CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIAS

PROGRAMA DE PÓS-GRADUAÇÃO *STRICTO SENSU*

CARLOS ROBERTO SCHIMIDT

DESENVOLVIMENTO DE GERÊNCIA *SNMP* PARA DISPOSITIVOS DE REDES TOTALMENTE ÓPTICAS

CAMPINAS – SP
2007

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

GRÃO-CHANCELER
Dom Bruno Gamberini

MAGNÍFICO REITOR
Prof. Pe. Wilson Denadai

VICE-REITOR
Profa. Dra. Angela de Mendonça Engelbrecht

PRÓ-REITORA DE PESQUISA E PÓS-GRADUAÇÃO
Profa.Dra. Vera Engler Cury

DIRETOR DO CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIAS
Prof. Dr. Orandi Mina Falsarella

COORDENADOR DO CURSO DE MESTRADO PROFISSIONAL EM GESTÃO DE REDES DE TELECOMUNICAÇÕES
Área de Concentração: Gestão de Redes e Serviços
Prof. Dr. Omar Carvalho Branquinho

CARLOS ROBERTO SCHIMIDT

DESENVOLVIMENTO DE GERÊNCIA *SNMP* PARA DISPOSITIVOS DE REDES TOTALMENTE ÓPTICAS

Dissertação apresentada ao Curso de Mestrado Profissional em Gestão de Redes de Telecomunicações do Centro de Ciências Exatas, Ambientais e de Tecnologias da Pontifícia Universidade Católica de Campinas como requisito parcial para obtenção do título de Mestre em Gestão de Redes de Telecomunicações.
Área de concentração: Gestão de Redes e Serviços.

Orientador: Prof. Dr. Marcelo Luís Francisco Abbade.

CAMPINAS – SP
2007

Ficha Catalográfica
Elaborada pelo Sistema de Bibliotecas e
Informação - SBI - PUC-Campinas

t658.4038 Schimidt, Carlos Roberto
C284g Gerenciamento SNMP para dispositivos de redes totalmente ópticas / Carlos Roberto Schimidt - Campinas: PUC-Campinas, 2007.
113p.

Orientador: Marcelo Luís Francisco Abbade.
Dissertação (mestrado) – Pontifícia Universidade Católica de Campinas, Centro de Ciências Exatas, Ambientais e de Tecnologias, Pós-Graduação em Engenharia Elétrica.
Inclui anexos e bibliografia.

1. Gerenciamento da informação. 2. Redes de informação. 3. Internet (Redes de computação). 4. Guias de ondas óticas. 5. Software livre. I. Abbade, Marcelo Luís Francisco. II. Pontifícia Universidade Católica de Campinas. Centro de Ciências Exatas, Ambientais e de Tecnologias. Pós-Graduação em Engenharia Elétrica. III. Título.

22.ed.CDD – t658.4038

CARLOS ROBERTO SCHIMIDT

DESENVOLVIMENTO DE GERÊNCIA SNMP PARA DISPOSITIVOS DE REDES TOTALMENTE ÓPTICAS

Dissertação apresentada ao Curso de Mestrado Profissional em Gestão de Redes de Telecomunicações do Centro de Ciências Exatas, Ambientais e de Tecnologias da Pontifícia Universidade Católica de Campinas como requisito parcial para obtenção do título de Mestre em Gestão de Redes de Telecomunicações
Área de Concentração: Gestão de Redes e Serviços .
Orientador: Prof. Dr. Marcelo Luís Francisco Abbade

Dissertação defendida e aprovada em 08 de fevereiro de 2007 pela Comissão Examinadora constituída dos seguintes professores:

Prof. Dr. Marcelo Luís Francisco Abbade
Orientador da Dissertação e Presidente da Comissão Examinadora
Pontifícia Universidade Católica de Campinas.

Profa. Dr. Omar Carvalho Branquinho
Pontifícia Universidade Católica de Campinas.

Prof. Dr. Edson Moschim
Universidade Estadual de Campinas

DEDICATÓRIA

À memória da minha avó Francisca, cuja
educação e orientação que me forneceu,
colaborou para a minha formação,
personalidade e caráter.

AGRADECIMENTOS

A Deus,

Pela saúde e condições para desenvolver e concluir este trabalho.

À minha amada esposa Lysiée,

Pela tolerância e compreensão em relação à minha ausência enquanto envolvido nas pesquisas e atividades que esta dissertação requereu.

À minha filha Bruna,

Que inspira-me a desenvolver-me e servir como exemplo para seu futuro.

Ao meu orientador Professor Doutor Marcelo Luís Francisco Abbade,

Pela chance de compor seu quadro de pesquisadores, pela liberdade de expressão, e também pela orientação que conjuminou na conclusão deste trabalho.

À minha cunhada Flávia,

Pelas dicas e correções gramaticais da língua inglesa.

Aos meus amigos,

Que por mais insignificante que pensem ter sido a colaboração para este trabalho, foram de suma importância para a qualidade do mesmo.

“A satisfação está no esforço feito para
alcançar o objetivo, e não em tê-lo alcançado. ”

(Gandhi)

RESUMO

SCHIMIDT, Carlos Roberto. **Desenvolvimento de Gerência *SNMP* para Dispositivos de Redes Totalmente Ópticas.** 2007. 113 f.. Dissertação (Mestrado em Engenharia Elétrica). Centro de Ciências Exatas, Ambientais e de Tecnologias. Pontifícia Universidade Católica de Campinas, Campinas.

Conversores ópticos de comprimento de onda são equipamentos que podem reduzir o congestionamento e aumentar a eficiência da próxima geração de redes ópticas com multiplexação por divisão em comprimentos de onda (*wavelength division multiplexing*, WDM). Atualmente, nosso grupo de pesquisas está desenvolvendo um protótipo deste conversor dentro do Projeto KyaTera promovido pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). Neste contexto, o objetivo deste trabalho foi o de implementar um *software* de código aberto capaz de gerenciar, de maneira dinâmica, a operação do conversor em uma rede totalmente óptica. Para isso, o trabalho foi dividido em duas etapas. Na primeira, as funções de gerenciamento do modelo *Telecommunications Management Network* (TMN) foram utilizadas para definir os parâmetros de gerenciamento do equipamento. A partir destas definições, estruturou-se e implementou-se uma *Management Information Base (MIB)* para o referido equipamento. A segunda etapa consistiu em desenvolver e implementar um agente *Simple Network Management Protocol* (*SNMP)* capaz de realizar a comunicação entre a gerência de redes e a *MIB* do equipamento conversor. Ambos os *software*s foram testados e funcionaram de acordo com o esperado. O *software* desenvolvido também pode ser utilizado para realizar a comunicação entre uma gerência centralizada e nós ópticos que não façam a conversão de comprimentos de onda ou, mediante pequenas adaptações, pode ser usado para gerenciar outros equipamentos de redes totalmente ópticas. Por essas razões, o trabalho pode ser útil para outros grupos de pesquisa ou para empresas que necessitem avaliar novas tecnologias em redes ópticas, como a implementação de algoritmos de alocação de comprimentos de onda ou de estratégias para a monitoração da qualidade de sinais. Com o intuito de auxiliar o trabalho de outros grupos que precisem realizar a gerência de equipamentos, um apêndice dessa dissertação também apresenta uma descrição detalhada dos procedimentos (nem sempre explicitamente indicados na literatura) necessários para atingir esse objetivo.

PALAVRAS-CHAVE: Gerenciamento de Rede; protocolo SNMP; conversor de comprimento de ondas; redes ópticas.

ABSTRACT

SCHIMIDT, Carlos Roberto. **Development of *SNMP* Management for All Network Optical Devices.** 2007. 113 f.. Dissertação (Mestrado em Engenharia Elétrica). Centro de Ciências Exatas, Ambientais e de Tecnologias. Pontifícia Universidade Católica de Campinas, Campinas.

Optical wavelength converters (OWC) are devices that can reduce congestion and improve the efficiency of the next generation of wavelength division multiplexing (WDM) optical networks. Currently, our research group is developing a prototype of this converter in the Kyatera Project funded by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). In this perspective, the objective of this work was to implement an open source software to dynamically manage the wavelength converter prototype in an all optical network. This task was performed in two phases. In the first one, the Telecommunications Management Network (TMN) functionalities were utilized to define the device parameters that should be managed. After this, a management information base (MIB) was structured for the OWC. In the second phase, a SNMP agent was developed and implemented in order to allow the communication between network management and the OWC MIB. Both software programs were tested and worked as expected. The software programs could also be utilized in the communication between a central management and optical nodes that do not perform wavelength conversions or after some adjustments, to manage other all-optical devices. Therefore, this effort may contribute with other research groups or companies that need to evaluate new optical network technologies, such as routing wavelength assignment algorithms and the monitoring of signal quality. For those interested, an appendix fully describes the procedures for managing new devices.

KEY WORDS: Management Network; SNMP protocol; wave length converter; network optical.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	=	Application Programming Interface
AGS	=	Acordo de Gerenciamento de Serviço
ANS	=	Acordo de Nível de Serviço
AON	=	All Optical Network
ASN.1	=	Abstract Syntax Notation One
CMIP	=	Common Management Information Protocol
COBIT	=	Control Objectives for Information and related Technology
CPU	=	Central Processing Unit
FAPESP	=	Fundação de Amparo à Pesquisa do Estado de São Paulo
FTP	=	File Transfer Protocol
FWM	=	Four-Wave Mixing
HEMS	=	High-level Entity Monitoring System
HTTP	=	Hypertext Transfer Protocol
IANA	=	Internet Assigned Numbers Autorithy
IEEE	=	Institute of Electrical and Eletronics Engineers
IETF	=	Internet Engineering Task Force
IP	=	Internet Protocol
ISACA	=	Information Systems Control Journal
ISO	=	International Organization for Standardization
ITU	=	International Telecommunication Union
LMT	=	Laboratório de Meios de Transmissão
MIB	=	Management Information Base
MRTG	=	Multi Router Traffic Grapher
OBPF	=	Optical Band-Pass Filter
OBS	=	Optical Burst Switched
OCS	=	Optical Circuit-Switched
OID	=	Object Identifier
OEO	=	Ópticas-Eletrônicas-Ópticas
OPS	=	Optical Packet-Switched
PC	=	Polarization Controller
PDU	=	Protocol Data Unit
PEN	=	Private Enterprise Number
PM	=	Power Meter
PUC	=	Pontifícia Universidade Católica
QAs	=	Q-Adapters
QoS	=	Quality of Service
RFC	=	Request for Comment
RMON	=	Remote Network Monitoring
SMI	=	Structure Management Information
SNMP	=	Simple Network Management Protocol
TCP	=	Transmission Control Protocol
TMN	=	Telecommunications Management Network
UDP	=	User Datagram Protocol
WDM	=	Wavelength Division Multiplexing

SUMÁRIO

CAPÍTULO 1 - INTRODUÇÃO

Com a tecnologia cada vez mais presente nas funcionalidades empresariais, a busca por níveis de disponibilidade e desempenho cada vez mais eficazes tem se mostrado cada vez maior, forçando a instauração de acordo de níveis de serviço para garantir que os recursos estejam disponíveis quando necessários, e não incorram em tempo demasiado para retornar o efeito desejado.

A preocupação com a qualidade e disponibilidade dos serviços é constante, tanto que novas abordagens tecnológicas procuram servir como instrumentos para atingir tais propósitos, como citado em publicação do *Information Systems Control Journal (ISACA)* : "...acordos de níveis de serviços (ANS) e acordos de gerenciamento de serviços (AGS), suportados por mecanismos como *Control Objectives for Information and related Technology (COBIT)* e *balanced scorecard* (BSC)." (AMELINCKX; HAES; GREMBERGEN, 2003, p1, tradução nossa).

Entretanto, somente grandes corporações possuem condição financeira a ponto de custear um projeto de gerenciamento de infra-estrutura de rede confiável e globalizado (LIMA, 1997), valendo-se de soluções comerciais de grandes fabricantes de softwares.

No entanto os pacotes comerciais limitam-se a proporcionar um gerenciamento generalizado, pois os agentes de gerenciamentos são desenvolvidos especificamente para atender necessidades de cada fabricante de equipamento.

Com base nesta alegação, destacam-se no mercado na atual conjuntura, os seguintes softwares de gerenciamento de redes, com seus respectivos fabricantes.

Transcend - Fabricado pela empresa *3COM*, para gerenciar sobretudo seus equipamentos.

CiscoWorks - Fabricado pela empresa *CISCO*, para gerenciar sobretudo seus equipamentos

HP-OpenView - Fabricado pela empresa *HP*, com inúmero agentes para o gerenciamento de equipamentos de outros fabricantes.

Spectrum - Desenvolvido inicialmente pela empresa *Cabletron*, que a posteori denominou-se *Enterasys*. Contudo este produto foi adquirido pela empresa Aprisma, que por sua vez foi incorporada à empresa *Concord*, e esta última adquirida pela empresa *Computer Associate*.

Tivoli Omegamon - A plataforma de gerenciamento *Tivoli* foi desenvolvida pela empresa *IBM*, entretanto com a aquisição da empresa *Candle*, está ocorrendo uma fusão das funcionalidades da plataforma *Tivoli* com a plataforma *Candle Omegamon* para tornarem-se um único produto.

Unicenter TNG - Conjunto de produtos desenvolvidos pela empresa *Computer Associate*, que contém uma plataforma de gerenciamento. Ainda não definido qual o destino da plataforma de gerenciamento, uma vez que esta empresa incorporou ferramentas das empresas adquiridas *Concord* e *Aprisma*, com a aquisição da empresa *Concord*.

Percebe-se que a maioria dos softwares de gerenciamento de redes, são direcionados para equipamentos e componentes que são de própria fabricação da empresa. Contudo todos estes softwares são baseados no protocolo de gerenciamento *Simple Network Management Protocol (SNMP)* que faz parte da arquitetura de protocolos denominada *Transmission Control Protocol / Internet Protocol (TCP/IP)*.

Existem boas publicações de fabricantes de equipamentos disponíveis na Internet para auxiliar no entendimento do gerenciamento de rede, contudo são bastante direcionadas para seus produtos, como é o caso da HP em "*HP OpenView Extensible SNMP Agent Administrator´s Guide*" (Hewlett-Packard Development Company, 2004) e da IBM em "*Managing AIX Server Farms*" (PARKER, 2005).

Determinado fabricante de equipamento que deseja desenvolver um novo produto e lançá-lo no mercado, sofre limitações para disponibilizar um bom gerenciamento de seu produto, uma vez que terá que construir sua base de gerenciamento seguindo apenas os padrões de gerenciamento *SNMP* descritos em documentos denominados *Request for Comment (RFC´*s).

O próprio padrão *SNMP* evoluiu conforme exigências e necessidades do mercado, disponibilizando extensibilidade (MAURO; SCHMIDT, p. 173, 2001) de forma a proporcionar atributos gerenciáveis personalizados de acordo com cada equipamento. Embora “proporcione aos usuários uma simples configuração de operações que permite que os mecanismos sejam gerenciados remotamente” (MAURO; SCHMIDT , 2001, 1 ed, p. 1, tradução nossa).

Entretanto, vale ressaltar que para os grandes desenvolvedores de soluções para gerenciamento de redes suportarem extensibilidade para o gerenciamento de produtos de outros fabricantes seria preciso incorrer em custos e esforços que talvez não tragam o retorno necessário.

Sendo assim, existe um árduo caminho a ser percorrido pelos novos fabricantes de equipamento, muitas vezes tendo que inicialmente desenvolver soluções proprietárias para o gerenciamento do dispositivo.

Por outro lado, as empresas que compram ou montam soluções para o gerenciamento de sua plataforma tecnológica, evitam ao máximo ter uma solução heterôgenea, pois teriam que investir em muitos profissionais com características e conhecimentos distintos, sem contar a dificuldade em diagnosticar problemas entre soluções incompatíveis.

Com o advento de redes totalmente ópticas, e o surgimento de equipamentos para tirar maior proveito desta tecnologia, emerge a necessidade de pesquisar um gerenciamento efetivo destas redes, e a utilização do padrão *SNMP* para gerenciar os dispositivos desta rede vem de encontro a manter a compatibilidade com os softwares de gerenciamento existentes no mercado.

Este trabalho está relacionado com o projeto do Laboratório de Meios de Transmissão (LMT) da PUC-Campinas junto ao Programa Tidia/KyaTera, promovido pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). O Programa *Tidia/KyaTera* tem por objetivo "incentivar a pesquisa científica e tecnológica em projetos cooperativos para o estudo de tecnologias da Internet Avançada, apoiado por uma rede de fibras ópticas que interliga os laboratórios participantes, favorecendo a demonstração experimental e a aplicação das tecnologias propostas." (FAPESP, 2006). Neste contexto, o grupo de pesquisadores do LMT propôs o desenvolvimento de um dispositivo chamado conversor de comprimentos de onda totalmente óptico, que tem por objetivo reduzir a probabilidade de bloqueio de chamadas em redes com mutiplexação por divisão em comprimentos de onda (*wavelength division multiplexing* - *WDM*) e, conseqüentemente, aumentar a eficiência de utilização destas redes.

Neste trabalho descreveremos o desenvolvimento e a implementação de um *software* para gerenciar esse dispositivo conversor. Ao longo do texto, também estaremos descrevendo detalhadamente um procedimento para implementações de uma base de gerenciamento de informações (*management information base*, *MIB*) e de um software chamado agente *SNMP*, que é utilizado pela gerência de redes para acessar essa base. Devido à escassez de detalhes presente na literatura atual (McGINNIS; PERKINS, 1996), (TOWNSEND, 1995), essa descrição pode ser relevante para outros grupos de pesquisa, em especial aqueles ligados ao Programa *Tidia/Kyatera*, e também por profissionais que pretendam desenvolver ferramentas de gerenciamento para seus dispositivos. Além disso, acreditamos que as ferramentas de gerência aqui apresentadas possam ser adaptadas, com certa facilidade, para gerenciar outros dispositivos dos nós de comutação de redes ópticas e/ou totalmente ópticas.

A organização do restante deste capítulo é a seguinte. Na seção 1.1 daremos uma introdução do conceito de comutação. Na seção 1.2 descreveremos uma rede totalmente óptica e na seção 1.3 a importância do conversor de comprimento de ondas dentro desta estrutura. Na seção 1.4 serão discutidos os objetivos e métodos do trabalho utilizados para o desenvolvimento e implementação do agente SNMP para o conversor de comprimento de ondas. Na

seção 1.5 é referente à justificativa do trabalho, ou seja, quais as contribuições que este trabalho fornecerá para a comunidade. Por último a seção 1.6 descreverá como foi estabelecida a organização desta dissertação.

1.1 Comutação

O tráfego nas redes de computadores ocorre através de comutação, seja por circuito, pacote ou mensagens (COLCHER; LEMOS; SOARES, p. 75-80, 1995).

A comunicação em uma rede comutada por circuito está subdividida em três etapas: o estabelecimento do circuito, a conversação e a desconexão do circuito.

No estabelecimento do circuito define-se uma rota fixa entre os nós por onde a comunicação será realizada através de um canal dedicado, e após um período indeterminado a conexão é encerrada.

Os canais de comunicação em comutação de circuitos podem ser alocados através de chaveamento espacial, chaveamento de freqüências ou chaveamento do tempo.

O chaveamento espacial estabelece um enlace físico permanente durante toda a comunicação entre dois nós através de chaves físicas existentes nos nós intermediários que interconectam suas portas.

No chaveamento de freqüências é estabelecido uma associação entre dois canais de freqüências em cada enlace, onde um nó intermediário ao receber sinal de uma onda portadora de determinada freqüência, realiza a filtragem e demodulação deste sinal para sua posterior modulação e transmissão na outra frequência associada.

No chaveamento de tempo, os nós intermediários possuem uma associação de um canal *Time Division Multiplex (TDM)* síncrono de uma linha

com o canal *TDM* de outra linha, demultiplexando o sinal de um circuito desejado para ser multiplexado e encaminhado para outro nó.

Uma rede comutada por circuito é ideal para veicular fluxo contínuo e constante da informação, ou seja, a transferência de dados entre dois pontos após o estabelecimento do circuito, com grande vantagem de não necessitar de empacotamento de bits para a transmissão, mas impede uma versatibilidade para estabelecer troca de comunicação entre diversos pontos.

Para melhor entendimento, pacote é onde as informações são preparadas para serem transmitidas até o nó destino, onde no contexto de redes, trata-se de um conjunto de bits compreendendo informação de controle, endereço de origem e destino dos nós envolvidos na transmissão.

Na Comutação por Pacote, o tamanho da unidade de transmissão de dados é limitado, onde as mensagens que excederem este limite devem ser quebradas em unidades menores denominadas pacotes (SOARES, 1995).

Cada pacote pode seguir caminhos distintos, uma vez que a cada roteador pelo qual o pacote passa uma decisão é tomada para definir qual a melhor rota o pacote deve ser reencaminhado de acordo com as métricas do protocolo de roteamento configurado no roteador. Explanação detalhada sobre protocolo de roteamento está fora do contexto deste trabalho, mas um melhor en tendimento do assunto pode ser verificado em GOUGH, 2001.

Nas redes comutadas por pacote, os pacotes chegam ao destino não na mesma ordem em que foram enviados, sendo necessário o armazenamento temporário destes pacotes no destino para que a ordem seja restabelecida.

Nas redes Comutadas por Mensagens as mensagens são roteadas completas pela rede, um nó por vez, onde as mesmas possuem uma identificação da origem e destino da informação em seu cabeçalho.

1.2 Enlaces Ópticos e Redes Totalmente Ópticas

A utilização da fibra óptica para interligação de nós da rede possibilitou uma largura de banda passante bastante maior que a oferecida por cabos e por enlaces sem-fio, além do incremento na taxa de transmissão entre os pontos interligados. De fato, a largura de banda oferecida na terceira janela das fibras (região de 1550 nm) é da ordem de 12 THz. Para aumentar a eficiência de exploração dessa largura de banda, foram introduzidos na década de 1990 os sistemas de multiplexação por divisão em comprimento de onda (*wavelength division multiplexing- WDM*), nos quais sinais modulados em diferentes portadoras são multiplexados em uma única fibra (RAMASWAMI; SIVARAJAN, 2002). A Figura 1 ilustra um enlace WDM.

