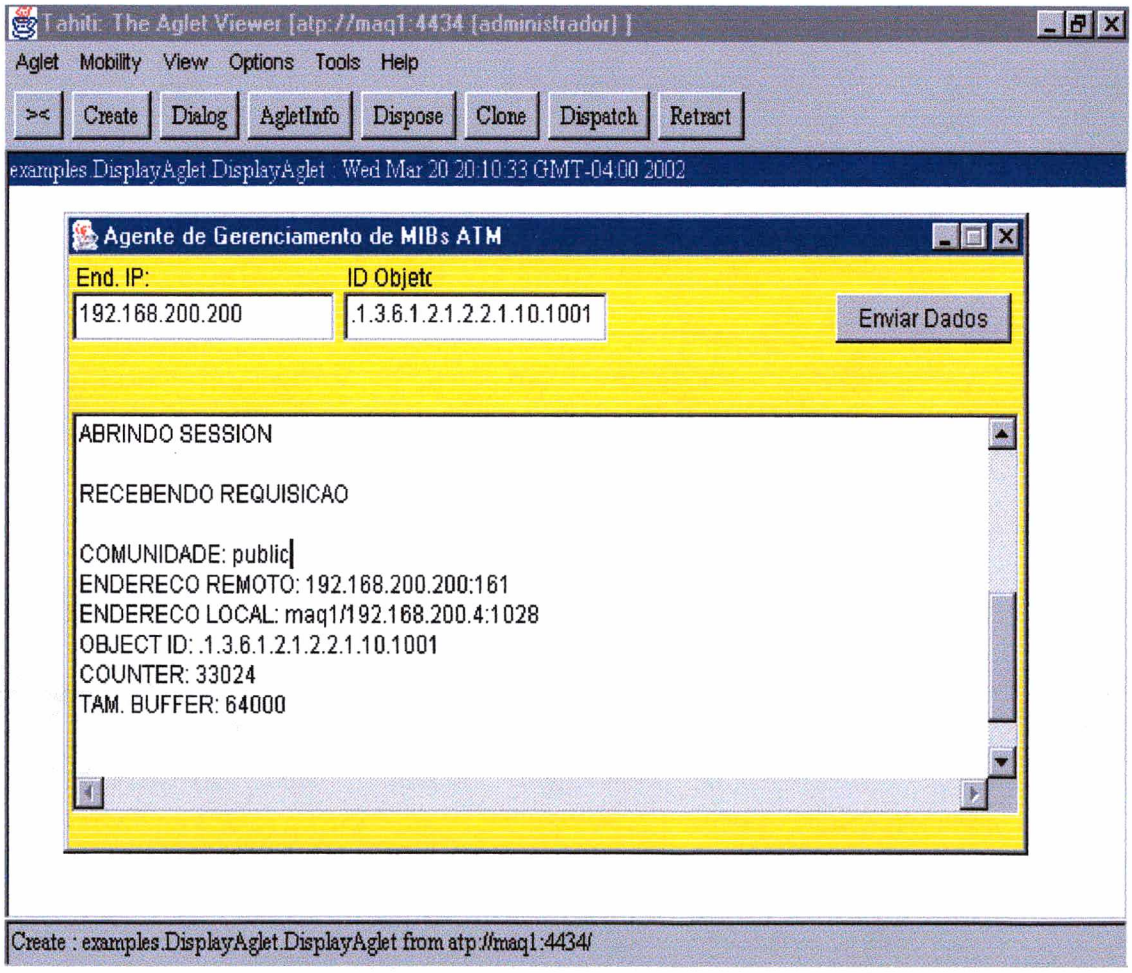


FIGURA 27 – Interface de gerência instanciada no TAHITI

Na Fig. 27 temos a execução do protótipo já no ambiente do TAHITI, que serve como estação de gerência móvel coletora das informações da rede. O agente é iniciado na estação e com o método *dispatch* é automaticamente enviado para a outra. O código do Quadro 6 mostra essa implementação.

```
public void run(){
    my_dialog = new frm_APIMib();
            my_dialog.setVisible(true);
            try     {URL url = new URL("atp://NOTEBOOK1:4434");
                    dispatch(url); }
            catch(MalformedURLException ds){}
            catch(Exception ds){}
            }
```

QUADRO 7 – Despachando o AGLET

Com as construções desses protótipos foram estabelecidas as condições de desenvolver o agente móvel proposto neste trabalho, para gerência de um *backbone ATM*, então foi definida a nova interface, campos da *MIB ATM* e a sua implementação.

7.6. A Definição das Informações Gerenciáveis para a INFOVIA-MT

Tanto para os elementos ativos workgroup, como para os puramente ATM, serão utilizados os protocolos SNMP, onde foram definidos quais as informações da MIB serão coletadas.

7.6.1. Definição dos Campos para Equipamentos ATM

O modelo proposto para gerenciamento de redes ATM é semelhante ao utilizado nas redes tradicionais, que envolvem a aplicação de gerenciamento e uma aplicação de

um agente que deve servir as requisições SNMP do gerente. A troca de mensagem utilizará o serviço de LANE.

Para os equipamentos com *core* ATM NNI *(Network Network Interface),* foi utilizado a proposta da *Integrated Local Management Interface* (ILMI) do *ATM Forum,* cujas mensagens SNMP são encapsuladas diretamente sobre o protocolo AAL5. A ATM interface MIB, descrita na ILMI 4.0, contém informações referentes a nível físico, nível ATM, conexões VPs e VCs, informações de registro de endereços e serviço de registro e outras conforme descrito por [GER00]. Foram visualizados os grupos na Fig. 28.

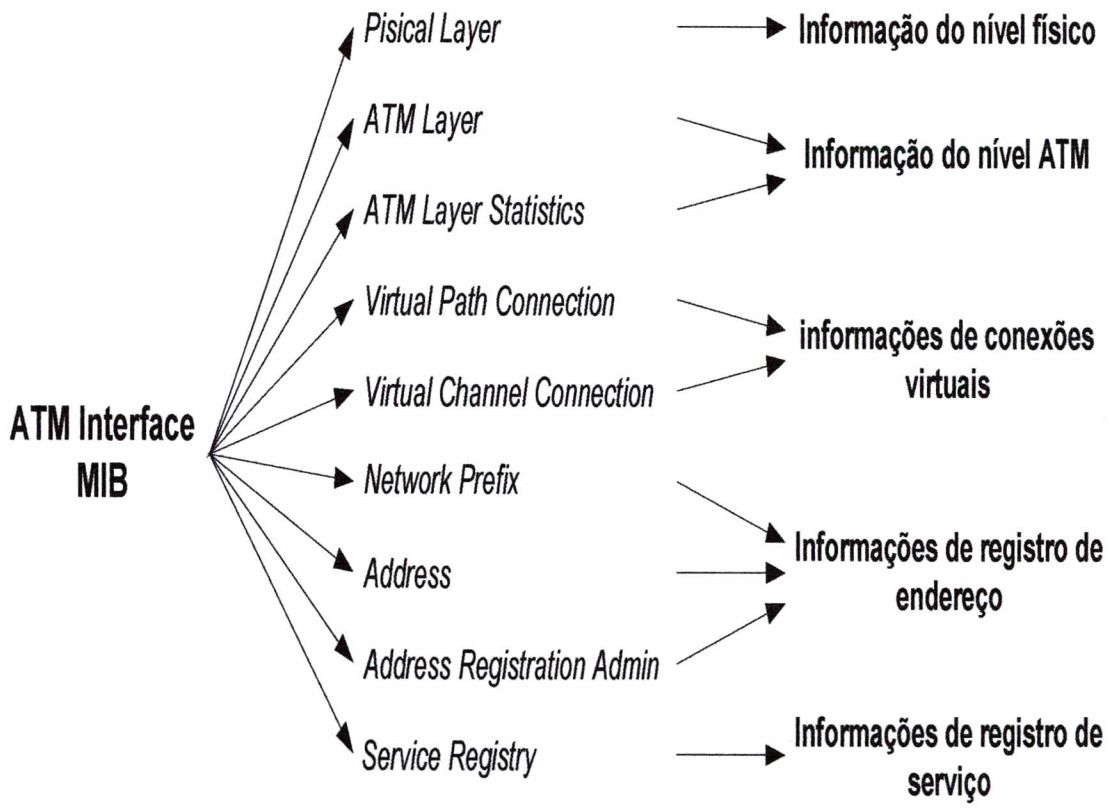


FIGURA 28 – Grupos do ATM

7.6.1.1. Informações do Nível Físico

Foram definidas as informações e estatísticas sobre *Phisical Layer* na interface monitorada para o agente da INFOVIA-MT. É composto pelos objetos descritos na tabela 2.

Identificador do Objeto	Função
AtmfPortindex	Usado para identificar uma interface ATM no equipamento que está sendo monitorado.
AtmfPortMyIfName	Identifica o Nome da Interface.
AtmfPortMyIfIdentifier	É um número inteiro que identifica a interface ATM.
AtmfMyIpNmAddress	Identifica o endereço IP da estação, para qual a estação de gerenciamento pode mandar mensagens de gerenciamento.
AtmfMyOsiNmNsapAddress	Identifica o endereço NSAP da estação para qual a estação de gerenciamento pode enviar mensagens de gerenciamento.