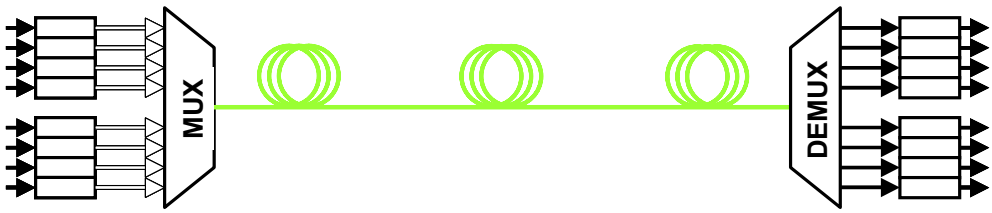

Figura 1. Enlace WDM ponto-a-ponto

A fim de padronizar a fabricação de equipamentos para enlaces e redes *WDM*, o *International Telecommunication Union (ITU-T*) elaborou a recomendação G.694.1 que estabeleceu 193.1 THz como freqüência central de uma grade de canais *WDM* com espaçamento de 100 GHz entre cada canal. Posteriormente, a recomendação G.694.1 reduziu este espaçamento para 50 GHz.

Outro fator limitante para o aproveitamento da largura de banda oferecida pelas fibras são as conversões ópticas-eletrônicas-ópticas (OEO) de sinal realizadas pelos elementos de redes, tais como roteadores e *switches*. Como a taxa de processamento destes sinais é relativamente baixa, a eficiência da rede não atinge o melhor nível de transmissão.

Em função disso foram criados equipamentos para realizar opticamente a comutação dos sinais, sem as conversões OEO desnecessárias. Redes que utilizam esses equipamentos em seus nós são denominadas totalmente ópticas (*all optical network- AON*) (RAMASWAMI, 2002). A Figura 2 indica uma configuração de nó totalmente óptico na qual os sinais provenientes de cada fibra são demultiplexados e enviados para um *switch* totalmente óptico. Cada um destes *switches* opera com sinais de um único comprimento de onda e os comuta opticamente para os multiplexadores de saída. Apesar de o controle destes *switches* ser eletrônico, a transmissão dos sinais não sofre conversão OEO e, portanto, ela é transparente à taxa de transmissão e ao formato de modulação utilizados.

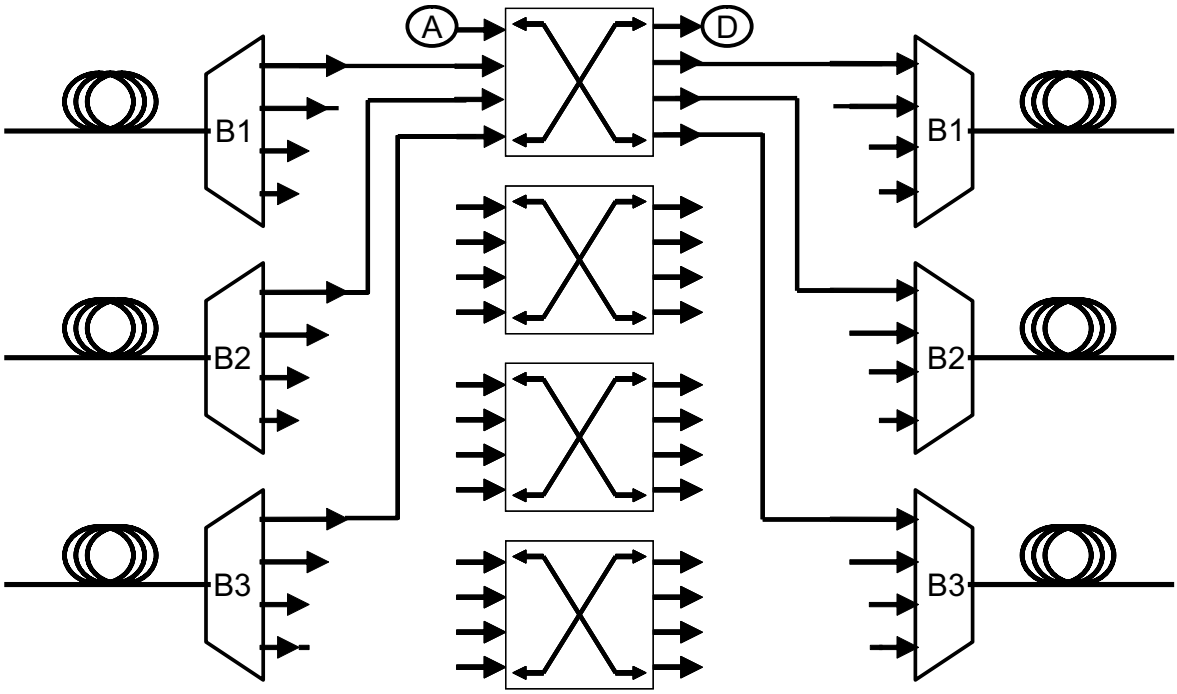

Figura 2. Representação de um nó totalmente óptico

Existem três estratégias básicas de utilização das redes totalmente ópticas: redes com comutação de circuitos ópticos (*optical circuit-switched, OCS*), redes com comutação de pacotes (*optical packet-switched OPS*) e redes com comutação de rajadas (*optical burst switched-OBS*).

A primeira dessas estratégias é a única que dispõe de tecnologia para ser implementada atualmente e consiste no estabelecimento de um circuito entre os nós clientes da rede. Em geral, este circuito é constituído por um conjunto de

enlaces de fibras e um determinado comprimento de onda (portadora óptica) em cada enlace. No entanto, as implementações atuais deste tipo de rede consideram que o mesmo comprimento de onda é utilizado em todos os enlaces, o que caracteriza uma condição chamada de continuidade de comprimentos de onda. Em média, cada circuito entre clientes é mantido por cerca de um ano (RAMASWAMI; SIVARAJAN, 2002).

Devido às suas complexidades tecnológicas, as redes *OPS* e *OBS* estão atualmente restritas a propostas teóricas e a plataformas de testes em laboratório e não possuem implementações comerciais. Por esta razão, este trabalho considerará apenas as redes *OCS*.

1.3 Conversores de Comprimento de Onda

É importante observar que a condição de continuidade de comprimento de onda também pode limitar a eficiência de utilização de uma rede totalmente óptica. Para exemplificarmos esta afirmação, consideramos a situação de uma rede hipotética de 4 nós ilustrada no diagrama apresentado na Figura 3. Nesta rede o nó B estabelece circuito com o nó D utilizando a freqüência[1] de portadora f_1.

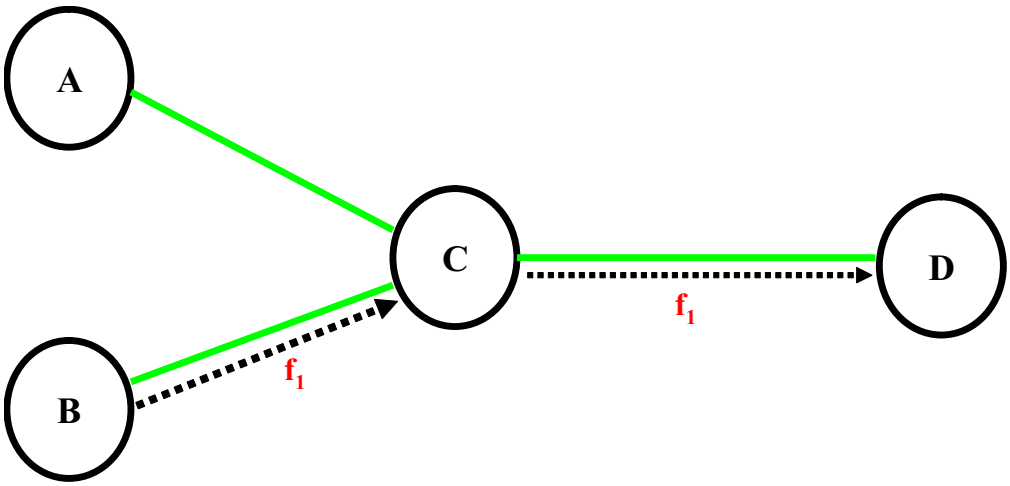

Figura 3. Nó B Transmitindo para o nó D em F_1

[1] A freqüência, f, está relacionada com o comprimento de onda, λ, através da bem conhecida relação f= c/ λ, na qual c é a velocidade de propagação da luz. Por este motivo, uma conversão de freqüências é equivalente a uma conversão de comprimentos de onda e, neste documento, utilizamos esses dois termos indistintamente.

Se o nó A precisar estabelecer um circuito de comunicação com o nó D, e utilizar a freqüência f_2, não haverá problema algum, pois no caminho em que o sinal percorrerá não existe utilização desta freqüência , conforme ilustrado no diagrama da Figura 4.

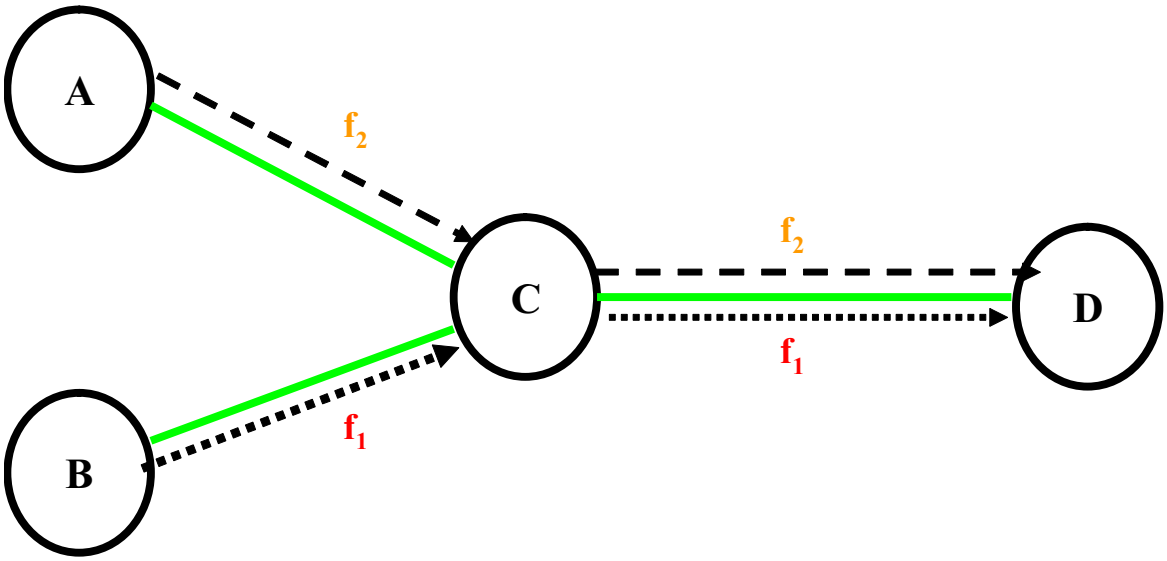

Figura 4. Nó A transferindo para nó D em F_2

No entanto, se o tráfego gerado no nó A aumentar pode ser que ele precise utilizar uma nova portadora para transmitir mais dados. Caso esta nova portadora esteja em f_1 a comunicação será possível entre os A e C mas não será possível entre os nós C e D, onde a portadora f_1 já está transmitindo dados do nó B. Dessa maneira, para respeitar a condição de continuidade de comprimento de onda, a gerência de rede deveria recusar o estabelecimento dessa chamada e isso reduziria a eficiência de aproveitamento da rede.

Uma alternativa para solucionar esse problema é utilizar um dispositivo que converta a portadora f_1 para uma outra portadora não utilizada no enlace entre os nós C e D, por exemplo, f_3. O dispositivo que realiza essa conversão é chamado de conversor de comprimentos de onda e existem várias tecnologias que podem ser utilizadas para implementar esta tecnologia.

Os conversores de comprimento de onda mais simples utilizam a conversão OEO, ou seja, o sinal em uma portadora de entrada é foto-detectado (conversão óptico-eletrônica) e, posteriormente, modula uma portadora de saída

(conversão eletro-óptica). Estes dispositivos são comerciais mas não podem ser utilizados em uma rede totalmente óptica.

Redes totalmente ópticas requerem conversores totalmente ópticos, que normalmente são baseados em efeitos não-lineares de fibras ópticas ou materiais semicondutores (RAMASWAMI; SIVARAJAN, 1996). O conversor considerado neste trabalho é baseado em um efeito que ocorre em fibras ópticas chamado mistura de quatro ondas (*four-wave mixing*, *FWM*). Abordaremos, de forma muito sucinta, os mecanismos físicos desta conversão no Capítulo CAPÍTULO 4 -.

No entanto, dentro do foco de gerência a que se propõe este trabalho, as características principais do conversor que devemos considerar são (SCHIMIDT; ABBADE, 2006) e (ABBADE, 2003):

a) o conversor deve responder automaticamente a uma demanda da rede, para converter um sinal em uma portadora f_i para uma portadora f_j;

b) Para realizar esta conversão, utilizando o efeito de *FWM* é necessário utilizar um sinal de potência elevada e constante que chamaremos de bombeio. A portadora do bombeio deve estar localizada em $(f_i+f_j)/2$;

c) O conversor pode degradar o sinal convertido e, portanto, devemos realizar alguma monitoração do sinal convertido para verificar se a degradação introduzida está em níveis aceitáveis. Neste trabalho a única monitoração realizada concerne à potência de saída do sinal convertido e numa situação em que o nó A precise encaminhar outra informação simultaneamente para o nó D em f_2 para poder utilizar este caminho. Isto é um problema que causa a sub-utilização da rede.

1.4 Objetivos e Métodos do Trabalho

Conforme indicado anteriormente esta dissertação tem como objetivos: a) descrever detalhadamente os procedimentos necessários para a implementação de uma *MIB* e de um agente *SNMP*, b) apresentar a

implementação destes dois softwares para o caso específico de um conversor totalmente óptico de comprimentos de onda.

Para tais realizações os seguintes passos foram utilizados. Primeiramente, realizamos uma pesquisa bibliográfica para compreender os conceitos de gerenciamento *SNMP*. Após isso, partimos do projeto do LMT junto ao programa Tidia/KyaTera para identificarmos quais atributos seriam passíveis de gerenciamento no conversor de comprimentos de onda totalmente óptico. Em seguida, utilizamos estas especificações para implementar a *MIB* do conversor e o agente *SNMP* correspondente. A *MIB* foi implementada fundamentalmente em acordo com as *RFC*'s 1098 e 1213 (McCLOGHRIE, 1991; CASE, 1998) que estabelece os padrões para *MIB-II e define o protocolo SNMP*. O agente *SNMP* foi escrito em linguagem C utilizando a ferramenta mib2c., que constitui um dos utilitários providos pelo *NET-SNMP* (NET-SNMP, 2005). Este trabalho também descreve os passos para se conseguir um número de registro junto ao *IANA* a ser utillizado na árvore de atributos da *MIB*, que é uma identificação única de um fabricante de hardware registrado.

1.5 Justificativa do Trabalho

Espera-se que os conversores de comprimento de ondas (RAMASWAMI; SIVARAJAN, 1996) aumentem a eficiência de utilização de redes totalmente ópticas. Como discutiremos nos capítulos seguintes, é desejável que a implementação deste equipamento execute uma série de tarefas de forma automatizada, como por exemplo, a verificação de quais canais estão disponíveis para a conversão e se a potência do sinal convertido está compatível com a potência dos sinais que não foram convertidos. Por essas razões, existem vários parâmetros do dispositivo que precisam ser controlados e gerenciados de maneira dinâmica.

O desenvolvimento deste trabalho está relacionado com a segunda dessas funções e é essencial para que o equipamento possa se comunicar com a gerência da rede. Além disso, o desenvolvimento do software de gerenciamento

aqui apresentado também apresenta importância fundamental para os testes do protótipo do conversor proposto pelos pesquisadores do LMT (ABBADE, 2003).

Finalmente, a descrição detalhada de todos os passos para desenvolver uma *MIB* para um novo equipamento e a estruturação do gerenciamento *SNMP* deste mecanismo baseada em software de código aberto, estarão agrupadas em um único documento de domínio público. Pretende-se com isto criar um material referencial para se idealizar a criação de um equipamento imaginando-se todos os passos necessários para proporcionar o devido gerenciamento do mesmo através do protocolo padrão de gerenciamento da arquitetura *TCP/IP* que é o *SNMP*.

1.6 Organização da Dissertação

Esta dissertação será dividida em 6 (seis) capítulos, No capítulo CAPÍTULO 2 - serão discutidas a importância e relevância de como o gerenciamento das redes e seus recursos influenciam nas decisões das empresas, visto que acordos de níveis de serviços são estabelecidos para garantir a disponibilidade e desempenho almejado pelos clientes. Também será comentado sobre a razão custo/ benefício de uma implementação eficaz do gerenciamento.

A seguir o capítulo 3 apresentará as ferramentas e softwares de código aberto escolhidos para mostrar o gerenciamento em funcionamento, constatando que é possível atingir eficácia para tal, baseando-se apenas em softwares que não requerem grande orçamento para implementação.

O capítulo 4 visa descrever a funcionalidade do Conversor de Ondas Totalmente Óptico, e a identificação de como os atributos da *MIB* devem ser estruturados para atender o gerenciamento das funcionalidades do equipamento.

A análise e discussão dos resultados esperados serão discutidos no capítulo 5, após o desenrolar das atividades previstas nos capítulos anteriores.

No último capítulo desta dissertação serão ressaltados os pontos relevantes deste trabalho, tanto no âmbito acadêmico como no comercial. Também serão descritos propostas para novos trabalhos que poderão agregar e complementar este trabalho.

No APÊNDICE A - serão descritos em detalhe os passos realizados para o desenvolvimento do agente *SNMP* para o conversor de comprimento de ondas.

No APÊNDICE B - serão mostrados os componentes do protótipo do conversor de comprimento de ondas e o inter-relacionamento entre eles, que influenciaram no raciocínio para a estruturação da *MIB*.

CAPÍTULO 2 - IMPORTÂNCIA DO GERENCIAMENTO DE REDE

O segmento para o gerenciamento de redes está subdividido em soluções de softwares e construção de equipamento gerenciável. Por um lado empresas elaboram gerentes *SNMP* e respectivos agentes personalizados, e por outro lado as empresas fabricantes de equipamentos preocupam-se em fazer com que seu produto seja gerenciável para conquistar o mercado.

Houve um tempo em que as empresas não se preocupavam em adquirir equipamentos gerenciáveis em decorrência da baixa razão custo-benefício. Isso acontecia porque os equipamentos gerenciáveis eram bem mais caros que os equipamentos não-gerenciáveis e ainda não era percebido o benefício trazido pelo gerenciamento das redes.

Com o advento de programas baseados em interface gráfica que alavancaram aplicações baseadas no modelo cliente-servidor, começou a ficar oneroso e dificultoso identificar os problemas que comprometem o desempenho das aplicações, uma vez que o processamento fica distribuído entre as estações e servidores diversos. Antes a identificação de problemas que comprometiam o desempenho era concentrada apenas no servidor, em virtude de que os usuários utilizavam terminais remotos para acessar os sistemas, e todo processamento era realizado no servidor.

Um outro aspecto a ser considerado, é que tanto no modelo cliente-servidor, como nos modelos que o sucedem atualmente (modelo 3 ou 4 camadas), tornou-se importante identificar onde se localiza a causa de uma indisponibilidade do sistema, uma vez que estes modelos de aplicações envolvem diversos recursos computacionais.

Com esta necessidade de gerenciamento cada vez mais premente, as empresas de tecnologia começaram cada vez mais a investir nas construções de módulos gerenciados para os hardwares, e softwares de gerenciamento capazes de administrarem ambientes heterôgeneos.

Os fabricantes de equipamento têm que construir uma *MIB* para os mesmos, e os desenvolvedores de soluções para gerenciar as redes tem que conhecer como tratar estas *MIB´*s.

Sendo assim, estas empresas procuraram se cadastrar no site da *Internet Assigned Numbers Authority* (*IANA*), que é a entidade responsável por centralizar a coordenação das funções da Internet Global, sendo que uma das funções é a atribuição do *Private Enterprise Number* (*PEN*) para cada empresa solicitante que esteja em conformidade com os critérios pré-estabelecidos. O *PEN* trata-se de um número para ser utilizado para criação de atributos personalizados na *MIB*. O processo para solicitação e obtenção *PEN* para a Pontifícia Universidade Católica de Campinas, será explicitado no APÊNDICE A -.

A título ilustrativo realizamos um levantamento para quantificar qual a parcela de participação do Brasil na solicitação destes identificadores. Para isso, foi realizado um acesso ao site do *IANA*, baixado a lista de todos *PEN´s* fornecidos, e após um processo de formatação e tabulação baseado no domínio internet, foi possível descobrir que o Brasil ocupa a 16ª colocação de empresas que possuem o *PEN*, conforme resumo estatístico demonstrado na Tabela 1.

Tabela 1. Estatísticas de PEN´s registrados no IANA de acordo com o domínio do país.

Domínio	Qtde.	Domínio	Qtde.
com	11730	tw	212
net	1729	**br**	**208**
de	1466	ru	196
org	621	kr	188
edu	529	cn	183
uk	527	at	178
fr	432	es	164
jp	387	pl	148
au	369	dk	119
nl	286	cz	114
it	283	be	113
ca	235	fi	109
se	221	il	107
ch	213	no	101

Vale ressaltar que a amostra da Tabela 1 foi resumida considerando os domínios que tivessem uma representatividade de pelo menos 100 empresas

cadastradas. Na Figura 5 é possível visualizar a representação gráfica destas informações.

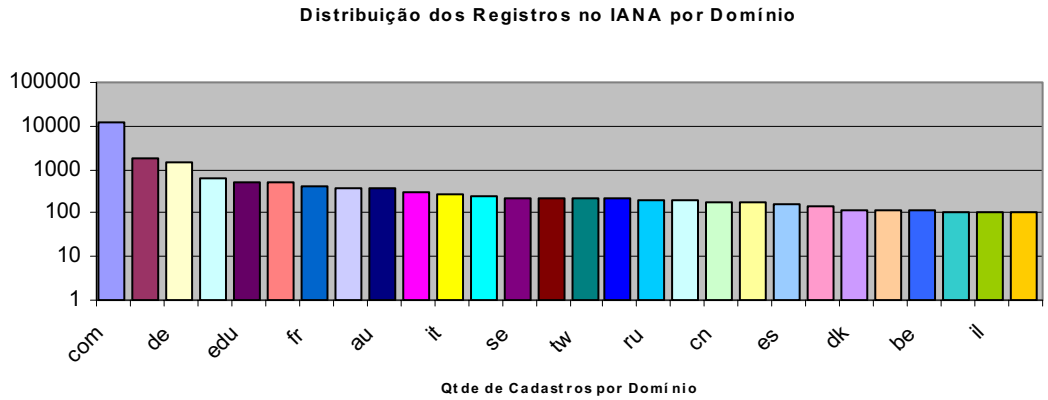

Figura 5. Distribuição dos Cadastros no IANA por Domínio

2.1 Gerência Integrada da Rede

O conceito de gerenciamento de rede é bastante abrangente e o *ITU-T* propôs uma série de recomendações conhecidas como a série *M.3000* ou Telecommunications Management Network (TMN)) (THE INTERNATIONAL ENGINEERING CONSORTIUM, 2006), representadas no Quadro 1.

Quadro 1. Série de Recomendações M.3000 do ITU-T.

Recomendações ITU-T	**Descrição**
M.3000	Visão Geral das Recomendações TMN (Overview of TMN Recommendations)
M.3010	Princípios para um Gerenciamento de Redes de Telecomunicações (Principles for a Telecommunications Management Network)
M.3020	Metodologia da Especificação da Interface TMN (TMN Interface Specification Methodology)
M.3100	Modelo Genérico de Informação da Rede (Generic Network Information Model)
M.3101	Sentenças de Conformidade dos Objetos Gerenciados para o Modelo Genérico de Informação da Rede (Managed Object Conformance Statements for the Generic Network Information Model)
M.3180	Catalogo de Informação de Gerenciamento do TMN (Catalogue of TMN Management Information)
M.3200	Serviços de Gerenciamento TMN (TMN Management Services: Overview)
M.3300	Capacidade de Gerenciamento Apresentadas na Interface F (TMN management Capabilities Presented at the F Interface)
M.3320	Estrutura de Requisitos de Gererenciamento para a Interface X do TMN (Management Requirements Framework for the TMN X Interface)
M.3400	Funções de Gerenciamento TMN (TMN Management Functions)

Estas recomendações foram adotadas pela *International Organization for Standardization (ISO)* como um padrão para infra-estrutura de gerenciamento de redes. As funcionalidades *TMN* são separadas por:

- Áreas Funcionais de Gerência
- Camadas de Gerência
- Serviços de Gerência

Uma abordagem associativa destas funcionalidades é apresentada de forma sucinta no item 3.1.1. Essa associação é possível porque o *TMN*, se baseia na orientação à objetos para o gerenciamento de atributos especificados na *MIB*.

O *TMN propõe* a utilização de elementos de mediação e de interfaces *QAs (Q-Adapters*) que habilitam o *TMN* a gerenciar elementos de rede (*Telecommunications Management Network Tutorial*, 2006).

Desta forma, este documento se propõe a criar uma *MIB* para ser gerenciada pelo protocolo *SNMP*, almejando um gerenciamento integrado da rede, com uma gerência distribuída, escalável e interoperável de acordo com a proposta do *TMN*.

3.1.1 Áreas Funcionais de Gerência

As áreas funcionais de gerência podem ser entendidas conforme resumo apresentado no Quadro 2.

Quadro 2. Áreas Funcionais de Gerência

Área Funcional	Descrição
Gerência de Falhas	permite a detecção, isolamento e correção de anomalias de uma rede de telecomunicações e de seus equipamentos
Gerência de Configuração	permite o controle, a identificação e a coleta de dados de NEs e de conexões entre eles, assim como o planejamento, a instalação e a configuração dos equipamentos da rede de forma a garantir os serviços requeridos pelos clientes de uma rede de telecomunicações
Gerência de Contabilização	habilita o uso dos serviços da rede para medição e determinação de custos, provendo facilidades de definição de parâmetros de bilhetagem e de coleta de registros de cobrança do uso de uma rede de telecomunicações
Gerência de Desempenho	permite a geração e a avaliação de relatórios de dados coletados de uma rede de telecomunicações de acordo com requisitos de qualidade de serviço requeridos pelos usuários da rede e de seus equipamentos
Gerência de Segurança	permite prevenir e detectar o uso impróprio ou não autorizado de recursos de uma rede de telecomunicações, assim como administrar a sua segurança

Quando estivermos verificando a situação dos componentes do Conversor de Comprimento de Ondas, e se os mesmos estão funcionais, estaremos praticando a gerência de falhas.

Já se precisarmos inventariar o Conversor de Comprimento de Ondas e suas características estaremos realizando a gerência de configuração.

Através da gerência de contabilização poderemos realizar se de tempos em tempos coletas de informações através do agente *SNMP* do Conversor de Comprimento de Ondas que podem ser graficamente representadas com a utilização do software *Multi Router Traffic Grapher (MRTG)*, que será explicado no APÊNDICE A -.

Quando estivermos verificando se a potência está adequada (conforme a descrição realizada no capítulo CAPÍTULO 4 -), estaremos praticando a gerência de desempenho.

A gerência de segurança visa assegurar que dados transmitidos tenham garantia de privacidade e confidenciabilidade.

3.1.2 Camadas de Gerência

As camadas de gerência são assim chamadas por haver uma hierarquia de sobreposição como se pode contemplar na Figura 6.

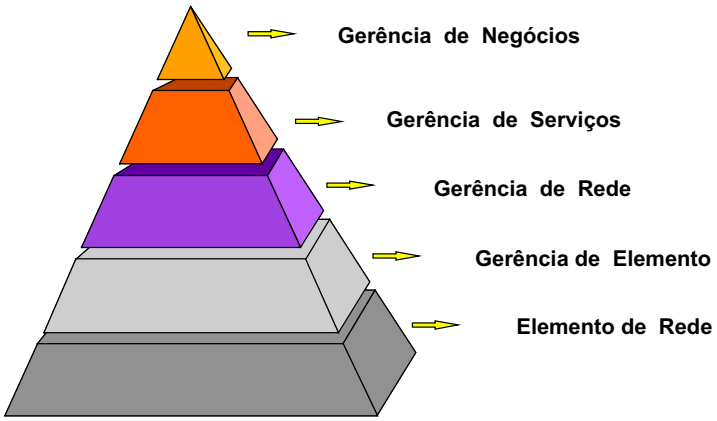

Figura 6. Camadas de Gerência

Estas camadas podem ser melhor entendidas através da descrição das respectivas responsabilidades descritas no Quadro 3.

Quadro 3. Responsabilidade das Camadas de Gerência.

Camada	Responsabilidade
Gerência de Negócios	Coordena o planejamento de alto nível, as cobranças, as estratégias, as decisões executivas, os contratos de negócios (BLA)
Gerência de Serviços	Utiliza informação disponibilizada pela camada de gerência de rede para gerenciar contratos de serviço de clientes existentes e em potencial, desde o aprovisionamento e a qualidade de serviço, até a gerência de falhas. A SML é também responsável pela interação com os provedores de serviços e com outros domínios administrativos, mantendo dados estatísticos para garantir a qualidade do serviço prestado.
Gerência de Rede	Tem a visibilidade de toda a rede baseada em informações de NEs disponibilizadas pelos OSs da camada de gerência de elemento de rede. A NML coordena todas as atividades de rede e suporta as requisições da SML.
Gerência de Elemento de Rede	Gerencia cada elemento de rede. A EML possui OSs, cada um dos quais é responsável pelas informações gerenciáveis de NEs específicos. De uma forma geral, um gerente de elemento de rede é responsável por um subconjunto dos elementos de rede, gerenciando os seus dados, atividades, registros, etc.
Elemento de Rede	Apresenta as informações gerenciáveis TMN de cada um dos NEs individualmente. A NEL faz a interface entre a informação gerenciável proprietária e a infra-estrutura TMN.

Desta forma o Conversor de Comprimento de Ondas é um elemento de rede, para o qual a *MIB* disponibilizará os itens necessários para gerência de seus componentes. Dentro da gerência de Rede, precisaremos contemplar a comunicação do Conversor de Comprimento de Ondas com os demais elementos de rede. Através da aquisição e acompanhamento de dados estatísticos, podem-se criar subsídios necessários para atender as gerências de serviços e de negócios.

3.1.3 Serviços de Gerência

Os serviços de gerência estão relacionados a seguir:

1. Administração de Usuário
2. Gerência de Provisionamento da Rede
3. Gerência da Força de Trabalho
4. Gerência da Qualidade do Serviço
5. Gerência de Tarifas, Tarifação e Contabilização
6. Administração de Tráfego
7. Gerência de Tráfego
8. Gerência de Manutenção
9. Administração de Segurança
10. Gerência de Logística

As informações disponibilizadas pela *MIB* do Conversor de Comprimento de Ondas serão organizadas de forma a se enquadrarem dentro das camadas de gerência e respectivos serviços de gerência.

3.1 Introdução

Neste capítulo discutiremos os conceitos necessários para o entendimento do protocolo *SNMP* e desenvolvimento da *MIB*. Em especial a seção 3.1.1, aborda as funcionalidades do protocolo *SNMP*, a seção 3.1.2 apresenta os modelos operacionais do *SNMP* e a seção 3.1.3 demonstra como uma *MIB* é estruturada. Também comentamos na seção 3.2 a utilização de programas de código livre e do *NET-SNMP* para o desenvolvimento do agente.

3.1.1 Caracterização do *SNMP*

O *SNMP* é um protocolo existente dentro da arquitetura *TCP/IP*, que atua na camada de aplicação e tem por propósito permitir o gerenciamento de todos os componentes de uma rede de computadores. Ele foi definido como padrão para o gerenciamento dos componentes de uma rede, através de diversos *RFC's* do *Internet Engineering Task Force (IETF)* mencionados no Quadro 5 deste documento.

Outras tentativas de estabelecer *frameworks* para o gerenciamento foram feitas, tais como *High-level Entity Monitoring System (HEMS)* e *Common Management Information Protocol (CMIP*), sendo este último específico para redes constituídas puramente do modelo *OSI*, arquitetado pela *ISO*.

O *SNMP* utiliza um esquema encontrado no *Abstract Syntax Notation One (ASN.1)*, que também é uma linguagem de programação específica para tratar os identificadores únicos existentes em uma *MIB.*

Estes identificadores únicos são denominados *OID* (*Object Identifier*), que não são limitados para objetos *SNMP* apenas, e podem ser utilizados para identificar qualquer tipo de item.

Uma *MIB* é constituída de especificações e descrições sobre os objetos gerenciáveis, e é mais facilmente entendida quando representada em forma de uma árvore roteável, como será mostrado na proposta para construção da árvore para atender o Conversor de Comprimento de Ondas, no capítulo CAPÍTULO 3 -.

A Figura 7 apresenta algumas ramificações da árvore de *OID* parando no nível *enterprises* que serve como base para continuação dos atributos específicos para cada empresa.

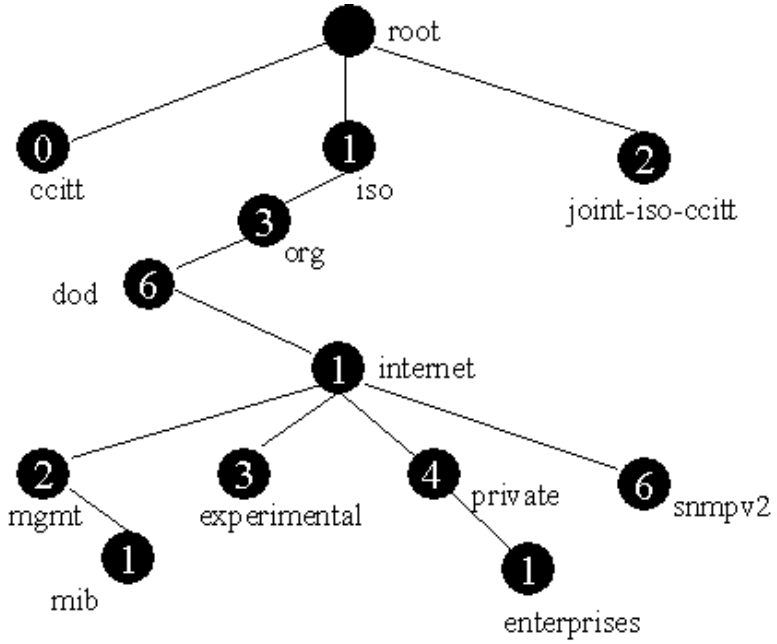

Figura 7. Árvore de OID base para Empresas.

Para gerenciar uma rede com o protocolo *SNMP* não é necessário se aprofundar em cada item existente numa *MIB*, mas para criar um gerenciamento de redes pró-ativo e reativo, o entendimento profundo dos *OID´s* se torna imprescindível.

Entretanto, este documento não se limita a esclarecer a *MIB*, mas se propõe a fornecer o conhecimento necessário para se desenvolver uma *MIB* para um novo equipamento, interpretando os *OID´s* descritos na *MIB*..

Para tanto, conhecimentos generalistas se fazem necessários em todas as áreas de tecnologia, tais como, banco de dados, lógica de programação, redes e sistemas operacionais. Isto porque a MIB funciona como um banco de

dados e o agente será um programa desenvolvido, que será executado sob o sistema operacional para acessar informações sobre os valores constantes nos *OID´s* através da rede.

O órgão que padroniza a estruturação e funcionamento do protocolo *SNMP* e das *MIB´s* é o *IETF*, no qual grupos de trabalho desenvolvem documentos, denominados *RFC´s*, que passam por diversos estágios de maturidade, identificados pelas categorias apresentadas no Quadro 4.

Quadro 4. Possíveis categorias que descrevem a situação de um documento RFC.

Categorias	Descrição da Situação
STANDARD	Padrão requerido, recomendado ou eletivo.
DRAFT STANDARD	Rascunho que pode ser requerido, recomendado ou eletivo.
PROPOSED STANDARD	Porposta de Padrão que pode ser recomendado ou Eletivo.
INFORMATIONAL	Apenas um documento informativo.
EXPERIMENTAL	Uso limitado.
HISTORIC	Não foi determinado como um padrão, porém o RFC é mantido a nível de histórico do grupo de trabalho
DRAFT STANDARD	Rascunho para a proposta de um padrão.
OBSOLETED	O documento ficou obsoleto devido a outro RFC tê-lo substituído em sua totalidade.

As principais RFC´s para o *SNMP* na atualidade estão ilustradas no Quadro 5.

Quadro 5. Principais RFC´s relacionadas com o SNMP.

RFC	Descrição
1155	Estrutura e Identificação da Informação de Gerenciamento para redes baseadas no TCP/IP
1156	Base de Informação de Gerenciamento para redes baseadas em TCP/IP
1157	Um Protocolo de Gerenciamento Simples de Rede
1441	Intrdoução à versão 2 da Estrutura de Gerenciamento de Rede baseada na Internet
1213	Base de Gerenciamento de Informações para Gerenciamento de Redes TCP/IP: MIB-II
3410	Introdução de Declaração da Aplicabilidade para a Estrutura de Gerenciamento de Rede baseada na Internet
3411	Uma arquitetura para Descrever Estruturas de Gerenciamento do SNMP
3412	Processamento de Mensagem e Comunicado para o SNMP
3413	Aplicação do SNMP
3414	Modelo de Segurança Baseado em Usuário para a versão 3 do SNMP
3415	Modelo de Controle de Acesso Baseado em Visão para o SNMP
3416	Versão 2 de Operações do Protocolo para o SNMP
3417	Mapeamento de Transporte para o SNMP
3418	MIB para o SNMP
3512	Configurando Redes e Mecanismos com o SNMP
3584	Co-existência entre as versões 1, 2 e 3 da Esrutura de Gerencimaneto de Rede baseada na Internet

Os documentos na íntegra estão disponibilizados na página de internet do IETF. (ROSE; McCLOGHRIE, 1990),(McCLOGHRIE/ ROSE, 1990),(SCHOFFSTALL; DAVIN, 1990), (CASE et al., 1993a),(McCLOGHRIE/ ROSE, 1991), (CASE et al., 2002a), (HARRINGTON; PRESUHN; WIJNEN, 2002),(CASE et al., 2002b), (LEVI; MEYER; STEWART, 2002), (BLUMENTHAL; WIJNEN, 2002), (WIJNEN; PRESUHN; McCLOGHRIE, 2002), (PRESUHN et al., 2002c), (PRESUHN et al., 2002b), (PRESUHN et al., 2002a), (MACFADEN et al., 2003), (CASE et al., 2003).

3.1.2 Conceituação Gerente-Agente

O modelo de gerenciamento *SNMP* é constituído basicamente de gerentes, agentes e eventos.

Quando os agentes são modelados de forma a fornecer as informações solicitadas pelos gerentes, desde que a informação exista dentro da MIB pela qual o agente responde, chamamos este modelo de *Pull Model*. O fluxo de solicitação da informação está ilustrado na Figura 8.

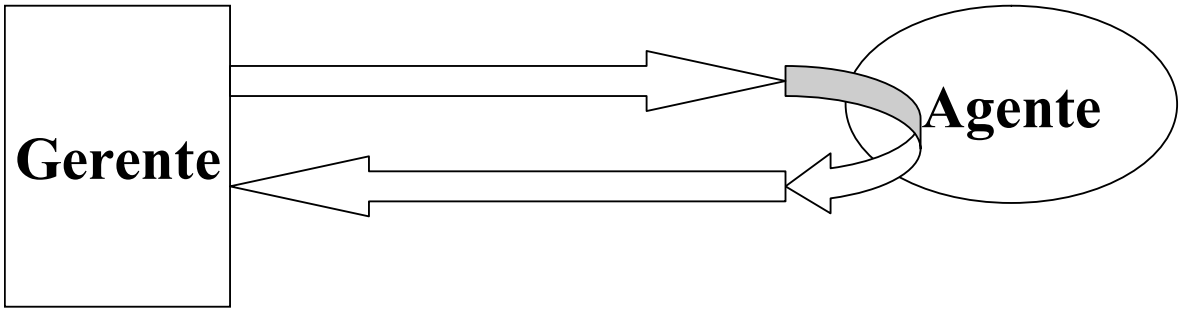

Figura 8. Modelo *Pull Model*.

Neste modelo o gerente realiza aquisições períodicas, que são denominadas de *polling*.

O outro modelo denomina-se *Push Model*, no qual o agente reage a eventos e exceções para o qual foi programado, enviando notificações para o gerente. A forma como o gerente obtém a informação necessária é representada na Figura 9.

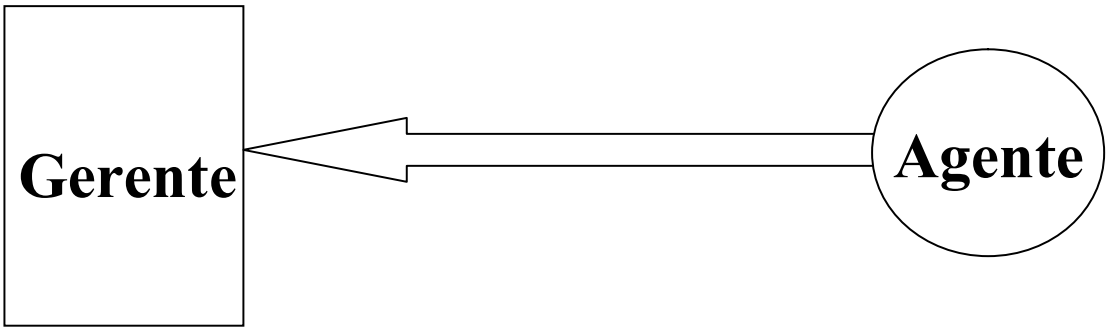

Figura 9. Modelo *Push Model*.

Entretanto, existem desvantagens significativas em ambos os modelos, e o grande desafio para o administrador de rede é conciliar o melhor destes 2 modelos, objetivando reduzir a latência de reação de um evento.

No *Pull Model* a latência para reação de um evento tende a ser maior. Para exemplificarmos isso, consideramos o diagrama representado na Figura 10, no qual o gerente executa requisições de 5 em 5 minutos, e um evento ocorre no agente após um minuto de um gerente ter realizado uma requisição. Somente após 4 minutos o gerente será capaz de identificar o evento, ou seja, quando for realizar uma nova requisição.

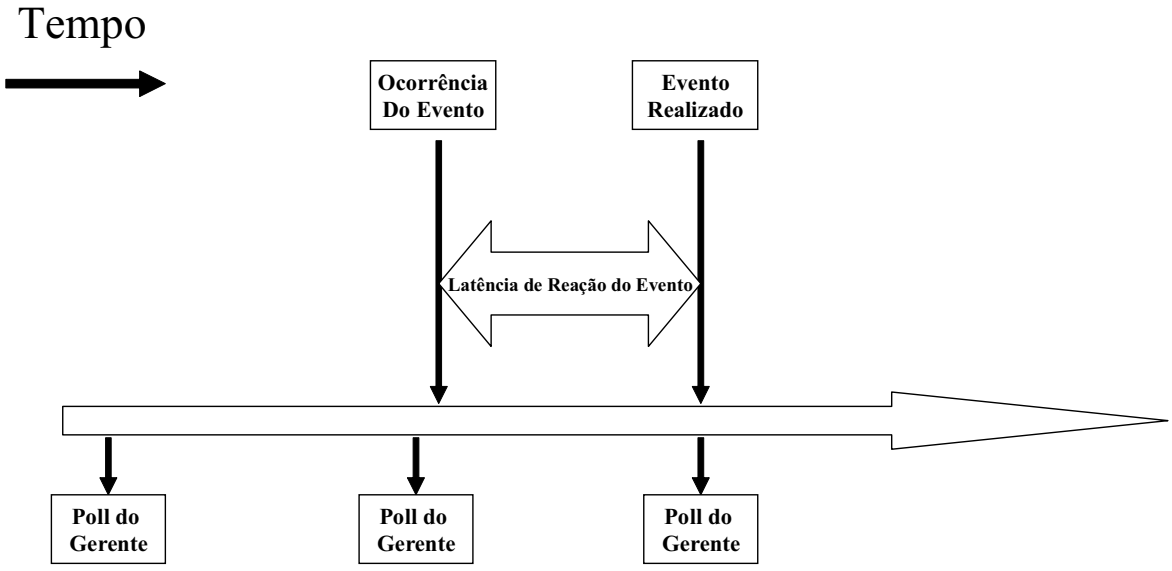

Figura 10. Latência para Reação de Evento no Modelo *Pull Model*.