TABELA 2 – Identificação dos objetos do Nível Físico

7.6.1.2. Informações do Nível ATM

Foram definidas as informações e estatísticas sobre *ATM Layer* na interface monitorada para o agente da INFOVIA-MT. É composto pelos objetos descritos na tabela 3.

Identificador do Objeto	Função
Index	Usado para identificar uma interface ATM no equipamento que está sendo monitorado.
MaxVPCs ***MaxVCCs***	Armazenam, respectivamente o número máximo de switched ou *permanent* VPCs e VCCs suportados pela interface.

ConfiguredVPCs ***ConfiguredVCCs***	Contém o número de VPCs e VCCs permanentes. Este número também informa a quantidade de linhas na tabela de *atmfVpcTable.*
MaxVpibits ***MaxVcibits***	Armazena o número máximo de bits de VPI e VCI que podem estar ativos na interface.
Unitype	Contém o tipo da UNI (2.0, 3.0, 3.1, 4.0, outro).
UniVersion	Informa se a UNI é Privada ou Pública.
DeviceType	Informa o tipo do equipamento (usuário ou nó de rede).
IlmiVersion	Informa a versão da ILMI.
NniSigVersion	Indica qual a ultima versão do *Private Network-Network Interface* (PNNI) suportado pela interface.

TABELA 3 – Ideintificação dos objetos da Camada ATM

7.6.1.3. Informações de Conexões Virtuais

Foram definidas as informações e estatísticas sobre as conexões virtuais na interface monitorada para o agente da INFOVIA-MT. É composto pelos objetos na tabela 4.

Identificador do Objeto	Função
PortIndex	Usado para identificar uma interface ATM no equipamento que está sendo monitorado.
OperStatus	Indica o *status* da conexão virtual.
ServiceCategory	Indica a categoria do serviço ATM.

TABELA 4 – Identificação dos objetos das conexões virtuais

7.6.1.4. Informações do Registro de Endereços

Foram definidas as informações e estatísticas sobre registro de endereços na interface do agente da INFOVIA-MT. É composto pelos objetos descritos na tabela 5.

Identificador do Objeto	Função
AtmfNetPrefixPort	Contém a identificação da interface ATM.
AtmfNetPrefixPrefix	Contém o prefixo usado para o endereço ATM.
AtmfAddressStatus	Indica a validade do endereço ATM pertencente a interface.

TABELA 5– Identificação dos objetos de registro de endereços

7.6.1.5. Informações dos Serviços

Foram definidas as informações e estatísticas sobre os serviços na interface monitorada para o agente da INFOVIA-MT. É composto pelos objetos da tabela 6.

Identificador do Objeto	Função
AtmfSrvcRegPort	Indica qual a interface na qual esta mensagem foi recebida.
AtmfSrvcRegServiceID	Determina qual o tipo de serviço.
atmfSrvcRegATMAddress	Indica o endereço ATM de onde o serviço reside.

TABELA 6– Identificação dos objetos de serviço

7.6.2. Definição dos Campos para Equipamentos *Workgroup*

As informações e estatísticas que serão monitoradas pelo agente móvel da INFOVIA-MT, são definidas para o *ObjetcID* ***.iso.org.dod.internet.mgmt.mib-2.system,*** conforme a tabela 7.

Identificador do Objeto	Função
SysContact	Pessoa responsável pelo sistema.
sysDescr	Descrição do sistema.
SysUpTime	Quanto tempo o sistema está operacional.

TABELA 7– Identificação dos objetos do Sistema

As informações e estatísticas que serão monitoradas pelo agente móvel da INFOVIA-MT, são definidas para o *ObjetcID* ***.iso.org.dod.internet.mgmt.mib-2.interfaces***, conforme a tabela 8.

Identificador do Objeto	Função
ifDescr	nome da interface.
ifInDiscards	taxa de descarte de entrada .
ifSpeed	largura de banda da interface .

TABELA 8– Identificação dos objetos de Interfaces

As informações e estatísticas que serão monitoradas pelo agente móvel da INFOVIA-MT, são definidas para o *ObjetcID* ***.iso.org.dod.internet.mgmt.mib-2.ip***, conforme a tabela 9.

Identificador do Objeto	Função
ipAdEntBcastAddr	o bit menos significativo do endereco IP de broadcast.
ipForwDatagrams	taxa de datagramas repassados.
ipRouteDest	endereço IP do destino.

TABELA 9– Identificação dos objetos do protocolo IP

As informações e estatísticas que serão monitoradas pelo agente móvel da INFOVIA-MT, são definidas para o *ObjetcID* ***.iso.org.dod.internet.mgmt.mib-2.icmp***, conforme a tabela 10.

Identificador do Objeto	Função
icmpInDestUnreachs	taxa de mensagens de Destino não Alcançado, recebidas.
IcmpInEchoReps	taxa de mensagens de Respostas a Eco recebidas.
IcmpInMsgs	taxa de recebimento de mensagens.

TABELA 10– Identificação dos objetos de do protocolo ICMP

As informações e estatísticas que serão monitoradas pelo agente móvel da INFOVIA-MT, são definidas para o *ObjetcID* ***.iso.org.dod.internet.mgmt.mib-2.tcp***, conforme a tabela 11.

Identificador do Objeto	Função
TcpActiveOpens	número de vezes que o sistema abriu uma conexão.
TcpAttemptFails	número de tentativas de conexões falhadas .
TcpCurrEstab	número de conexões de transporte corretamente abertas.
TcpEstabResets	número de reinicializações de conexões estabelecidas.

TABELA 11– Identificação dos objetos do protocolo TCP

As informações e estatísticas que serão monitoradas pelo agente móvel da INFOVIA-MT, são definidas para o *ObjetcID* ***.iso.org.dod.internet.mgmt.mib-2.udp***, conforme a tabela 12.

Identificador do Objeto	Função
UdpInDatagrams	taxa de datagramas recebidos.
UdpInErrors	taxa de datagramas UDP recebidos com erro.
UdpNoPorts	taxa de datagramas enviados.

TABELA 12– Identificação dos objetos do protocolo UDP

As informações e estatísticas que serão monitoradas pelo agente móvel da INFOVIA-MT, são definidas para o *ObjetcID* ***.iso.org.dod.internet.mgmt.mib-2.egp***, conforme a tabela 13.

Identificador do Objeto	Função
egpOutErrors	taxa de mensagens não enviadas, devido a ocorrência de erros.
egpOutMsgs	taxa de mensagens enviadas.

TABELA 13– Identificação dos objetos do protocolo EGP

As informações e estatísticas que serão monitoradas pelo agente móvel da INFOVIA-MT, são definidas para o *ObjetcID* ***.iso.org.dod.internet.mgmt.mib-2.snmp***, conforme a tabela 14.

Identificador do Objeto	Função
snmpOutPkts	taxa de pacotes SNMP enviados.
SnmpOutTraps	taxa de traps enviadas.

TABELA 14– Identificação dos objetos do protocolo SNMP

7.7. Aplicabilidade do Agente na Gerência do Backbone ATM na Prática

As aplicabilidades dessas tecnologias enfatizando os resultados obtidos para gerência de redes ATM. Foram definidas 03 (três) estações de gerência distribuída, onde o agente irá percorrer as estações, buscando as informações dos elementos ativos com *core* ATM e os equipamentos *workgroup*, definidas na sessão anterior, para monitorar a rede de forma distribuída. Toda vez que o agente identificar um excesso de pacotes na *switch* que está conectado a estação, ele automaticamente toma a decisão de mover-se para outro *host,* e com isso o agente coletor móvel estará sempre sendo executado na à partir da estação conectado ao *switch* sem *gargalos*, ou seja, minimizando perdas de pacotes. Observe o esquema da gerência do *backbone* ATM na Fig. 29.

O agente permanecerá coletando informações à partir daquele *host*, depois de um tempo determinado através da classe *Timer* , ele fará a verificação de possíveis gargalos, assim movendo-se automaticamente para outro *host.*

Na aplicação de gerência da INFOVIA-MT foram realizados testes passando como parâmetro de mobilidade o tempo de 20 segundos após a sua execução. Maiores detalhes serão descritos no ANEXO 1 – Fontes da Aplicação.

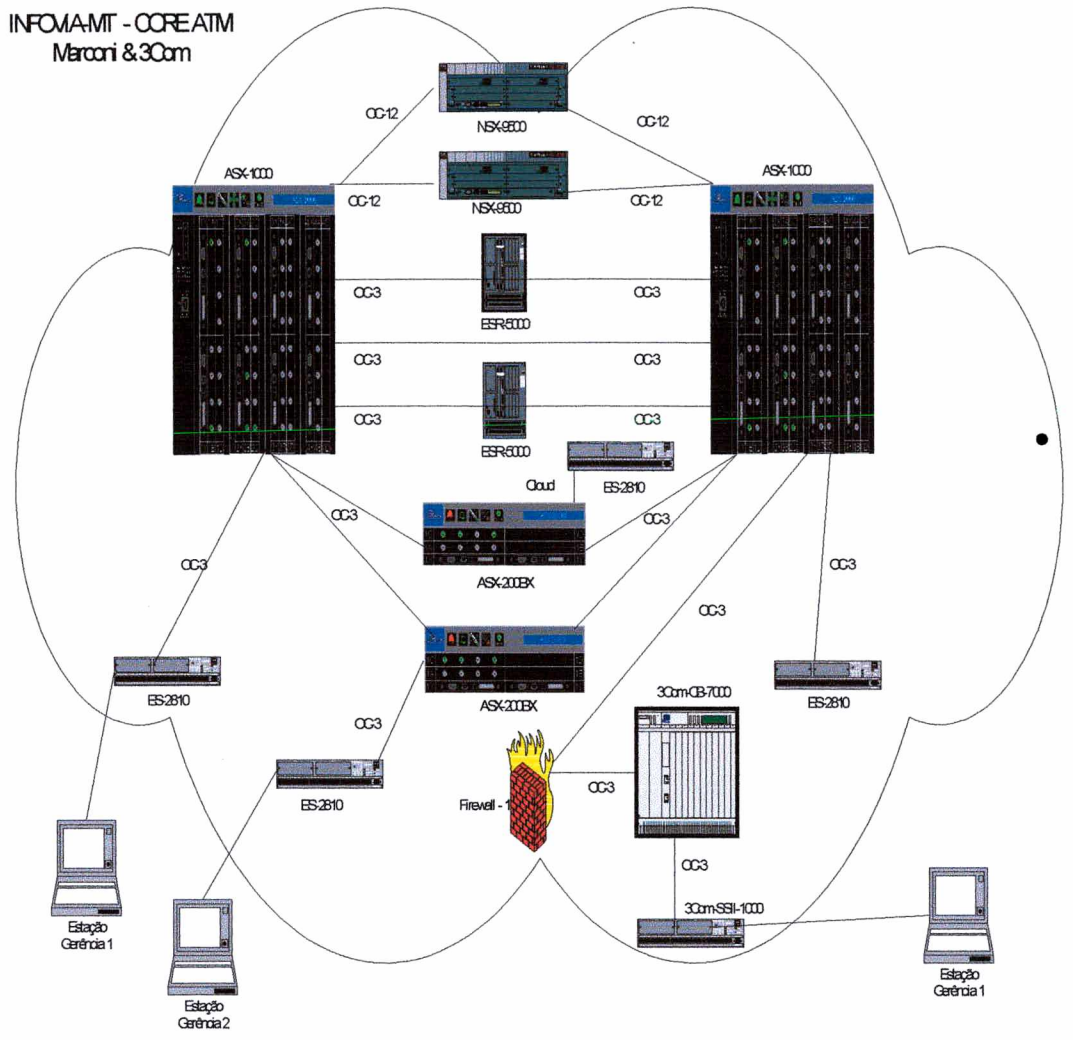


FIGURA 29 – Mobilidade na gerência da Rede ATM – INFOVIA-MT

7.8. Interface do Agente Móvel para a INFOVIA-MT

Para agente móvel foi desenvolvida uma interface que possibilita a visualização das informações contidas na MIB dos elementos ativos, puramente ATM e também dos *workgroup*.

Com objetivo de facilitar a visualização da coleta de informações realizada pelo agente, foi escolhida uma interface baseada em estrutura de árvore, onde podemos selecionar os Elementos Ativos ATM e os Elementos Ativos *Workgroup*, conforme a Fig. 30.

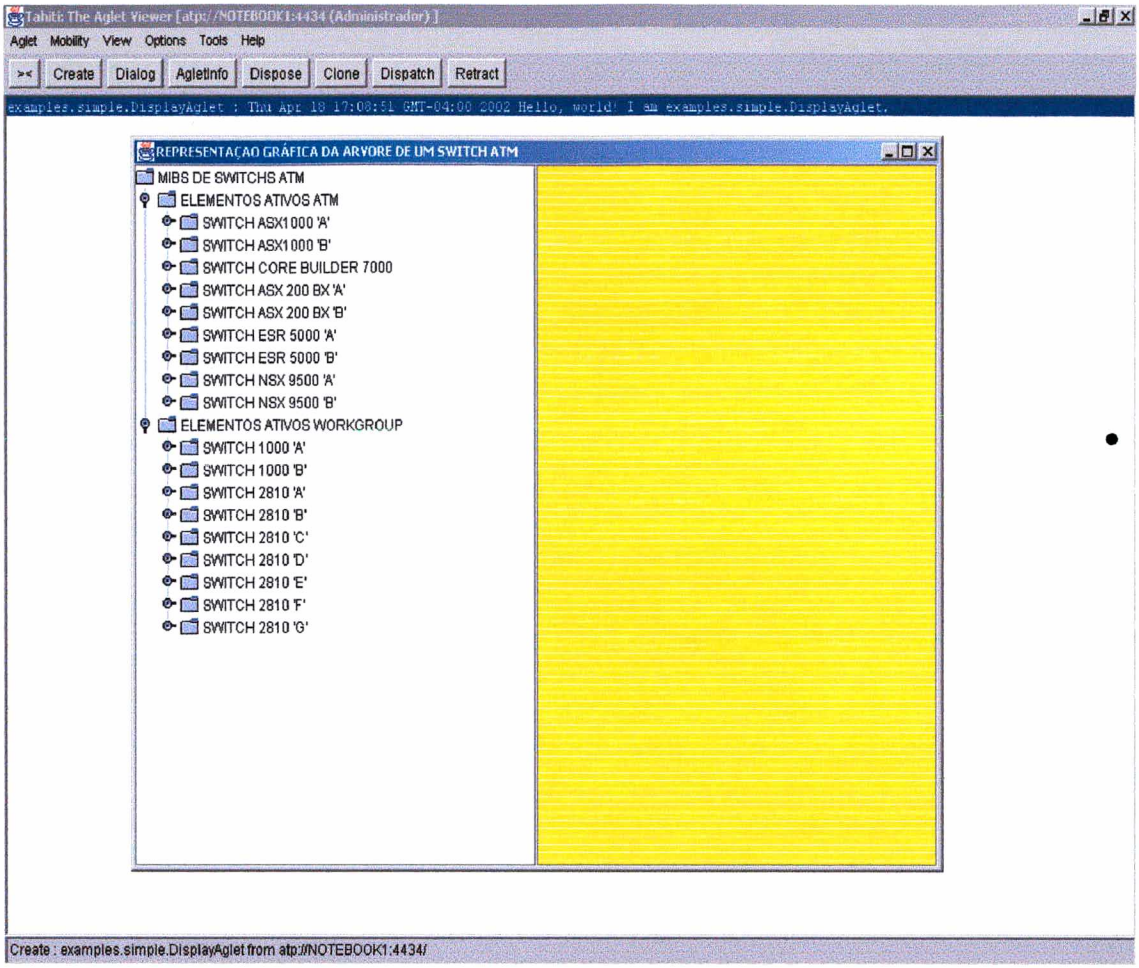


FIGURA 30 – Interface do Agente Móvel para INFOVIA-MT

Foi desenvolvido em *Java,* usando as classes de gerência de redes das API do *AdventNetSNMP* e as classes dos AGLETS da IBM, que possibilitam a mobilidade do código, citadas nas sessões anteriores.

Independente da estação móvel coletora onde está sendo executado o agente, ele poderá sempre coletar as informações. Como esse é o **Ambiente de Produção** do **Centro de Processamento de Dados do Estado de Mato Grosso**, este agente móvel trabalha somente com a primitiva *GET*, somente leitura, ou seja, não havendo nenhuma possibilidade do mesmo vir a danificar alguma configuração da rede. Para visualizar as informações coletadas de um determinado equipamento, basta selecioná-lo e escolher qual a informação, selecionando a folha correspondente do equipamento na MIB, será aberta uma sessão SNMP e será feita a coleta dessa informação pelo agente móvel, logo após a sessão. São visualizdas as informações dos elementos puramente ativos ATM na

Fig. 31, busca pela variável *SwitchSTPConfig,* cujo o identificador do objeto é **.1.3.6.1.4.1.1012.81.1.5.1.1.1**, que representa se o serviço de STP está habilitado no equipamento. No caso do *SWITCH ESR 5000 "A",* o agente trouxe a informação que este serviço está desabilitado *(Disable)* – INFORMAÇÃO = *Integer*: *4* .