Se o intervalo da requisição for diminuído para reduzir a latência, haverá um consumo maior dos recursos da rede que poderiam ser utilizados para

outras aplicações. Ou seja, ocorrerá um impacto negativo no desempenho e no tempo de resposta das aplicações que se comunicam pela rede.

No *Push Model* o fato do agente enviar informações para o gerente, pode ocasionar um congestionamento na rede, face aos inúmeros pacotes que podem ser enviados ao gerente até que o problema seja solucionado. Outro fator neste modelo, é que se houver falha de comunicação da interface de rede, o gerente jamais saberá desta informação.

Existem dois modelos para decidir quando um evento ocorre. O primeiro é denominado *edge triggered*, no qual o evento ocorre quando um valor monitorado atinge uma faixa, na qual existe um limite para indicar quando o alarme ocorrerá, e outro para desarmar o alarme. Ou seja, quando o evento ocorrer, não será considerado um novo evento enquanto o alarme não baixar do limite que irá desarmar.

Já no modelo *level triggered*, o evento ocorre apenas em cada intervalo pré-definido, desde que esteja acima da linha do limite estabelecido.

3.1.3 Estruturação de uma MIB

A estruturação de uma *MIB* é baseada no documento *Structure of Management Information (SMI*), que por sua vez exige a utilização da linguagem *ASN.1* que visa a padronização da escrita dos módulos em uma *MIB*.

Todas regras e metodologias necessárias para o desenvolvimento de uma *MIB* estão descritos nas *RFC´*s 1441 (CASE, 1993a), 1901 (CASE, 1993b), 1909 (McCLOGHRIE, 1996), 1910 (WATERS, 1996), 2578 (McCLOGHRIE, 1999) e 3584 (CASE, 2003).

Os tipos de dados utilizados pelo *SNMP* são classificados em:

Tipos Universais – baseados na classe universal, são os tipos permitidos para se definir objetos na *MIB*. (*integer, octect string, null, object identifier e sequence*)

Tipos de Aplicação - utilizados para dar funcionalidade à aplicação do protocolo (*display string, IP address, Phys Address, Counter, Gauge, Timeticks*).

A *MIB* é baseada num modelo orientado a objeto, no qual cada objeto pode conter inúmeras instâncias. O caminho completo para identificação de um objeto é representado pelo *OID*. *MIB´s* iniciais são definidas nos *RFC´s* desenvolvidos pelo grupos de trabalho do *IETF*.

Para se desenvolver uma *MIB* específica para o fabricante, o *OID* será iniciado por 1.3.6.1.4.1, que é a representação numérica do caminho *iso.org.dod.internet.private.enterprise*.

A partir deste ponto se faz necessário a obtenção do *PEN* para continuar a estruturação da *MIB*. A Figura 11 ilustra o *PEN* de 3 empresas tradicionais de tecnologia.

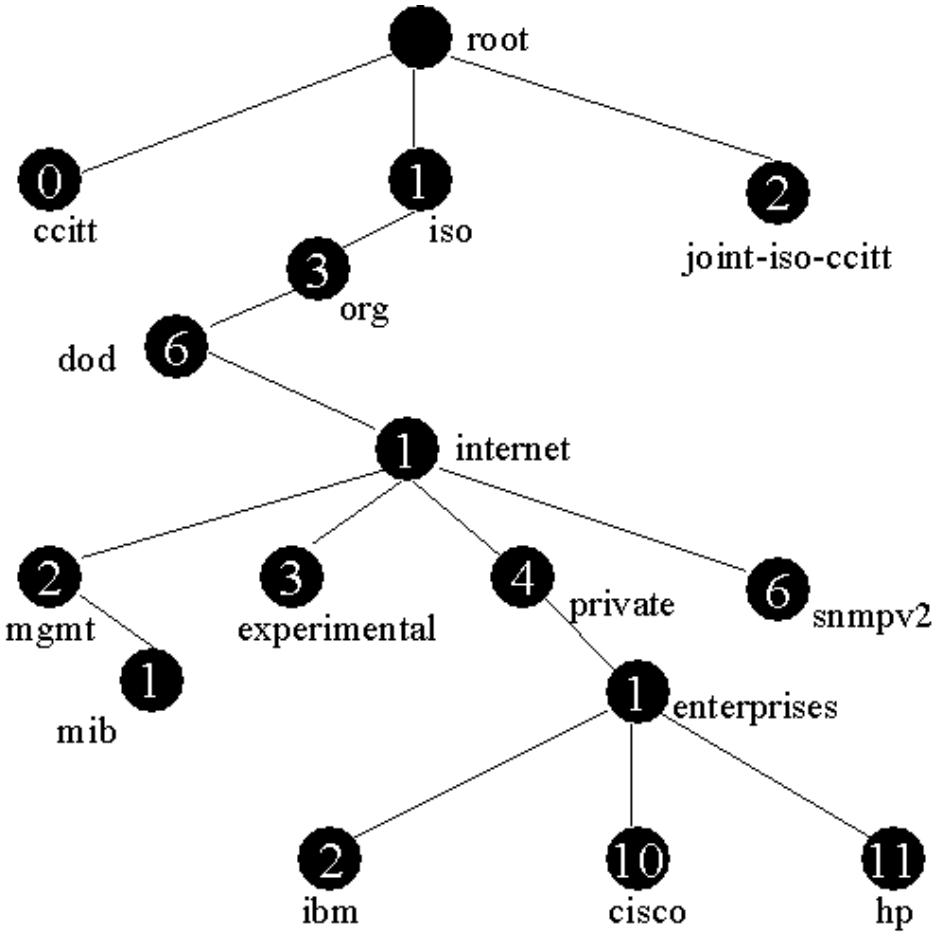

Figura 11. Árvore OID de MIB base para empresas.

A representação por OID destas empresas está demonstrada no Quadro 6**.**

Quadro 6. Representação dos OID´s base sob a árvore enterprise para as empresas IBM, CISCO e HP.

OID	Empresa	Representação Nominal
1.3.6.1.4.1.2	IBM	iso.org.dod.internet.private.enterprise.ibm
1.3.6.1.4.1.10	CISCO	iso.org.dod.internet.private.enterprise.cisco
1.3.6.1.4.1.11	HP	iso.org.dod.internet.private.enterprise.hp

Em nosso trabalho, após o cadastro da PUC-Campinas no *IANA*, obtivemos o *PEN* 23955. Sendo assim a árvore *MIB* base para desenvolver os atributos para esta universidade ficaria como apresentado na Figura 12.

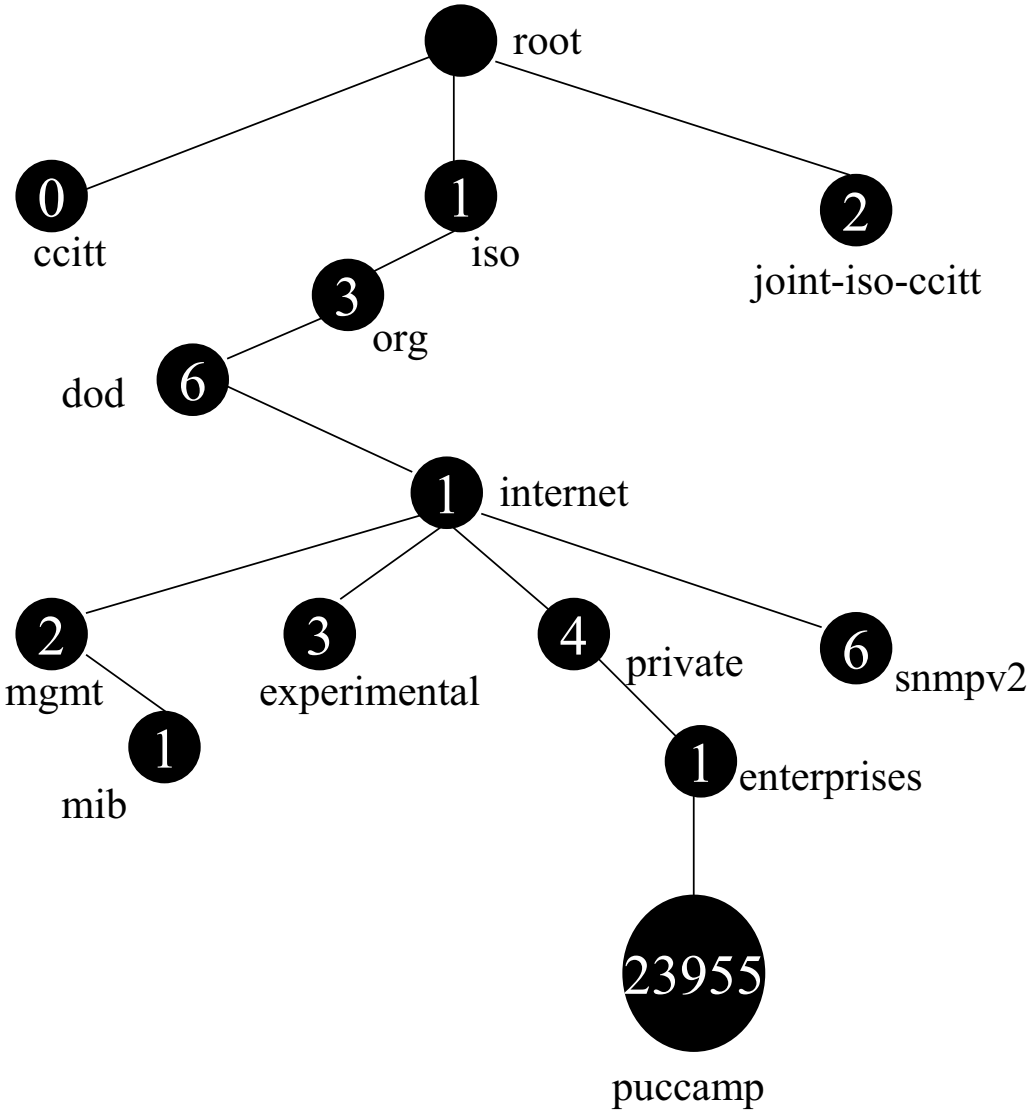

Figura 12. Árvore *MIB* com *OID* base para PUC-Campinas.

A partir deste ponto pode-se estender a árvore *MIB* para especificar os atributos necessários para um determinado equipamento desenvolvido pela PUC-Campinas.

Para tanto é necessário conhecer bem as funcionalidades do equipamento, realizando uma modelagem dos objetos inerentes ao mesmo para depois converter em um modelo de *MIB*. Entretanto isto será objeto de discussão do capítulo CAPÍTULO 4 - interessando no momento entender os passos essenciais para o desenvolvimento da *MIB*.

3.2 Agente SNMP de Código Livre

No ano de 1.992 um grupo de rede da *Carnegie-Mellon University* desenvolveu um pacote de programas e utilitários para o protocolo *SNMP*, cujo código foi disponibilizado publicamente. A princípio este pacote era conhecido como *UCD-SNMP*.

Com o passar dos anos uma força tarefa institucionalizou o código, tendo colaboração espontânea de muitos profissionais de tecnologia. Hoje este pacote de programas é conhecido como *NET-SNMP* (NET-SNMP, 2005), sendo publicado e divulgado no *site* http://www.net-snmp.org.

Este agente foi escolhido por conter *Application Programming Interface (API) (*WIKIPEDIA FOUNDATION, 2006a) que possibilitam customizar o agente *SNMP* para um determinado equipamento, através da utilização de uma *API* denominada *MIB2C*, que lê um arquivo de *MIB* escrito na linguagem *ASN.1* e cria um esqueleto de código fonte em linguagem *C*, que possibilitará a customização do agente *SNMP*.

Entretanto, este método acarreta em deixar o agente *SNMP* mais pesado em seu processamento e também dificulta a manutenção para atributos específicos de um novo equipamento, pois a cada nova alteração requer compilação global do agente *SNMP*.

Em virtude disto, existe a possibilidade de se criar um agente estendido, informando no arquivo de configuração do *NET-SNMP* para utilizar este agente para responder por determinados *OID´s*.

Nesta dissertação a customização do *SNMP* para o Conversor de Comprimento de Ondas utilizará o método de agente estendido que permite uma maior flexibilidade para manutenção do agente.

O Conversor de Comprimentos de Onda Totalmente Óptico é um equipamento que tem como objetivo auxiliar a comutação de dados de uma forma totalmente óptica, ou seja, diferenciando-se dos equipamentos existentes para redes ópticas na atualidade, que realizam a comutação internamente de forma mecânica.

Entretanto, a comunicação para passar por este meio físico compartilhado poderá sofrer uma alteração de sua freqüência caso a freqüência inicial já esteja utilizada por um outro circuito.

A idéia é fazer um transmissão simultânea das freqüências, multiplexando-as em um única banda e desmultiplexando-as do outro lado.

Este capítulo tem como objetivo descrever como funciona o conversor de comprimento de ondas para determinar os atributos que serão necessários para o gerenciamento deste equipamento. A seção 4.1 abordará os princípios de operação do conversor, seguido pela seção 4.2 que identifica as funcionalidades deste conversor para que na seção 4.3 sejam determinados os atributos que comporão a *MIB* do conversor. A seção 4.4 propõe como a árvore MIB deve estruturar os atributos, A seção 4.5 descreve como a MIB foi construída e a seção 4.6 aborda os passos para a construção do agente, finalizando com a seção 4.7 que esclarece o relacionamento das funcionalidades do conversor com a *MIB* desenvolvida.

4.1 Operação do Conversor de Comprimentos de Onda Totalmente Óptico

Nesta seção, descreveremos o princípio de operação do conversor de comprimentos de onda totalmente óptico proposto pela PUC-Campinas junto ao Programa *Tidia/Kyatera*, que faz parte do foco deste trabalho.

Quando dois sinais nas portadoras ópticas f_1 e f_2 ($f_2 > f_1$), propagam-se por uma fibra óptica, um efeito não-linear conhecido como Mistura de Quatro-Ondas proporciona o surgimento de dois sinais laterais em bandas laterais:

$f_{-} = f_1 - \Delta f$ (1a) e (1)

$f_{+} = f_2 + \Delta f$ (1b), (2)

na qual Δf= f_2- f_1. A potência do sinal nessas bandas laterais são dadas por (SHIBATA; BRAUN; WAARTS, 1987).

$P_{-} = k_{-} P_1^2 P_2$ (2a) e (3)

$P_{+} = k_{+} P_1 P_2^2$ (2b), (4)

sendo P_1 e P_2 as potências, respectivamente, dos sinais em f_1 e f_2. Os termos k_- e k_+ dependem da dispersão cromática nessas freqüências, de Δf, do comprimento, da atenuação e do parâmetro não-linear da fibra, de acordo com a descrição apresentada em SHIBATA, 1987. Além do mais, se P_1 e P_2 forem suficientemente baixas (SHIBATA; BRAUN; WAARTS, 1987), k_- e k_+ podem ser consideradas constantes.

Neste caso, se f_1 transmitir uma seqüência de bits e inserirmos em f_2 um sinal de potência constante, a potência do sinal em f_+ será alta quando o sinal em f_1 transmitir um bit 1 e nula quando o sinal em f_1 transmitir um bit 0. Dessa forma, se filtrarmos apenas a banda ao redor de f_+, obteremos uma cópia do sinal em f_1, o que caracteriza uma conversão de freqüências totalmente óptica. De maneira análoga, a conversão também pode ser obtida em f-. De agora em diante, chamaremos o sinal de potência constante de sinal de bombeio.

O processo de conversão descrito acima está ilustrado na Figura 13 Para que um equipamento utilize esse princípio em uma rede *WDM* real e converta um sinal de dados em f_{in} para uma nova portadora f_{out} é necessário ajustar a portadora do sinal de bombeio, f_P, para:

$f_P = (f_{in} + f_{out})/2$ (5)

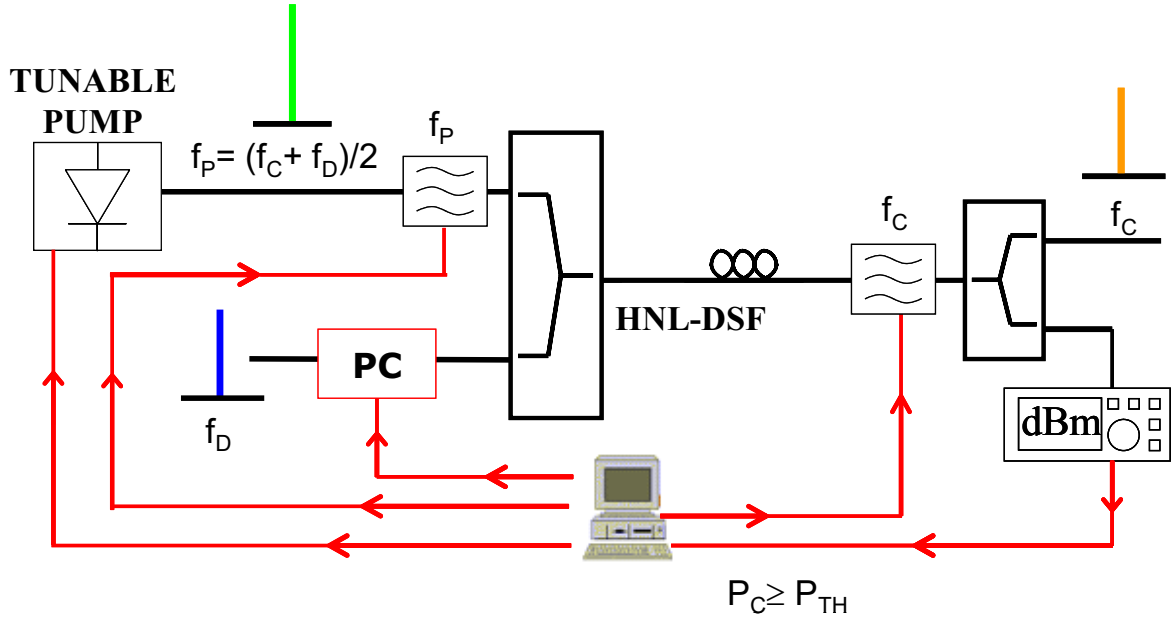

Figura 13. Conversor de Comprimento de Ondas FWM. PC= Controle de Polarização; OBPF= Filtro óptico de banda passante; PM= Medidor de Potência.

Essa relação pode ser prontamente verificada a partir de (1). Além disso, o filtro *OBPF1*, que tem por propósito reduzir o ruído do bombeio e *OBPF2* devem ser automática e respectivamente posicionados ao redor das freqüências f_P e f_{out}.

Um ponto importante é que (2) é válida quando os sinais em f_{in} e f_{out} têm os mesmos estados de polarização. Caso essa condição não seja satisfeita, a potência do sinal de saída será menor que a prevista em (2), dependendo da diferença entre os ângulos dos estados de polarização do sinal de dados e do bombeio.

Em uma rede *WDM* o bombeio estará sendo gerado localmente no nó do conversor ao passo que o sinal de dados será proveniente de uma outra localidade que tipicamente estará, pelo menos, a vários quilômetros de distância. Desta forma, não há garantias de que os dois sinais estarão no mesmo estado de polarização.

A fim de superar esse problema, a proposta realizada em (ABBADE, 2003), sugere a utilização de controle de realimentação de potência. Nesse procedimento, assim como sugerido na Figura 13, a potência de saída do

conversor, P_{out}, é monitorada e, caso seja menor que uma potência de limiar, P_{Th}, um comando é enviado para o controlador de polarização PC que gira o estado de polarização do sinal em f_{in}. O controle de realimentação cessa quando a) $P_{out} \geq P_{Th}$ ou b) $P_{out} < P_{Th}$ após um certo tempo. Neste segundo caso, a gerência do conversor deve informar a gerência de redes que a conversão não é permitida.

De fato, não é interessante que essa fase de teste de potência ocorra durante a transmissão de dados. Então, a proposta sugerida em (ABBADE, 2003) e adotada neste trabalho considera que durante esta fase de testes o sinal enviado pelo cliente que requer a conversão seja um sinal de testes, possivelmente uma seqüência de bits 0101010. Apenas se a fase de teste for bem sucedida ($P_{out} \geq P_{Th}$) é que a gerência da rede solicitará para o cliente enviar seus dados.

4.2 Identificação das Funcionalidades do Conversor

Para identificar os atributos que comporão a *MIB*, precisamos entender melhor o comportamento do conversor numa situação em que o sinal precise ser convertido.

Para melhor exemplificar tomemos como base a representação da Figura 14, onde Campinas e Curitiba desejam se comunicar com Rio de Janeiro, mas para isso existe São Paulo como um nó de chaveamento.