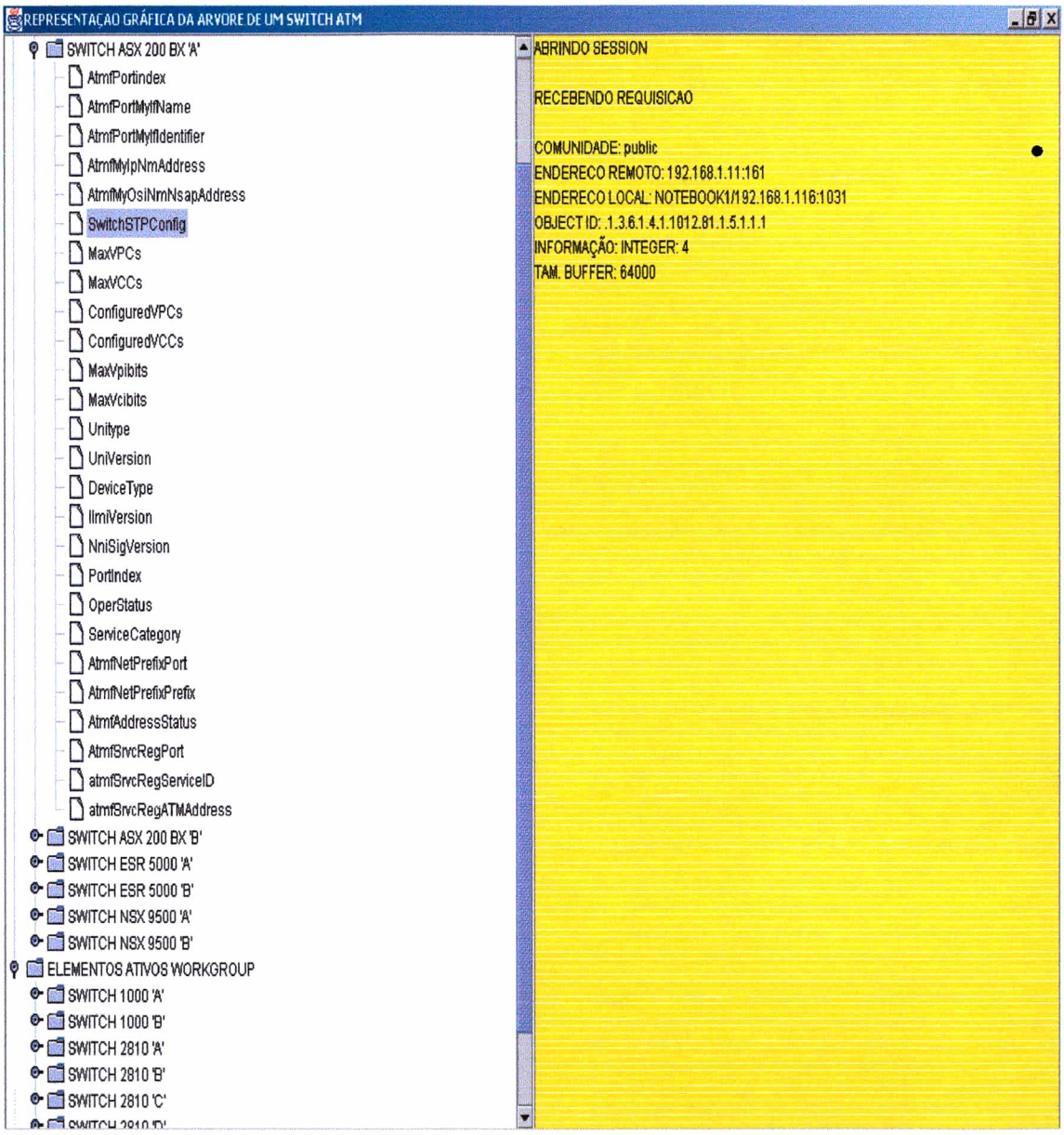


FIGURA 31 – Interface do Agente Móvel para INFOVIA-MT – ATM

A Fig. 32 mostra as informações dos *swithes workgroup*, busca pela variável *SwitchSTPConfig,* cujo identificador do objeto é **.1.3.6.1.4.1.1.1**, que representa o tempo total de operação do equipamento à partir do último *reboot.*

No caso do *SWITCH 1000 "A"*, o agente trouxe a informação que o equipamento está ligado a exatamente *21 HOURS, 33 MINUTES, 28 SECONDS.*

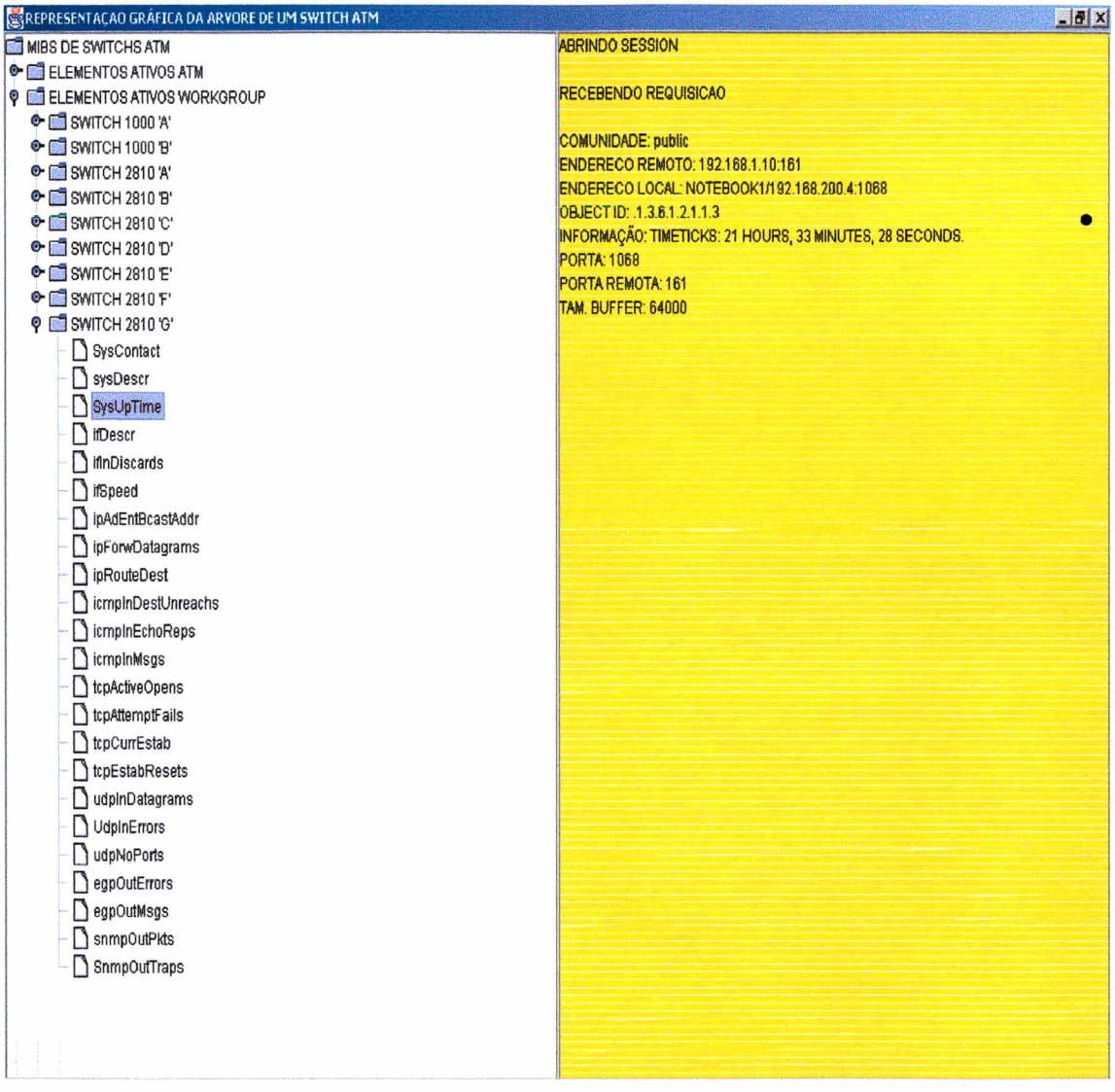


FIGURA 32 – Interface do Agente Móvel para INFOVIA-MT – *Workgroup*

Especificamente a mobilidade de código para a INFOVIA-MT é muito interessante, principalmente porque elementos ativos se encontram distantes fisicamente da sala de operações da INFOVIA-MT dificultando a gerência da rede, contudo pode-se utilizar a mobilidade para facilitar e operacionalizar essa gerência, possibilitando que possíveis falhas sejam identificadas no menor tempo possível.

8. CONCLUSÃO

A tecnologia ATM possui características de transporte de informação com altas taxas (desde *Mbit/s* até *Gbt/s*), grande escalabilidade, facilidade de multiplexação e comutação, facilidade pela integração de serviços e tem a capacidade de suportar novos serviços ou inovações tecnológicas que é um dos pontos mais fortes desta tecnologia, pois o ATM permite integrar de forma eficiente e simples, os serviços com as mais variadas características, por exemplo, taxa de bit constante ou variável, características de atraso constante ou variável, além de oferecer uma alta confiabilidade.

Este trabalho proporcionou uma perfeita interação entre Arquitetura de Redes, gerência de redes e Sistemas distribuídos, onde foi enfatizada a fundamentação teórica, e além disso, foi aplicado na prática em nosso ambiente de trabalho, analisando a maior rede de comunicação de dados do estado de Mato Grosso, a INFOVIA-MT.