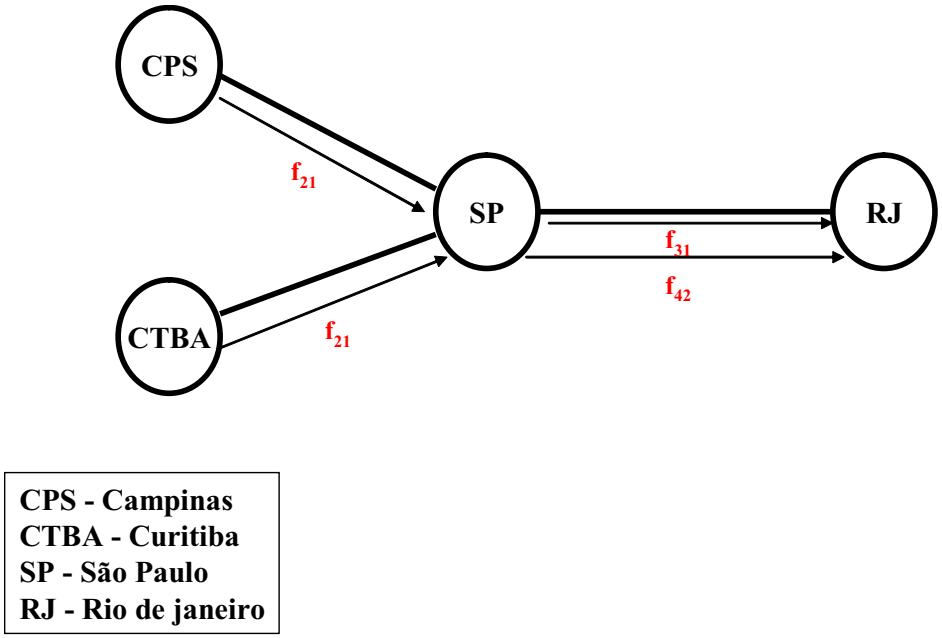

Figura 14. Freqüências de Comunicação do Conversor.

Vamos supor que Campinas e Curitiba enviaram suas transmissões na mesma freqüência de 192,10 THz representadas na Figura 14 por f_{21}, mas que São Paulo já tem estabelecido com Rio de Janeiro um canal de transmissão utilizando esta mesma freqüência f_{31}.

O Conversor de Comprimento de Onda visa resolver este problema, identificando os canais de transmissão que estão disponíveis entre São Paulo e Rio de Janeiro, que neste exemplo supondo que sejam os canais com freqüências 193,10 e 194,20 THz, representados por f_{31} e f_{42}, respectivamente, geraria-se uma freqüência de bombeio para fazer com que os dados de uma determinada freqüência de origem fossem movidos para a freqüência que se encontra disponível. Este cálculo é realizado de acordo com a equação (5) apresentada anteriormente.

Desta forma, segundo a Figura 13 poderíamos chegar no cálculo de freqüência de bombeio conforme mostrado na Tabela 2.

Tabela 2. Cálculo de Freqüência de Bombeio.

Freqüências	THz	Canal	Freq. Bombeio	Freqüência Associada	Freqüência Saída
f_{21}	192,10	21	192,60	f_{31}	193,10
f_{21}	192,10	22	193,15	f_{42}	194,20

As freqüências que serão utilizadas são as padronizadas pelo ITU-T, conforme apresentado na Tabela 3.

Tabela 3. Tabelas de Freqüências do ITU-T.

Canal	Freqüência	Canal	Freqüência	Canal	Freqüência
21	192,10	31	193,10	41	194,10
22	192,20	32	193,20	42	194,20
23	192,30	33	193,30	43	194,30
24	192,40	34	193,40	44	194,40
25	192,50	35	193,50	45	194,50
26	192,60	36	193,60	46	194,60
27	192,70	37	193,70	47	194,70
28	192,80	38	193,80	48	194,80
29	192,90	39	193,90	49	194,90
30	193,00	40	194,00	50	195,00

As necessidades de gerenciamento e monitoramento observadas neste contexto requerem o conhecimento de:

1. Freqüências de Entrada (Canais de Entrada) - Conhecer a freqüência que está entrando para após alocação de uma freqüência de saída, calcular a freqüência de bombeio.

2. Freqüências de Saídas (Canais de Saída) - Deve-se ter um inventário de todos canais que o equipamento estará apto para transmissão.

3. Freqüências de Saídas Disponíveis (Canais de Saída Disponíveis) - dos canais aptos, surgirá a necessidade de saber quais destes não estão sendo utilizados para transmissão.

4. Potência do Sinal de Saída - conhecer a potência do sinal que está sendo transmitido.

5. Limiar Minímo de Potência - estabelecer uma linha de tolerância miníma de potência de saída, pois se a potência estiver abaixo deste limiar, um alerta deve ser gerado e uma ação tomada para elevar a potência para patamares confiáveis.

6. Estabelecimento de Comunicação (Canal de Entrada x Canal de Saída) - conhecer qual associação foi estabelecida entre o canal de entrada x canal de saída para fechar o circuito.

Com base nestas funcionalidades será elaborada uma *MIB* com atributos específicos para o gerenciamento adequado do Conversor de Comprimento de Ondas para Redes Totalmente Ópticas.

4.3 Identificação dos atributos da *MIB*

Para desenvolvermos a *MIB* para o Conversor de Comprimento de Ondas precisamos criar categorias específicas e organizá-las dentro de uma árvore de registros.

As categorias de objetos podem ser organizadas da seguinte forma:

1. **Ações** -controla um sistema. Estes tipos de objetos podem ser utilizados para a realização de comandos

2. **Estatística** - proporciona obter dados quantitativos sobre um equipamento por um determinado período.

3. **Status** - indica a atual condição do sistema.

4. **Componentes** - anuncia os mecanismos físicos ou serviços que estão sendo gerenciado pelo *SNMP*.

5. **Atributos** – contém as propriedades de um objeto modelado que visa a identificar unicamente uma instância de um mecanismo ou serviço.

Dentro destas categorias para o modelo do Conversor de Comprimento de Ondas, determinaram-se os atributos necessários para o desenvolvimento da *MIB* para o conversor de comprimento de ondas, representados no Quadro 7.

Quadro 7. Necessidades de Gerenciamento do Conversor por Categoria.

Categorias	Necessidades
Ações	Realizar um reboot no conversor; Desabilitar um canal de comunicação
Estatísticas	Pacotes trafegados em cada canal; Tempo que o Conversor está ligado; Erros ocorridos; Utilização da Capacidade de Processamento e de Memória
Status	Condições dos canais de comunicação; Indicação de falhas de componentes (ventoinhas, fontes)
Componentes	CPU, Memória, Canais, Fontes, Ventoinhas, Temperatura
Atributos	Dependerá da quantidade de componentes e serviços existentes

Descrevendo um pouco melhor, a situação que requeira um reboot seria quando por algum motivo inesperado o conversor não esteja responsivo. A desabilitação um canal pode ser uma opção necessária para alguma condição de manutenção ou teste.

As estatísticas de pacotes trafegados em cada canal se referem aos pacotes de sinalização utilizados para estabelecer se a freqüencia deve ou não ser convertida, uma vez que não é possível contabilizar o tráfego que ocorre num circuito óptico estabelecido. Sempre é útil a informação desde quando um equipamento está ligado, pois esta informação pode ser associada com incidentes periódicos, como por exemplo, uma condição em que a memória do equipamento seja totalmente utilizada. Por isso é importante conhecer as estatísticas de utilização de memória e processador para ajudar em situações de diagnósticos e colaborar para planejamento de aumento de capacidade do equipamento.

É útil conhecer o *status* de um componente do equipamento porque permite ações pró-ativas de forma a manter maiores indíces de disponibilidade.

Também uma constante preocupação na atualidade é manter o inventário dos equipamentos e componentes de uma rede, que colabora para a gerência de configuração, numa situação em que o inventário aponte que o equipamento se encontra com menos módulos de memória do que na instalação inicial, por exemplo.

Com base nas categorias descritas no Quadro 7, definiram-se os itens para constituição da *MIB* para o conversor de comprimento de ondas apresentados no Quadro 8.

Quadro 8. Representação dos OID´s do Conversor por Categoria. (continua)

Nomenclatura Textual	Descrição	Categoria
WaveLenghtConv	Árvore Destinada ao Conversor de Comprimento de Ondas	N.A.
system	Grupo Destinado para atributos do sistema	N.A.
systemUpTime	Indica o tempo em que o Conversor foi ligado	Estatística
systemFreeMemory	Memória Não Utilizada Disponível para o Conversor	Estatística
systemPhysMemory	Memória Física Total do Conversor	Componente
systemMaxUserMem	Máximo de Memória Permitido para Utilização - Pode ser utilizado para limitar a memória do conversor em situações de simulação	Ação
systemIdleCPU	Informa Porcentagem de ociosidade da CPU	Estatística
systemRestart	Se configurado para TRUE, o conversor sofre uma reinicialização	Ação
statusChannelTable	Tabela dos Canais Livres, utilizada para informar quais canais podem ser utilizados dado uma necessidade de estabelecer um circuito	N.A.
statusChannelEntry	Identifica cada entrada na tabela de canais livres	N.A.
statusChannelID1	Indice da tabela de canais livres. Representa fisicamente o canal no conversor.	N.A.
statusChannelStatus	Indica o status do canal - 0 (livre), 1(ocupado),2(com falha) - Permite mudar o status de forma a gerenciar o canal	Ação/Status
statusChannelNr	Número do canal livre	Estatística
freeChannelfree	Reservado para liberação do canal em evento ainda não previsto	Ação
channelTable	Tabela de Situação dos Canais	N.A.
channelEntry	Identifica cada entrada na Tabela de Situação do Canais	N.A.
channelID	Indíce da Tabela de Situação do Canais - Informará o número do canal	Componente
channelInFreq	Freqüência de Entrada do Canal	Status
channelOutFreq	Freqüência de Saída do Respectivo Canal	Status
channelPumpFreq	Freqüência de Bombeio do Canal	Status
channelOutPower	Potência de Saída do Canal	Status
channelStatus	Status do Canal - 0 (livre), 1(ocupado),2(com falha),3(desativado pelo Administrador)	Status
convStatus	Grupo para Informação da Situação de Componentes	N.A.
convStatusFan1	Status de funcionamento do Cooler Principal - 0(OK),1(Falha)	Status
convStatusFan2	Status de funcionamento do Cooler Redundante - 0(OK),1(Falha)	Status
convStatusPower1	Status de funcionamento da Fonte Principal - 0(OK),1(Falha)	Status
convStatusPower2	Status de funcionamento da Fonte Redundante - 0(OK),1(Falha)	Status
convStatusCPU	Status de funcionamento da CPU - 0(OK),1(Falha)	Status
convStatusMem	Status de funcionamento da Memória - 0(OK),1(Falha)	Status
convAttr	Grupo com Intuito de Inventariar o Conversor	
convAttrPhysMem	Memória Física Instalada no Conversor	Atributos
convAttrClockCPU	Velocidade do Barramento do Processador	Atributos

Quadro 8. Representação dos OID´s do Conversor por Categoria. (conclusão)

Nomenclatura Textual	Descrição	Categoria
convAttrNrChannel	Números de Canais Existentes Fisicamente	Atributos
convChStatTable	Tabela de Estatísticas de Tráfego	
convChStatEntry	Identifica da entrada na Tabela de Estatística de Tráfego	Estatística
convChStatNr	Número do Canal	Estatística
convChStatPctIn	Quantidade de pacotes que Entraram no respectivo canal	Estatística
convChStatPctOut	Quantidade de pacotes que Sairam do respectivo canal	Estatística
convChStatErr	Quantidade de Erros ocorridos para o respectivo canal	Estatística
convChStatDrop	Quantidade de Pacotes dropados para o canal	Estatística

4.4 Propostas para a árvore da MIB

Uma vez definidos os itens para compor a *MIB* do Conversor de Comprimentos de Ondas, e tendo atribuído o *PEN* 23955 para a PUC de Campinas, a árvore proposta considerando apenas o primeiro nível de objetos ficaria como apresentada na Figura 15.

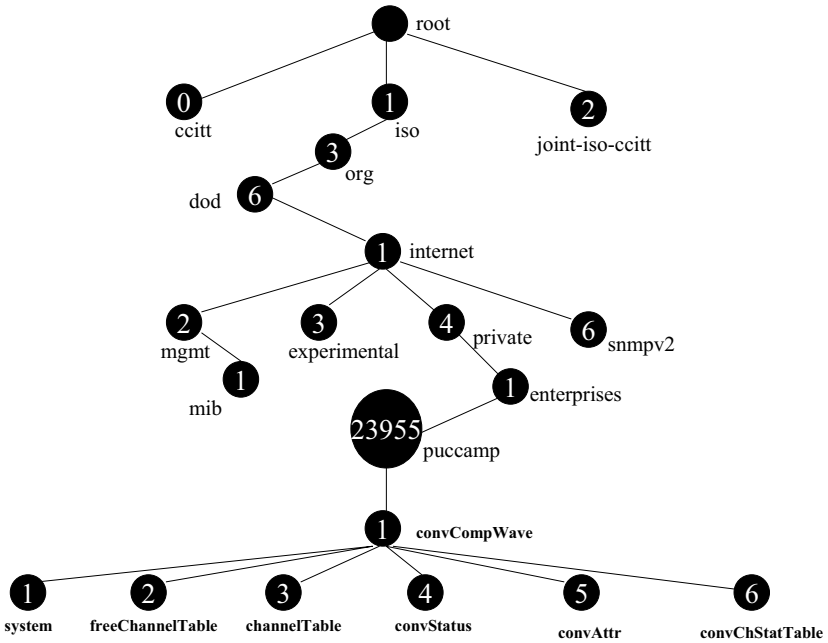

Figura 15. Árvore *MIB* dos *OID´s* Principais do Conversor.

A representação numérica e textual dos *OID´s* dos atributos especificados para o conversor de comprimento de ondas pode ser visualizada no Quadro 9.

Quadro 9. OID´s do Conversor de Comprimento de Ondas (continua)

Nomenclatura Numérica	Nomenclatura Textual
1.	iso
1.3.	iso.org
1.3.6.	iso.org.dod
1.3.6.1.	iso.org.dod.internet
1.3.6.1.4.	iso.org.dod.internet.private
1.3.6.1.4.1.	iso.org.dod.internet.private.enterprises
1.3.6.1.4.1.23955.	iso.org.dod.internet.private.enterprises.puccamp
1.3.6.1.4.1.23955.1	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv
1.3.6.1.4.1.23955.1.1	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.system
1.3.6.1.4.1.23955.1.1.1	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.system.systemUpTime
1.3.6.1.4.1.23955.1.1.2	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.system.systemFreeMemory
1.3.6.1.4.1.23955.1.1.3	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.system.systemPhysMemory
1.3.6.1.4.1.23955.1.1.4	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.system.systemIdleCPU
1.3.6.1.4.1.23955.1.1.5	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.system.systemRestart
1.3.6.1.4.1.23955.1.2	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.statusChannelTable
1.3.6.1.4.1.23955.1.2.1	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.statusChannelTable.statusChannelEntry
1.3.6.1.4.1.23955.1.2.1.1	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.statusChannelTable.statusChannelEntry.statusChannelID1
1.3.6.1.4.1.23955.1.2.1.2	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.statusChannelTable.statusChannelEntry.statusChannelStatus
1.3.6.1.4.1.23955.1.2.1.3	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.statusChannelTable.statusChannelEntry.statusChannelNr
1.3.6.1.4.1.23955.1.2.1.4	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.statusChannelTable.statusChannelEntry.statusChannelfree
1.3.6.1.4.1.23955.1.3	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.channelTable
1.3.6.1.4.1.23955.1.3.1	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.channelTable.channelEntry
1.3.6.1.4.1.23955.1.3.1.1	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.channelTable.channelEntry.channelID
1.3.6.1.4.1.23955.1.3.1.2	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.channelTable.channelEntry.channelOutFreq
1.3.6.1.4.1.23955.1.3.1.3	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.channelTable.channelEntry.channelPumpFreq
1.3.6.1.4.1.23955.1.3.1.4	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.channelTable.channelEntry.channelOutPower

Quadro 9. OID´s do Conversor de Comprimento de Ondas (conclusão)

Nomenclatura Numérica	Nomenclatura Textual
1.3.6.1.4.1.23955.1.3.1.5	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.channelTable.channelEntry.channelInFreq
1.3.6.1.4.1.23955.1.3.1.6	iso.org.dod.internet.private.enterprises.puccamp.WaveLenghtConv.channelTable.channelEntry.channelStatus
1.3.6.1.4.1.23955.4	iso.org.dod.internet.private.enterprises.puccamp. convStatus
1.3.6.1.4.1.23955.4.1	iso.org.dod.internet.private.enterprises.puccamp.convStatus.convStatusFan1
1.3.6.1.4.1.23955.4.2	iso.org.dod.internet.private.enterprises.puccamp.convStatus.convStatusFan2
1.3.6.1.4.1.23955.4.3	iso.org.dod.internet.private.enterprises.puccamp.convStatus.convStatusPower1
1.3.6.1.4.1.23955.4.4	iso.org.dod.internet.private.enterprises.puccamp.convStatus.convStatusPower2
1.3.6.1.4.1.23955.4.5	iso.org.dod.internet.private.enterprises.puccamp.convStatus.convStatusCPU
1.3.6.1.4.1.23955.4.6	iso.org.dod.internet.private.enterprises.puccamp.convStatus.convStatusMem
1.3.6.1.4.1.23955.5	iso.org.dod.internet.private.enterprises.puccamp. convAttr
1.3.6.1.4.1.23955.5.1	iso.org.dod.internet.private.enterprises.puccamp.convAttr.convAttrPhysMem
1.3.6.1.4.1.23955.5.2	iso.org.dod.internet.private.enterprises.puccamp.convAttr.convAttrClockCPU
1.3.6.1.4.1.23955.5.3	iso.org.dod.internet.private.enterprises.puccamp.convAttr.convAttrNrChannel
1.3.6.1.4.1.23955.6	iso.org.dod.internet.private.enterprises.puccamp.convAttr.convChStatTable
1.3.6.1.4.1.23955.6.1	iso.org.dod.internet.private.enterprises.puccamp.convAttr.convChStatTable
1.3.6.1.4.1.23955.6.1.1	iso.org.dod.internet.private.enterprises.puccamp.convAttr.convChStatTable.convChStatEntry
1.3.6.1.4.1.23955.6.1.2	iso.org.dod.internet.private.enterprises.puccamp.convAttr.convChStatTable.convChStatEntry.convChStatNr
1.3.6.1.4.1.23955.6.1.3	iso.org.dod.internet.private.enterprises.puccamp.convAttr.convChStatTable.convChStatEntry.convChStatPctIn
1.3.6.1.4.1.23955.6.1.4	iso.org.dod.internet.private.enterprises.puccamp.convAttr.convChStatTable.convChStatEntry.convChStatPctOut
1.3.6.1.4.1.23955.6.1.5	iso.org.dod.internet.private.enterprises.puccamp.convAttr.convChStatTable.convChStatEntry.convChStatErr
1.3.6.1.4.1.23955.6.1.6	iso.org.dod.internet.private.enterprises.puccamp.convAttr.convChStatTable.convChStatEntry.convChStatDrop

4.5 Construção da MIB para o Conversor

A *ASN.1*, que é uma linguagem definida pelo *ITU-T* para a descrição de tipos de dados e de valores, muito utilizada na especificação de dados em protocolos e serviços de telecomunicações, especialmente os que envolvem os

níveis de aplicação é o padrão utilizado para escrita de uma *MIB*. Entretanto o principal desafio é após colocar a árvore da *MIB* nesta notação, é ter a absoluta certeza que a sintaxe está correta, ou seja, validar a estrutura.

Dentro deste contexto, foram pesquisadas ferramentas na Internet que auxiliassem o desenvolvimento de uma *MIB*. Tais ferramentas são organizadas nas seguintes categorias:

- ***MIB Builder*** - Ferramenta para construir a *MIB* através de interface gráfica

- ***MIB Compiler*** - Ferramenta para compilar a *MIB*, e portanto validar a mesma

- ***MIB Explorer*** - Ferramenta para explorar a *MIB*. Uma vez a *MIB* pronta, informa-se o endereço do equipamento, e começa-se a consultar ou alterar os atributos *SNMP*.

Infelizmente, não foi encontrada nenhuma ferramenta destas categorias, que tenham o benefício de interface gráfica baseadas em código aberto. Então foi feito uma prévia avaliação de 3 pacotes de programas comerciais, cujo propósito é a construção de *MIB´s*, apresentados no Quadro 10.

Quadro 10. Programas Comerciais para Desenvolvimento de *MIB´s* e Agentes *SNMP*.

Ferramenta	Empresa	Site
MIB Smithy - Professional Edition	Muonics	http://www.muonics.com/
NuDesign Visual MIBuilder 4.5	NuDesign Team Inc.	http://www.ndt-inc.com
MG-SOFT Visual MIB Builder Version 6.0 Build 88	MG-Soft Corporation	http://www.mg-soft.com

Estas ferramentas possuem recursos que facilitam e agilizam o processo de criação de *MIB´s*. Dentre algumas características interessantes que consegui avaliar:

- Validação dos atributos à medida que se vai estruturando a *MIB*;

- Conversão de MIB versão *SNMP v1* para *SNMP v2*

- Possibilidade de criar uma *MIB* na versão *SNMP v3*

- Recursos *drag-and-drop* (arrastar e soltar) com o mouse para estruturar a árvore *MIB*

- Permitem a importação de outras *MIB´s*

Em nenhuma destas versões foi possível o desenvolvimento de uma *MIB* completa para o conversor, devido a limitações impostas para uma versão apenas para avaliação do produto.

Os custos destas ferramentas estão em torno de *US$* 1.000 no mínimo, e a meta deste trabalho não é de utilizar ferramentas comerciais, mas estamos interessados em avaliá-las de forma rápida a fim de comparar os esforços com ferramentas de código aberto para o mesmo fim.

Neste ponto vale ressaltar que encontra-se disponível uma gama de utilitários para o desenvolvimento de *MIB´s* e agentes, organizada pelo mesmo grupo responsável pela manutenção do projento *NET-SNMP*. Contudo estes não dispõem de facilidades e interfaces gráficas disponíveis nos aplicativos comerciais.

O Quadro 11 indica a listagem em *ASN.1* para a *MIB* do Conversor de Comprimento de Ondas para Redes Totalmente Ópticas .