8.1. Resultados Obtidos

Um estudo sobre a tecnologia ATM baseado nos conceitos de sua arquitetura, suas características e enfatizando a INFOVIA-MT na sua real configuração e para demonstrar o esquema do *backbone*, suas principais configurações e de seus equipamentos.

A gerência e monitoramento de redes, apresentando o embasamento teórico que na prática demonstradas com as ferramentas de gerência e monitoramento proprietárias, via console e via *web browser*, que são hoje utilizadas na INFOVIA-MT para configuração, detecção de erros e ganho de performance.

Após a compreensão do processo de gerência, estudos sobre as MIB's, dos equipamentos da INFOVIA-MT, utilizando a ferramenta *MibBrowser* do *AdventNet*, dando ênfase para ATM MIB dos equipamentos com núcleo ATM e RFC1213 MIB

para equipamentos de borda. Foi desenvolvido um protótipo de busca de informações utilizando as classes do SNMP.

A utilização dos agentes móveis com os AGLETS da IBM, suas classes e métodos, que proporcionaram além da compreensão da parte conceitual a possibilidade de desenvolver o agente proposto, enfatizando principalmente a mobilidade de código.

A implementação de agentes móveis na gerência da INFOVIA-MT, foi demonstrada através de uma interface, que foi instanciada em servidores rodando *TAHITI,* onde este agente busca informações da rede e se move pela mesma.

Como este trabalho foi implantado em um ambiente de produção da INFOVIA-MT, utilizamos somente a primitiva de leitura, onde o agente não acarreta nenhum dano a rede, ficando assim como subsídio para novos trabalhos que utilização a primitiva de gravação, que gostaríamos de fazer para um projeto futuro maior descrito na próxima sessão.

A mobilidade de código do agente móvel desenvolvido para a INFOVIA-MT é importante, principalmente, porque os elementos ativos se encontram distantes fisicamente da sala de operações da INFOVIA-MT, dificultando a gerência da rede, contudo pode-se utilizar dessa mobilidade para facilitar e operacionalizar essa gerência, possibilitando que possíveis falhas sejam identificadas no menor tempo possível.

Este trabalho proporcionou na realidade prática a possibilidade de fazer um comparativo entre a gerência de redes utilizando agentes e os métodos tradicionais, onde os resultados obtidos foram satisfatórios em relação ao desempenho e mobilidade do agente. Em relação ao desempenho, com a utilização dos agentes móveis pode-se utilizar máquinas com pouco poder de processamento, ao contrário da maioria dos softwares de gerência em outras plataformas que utilizam muito o processador. A mobilidade vem falicitar a atividade de gerência como um todo, como exemplo, pode-se quantificar no caso específico do agente da INFOVIA-MT, o tempo gasto para o deslocamento de um técnico até o local, onde está instalado o elemento ativo, e a

necessidade de um equipamento para a conectar a *console* via *Hiper Terminal*, realizando as consultas na *mib* do ativo através de comandos. Neste caso o custo com deslocamento, onde o técnico deve aguardar a disponibilidade do setor de transporte da empresa, podendo chegar até um dia inteiro de espera, e dependendo da criticidade da consulta, o tempo de resposta é muito importante, podendo afetar o desempenho da rede.

8.2. Perspectivas Futuras

Durante a realização deste trabalho e enaltecidos com o poder das ferramentas aqui utilizadas, apresentamos aqui as nossas perspectivas para um futuro trabalho em outro nível de aprofundamento, excelência, único e principalmente de tempo disponível para execução das pesquisas e elaboração dos protótipos.

Um estudo mais profundo da MIB ATM para os equipamentos puramente ATM e da RFC1213 MIB para equipamentos de outras arquiteturas com *uplinks* ATM gerência de redes ATM com SNMP, utilizando também a primitiva SET sem correr riscos de danificar configurações da rede em um ambiente de produção.

A integração de todas essas informações gerenciais em Banco de Dados Distribuídos, onde teríamos replicadas e fragmentadas as informações, para aumentar a disponibilidade e performance da rede.

A utilização técnicas de para minimizar a estatísticas de falhas e ao mesmo tempo maximizar as estatísticas de performance, encontraremos fórmulas estatísticas que decidirão quando deveremos mover-se para outro *host*, ou até mesmo na configuração de um determinado recurso do equipamento para alcançar índices aceitáveis.

Técnicas de Inteligência Computacional usando os conceitos de Aprendizagem de Máquina, onde de posse de uma base de conhecimento o agente possa aprender, reagir e tomar decisões para melhorar a sua performance.

Implementar no agente as técnicas de priorização de pacotes com a aplicação de recursos de qualidade de serviço disponível na tecnologia ATM, possibilitando assim estabelecer acordos de nível de serviço - SLA (*Service Level Agrement*), onde o agente garantirá a o recurso necessário para que isto aconteça.

A expansão da mobilidade na interação com agentes de plataformas distintas e possivelmente na interoperabilidade com outras tecnologias como CORBA, RMI, Servlets com JSP (*Java Server Pages*).

A implementação de software que tenha a integração de todas as tecnologias aqui citadas, possibilitando um aumento considerável de performance do processo de gerência e monitoramento de redes ATM.

9. REFERÊNCIAS BIBLIOGRÁFICAS

[BIE98] BIESZCZAD, A. Mobile Agents for Network Management, IEEE Communications Surveys. Fourth Quarter 1998. Vol. 1 No.1 - Available at http://www.comsoc.org/pubs/surveys.

[CHA93] CHANG, Y.; SU D.e WAKID, S. The Generic Flow Control (GFC) Protocol: A Performance Assessment. Proceedings of International Conference on Network Protocols (ICNP93), October, 19-22, 1993, também disponível em; www:/isdn.ucsl.nist.gov/misc/hsnt/journals/gfcpaper.html

[CHE97] CHENG, L. Quality of service based on both call admission and cell scheduling. Computer Networks and ISDN Systems 29(5), 555-567, abr.1997.

[CON00] PLATAFORMA CONCÓRDIA. Abril, 2000. Site http://www.mitsubishielectric.com

[COS99] 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.

[DAN00] OLIVEIRA, DANIELA VANASSI DE. Mobilidade em Gerência de Redes SNMP. Dissertação de mestrado do CPGCC-UFSC. Florianópolis, outubro, 2000.

[DZI97] DZIONG, Z. ATM Network Resource Management. McGraw-Hill series on computer communications, McGraw-Hill, New York,1997, 315p.

[GER00] GERÊNCIA DE REDES. Fevereiro, 2000. Site http://penta2.ufrgs.br

[GOT97] GOTTFRIED LUDERER, Network Management Agents Supported by a Java Environment, ISINM'97, SanDiego CA. 1997.

[GOY97] GOYAL, et al. Improving the Performance of TCP over the ATM-UBR service. Submitted to Computer Communications, 1997. Disponível em, http://www.cis.ohio-state.edu/~jain/

[ITU93] ITU-T. Recommendation I.321: B-ISDN protocol reference model. Abril de 1993.

[ITU93a] ITU-T. Recommendation I.363: B-ISDN ATM adaptation layer (AAL) specification. Março de 1993.

[ITU93b] ITU-T. Recommendation I.413: B-ISDN User-Network Interface (UNI). Março de 1993.

[ITU93c] ITU-T. Recommendation I.350: General aspects of quality of service and network performance in digital networks, including ISDN. Março de 1993.

[ITU96] ITU-T. Recommendation I.371: Traffic Control and Congestion Control in B-ISDN. Agosto de 1996.

[MAR98] JÚNIOR, MÁRIO LEMES PROENÇA. Uma Ferramenta para Auxílio no Gerenciamento de Redes com Backbone ATM. Dissertação de Mestrado. Porto Alegre, março 1998.

[MON94] MONTEIRO, J. A. S. Rede Digital de Serviços Integrados de Faixa Larga (RDSI-FL). Editado por: Silvio Lemos Meira em, IX Escola de Computação, Recife, julho, 1994, 206p.

[OSH98] OSHIMA, Mand Karjoth Aglet Specification 1.0, Proceedings of 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98). 1998.

[PRY95] PRYCKER, M. de. Asynchronous Transfer Mode - Solution for Broadband ISDN. Third Edition, Prentice Hall, London, 1995, 380p.

[ROC95] ROCHOL, J. Comunicação Digital. Texto Didático, Departamento de Informática Aplicada, Instituto de Informática da UFRGS, 1995, 215p

[SCH98] SCHRAMM, C., BIESZCZAD, A. AND PAGUREK, B. Application-Oriented Network Modeling with Mobile Agents. Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'98). New Orleans, Louisiana, Feb. 1998.

[SID93] SIDI, M.; LIU, W. Z.; CIDON, I.; GOPAL, I. Congestion Control through input rate regulation. IEEE Transactions on Communications. 41(3), 471-477, Mar. 1993

[TAF95] THE ATM FORUM. LAN Emulation Over ATM, Version 1.0, AF-LANE-0021.000, [S.I.], jan. 1995.

[TAF97] THE ATM FORUM. Remote Monitorimg MIB Extensions for ATM Networks, af-nm-test-0080.000, [S.I.], mai. 1997.