Quadro 11. *MIB* em *ASN.1* para o Conversor de Comprimentos de Onda Totalmente Óptico.

(continua)

```
PUCCAMP-CONV-MIB DEFINITIONS ::= BEGIN

IMPORTS
        MODULE-IDENTITY, OBJECT-TYPE, enterprises,
        Counter64, Integer32, Gauge32, TimeTicks
                FROM SNMPv2-SMI
        TruthValue
                FROM SNMPv2-TC;

puccampModIdentity MODULE-IDENTITY
        LAST-UPDATED "200611122045Z"
        ORGANIZATION
                "Pontificia Universidade Catolica de Campinas"
        CONTACT-INFO
                "Carlos Roberto Schimidt
           Endereco:   Rodovia Dom Pedro I, km 136
                    Parque das Universidades
        CEP: 13086-900 - Campinas - Sao Paulo - Brazil

           email:    carlos.schimidt@puc-campinas.br"
```

Quadro 11. *MIB* em *ASN.1* para o Conversor de Comprimentos de Onda Totalmente Óptico.

(continuação)

```
        DESCRIPTION

                "Esta MIB foi desenvolvida por Aluno da PUC-CAMP para
        especificar os atributos de um Conversor de Comprimentos
        de ondas para redes totalmente opticas"

::= { puccamp 2 }

puccamp              OBJECT IDENTIFIER ::= { enterprises 23955 }
WaveLenghtConv      OBJECT IDENTIFIER ::= { puccamp 1 }
system               OBJECT IDENTIFIER ::= { WaveLenghtConv 1 }
convStatus           OBJECT IDENTIFIER ::= { WaveLenghtConv 4 }
convAttr             OBJECT IDENTIFIER ::= { WaveLenghtConv 5 }

systemUpTime  OBJECT-TYPE
        SYNTAX     TimeTicks
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Mostra quanto tempo o conversor esta funcionando"
        ::= { system 1 }

systemFreeMemory  OBJECT-TYPE
        SYNTAX     Gauge32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Apresenta quanto existe de memoria livre no conversor"
        ::= { system 2 }

systemPhysMemory  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Total de memoria existente no conversor"
        ::= { system 3 }

systemIdleCPU  OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Percentagem de CPU disponivel para o sistema"
        ::= { system 4 }

systemRestart  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-write
        STATUS     current
        DESCRIPTION
                "Forcar reboot do conversor. Valor padrao e 0, se alterado para
        0 o conversor sofrera uma reinicializacao"
        DEFVAL { 0 }
        ::= { system 5 }

statusChannelTable  OBJECT-TYPE
        SYNTAX SEQUENCE OF FreeChannelEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "Tabela que controla a disponibilidade dos canais"
        ::= { WaveLenghtConv 2 }

statusChannelEntry  OBJECT-TYPE
        SYNTAX  FreeChannelEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "Descreve as entradas da tabela freeChannel"
        INDEX { statusChannelID1 }
        ::= { statusChannelTable 1 }
```

Quadro 11. *MIB* em *ASN.1* para o Conversor de Comprimentos de Onda Totalmente Óptico.

(continuação)

```
statusChannelEntry ::= SEQUENCE {
        statusChannelID1
                Integer32,
        statusChannelStatus
                Integer32,
        statusChannelNr
                Integer32,
        statusChannelfree
                Integer32
}

statusChannelID1  OBJECT-TYPE
        SYNTAX     Integer32 (1..32)
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "Indice da tabela freeChannel"
        ::= { statusChannelEntry 1 }

statusChannelStatus  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-create
        STATUS     current
        DESCRIPTION
                "Status do respectivo canal. 0 - livre, 1 - ocupado"
        ::= { statusChannelEntry 2 }

statusChannelNr  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-create
        STATUS     current
        DESCRIPTION
                "numero do canal associado de acordo com a tabela do ITU"
        ::= { statusChannelEntry 3 }

statusChannelfree  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-create
        STATUS     current
        DESCRIPTION
                "Reservado para futura implementacao onde seja necessario
        liberar o canal por situacoes anormais nao previstas"
        ::= { statusChannelEntry 4 }

channelTable  OBJECT-TYPE
        SYNTAX SEQUENCE OF ChannelEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "Tabela que possui as associacoes dos canais, com as
        associacoes das frequencias utilizadas, ou seja, a frequencia de
        entrada se houve conversao a frequencia de bombeio, e a frequencia de
        saida, mais a potencia medida em miliwatts"
        ::= { WaveLenghtConv 3 }

channelEntry  OBJECT-TYPE
        SYNTAX  ChannelEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "Descreve as entradas da tabela channelTable"
        INDEX { channelID }
        ::= { channelTable 1 }

ChannelEntry ::= SEQUENCE {
        channelID
                Integer32,
        channelInFreq
                OCTET STRING,
        channelOutFreq
                OCTET STRING,
        channelPumpFreq
```

Quadro 11. *MIB* em *ASN.1* para o Conversor de Comprimentos de Onda Totalmente Óptico.

(continuação)

```
                        OCTET STRING,
                channelOutPower
                        OCTET STRING
        }

        channelID  OBJECT-TYPE
                SYNTAX     Integer32 (1..32)
                MAX-ACCESS read-only
                STATUS     current
                DESCRIPTION
                        "Indice da tabela channelTable"
                ::= { channelEntry 1 }

        channelInFreq  OBJECT-TYPE
                SYNTAX     OCTET STRING
                MAX-ACCESS read-write
                STATUS     current
                DESCRIPTION
                        "Frequencia de entrada efetiva"
                ::= { channelEntry 2 }

        channelOutFreq  OBJECT-TYPE
                SYNTAX     OCTET STRING
                MAX-ACCESS read-write
                STATUS     current
                DESCRIPTION
                        "Frequencia de saida. Sera a mesma de entrada se nao houver
                conversao do sinal."
                ::= { channelEntry 3 }

        channelPumpFreq  OBJECT-TYPE
                SYNTAX     OCTET STRING
                MAX-ACCESS read-write
                STATUS     current
                DESCRIPTION
                        "Frequencia de bombeio utilizada para a conversao do sinal
                quando necessario"
                ::= { channelEntry 4 }

        channelOutPower  OBJECT-TYPE
                SYNTAX     OCTET STRING
                MAX-ACCESS read-write
                STATUS     current
                DESCRIPTION
                        "Potencia apurada quando necessario a conversao do sinal"
                ::= { channelEntry 5 }

        convStatusFan1  OBJECT-TYPE
                SYNTAX     Integer32
                MAX-ACCESS read-only
                STATUS     current
                DESCRIPTION
                        "Apresenta o status da ventoinha. 0 - normal - 1 - com defeito,
                2 - desligado pelo administrador"
                ::= { convStatus 1 }

        convStatusPower  OBJECT-TYPE
                SYNTAX     Integer32
                MAX-ACCESS read-only
                STATUS     current
                DESCRIPTION
                        "Apresenta o status da fonte de energia. 0 - normal - 1 - com
                defeito,
                2 - desligado pelo administrador"
                ::= { convStatus 2 }

        convStatusCPU  OBJECT-TYPE
                SYNTAX     Integer32
                MAX-ACCESS read-only
                STATUS     current
                DESCRIPTION
                        "Apresenta o status da CPU. 0 - normal - 1 - com defeito,
                2 - desligado pelo administrador"
```

Quadro 11. *MIB* em *ASN.1* para o Conversor de Comprimentos de Onda Totalmente Óptico.

(continuação)

```
              ::= { convStatus 3 }

            convStatusMem  OBJECT-TYPE
                    SYNTAX     Integer32
                    MAX-ACCESS read-only
                    STATUS     current
                    DESCRIPTION
                            "Apresenta o status da Memoria. 0 - normal, 1 - apresentado
                    erros,
                    2 - desligado pelo administrador"
                    ::= { convStatus 4 }

            convAttrPhysMem  OBJECT-TYPE
                    SYNTAX     Integer32
                    MAX-ACCESS read-only
                    STATUS     current
                    DESCRIPTION
                            "Memória Física Instalada no Conversor"
                    ::= { convAttr 1 }

            convAttrClockCPU  OBJECT-TYPE
                    SYNTAX     Integer32
                    MAX-ACCESS read-only
                    STATUS     current
                    DESCRIPTION
                            "Velocidade do Barramento do Processador "
                    ::= { convAttr 2 }

            convAttrNrChannel  OBJECT-TYPE
                    SYNTAX     Integer32
                    MAX-ACCESS read-only
                    STATUS     current
                    DESCRIPTION
                            "Números de Canais Existentes Fisicamente. Embora a fibra
                    optica permita multiplos canais devido a multiplexacao, a limitacao
                    sera em funcao da capacidade de
                    processamento e memoria do equipamento"
                    ::= { convAttr 3 }

            convChStatTable  OBJECT-TYPE
                    SYNTAX SEQUENCE OF ConvChStatEntry
                    MAX-ACCESS not-accessible
                    STATUS     current
                    DESCRIPTION
                            "Tabela de Estatísticas de Tráfego"
                    ::= { WaveLenghtConv 6 }

            convChStatEntry  OBJECT-TYPE
                    SYNTAX  ConvChStatEntry
                    MAX-ACCESS not-accessible
                    STATUS     current
                    DESCRIPTION
                            "Identifica da entrada na Tabela de Estatística de Tráfego"
                    INDEX { convChStatNr }
                    ::= { convChStatTable 1 }

            ConvChStatEntry ::= SEQUENCE {
                    convChStatNr
                            Integer32,
                    convChStatPctIn
                            Counter64,
                    convChStatPctOut
                            Counter64,
                    convChStatErr
                            Counter64,
                    convChStatDrop
                            Counter64
            }

            convChStatNr  OBJECT-TYPE
                    SYNTAX     Integer32 (1..32)
                    MAX-ACCESS read-only
                    STATUS     current
```

Quadro 11. *MIB* em *ASN.1* para o Conversor de Comprimentos de Onda Totalmente Óptico.

(conclusão)

```
              DESCRIPTION
                        "Número do Canal de saida para qual as estatisticas
                correspondem"
                ::= { convChStatEntry 1 }

            convChStatPctIn  OBJECT-TYPE
                SYNTAX     Counter64
                MAX-ACCESS read-only
                STATUS     current
                DESCRIPTION
                        "Quantidade de pacotes de entrada no respectivo canal"
                ::= { convChStatEntry 2 }

            convChStatPctOut  OBJECT-TYPE
                SYNTAX     Counter64
                MAX-ACCESS read-only
                STATUS     current
                DESCRIPTION
                        "Quantidade de pacotes que Sairam do respectivo canal"
                ::= { convChStatEntry 3 }

            convChStatErr  OBJECT-TYPE
                SYNTAX     Counter64
                MAX-ACCESS read-only
                STATUS     current
                DESCRIPTION
                        "Quantidade de Erros ocorridos para o respectivo canal"
                ::= { convChStatEntry 4 }

            convChStatDrop  OBJECT-TYPE
                SYNTAX     Counter64
                MAX-ACCESS read-only
                STATUS     current
                DESCRIPTION
                        "Quantidade de Pacotes dropados para o canal"
                ::= { convChStatEntry 5 }
            END
```

Uma forma de validar gratuitamente a sintaxe da *MIB* é através de *URL´s* disponíveis na Internet próprias para este fim. A adotada para a análise da *MIB* do conversor foi http://www.simpleweb.org/ietf/mibs/validate/, em virtude desta *URL* fazer parte de uma organização que provê serviço gratuito de informação sobre gerenciamento de redes para as comunidades de pesquisa. O arquivo foi submetido para análise conforme Figura 16.

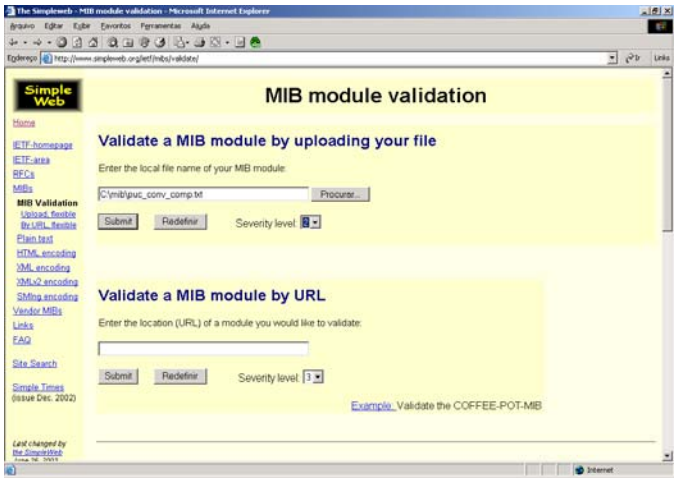

Figura 16. Submissão para Validação da *MIB* para o Conversor.

Após a validação da *MIB*, o resultado obtido é ilustrado na Figura 17.

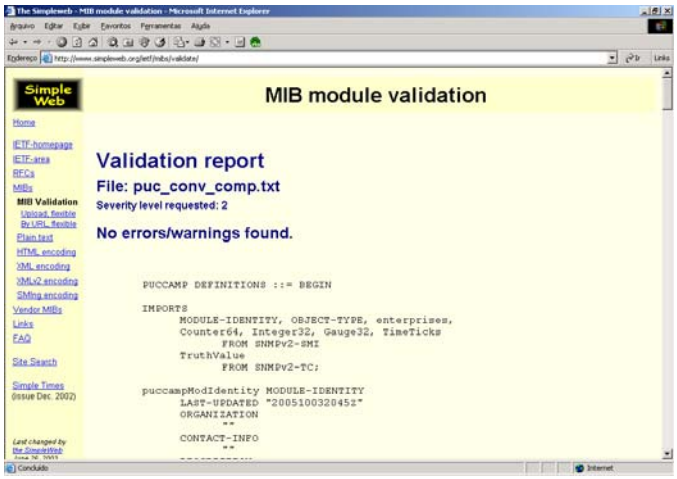

Figura 17. Relatório de Validação da *MIB* sem erros.

4.6 Criação do Agente *SNMP*

Após a elaboração da *MIB* conforme determinado na linguagem *ASN.1*, foi necessário criar um agente *SNMP* estendido para permitir a gerência do conversor.

Para executar esta tarefa, realizamos os seguintes passos. Primeiramente, geramos um esqueleto do código C, utilizando o utilitário mib2c disponível como um dos utilitários do *NET-SNMP*. Após isso, realizamos a instrumentação do código C, que corresponde a inserir linhas de código no esqueleto do código. Após essa etapa, o código C foi compilado como sub-agente

sendo executado com sucesso, pois conectou-se com o agente master *SNMP* e tornou-se responsável pelo tratamento dos atributos do conversor.

O código desenvolvido do agente *SNMP* está apresentado no APÊNDICE A -, ao passo que os resultados dos testes realizados estão apresentados no Capítulo CAPÍTULO 5 -. O Apêndice A apresenta uma descrição detalhada da execução de cada um destes passos e tem por objetivo ser um tutorial para outros pesquisadores e profissionais que precisem implementar agentes *SNMP*.

4.7 Relacionamento das Funções do Conversor com a *MIB*

Assumimos que haverá um nó na rede totalmente óptica que assumirá a função de gerência da rede. Este nó é comumente denominado *Network Management Station (NMS)*, e dentro do contexto deste trabalho terá a função de controlar e gerenciar todos os caminhos e freqüências numa rede totalmente óptica. Sendo assim, o tratamento dos atributos da *MIB* do conversor de comprimento de ondas serão manipulados essencialmente pelo nó *NMS*. A seguir apresentamos um exemplo para ilustrar como os softwares desenvolvidos poderiam atuar junto à gerência de redes para realizar uma conversão de comprimentos de onda.

Quando o sub-agente do conversor de comprimento de ondas é iniciado, as tabelas *channelTable* e *statusChannelTable* da *MIB* são carregadas conforme apresentado nas Tabela 4 e Tabela 5

Tabela 4. Tabela *channelTable* em seu estado inicial quando o agente *SNMP* é inicializado.

channelID	channelInFreq	channelOutFreq	channelPumpFreq	channelOutPower
21	0	192,1	0	0
22	0	192,2	0	0
23	0	192,3	0	0
24	0	192,4	0	0
25	0	192,5	0	0

Tabela 5. Tabela *statusChannelTable* em seu estado inicial quando o agente *SNMP* é inicializado.

statusChannelID1	statusChannelStatus	statusChannelNr
0	**0**	21
1	**0**	22
2	**0**	23
3	**0**	24
4	**0**	25

Quando ocorrer a comunicação de um nó B para um nó D na freqüência f_1, conforme ilustrado anteriormente na Figura 4, as tabelas da *MIB* serão atualizadas conforme demonstrado nas tabelas Tabela 6 e Tabela 7, ou seja, como não havia nenhuma freqüência sendo utilizada, todas estão livres para serem utilizadas, portanto não haverá a necessidade de conversão do sinal. Então o valor do atributo *statusChannelStatus* da tabela *statusChannelTable* é alterado para "1" indicando que a freqüência associada ao canal 21 está alocada, e na tabela *channelTable* será associado com a freqüência de saída o correspondente à freqüência de entrada sem conversão alterando o valor de *channelInFreq* para o canal 21 de 0 para 192,1. Na hipótese de que a potência de saída medida foi 1,1 mW, este valor é registrado no atributo *channelOutPower* para o canal 21. Como não houve necessidade de conversão de sinal, o atributo *channelPumpFreq* permanece zerado.

Tabela 6. Tabela *statusChannelTable* após ocorrer a primeira transmissão.

statusChannelID1	statusChannelStatus	statusChannelNr
0	**1**	21
1	**0**	22
2	**0**	23
3	**0**	24
4	**0**	25

Tabela 7. Tabela *channelTable* após ocorrer a primeira transmissão.

channelID	channelInFreq	channelOutFreq	channelPumpFreq	channelOutPower
21	192,1	192,1	0	1,1
22	0	192,2	0	0
23	0	192,3	0	0
24	0	192,4	0	0
25	0	192,5	0	0

Quando ocorrer a comunicação de um nó A para um nó D na freqüência f_2, as tabelas da MIB serão atualizadas conforme demonstrado nas tabelas Tabela 8 e Tabela 9, ou seja, como esta freqüência também não estava sendo utilizada, o procedimento segue-se similar à primeira transmissão, alterando o atributo *statusChannelStatus* da tabela *statusChannelTable* para 1, e na tabela *channelTable* os atributos referentes ao canal 22 ficam: *channelInFreq* = 192,2, *channelOutFreq* = 192,2, *channelPumpFreq* = 0, e *channelOutPower* = 1,5, assumindo que a potência de saída medida foi 1,5 mW.

Tabela 8. Tabela *statusChannelTable* após ocorrer a segunda transmissão.

statusChannelID1	statusChannelStatus	statusChannelNr
0	**1**	21
1	**1**	22
2	**0**	23
3	**0**	24
4	**0**	25

Tabela 9. Tabela channelTable após ocorrer a segunda transmissão.

channelID	channelInFreq	channelOutFreq	channelPumpFreq	channelOutPower
21	192,1	192,1	0	1,1
22	192,2	192,2	0	1,5
23	0	192,3	0	0
24	0	192,4	0	0
25	0	192,5	0	0

Quando o nó A iniciar outra transmissão para o nó D na freqüência f_1, chegamos na situação indicada como um problema, pois esta freqüência já está sendo utilizada entre o trecho compreendido entre os nós C e D. O conversor então irá trocar alguns pacotes de sinalização com o nó que originou a transmissão de forma a verificar o limiar da potência e negociar a conversão do comprimento de onda da freqüência f_1 de origem para a primeira freqüência que estiver disponível para o trecho compreendido entre os nós C e D, que no nosso exemplo seria a freqüência f_3. Após todo este processo as tabelas da MIB ficariam preenchidas conforme demonstrado nas tabelas Tabela 10 e Tabela 11, ou seja, o atributo *statusChannelStatus* da tabela *statusChannelTable* para o canal 23 tem seu valor alterado para 1 indicando a alocação desta freqüência, e a primeira

freqüência de saída disponível na tabela channelTable é selecionada para a conversão, que neste exemplo é a freqüência 192,3. O algoritmo para cálculo da freqüência de bombeio determina que a mesma deve ser 192,2 para que uma freqüência de entrada em 192,1 seja convertida para um freqüência de saída de 192,3. Então o valor 192,2 é registrado no aributo channelPumpFreq, e assumimos que a potência medida nesta transmissão foi de 1 mW, portanto o atributo channelOutPower é alterado para 1.

Tabela 10. Tabela *statusChannelTable* após alocar a freqüência de saída.

statusChannelID1	statusChannelStatus	statusChannelNr
0	**1**	21
1	**1**	22
2	**1**	23
3	**0**	24
4	**0**	25

Tabela 11. Tabela *channelTable* após a conversão do sinal de f_1 para f_3.

channelID	channelInFreq	channelOutFreq	channelPumpFreq	channelOutPower
21	192,1	192,1	0	1,1
22	192,2	192,2	0	1,5
23	192,1	192,3	192,2	1
24	0	192,4	0	0
25	0	192,5	0	0

Dentro deste raciocínio, o nó NMS com a estrutura proporcionada pela MIB do conversor conseguirá proporcionar maior eficácia e utilização de uma rede totalmente óptica.

O conversor de comprimentos de onda em desenvolvimento na PUC-Campinas ainda não está concluído porque depende da chegada de equipamentos importados, que estão sendo financiados pela FAPESP. A fim de verificarmos a operação dos softwares de gerência desenvolvidos nesse trabalho pensamos em fazer alguns testes mais simplificados com os equipamentos que temos em nossos laboratórios. No entanto, isso necessitaria a integração com os softwares de controle destes equipamentos e estes softwares ainda estão em fase de desenvolvimento.

Por essas razões, os testes apresentados neste capítulo restringem-se a mostrar que a *MIB* foi compilada com sucesso e que o agente *SNMP* desenvolvido responde corretamente aos comandos do *SNMP*.

5.1 Compilação da MIB

De acordo com o resultado retornado pela *URL* de verificação da *MIB*, a MIB desenvolvida para o conversor de comprimento de ondas foi compilada com sucesso, pois apontava os erros de forma a facilitar a correção dos mesmos. Primeiramente os erros são enumerados conforme ilustrado na Figura 18.

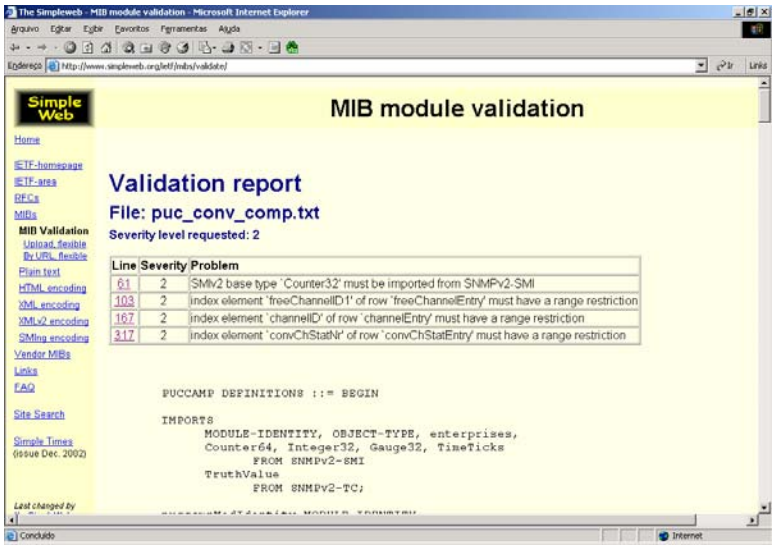

Figura 18. Relatório de erros na verificação da *MIB.*

Uma vez apresentados os erros, foi possível clicar sobre a referência do erro e ser redirecionado para a linha correspondente ao mesmo, conforme ilustrado na Figura 19.

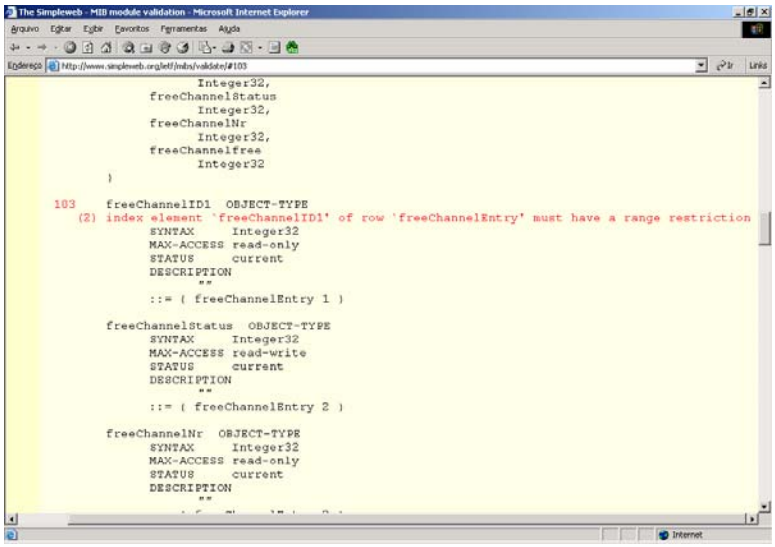

Figura 19. Destaque do erro no código da *MIB.*

5.2 Testes com comandos para o Agente SNMP

Para testar o agente SNMP, executamos comandos de consulta e manipulação no *SNMP. Todos esses comandos* funcionaram dentro das expectativas e do contexto para tratamento dos atributos da *MIB* do conversor de comprimento de ondas. A seguir, apresentamos alguns exemplos.

No caso de precisar saber se existe freqüência 192,1 sendo utilizada, executamos o comando *snmpget* para obtermos esta informação, consultando as tabelas *statusChannelTable* e *channelTable* conforme ilustrado no Quadro 12.

Quadro 12. Comando snmpget extraindo informações das tabelas *statusChannelTable* e *channelTable* da *MIB* do conversor de comprimento de ondas

```
# snmpget -v 2c -c pub_puccamp 192.168.0.178 channelInFreq.21 channelOutFreq.21
channelPumpFreq.21 channelOutPower.21
PUCCAMP-CONV-MIB::channelInFreq.21 = STRING: "0"
PUCCAMP-CONV-MIB::channelOutFreq.21 = STRING: "0"
PUCCAMP-CONV-MIB::channelPumpFreq.21 = STRING: "0"
PUCCAMP-CONV-MIB::channelOutPower.21 = STRING: "0"

# snmpget -v 2c -c pub_puccamp 192.168.0.178 statusChannelStatus.0 statusChannelNr.0
PUCCAMP-CONV-MIB::statusChannelStatus.0 = INTEGER: "0"
PUCCAMP-CONV-MIB::statusChannelNr.0 = INTEGER: "21"
```

Como foi verificado, não existia utilização para esta freqüência, então através do comando *snmpset* simulamos o registro dos valores nos atributos das tabelas, conforme ilustrado no Quadro 13.

Quadro 13. Comando snmpset alterando os valores dos atributos das tabelas *statusChannelTable* e *channelTable* da *MIB* do conversor de comprimento de ondas

```
# snmpset -v 2c -c priv_puccamp 192.168.0.178 channelInFreq.21 = 192,1 channelOutFreq.21 =
192,1 channelPumpFreq.21 = 0 channelOutPower.21 = 1,5
PUCCAMP-CONV-MIB::channelInFreq.21 = STRING: "192,1"
PUCCAMP-CONV-MIB::channelOutFreq.21 = STRING: "192,1"
PUCCAMP-CONV-MIB::channelPumpFreq.21 = STRING: "0"
PUCCAMP-CONV-MIB::channelOutPower.21 = STRING: "1,5"

# snmpset -v 2c -c priv_puccamp 192.168.0.178 statusChannelStatus.0 = 1
PUCCAMP-CONV-MIB::statusChannelStatus.0 = INTEGER: "1"
```

No caso de consultarmos a *MIB* e idenficar que a freqüência 192,1 já está sendo utilizada, o retorno do comando *snmpget* para consulta das tabelas *statusChannelTable* e *channelTable* retornou como ilustrado no Quadro 14.

Quadro 14. Comando snmpget extraindo informações das tabelas *statusChannelTable* e *channelTable* da *MIB* do conversor de comprimento de ondas, numa situação em que a freqüência já está alocada

```
# snmpget -v 2c -c pub_puccamp 192.168.0.178 channelInFreq.21 channelOutFreq.21
channelPumpFreq.21 channelOutPower.21
PUCCAMP-CONV-MIB::channelInFreq.21 = STRING: "192,1"
PUCCAMP-CONV-MIB::channelOutFreq.21 = STRING: "192,10"
PUCCAMP-CONV-MIB::channelPumpFreq.21 = STRING: "0"
PUCCAMP-CONV-MIB::channelOutPower.21 = STRING: "1,5"

# snmpget -v 2c -c pub_puccamp 192.168.0.178 statusChannelStatus.0 statusChannelNr.0
PUCCAMP-CONV-MIB::statusChannelStatus.0 = INTEGER: "1"
PUCCAMP-CONV-MIB::statusChannelNr.0 = INTEGER: "21"
```

Nesta situação, foi simulado a situação em que a conversão do sinal é necessária executando o comando *snmpset* conforme ilustrado no Quadro 15.

Quadro 15. Comando snmpset alterando os valores dos atributos das tabelas *statusChannelTable* e *channelTable* da *MIB* do conversor de comprimento de ondas na situação em que a conversão de sinal é necessária

```
# snmpset -v 2c -c priv_puccamp 192.168.0.178 channelInFreq.23 = 192,1 channelOutFreq.23 =
192,1 channelPumpFreq.23 = 0 channelOutPower.23 = 1,2
PUCCAMP-CONV-MIB::channelInFreq.21 = STRING: "192,1"
PUCCAMP-CONV-MIB::channelOutFreq.21 = STRING: "192,3"
PUCCAMP-CONV-MIB::channelPumpFreq.21 = STRING: "192,2"
PUCCAMP-CONV-MIB::channelOutPower.21 = STRING: "1,2"

# snmpset -v 2c -c priv_puccamp 192.168.0.178 statusChannelStatus.2 = 1
PUCCAMP-CONV-MIB::statusChannelStatus.2 = INTEGER: "1"
```

Contudo, o conversor de comprimento de ondas pode viabilizar a troca de informação, que não seria permitida sem a utilização do mesmo, permitindo

que o nó A conecte com o nó C na freqüência f_1, e convertendo esta freqüência entre C e D para f_3, conforme ilustrado na Figura 20.

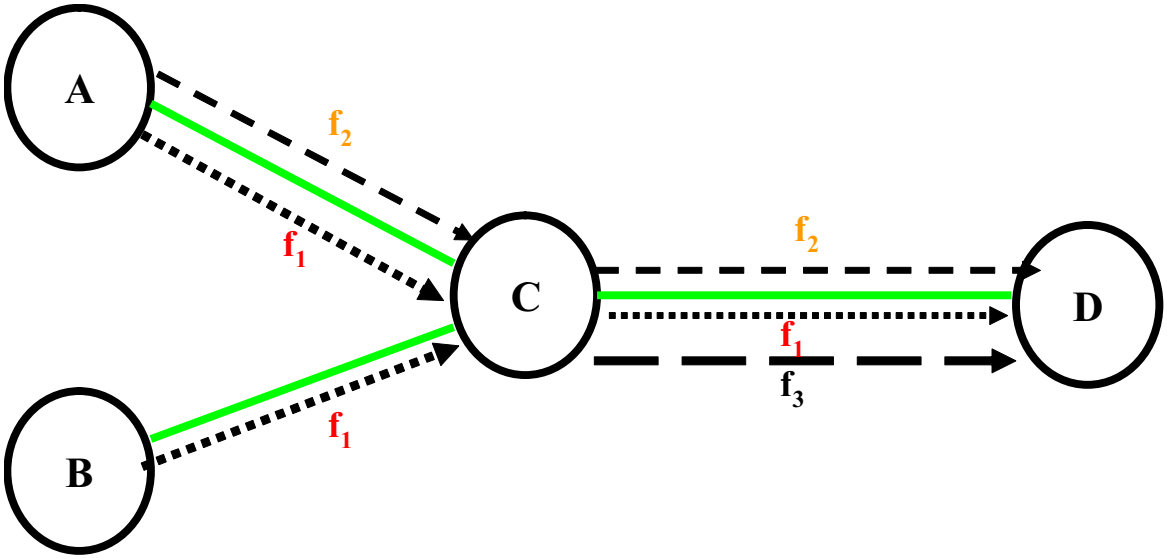

Figura 20. Nó A transmitindo para nó D utilizando conversão de F_1 para F_3.

Em vista disto o conversor de comprimento de ondas para redes totalmente ópticas, que visa resolver o problema declarado neste trabalho, tornando a comunicação das redes mais eficientes , e a *MIB* desenvolvida para o mesmo auxilia o gerenciamento dos caminhos da rede.

CAPÍTULO 6 - CONCLUSÃO E PROPOSTA DE NOVOS TRABALHOS

6.1 Conclusão

Os objetivos deste trabalho que consistiram a) na estruturação da *MIB* para um conversor de comprimento de ondas para redes totalmente ópticas e b) na manipulação de seus atributos através do agente *SNMP* desenvolvido para este fim foram plenamente atingidos.

A *MIB* foi compilada sem erros e de forma coerente com os padrões e *RFC´s* do protocolo *SNMP*, e o agente *SNMP* desenvolvido respondeu bem aos comandos de leitura e manipulação dos atributos da *MIB*.

A etapa de instrumentação do código em linguagem C para o agente *SNMP* não foi direta devido ao fato de o utilitário *mib2c* não gerar código livre de erros. No entanto, graças aos fóruns existentes na *Internet* foi possível obter respostas e soluções através da correspondência com outros participantes.

O algoritmo determinado para conversão das freqüências pelo conversor de comprimento de ondas colaborou efetivamente para o gerenciamento de dispositivos para redes totalmente ópticas. Este algoritmo consiste em encontrar a freqüência que será utilizada para o bombeio através da divisão da soma das freqüências de entrada e saída.

A constatação do funcionamento do agente *SNMP* respondendo à uma simulação do funcionamento real de um conversor de comprimento de ondas permitiu demonstrar a contribuição deste trabalho.

Através da metodologia proposta para a estruturação da *MIB* que procurou se adequar aos conceitos do *TMN*, verificou-se uma forma coerente de planejamento e elaboração dos atributos gerenciáveis.

Integrações com outros programas e controladores não foram possíveis devido à não existência dos mesmos que se encontram em fase de desenvolvimento por alunos de graduação.

O conteúdo deste trabalho teve artigo publicado em anais do III Workshop TIDIA (SCHIMIDT; ABBADE, 2006).

6.2 Proposta de novos trabalhos

Futuramente, quando da implementação do Conversor de Comprimento de Ondas, acredito ser possível separar um canal exclusivo para o gerenciamento da rede, dentro da banda destinada à multiplexação das freqüências geradas pelo conversor.

Dentro deste contexto, acredito que este trabalho pode ser complementado, estruturando agentes para trabalhar dentro destas características. Quando isto for possível, ou quando o conversor estiver preparado para entrar em escala comercial, desenvolvimento da *MIB* para o Conversor para a versão 3 do *SNMP*, considerando novas funcionalidades que estão em fase de pesquisa, para agregar atributos elevar o nível de segurança do gerenciamento.

Seria conveniente que pesquisadores que se interessem por gerenciamento, e que tenham noções avançadas de desenvolvimento de aplicativos desenvolvam interfaces gráficas e utilitários para o desenvolvimento de agentes, e em virtude disto planejo aprofundar os estudos dentro do projeto *NET-SNMP*, chegando inclusive a escrever um livro que possam estimular os pesquisadores nesta área.

Após o funcionamento deste conversor, poderá ser proposto junto ao *IETF* a criação de um novo *RFC* específico, podendo os objetos gerenciáveis deste tipo de equipamento estarem sob uma outra árvore, que não exclusiva de uma empresa, de forma padronizada.

Outra proposta seria o estudo da integração deste trabalho com a qualidade de roteamento de serviço proposta por Daheb e Pujolle (2005, v.4, p5), onde o gerenciamento de acordo de níveis de serviços ocorre em um nó da rede, assim como proposto dentro deste trabalho.

Com a evolução e desenvolvimento de *chip´*s e barramentos ópticos (INTEL, 2006), pode-se no futuro diminuir o retardo que ocorre para cálculo da freqüência de bombeio que neste no momento depende de processamento eletrônico, além de que um único *chip* poderá realizar a conversão da freqüência sem a necessidade de utilizar uma fibra óptica de 3 Km.

CAPÍTULO 7 - REFERÊNCIAS

ABBADE, M. L. F. Conversor Óptico de Comprimentos de Onda. Projeto de Pesquisa vinculado ao Programa Tidia/KyaTera, Número de processo 03/08320-2R, FAPESP, 2003.

ABOUL-MAGD . O. Automatic Switched Optical Networks (ASON) and its Related Protocols. Internet Engineering Task Force, SET. 2001. Disponível em: <http://www.ietf.org/proceedings/01mar/slides/ipo-2/index.html>. Acesso em 15 mar. 2006.

BLUMENTHAL, D.J. et al. All-optical label swapping networks and technologies. IEEE J. Light. Tech., vol. 18, pp. 2058–2075, Dez. 2000.

BLUMENTHAL, U.; WIJNEN, B. User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). Network Working Group, 2002. Disponível em: <http://www.ietf.org/rfc/rfc3414.txt?number=3414>. Acesso em 15 jun. 2005.

CARPENTER, G.; WIJNEN, B. SNMP-DPI: Simple Network Management Protocol Distributed Program Interface.[S.I.]: Network Working Group, may 1991. Disponível em:<http://www.ietf.org/rfc/rfc1228.txt?number=1228>. Acesso em 15 jun. 2005.

CASE, J.D. et al. Simple Network Management Protocol. [S.I.]: Network Working Group, 1988. Disponível em: <http://www.ietf.org/rfc/rfc1067.txt?number=1067>. Acesso em 15 jun. 2005.

CASE, J.D. et al. Simple Network Management Protocol (SNMP). [S.I.]: Network Working Group, 1989. Disponível em: <http://www.ietf.org/rfc/rfc1098.txt?number=1098>. Acesso em 15 jun. 2005.

CASE, J.D. et al. FDDI Management Information Base.[S.I.]: Network Working Group, 1992. Disponível em:<http://www.ietf.org/rfc/rfc1285.txt?number=1285>. Acesso em 15 jun. 2005.

CASE, J. D. et al. Introduction to version 2 of the Internet-standard Network Management Framework. Network Working Group, 1993a. Disponível em:<http://www.ietf.org/rfc/rfc1441.txt?number=1441>. Acesso em 15 jun. 2005.

CASE, J. D. et al. Introduction to Community-based SNMPv2. Network Working Group, 1993b. Disponível em:<http://www.ietf.org/rfc/rfc1901.txt?number=1901>. Acesso em 15 jan. 2005.

CASE, J. D. et al. Introduction and Applicability Statements for Internet Standard Management Framework. Network Working Group, 2002a. Disponível em:<http://www.ietf.org/rfc/rfc3410.txt?number=3410>. Acesso em 15 jan. 2005.

CASE, J. D. et al. Message Processing and Dispatching for the Simple Network Management Protocol (SNMP). Network Working Group, 2002b. Disponível em:<http://www.ietf.org/rfc/rfc3412.txt?number=3412>. Acesso em 15 jan. 2005.

CASE, J. et al. Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework. Network Working Group, 2003. Disponível em:<http://www.ietf.org/rfc/rfc3584.txt?number=3584>. Acesso em 15 jun. 2005.

DAHEB, B.; PUJOLLE, G. Quality of Service Routing for Service Level Agreement Conformance in Optical Networks. In: IEEE GLOBECOM, 2005, St. Louis. Proceedings of IEEE GLOBECOM 2005. Missouri: Renaissance Grand Hotel, 2005. v. 4. p5.

FRYE, R. et al. Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework. Network Working Group, 2000. Disponível em:<http://www.ietf.org/rfc/rfc2576.txt?number=2576>. Acesso em 15 jun. 2005.

FUNDAÇÃO DE AMPARO À PESQUISA DO ESTADO DE SÃO PAULO. O que é o TIDIA. Disponível em: <http://www.tidia.fapesp.br/portal/document.2006-08-10.2085223688>. Acesso em 18 nov. 2006a.

FUNDAÇÃO DE AMPARO À PESQUISA DO ESTADO DE SÃO PAULO. Projeto Kyatera. Disponível em: <www.kyatera.fapesp.br/portal/index_br.html>. Acesso em 18 nov. 2006b.

GOUGH, C. Cisco CCNP Routing Exam Certification Guide. Cisco Press, 2001.

GREMBERGEN, V. W.; HAES, S.; AMELINCKX, I. Using COBIT and the Balanced Scorecard as Instruments for Service Level Management. Information Systems Control Journal, V. 4 2003.

HARRINGTON, D.; PRESUHN, R.; WIJNEN, B. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. Network Working Group, 2002. Disponível em:<http://www.ietf.org/rfc/rfc3411.txt?number=3411>. Acesso em: 15 jun. 2005.

HEWLETT-PACKARD. HP OpenView Extensible SNMP Agent Administrator´s Guide, Hewlett-Packard Development Company, 2004. Disponível em: <http://docs.hp.com/en/5990-8153/5990-8153.pdf>. Acesso em: 03 maio 2005.

INTEL CORPORATION. Optical Interconnects for Chips Possible. Disponível em: < http://www.intel.com/technology/techresearch/research/rs03044.htm>. Acesso: 20 novembro 2006.

KASTENHOLZ, F. SNMP Communications Services.[S.l.]: Network Working Group, 1991. Disponível em:<http://www.ietf.org/rfc/rfc1270.txt?number=1270>. Acesso em: 15 jun. 2005.

LEVI, D.; MEYER, P.; STEWART, B. Simple Network Management Protocol (SNMP) Applications. Network Working Group, 2002. Disponível em:<http://www.ietf.org/rfc/rfc3413.txt?number=3413>. Acesso em 15 jan. 2005.

LIMA, M. M. A. E. Gerenciamento de Redes TCP/IP. In: Boletim bimestral sobre Tecnologia de redes, v. 1 , n. 7. dez. 1997. Disponível em: <http://www.rnp.br/newsgen/9712/gerencia.html>. Acesso em: 03 maio 2005.

MACFADEN, M. et al. Configuring Networks and Devices with Simple Network Management Protocol (SNMP). Network Working Group, 2003. Disponível em: <http://www.ietf.org/rfc/rfc3512.txt?number=3512>. Acesso em 15 jun. 2005.

MAURO, D. R.; SCHMIDT, K. J. Essential SNMP. 1 ed. Sebastopol: O´Reilly & Associates,Inc., 2001.

McCLOGHRIE, K.;ROSE, M.T. Management Information Base for network management of TCP/IP-based internets. [S.I.]: Network Working Group, 1988a. Disponível em: <http://www.ietf.org/rfc/rfc1066.txt?number=1066>. Acesso em 15 jun. 2005.

McCLOGHRIE, K.;ROSE, M.T. Structure and identification of management information for TCP/IP-based internets. [S.I.]: Network Working Group, 1988b. Disponível em: <http://www.ietf.org/rfc/rfc1065.txt?number=1065>. Acesso em 15 jun. 2005.

McCLOGHRIE,K.;ROSE, M.T. Management Information Base for Network Management of TCP/IP-based internets. Network Working Group, 1990. Disponível em: <http://www.ietf.org/rfc/rfc1156.txt?number=1156>. Acesso em 15 jun. 2005.

McCLOGHRIE,K.;ROSE, M.T. Management Information Base for Network Management of TCP/IP-based internets:MIB-II.[S.I.]: Network Working Group, 1991. Disponível em: <http://www.ietf.org/rfc/rfc1213.txt?number=1213>. Acesso em 15 jun. 2005.