[TAH00] PLATAFORMA TAHITI. Dezembro 2000 http://www.trl.ibm.com/aglets/tahiti/tahiti.html

[TAN97] TANENBAUM, Andrew S., 1944- Redes de Computadores/ Andrew S. Tanenbaum: Tradução [ds 3.ed.original] Insight Serviços de Informática. Rio de Janeiro: Campus, 1997.

[VOY00] PLATAFORMA VOYAGER. Abril, 2000. Site http://www.objectspace.com

ANEXO 1 - FONTES

Arquivo Fonte do agente da INFOVIA-MT para a solução proposta.

```
//*****************************************************************************//

package examples.DisplayAglet;
import com.ibm.aglet.*;
import com.adventnet.snmp.beans.*;
import com.adventnet.snmp.mibs.*;
import com.adventnet.snmp.snmp2.*;
import com.adventnet.snmp.snmp2.usm.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.tree.*;

/********************* Formulario Principal ********************************/
class SimpleTreeFrame extends JFrame{
  JTree tree;
  JTextArea res = new JTextArea();

  //********************* Inicio do Método Construtor**********************//
  public SimpleTreeFrame(){
      Container contentPane = getContentPane();
      contentPane.setLayout(new GridLayout(1,2));
      setTitle("REPRESENTAÇAO GRÁFICA DA ARVORE DE UM SWITCH ATM");
      setBounds(100,50,600,500);
      addWindowListener(new WindowAdapter(){
         public void windowClosing(WindowEvent e){
                        System.exit(0);
         }
      });
      DefaultMutableTreeNode root = new DefaultMutableTreeNode("MIBS DE SWITCHS
ATM");
      DefaultMutableTreeNode ModelosSwitch = new DefaultMutableTreeNode("ELEMENTOS
ATIVOS ATM");
      root.add(ModelosSwitch);
      /********************* Switch asx1000 'A'*************************************/
      DefaultMutableTreeNode Switchs = new DefaultMutableTreeNode("SWITCH ASX1000
'A'");
      DefaultMutableTreeNode Folhas = new
DefaultMutableTreeNode(LeafAtm[0]);Switchs.add(Folhas);
      for(int i = 1; i < 26; i++){
         Folhas = new DefaultMutableTreeNode(LeafAtm[i]);Switchs.add(Folhas);
      }
      ModelosSwitch.add(Switchs);
    //************************* Switch asx1000 'B'*****************************//
      Switchs = new DefaultMutableTreeNode("SWITCH ASX1000 'B'");
      for(int i = 0; i < 26; i++){
         Folhas = new DefaultMutableTreeNode(LeafAtm[i]);Switchs.add(Folhas);
      }
      ModelosSwitch.add(Switchs);
```

```
//**********************************SWITCH CORE BUILDER 7000 **************//
   Switchs = new DefaultMutableTreeNode("SWITCH CORE BUILDER 7000");
   for(int i = 0; i < 26; i++){
      Folhas = new DefaultMutableTreeNode(LeafAtm[i]);Switchs.add(Folhas);
   }
   ModelosSwitch.add(Switchs);
//**********************************SWITCH ASX 200 BX 'A' **************//
   Switchs = new DefaultMutableTreeNode("SWITCH ASX 200 BX 'A'");
   for(int i = 0; i < 26; i++){
      Folhas = new DefaultMutableTreeNode(LeafAtm[i]);Switchs.add(Folhas);
   }
   ModelosSwitch.add(Switchs);
//**********************************SWITCH ASX 200 BX 'B'    **************//
   Switchs = new DefaultMutableTreeNode("SWITCH ASX 200 BX 'B'");
   for(int i = 0; i < 26; i++){
      Folhas = new DefaultMutableTreeNode(LeafAtm[i]);Switchs.add(Folhas);
   }
   ModelosSwitch.add(Switchs);
      //*********************************SWITCH ESR 5000 'A'**********************//
   Switchs = new DefaultMutableTreeNode("SWITCH ESR 5000 'A'");
   for(int i = 0; i < 26; i++){
      Folhas = new DefaultMutableTreeNode(LeafAtm[i]);Switchs.add(Folhas);
   }
   ModelosSwitch.add(Switchs);
      //*********************************SWITCH ESR 5000 'B'**********************//
   Switchs = new DefaultMutableTreeNode("SWITCH ESR 5000 'B'");
   for(int i = 0; i < 26; i++){
      Folhas = new DefaultMutableTreeNode(LeafAtm[i]);Switchs.add(Folhas);
   }
   ModelosSwitch.add(Switchs);
      //*********************************SWITCH NSX 9500 'A'**********************//
   Switchs = new DefaultMutableTreeNode("SWITCH NSX 9500 'A'");
   for(int i = 0; i < 26; i++){
      Folhas = new DefaultMutableTreeNode(LeafAtm[i]);Switchs.add(Folhas);
   }
   ModelosSwitch.add(Switchs);

      //*********************************SWITCH NSX 9500 'B'**********************//
   Switchs = new DefaultMutableTreeNode("SWITCH NSX 9500 'B'");
   for(int i = 0; i < 26; i++){
      Folhas = new DefaultMutableTreeNode(LeafAtm[i]);Switchs.add(Folhas);
   }
   ModelosSwitch.add(Switchs);

      //*************************************************************************//
   ModelosSwitch = new DefaultMutableTreeNode("ELEMENTOS ATIVOS WORKGROUP");
   root.add(ModelosSwitch);
      //*********************************SWITCH 1000 'A'**************************//
   Switchs = new DefaultMutableTreeNode("SWITCH 1000 'A'");
   for(int i = 0; i < 23; i++){
      Folhas = new DefaultMutableTreeNode(LeafWork[i]);Switchs.add(Folhas);
   }
   ModelosSwitch.add(Switchs);

//*********************************SWITCH 1000 'B'**************************//
   Switchs = new DefaultMutableTreeNode("SWITCH 1000 'B'");
   for(int i = 0; i < 23; i++){
      Folhas = new DefaultMutableTreeNode(LeafWork[i]);Switchs.add(Folhas);
```

```
    }
    ModelosSwitch.add(Switchs);

//*****************************SWITCH 2810 'A'*****************************//
    Switchs = new DefaultMutableTreeNode("SWITCH 2810 'A'");
    for(int i = 0; i < 23; i++){
       Folhas = new DefaultMutableTreeNode(LeafWork[i]);Switchs.add(Folhas);
    }
    ModelosSwitch.add(Switchs);

//*****************************SWITCH 2810 'B'*****************************//
    Switchs = new DefaultMutableTreeNode("SWITCH 2810 'B'");
    for(int i = 0; i < 23; i++){
       Folhas = new DefaultMutableTreeNode(LeafWork[i]);Switchs.add(Folhas);
    }
    ModelosSwitch.add(Switchs);

//*****************************SWITCH 2810 'C'*****************************//
    Switchs = new DefaultMutableTreeNode("SWITCH 2810 'C'");
    for(int i = 0; i < 23; i++){
       Folhas = new DefaultMutableTreeNode(LeafWork[i]);Switchs.add(Folhas);
    }
    ModelosSwitch.add(Switchs);

//*****************************SWITCH 2810 'D'*****************************//
    Switchs = new DefaultMutableTreeNode("SWITCH 2810 'D'");
    for(int i = 0; i < 23; i++){
       Folhas = new DefaultMutableTreeNode(LeafWork[i]);Switchs.add(Folhas);
    }
    ModelosSwitch.add(Switchs);

//*****************************SWITCH 2810 'E'*****************************//
    Switchs = new DefaultMutableTreeNode("SWITCH 2810 'E'");
    for(int i = 0; i < 23; i++){
       Folhas = new DefaultMutableTreeNode(LeafWork[i]);Switchs.add(Folhas);
    }
    ModelosSwitch.add(Switchs);

//*****************************SWITCH 2810 'F'*****************************//
    Switchs = new DefaultMutableTreeNode("SWITCH 2810 'F'");
    for(int i = 0; i < 23; i++){
       Folhas = new DefaultMutableTreeNode(LeafWork[i]);Switchs.add(Folhas);
    }
    ModelosSwitch.add(Switchs);

//*****************************SWITCH 2810 'G'*****************************//
    Switchs = new DefaultMutableTreeNode("SWITCH 2810 'G'");
    for(int i = 0; i < 23; i++){
       Folhas = new DefaultMutableTreeNode(LeafWork[i]);Switchs.add(Folhas);
    }
    ModelosSwitch.add(Switchs);
//*************************************************************************//
    tree = new JTree(root);

//************* Evento de Seleçao dos Nós das Arvores**********************//
    tree.addTreeSelectionListener(new TreeSelectionListener(){
       public void valueChanged(TreeSelectionEvent e){
        res.setText("");
```

```
                TreePath path = tree.getSelectionPath();
            if (path == null) return;
            DefaultMutableTreeNode SelectNode =
(DefaultMutableTreeNode)path.getLastPathComponent();
            if (SelectNode.isLeaf()){
               String lea = (String)SelectNode.getUserObject();
               String swi = (String)SelectNode.getParent().toString();
               // SWITCHS ATM
               if (swi.equals("SWITCH ASX1000 'A'")){
                  for(int i = 0; i < 26; i++){
                     if (lea.equals(Atm_ASX_1000_A.getFolha()[i]))
                        APIMib(new String[]
{Atm_ASX_1000_A.getIPSwitch(),Atm_ASX_1000_A.getOIDFolha()[i]});
                  }
               }
               if (swi.equals("SWITCH ASX1000 'B'")){
                  for(int i = 0; i < 26; i++){
                     if (lea.equals(Atm_ASX_1000_B.getFolha()[i]))
                        APIMib(new String[]
{Atm_ASX_1000_B.getIPSwitch(),Atm_ASX_1000_B.getOIDFolha()[i]});
                  }
               }
               if (swi.equals("SWITCH CORE BUILDER 7000")){
                  for(int i = 0; i < 26; i++){
                     if (lea.equals(Atm_CORE_BUILDER_7000.getFolha()[i]))
                        APIMib(new String[]
{Atm_CORE_BUILDER_7000.getIPSwitch(),Atm_CORE_BUILDER_7000.getOIDFolha()[i]});
                  }
               }
               if (swi.equals("SWITCH ASX 200 BX 'A'")){
                  for(int i = 0; i < 26; i++){
                     if (lea.equals(Atm_ASX_200_BX_A.getFolha()[i]))
                        APIMib(new String[]
{Atm_ASX_200_BX_A.getIPSwitch(),Atm_ASX_200_BX_A.getOIDFolha()[i]});
                  }
               }
               if (swi.equals("SWITCH ASX 200 BX 'B'")){
                  for(int i = 0; i < 26; i++){
                     if (lea.equals(Atm_ASX_200_BX_B.getFolha()[i]))
                        APIMib(new String[]
{Atm_ASX_200_BX_B.getIPSwitch(),Atm_ASX_200_BX_B.getOIDFolha()[i]});
                  }
               }
               if (swi.equals("SWITCH ESR 5000 'A'")){
                  for(int i = 0; i < 26; i++){
                     if (lea.equals(Atm_ESR_5000_A.getFolha()[i]))
                        APIMib(new String[]
{Atm_ESR_5000_A.getIPSwitch(),Atm_ESR_5000_A.getOIDFolha()[i]});
                  }
               }
               if (swi.equals("SWITCH ESR 5000 'B'")){
                  for(int i = 0; i < 26; i++){
                     if (lea.equals(Atm_ESR_5000_B.getFolha()[i]))
                        APIMib(new String[]
{Atm_ESR_5000_B.getIPSwitch(),Atm_ESR_5000_B.getOIDFolha()[i]});
                  }
               }
               if (swi.equals("SWITCH NSX 9500 'A'")){
```

```
                for(int i = 0; i < 26; i++){
                    if (lea.equals(Atm_NSX_9500_A.getFolha()[i]))
                        APIMib(new String[]
{Atm_NSX_9500_A.getIPSwitch(),Atm_NSX_9500_A.getOIDFolha()[i]});
                }
            }
            if (swi.equals("SWITCH NSX 9500 'B'")){
                for(int i = 0; i < 26; i++){
                    if (lea.equals(Atm_NSX_9500_B.getFolha()[i]))
                        APIMib(new String[]
{Atm_NSX_9500_B.getIPSwitch(),Atm_NSX_9500_B.getOIDFoiha()[i]});
                }
            }
            //**************************************SWITCHS
WORKGROUPS*****************************************//
            if (swi.equals("SWITCH 1000 'A'")){
                for(int i = 0; i < 23; i++){
                    if (lea.equals(Work_SWITCH_1000_A.getFolha()[i]))
                        APIMib(new String[]
{Work_SWITCH_1000_A.getIPSwitch(),Work_SWITCH_1000_A.getOIDFolha()[i]});
                }
            }
            if (swi.equals("SWITCH 1000 'B'")){
                for(int i = 0; i < 23; i++){
                    if (lea.equals(Work_SWITCH_1000_B.getFolha()[i]))
                        APIMib(new String[]
{Work_SWITCH_1000_B.getIPSwitch(),Work_SWITCH_1000_B.getOIDFolha()[i]});
                }
            }
            if (swi.equals("SWITCH 2810 'A'")){
                for(int i = 0; i < 23; i++){
                    if (lea.equals(Work_SWITCH_2810_A.getFolha()[i]))
                        APIMib(new String[]
{Work_SWITCH_2810_A.getIPSwitch(),Work_SWITCH_2810_A.getOIDFolha()[i]});
                }
            }
            if (swi.equals("SWITCH 2810 'B'")){
                for(int i = 0; i < 23; i++){
                    if (lea.equals(Work_SWITCH_2810_B.getFolha()[i]))
                        APIMib(new String[]
{Work_SWITCH_2810_B.getIPSwitch(),Work_SWITCH_2810_B.getOIDFolha()[i]});
                }
            }
            if (swi.equals("SWITCH 2810 'C'")){
                for(int i = 0; i < 23; i++){
                    if (lea.equals(Work_SWITCH_2810_C.getFolha()[i]))
                        APIMib(new String[]
{Work_SWITCH_2810_C.getIPSwitch(),Work_SWITCH_2810_C.getOIDFolha()[i]});
                }
            }
            if (swi.equals("SWITCH 2810 'D'")){
                for(int i = 0; i < 23; i++){
                    if (lea.equals(Work_SWITCH_2810_D.getFolha()[i]))
                        APIMib(new String[]
{Work_SWITCH_2810_D.getIPSwitch(),Work_SWITCH_2810_D.getOIDFolha()[i]});
                }
            }
            if (swi.equals("SWITCH 2810 'E'")){
```

```
                for(int i = 0; i < 23; i++){
                    if (lea.equals(Work_SWITCH_2810_E.getFolha()[i]))
                        APIMib(new String[]
{Work_SWITCH_2810_E.getIPSwitch(),Work_SWITCH_2810_E.getOIDFolha()[i]});
                }
            }
            if (swi.equals("SWITCH 2810 'F'")){
                for(int i = 0; i < 23; i++){
                    if (lea.equals(Work_SWITCH_2810_F.getFolha()[i]))
                        APIMib(new String[]
{Work_SWITCH_2810_F.getIPSwitch(),Work_SWITCH_2810_F.getOIDFolha()[i]});
                }
            }
            if (swi.equals("SWITCH 2810 'G'")){
                for(int i = 0; i < 23; i++){
                    if (lea.equals(Work_SWITCH_2810_G.getFolha()[i]))
                        APIMib(new String[]
{Work_SWITCH_2810_G.getIPSwitch(),Work_SWITCH_2810_G.getOIDFolha()[i]});
                }
            }
        }
    }
});

    res.setBackground(Color.yellow);
    int mode = TreeSelectionModel.SINGLE_TREE_SELECTION;
    tree.getSelectionModel().setSelectionMode(mode);
    tree.putClientProperty("JTree.lineStyle","Angled");
    JScrollPane scr = new JScrollPane(tree);
    JScrollPane scr1 = new JScrollPane(res);
    contentPane.add("Lefth",scr);
    contentPane.add("Rigth",scr1);
}
//*************************************Finalizaçao do Método Construtor*****************************//

//**************Inicio do método que acessa a Mib: parametro de duas posições(IP,OID)********//

private void APIMib(String[] marg){
  try{
    this.setCursor(Cursor.WAIT_CURSOR);
    SnmpAPI api = new SnmpAPI();
    SnmpPDU pdu = new SnmpPDU();
    SnmpVar var = null;
    SnmpOID oid = new SnmpOID(marg[1]);
    api.start();
    api.setDebug(true);
    SnmpSession session = new SnmpSession(api);
    try{
       session.open();
       res.append("ABRINDO SESSION\n\n");
    }catch(SnmpException e){
       res.append("Erro abrindo o socket: "+e+'\n');
         this.setCursor(Cursor.DEFAULT_CURSOR);
    }
    session.setPeername(marg[0]);
    try{
      var = session.get(oid);
      res.append("RECEBENDO REQUISICAO\n\n");
```

```
      }catch(SnmpException e){
         res.append("Erro enviando requisição SNMP: "+e+'\n');
         this.setCursor(Cursor.DEFAULT_CURSOR);
      }
      pdu.setCommand(api.GET_REQ_MSG);//aqui
      pdu.addNull(oid);
      try{
         pdu = session.syncSend(pdu);
      }catch(SnmpException e){
         res.append("Erro recebendo requisição SNMP sincronizada: "+e+'\n');
      }
      try{
         address = InetAddress.getLocalHost();
      }catch(UnknownHostException w){
         this.setCursor(Cursor.DEFAULT_CURSOR);
      }
      res.append("COMUNIDADE: "+pdu.getCommunity()+'\n');
      res.append("ENDERECO REMOTO:
"+pdu.getRemoteHost()+":"+session.getRemotePort()+'\n');
      res.append("ENDERECO LOCAL: "+address.toString()+":"+session.getLocalPort()+'\n');
      res.append(oid.toTagString().toUpperCase()+'\n');
      res.append(var.toTagString().toUpperCase()+'\n');
      res.append("PORTA: "+String.valueOf(session.getLocalPort())+'\n');
      res.append("PORTA REMOTA: "+String.valueOf(session.getRemotePort())+'\n');
      res.append("TAM. BUFFER: "+String.valueOf(session.getPacketBufferSize())+"\n\n\n");
      session.close();
      api.close();
    }finally{
      this.setCursor(Cursor.DEFAULT_CURSOR);
    }

  }

  //************Declaraçao de Objetos que serão utilizados na aplicaçao***********//
  InetAddress address;
  //**************************Vetor das Folhas do Atm*****************************//
  private String[] LeafAtm = {"SwitchSTPConfig",
                              "AtmfPortMyIfName",
                              "AtmfPortMyIfIdentifier",
                              "AtmfMyIpNmAddress",
                              "AtmfMyOsiNmNsapAddress",
                              "Index",
                              "MaxVPCs",
                              "MaxVCCs",
                              "ConfiguredVPCs",
                              "ConfiguredVCCs",
                              "MaxVpibits",
                              "MaxVcibits",
                              "Unitype",
                              "UniVersion",
                              "DeviceType",
                              "IlmiVersion",
                              "NniSigVersion",
                              "PortIndex",
                              "OperStatus",
                              "ServiceCategory",
                              "AtmfNetPrefixPort",
                              "AtmfNetPrefixPrefix",
```

```
                    "AtmfAddressStatus",
                    "AtmfSrvcRegPort",
                    "atmfSrvcRegServiceID",
                    "atmfSrvcRegATMAddress"};
//*******************************Vetor das Folhas do Work************************//
private String[] LeafWork ={"SysContact",
                    "sysDescr",
                    "SysUpTime",
                    "ifDescr",
                    "ifInDiscards",
                    "ifSpeed",
                    "ipAdEntBcastAddr",
                    "ipForwDatagrams",
                    "ipRouteDest",
                    "icmpInDestUnreachs",
                    "icmpInEchoReps",
                    "icmpInMsgs",
                    "tcpActiveOpens",
                    "tcpAttemptFails",
                    "tcpCurrEstab",
                    "tcpEstabResets",
                    "udpInDatagrams",
                    "UdpInErrors",
                    "udpNoPorts",
                    "egpOutErrors",
                    "egpOutMsgs",
                    "snmpOutPkts",
                    "SnmpOutTraps"};

//*****************************Vetor de IPs dos Switchs*************************//
private String[] IPSwitch   =   {"192.168.1.1",
                                 "192.168.1.2",
                                 "192.168.1.3",
                                 "192.168.1.4",
                                 "192.168.1.5",
                                 "192.168.1.11",
                                 "192.168.1.12",
                                 "192.168.1.9",
                                 "192.168.1.10",
                                 "192.168.1.20",
                                 "192.168.1.40",
                                 "192.168.1.41",
                                 "192.168.1.42",
                                 "192.168.1.43",
                                 "192.168.1.44",
                                 "192.168.1.47",
                                 "192.168.1.49",
                                 "192.168.1.52"};

//*****************************Vetor dos IDs das Folhas*************************//

//*************************************FOLHAS ATM*******************************//

private String[] OIDLeaf    =   {".1.3.6.1.4.1.1012.81.1.5.1.1.1",
                                 ".1.3.6.1.4.1.1012.81.1.5.1.1.2",
                                 ".1.3.6.1.4.1.1012.81.1.5.1.1.3",
                                 ".1.3.6.1.4.1.1012.81.1.5.1.1.4",
                                 ".1.3.6.1.4.1.1012.81.1.5.1.1.5",
```

```
                                        ".1.3.6.1.4.1.1012.81.1.5.1.1.6",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.1.7",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.1.8",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.1.9",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.1.10",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.2.1",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.4.3",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.4.4",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.4.5",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.5.1",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.5.2",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.5.3",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.6.1",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.6.2",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.6.3",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.7.1",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.7.2",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.7.3",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.8.1",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.8.2",
                                        ".1.3.6.1.4.1.1012.81.1.5.1.8.3"};

//*************************************FOLHAS ATM*********************************//

  private String[] OIDLeafWork     =     {".1.3.6.1.2.1.1.1",
                                          ".1.3.6.1.2.1.1.2",
                                          ".1.3.6.1.2.1.1.3",
                                          ".1.3.6.1.2.1.1.4",
                                          ".1.3.6.1.2.1.1.5",
                                          ".1.3.6.1.2.1.1.6",
                                          ".1.3.6.1.2.1.1.7",
                                          ".1.3.6.1.2.1.2.1",
                                          ".1.3.6.1.2.1.4.3",
                                          ".1.3.6.1.2.1.4.4",
                                          ".1.3.6.1.2.1.4.5",
                                          ".1.3.6.1.2.1.5.1",
                                          ".1.3.6.1.2.1.5.2",
                                          ".1.3.6.1.2.1.5.3",
                                          ".1.3.6.1.2.1.6.1",
                                          ".1.3.6.1.2.1.6.2",
                                          ".1.3.6.1.2.1.6.3",
                                          ".1.3.6.1.2.1.7.1",
                                          ".1.3.6.1.2.1.7.2",
                                          ".1.3.6.1.2.1.7.3",
                                          ".1.3.6.1.2.1.8.1",
                                          ".1.3.6.1.2.1.8.2",
                                          ".1.3.6.1.2.1.8.3"};


 //************************** Vetor de Objetos para cada switch************************//

  private SwitchLeaf Atm_ASX_1000_A      = new SwitchLeaf(LeafAtm,OIDLeaf,IPSwitch[0]);
  private SwitchLeaf Atm_ASX_1000_B      = new SwitchLeaf(LeafAtm,OIDLeaf,IPSwitch[1]);
  private SwitchLeaf Atm_CORE_BUILDER_7000= new
SwitchLeaf(LeafAtm,OIDLeaf,IPSwitch[2]);
  private SwitchLeaf Atm_ASX_200_BX_A    = new SwitchLeaf(LeafAtm,OIDLeaf,IPSwitch[3]);
  private SwitchLeaf Atm_ASX_200_BX_B    = new SwitchLeaf(LeafAtm,OIDLeaf,IPSwitch[4]);
  private SwitchLeaf Atm_ESR_5000_A      = new SwitchLeaf(LeafAtm,OIDLeaf,IPSwitch[5]);
```

```
  private SwitchLeaf Atm_ESR_5000_B      = new SwitchLeaf(LeafAtm,OIDLeaf,IPSwitch[6]);
  private SwitchLeaf Atm_NSX_9500_A      = new SwitchLeaf(LeafAtm,OIDLeaf,IPSwitch[7]);
  private SwitchLeaf Atm_NSX_9500_B      = new SwitchLeaf(LeafAtm,OIDLeaf,IPSwitch[8]);

  private SwitchLeaf Work_SWITCH_1000_A = new
SwitchLeaf(LeafWork,OIDLeafWork,IPSwitch[9]);
  private SwitchLeaf Work_SWITCH_1000_B = new
SwitchLeaf(LeafWork,OIDLeafWork,IPSwitch[10]);
  private SwitchLeaf Work_SWITCH_2810_A = new
SwitchLeaf(LeafWork,OIDLeafWork,IPSwitch[11]);
  private SwitchLeaf Work_SWITCH_2810_B = new
SwitchLeaf(LeafWork,OIDLeafWork,IPSwitch[12]);
  private SwitchLeaf Work_SWITCH_2810_C = new
SwitchLeaf(LeafWork,OIDLeafWork,IPSwitch[13]);
  private SwitchLeaf Work_SWITCH_2810_D = new
SwitchLeaf(LeafWork,OIDLeafWork,IPSwitch[14]);
  private SwitchLeaf Work_SWITCH_2810_E = new
SwitchLeaf(LeafWork,OIDLeafWork,IPSwitch[15]);
  private SwitchLeaf Work_SWITCH_2810_F = new
SwitchLeaf(LeafWork,OIDLeafWork,IPSwitch[16]);
  private SwitchLeaf Work_SWITCH_2810_G = new
SwitchLeaf(LeafWork,OIDLeafWork,IPSwitch[17]);
}
class SwitchLeaf {

   /********************************* Creates new SwitchLeaf *********************************/
   public SwitchLeaf() {

   }
   public SwitchLeaf(String[] leaf,String[] oidLeaf,String ipSwitch) {
      Folha    = leaf;
      OIDFolha = oidLeaf;
      IPSwitch = ipSwitch;
   }

   public String[] getFolha(){
      return Folha;
   }
   public String getIPSwitch(){
      return IPSwitch;
   }
   public String[] getOIDFolha(){
      return OIDFolha;
   }
   private String[] Folha;
   private String IPSwitch;
   private String[] OIDFolha;
}

 /********************************* Classe de criação do Aglet *********************************/

public class SimpleTree extends Aglet{
        public void OnCreate(){
                Frame f = new SimpleTreeFrame();
                f.show();
        }
        public void run() {
        if (!tm.isRunning())
```

```
            tm.start();
        }
Timer tm = new Timer(20000, new ActionListener(){
public void actionPerformed(ActionEvent ev){
  dispatch("atp://NOTEBOOK:1433");
}
});
}
```