McCLOGHRIE,K.;ROSE, M. T. A Convention for Describing SNMP-based Agents.[S.I.]: Network Working Group, 1992. Disponível em: <http://www.ietf.org/rfc/rfc1303.txt?number=1303>. Acesso em 15 jun. 2005.

McCLOGHRIE,K. An Administrative Infrastructure for SNMPv2. Network Working Group, 1996. Disponível em:<http://www.ietf.org/rfc/rfc1909.txt?number=1909>. Acesso em 15 jun. 2005.

McCLOGHRIE,K. et al. Structure of Management Information Version 2 (SMIv2). Network Working Group, 1999. Disponível em:<http://www.ietf.org/rfc/rfc2578.txt?number=2578>. Acesso em 15 jun. 2005.

McGINNIS, E.; PERKINS, D. Understanding SNMP MIBs. New Jersey: Prentice Hall, 1996.

NET-SNMP. NET-SNMP Tutorial. Disponível em: <http://www.net-snmp.org/tutorial-5>. Acesso em: 03 maio 2005.

OETICKER, T.; RAND, D. MRTG: The Multi Router Traffic Grapher. Disponível em: <http://people.ee.eTHz.ch/~oetiker/webtools/mrtg>. Acesso em: 03 maio 2005.

PARKER. A. et al. Managing AIX Server Farms. 1 ed. Austin: IBM Corporation, International Technical Support Organization, 2002. Disponível em: <http://www.redbooks.ibm.com/redbooks/pdfs/sg246606.pdf>. Acesso em: 03 maio 2005.

PERKINS, D. T. RMON - Remote Monitoring of SNMP-Managed. Prentice Hall, 1999.

PONTIFíCIA UNIVERSIDADE CATÓLICA DE CAMPINAS. Laboratório dos Meios de Transmissão. Conversor de Comprimento de Ondas para Redes Totalmente òpticas. Disponível em <http://www.kyatera.fapesp.br/portal/V.part/Laboratorios/lmt>. Acesso em 11 jun. 2006.

PRESUHN, R. et al. Management Information Base (MIB) for the Simple Network Management Protocol (SNMP). Network Working Group, 2002a. Disponível em: <http://www.ietf.org/rfc/rfc3418.txt?number=3418>. Acesso em 15 jun. 2005.

PRESUHN, R. et al. Transport Mappings for the Simple Network Management Protocol (SNMP). Network Working Group, 2002b. Disponível em: <http://www.ietf.org/rfc/rfc3417.txt?number=3417>. Acesso em 15 jun. 2005.

PRESUHN, R. et al. Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). Network Working Group, 2002b. Disponível em: <http://www.ietf.org/rfc/rfc3416.txt?number=3416>. Acesso em 15 jun. 2005.

RAMASWAMI, R.; SIVARAJAN, K.N. Design of logical topologies for wavelength-routed optical networks. IEEE JSAC/JLT Special Issue on Optical Networks, vol. 14, no. 5, pp. 840-851, Jun. 1996.

RAMASWAMI, R.; SIVARAJAN, K.N. Optical Networks: A Practical Perspective. 2 ed. MKP- Academic Press, 2002.

ROSE, M.T. Management Information Base for network management of TCP/IP-based internets: MIB-II.[S.I.]: Network Working Group, 1990. Disponível em: <http://www.ietf.org/rfc/rfc1158.txt?number=1158>. Acesso em 15 jun. 2005.

ROSE, M.T.;McCLOGHRIE,K. Structure and identification of management information for TCP/IP-based internets. [S.I.]: Network Working Group, 1990. Disponível em: <http://www.ietf.org/rfc/rfc1155.txt?number=1155>. Acesso em 15 jun. 2005.

ROSE, M.T. Convention for defining traps for use with the SNMP.[S.l.]: Network Working Group, 1991a. Disponível em: <http://www.ietf.org/rfc/rfc1215.txt?number=1215>. Acesso em 15 jun. 2005.

ROSE, M.T. SNMP MUX protocol and MIB.[S.l.]: Network Working Group, 1991b. Disponível em:<http://www.ietf.org/rfc/rfc1227.txt?number=1227>. Acesso em 15 jun. 2005.

SCHIMIDT, C. R.; ABBADE, M. L. F. Management Software for All-Optical Wavelength Converters. In: III Worshop FAPESP TIDIA, 2006, São Paulo. Proceedings of III Workshop TIDIA. São Paulo: Centro de Convenções Rebouças, 2006. p. 20-22. 1 CD-ROM.

SCHOFFSTALL, M.L.;DAVIN, J. Simple Network Management Protocol (SNMP). [S.l.]: Network Working Group, 1990. Disponível em: <http://www.ietf.org/rfc/rfc1157.txt?number=1157>. Acesso em 15 jun. 2005.

SNMPv3 White Paper. Disponível em: <http://www.snmp.com/snmpv3/v3white.shtml>. Acesso em: 03 maio 2005.

STEINBERG, L. Troubleshooting SNMP. Osborne - McGraw-Hill, 2000.

SHIBATA, N.; BRAUN, R.P.; WAARTS, R.G. Phase-mismatch dependence of efficiency of wave generation through FWM in a single-mode optical fiber, Journal of Quantum Electron., v. QE-23, p. 1205-1210, Jul. 1987.

SOARES, L.F.G.; LEMOS, G.; COLCHER, S. Redes de Computadores: Das LAN´s, MAN´s e WAN´s às Redes ATM. 1. ed. Rio de Janeiro: Campus, 1995.

SONG, S. et al. Intensity-dependent effects on FWM in optic fibers, Journal of Lightwave Technology, v. 17, no 11, p. 2285-2290, Nov. 1999.

STALLINGS, W. SNMP, SNMPV2, SNMPV3, AND RMON 1 AND 2. 3 ed. Addison Wesley Logman, 1999.

TOWNSEND, R. L. SNMP Application Develeper´s Guide. John Wiley Professional,1995.

THE INTERNATIONAL ENGINEERING CONSORTIUM. Telecommunications Management Network (TMN) Tutorial. Disponível em: < http://www.iec.org/online/tutorials/tmn/index.html>. Acesso em 02 fev. 2006.

TUTORIAL GPIB. Disponível em: <http://www.hit.bme.hu/people/papay/edu/GPIB/tutor.htm>. Acesso em: 03 abr. 2006.

WATERS, G. User-based Security Model for SNMPv2. Network Working Group, 1996. Disponível em:<http://www.ietf.org/rfc/rfc1910.txt?number=1910>. Acesso em 15 jun. 2005.

WIJNEN, B.; PRESUHN, R.; McCLOGHRIE, K. View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP). Network Working Group, 2002. Disponível em:<http://www.ietf.org/rfc/rfc3415.txt?number=3415>. Acesso em 15 jun. 2005.

WIKIMEDIA FOUNDATION. Application Programming Interface. Disponível em: < http://pt.wikipedia.org/wiki/API>. Acesso em 10 nov. 2006a.

WIKIMEDIA FOUNDATION. Wavelength-division multiplexing. Disponível em: <http://en.wikipedia.org/wiki/Wavelength_division_multiplexing>. Acesso em 10 out. 2006b.

APÊNDICE A - DESENVOLVIMENTO DO AGENTE *SNMP*

Este apêndice apresentará com maior nível de detalhe as etapas para a criação do sub-agente *SNMP* para o conversor de comprimento de ondas para redes totalmente ópticas, que compreende desde o cadastro do *PEN* para a PUC de Campinas até a exploração da *MIB* com alguns softwares comerciais, passando pela criação do código fonte do sub-agente e comandos *SNMP* para consulta e alteração de atributos da *MIB*, e também a criação de gráficos estatísticos com o *MRTG*.

No ítem A.1 serão descritos os passos que foram realizados para obtenção do *PEN* para a PUC de Campinas. O ítem A.2 descreverá os procedimentos realizados para o desenvolvimento do agente *SNMP*, enquanto que o ítem A.3 apresenta o código em linguagem C instrumentado para o conversor de comprimento de ondas. Uma demonstração do programa *MRTG* gerando gráficos estatísticos de atributos da *MIB* do conversor será descrita no ítem A.4, e por último o ítem A.5 apresentará alguns programas comerciais que auxiliam e facilitam o desenvolvimento de *MIB´s* e agentes *SNMP*.

A.1 Cadastro da Universidade no IANA

Qualquer empresa, entidade ou universidade pode solicitar o registro junto ao *IANA* para obtenção do *PEN*.

Recomenda-se realizar uma consulta aos números já cadastrados para verificar se a empresa não se encontra cadastrada, através da página de internet http://www.iana.org/assignments/enterprise-numbers.

Uma vez constatado que a empresa não possui um *PEN,* pode-se solicitar um através do preenchimento de formulário eletrônico disponível no endereço de internet http://www.iana.org/cgi-bin/enterprise.pl e visualizado na Figura 21.

Uma vez acessado este endereço será mostrado a página que possui o formulário que foi devidamente preenchido para a realização do cadastro da PUC Campinas, onde os passos são ilustrados em 3 figuras, sendo que a Figura 22 apresenta a primeira parte do preenchimento do formulário, a Figura 23 apresenta a segunda parte do preenchimento do formulário e a Figura 24 representa o retorno da solicitação efetuada com sucesso..

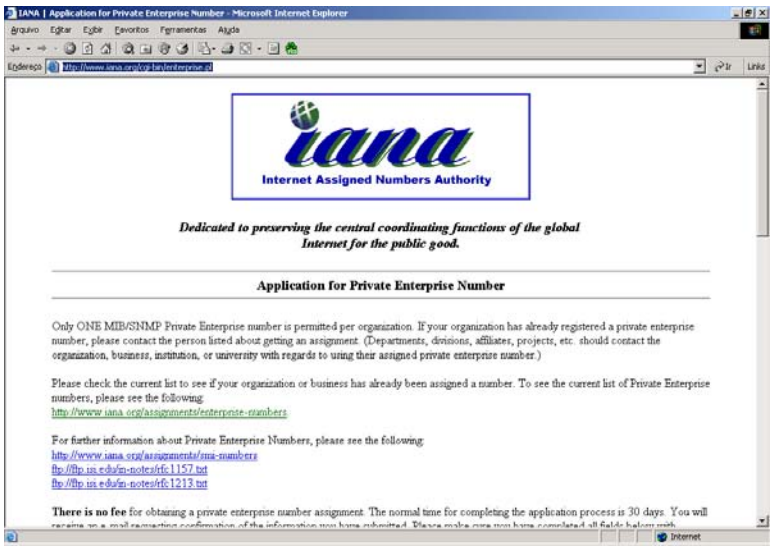

Figura 21. *Site* do Formulário de Solicitação do *PEN*.

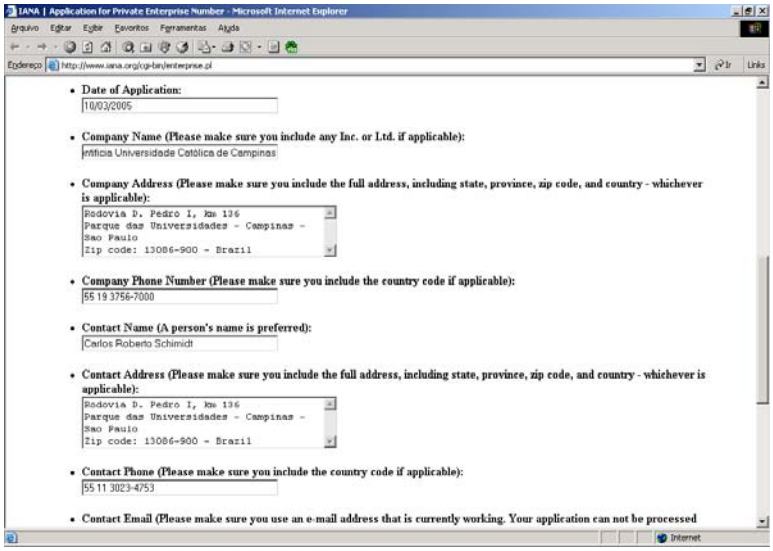

Figura 22. Primeira Parte do Preenchimento do Formulário de Solicitação de *PEN*.

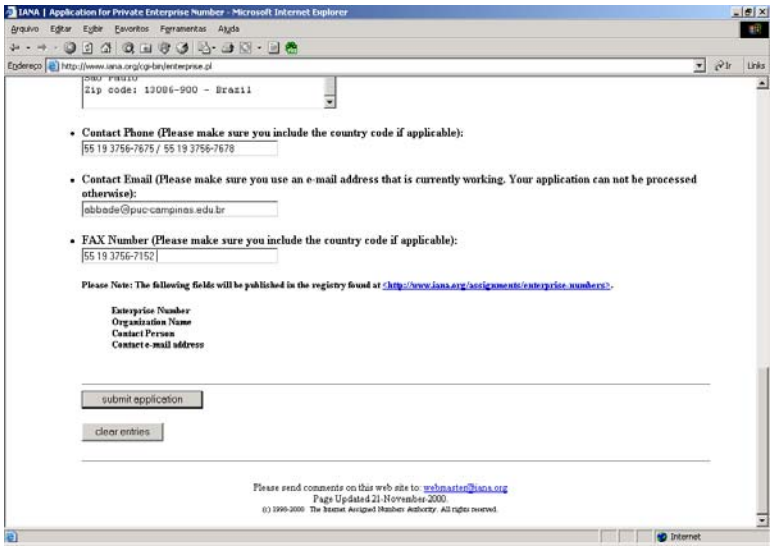

Figura 23. Segunda Primeira Parte do Preenchimento do Formulário de Solicitação de PEN.

Após efetuar o clique no botão *submit application* retornará a tela confirmando o envio do formulário conforme mostrado na Figura 24.

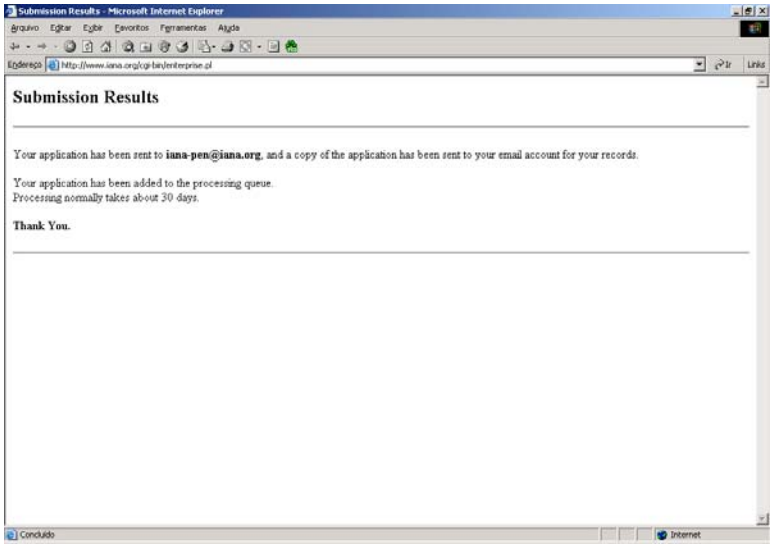

Figura 24. Retorno com sucesso da solicitação de *PEN*.

O contato cadastrado recebe um email com os dados cadastrados, onde deve-se responder o mesmo para efetivar o cadastro. O modelo do email recebido pode ser visualizado na Figura 25.

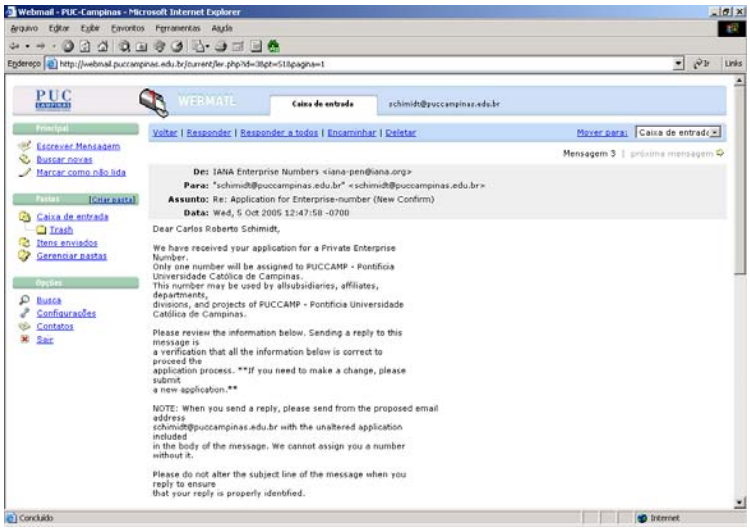

Figura 25. Primeiro email de confirmação de solicitação do *PEN*.

Após a resposta, recebe-se do *IANA* um outro email de confirmação, informando que efetivação do cadastro poderá levar até 30 dias. O modelo deste outro email recebido pode ser visualizado na Figura 26.

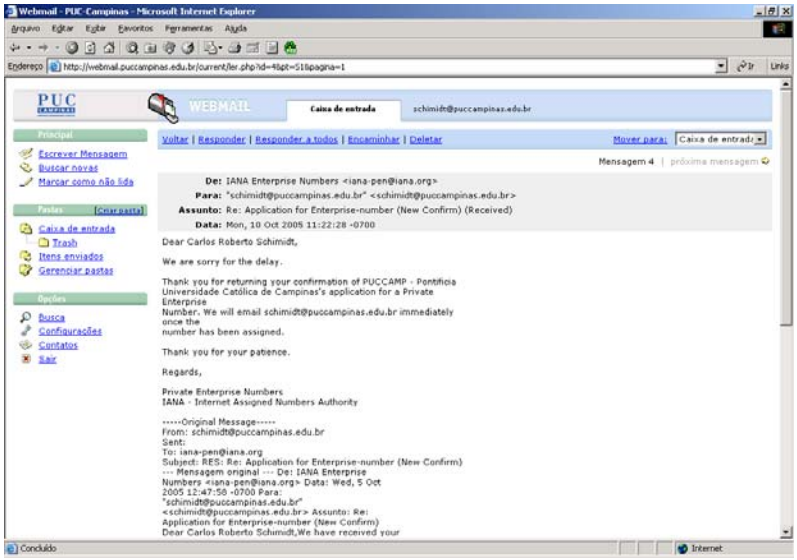

Figura 26. Segundo email de confirmação de solicitação do *PEN*.

Após a confirmação deste outro email, em aproximadamente 4 dias chegou outro email informando o *PEN* para a PUC-CAMP, cujo número designado foi 23955. Este modelo de email também pode ser visualizado na Figura 27.

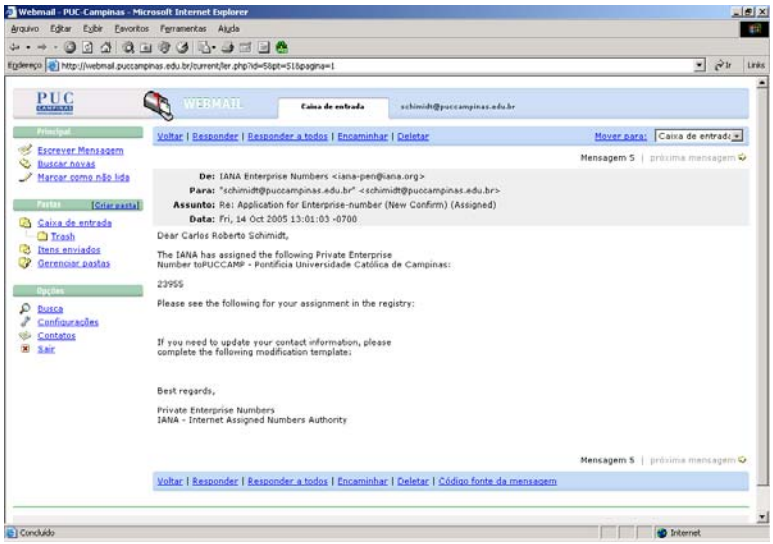

Figura 27. Email informando *PEN* da Puc-Campinas.

Foi realizada uma consulta ao site para verificação da designação do número para a PUC-CAMP, onde os dados estavam corretos, conforme pode ser visualizado na Figura 28.

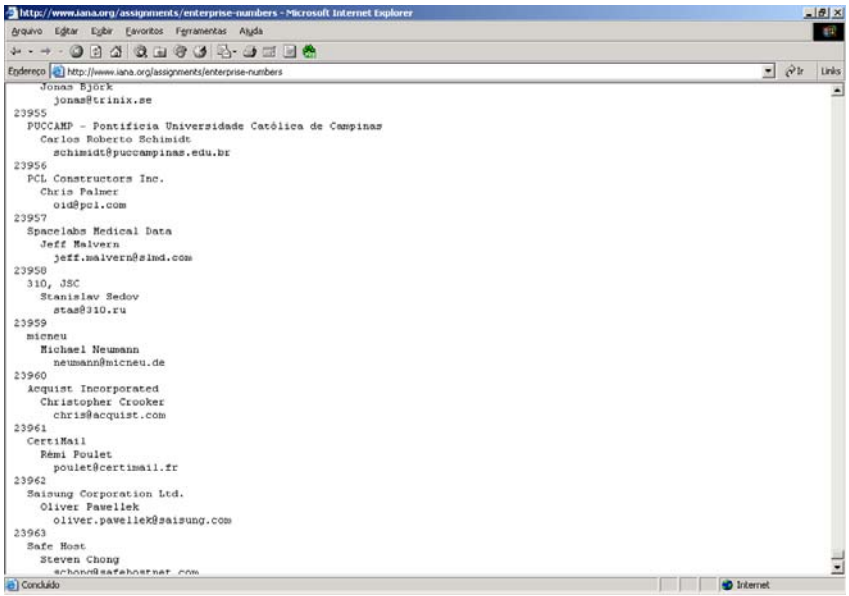

Figura 28. Confirmação do Cadastro do *PEN* para a PUC-Campinas.

O contato responsável pelo cadastro é associado com a entidade, conforme ilustrado na figura 28, mas posteriormente pode-se solicitar a alteração do contato.

É bom ressaltar que não existe nenhuma consistência para atribuição do *PEN*, onde a solicitação poderia ser feita até mesmo por uma pessoa física.

A.2 UTILIZAÇÃO DO NET-SNMP PARA DESENVOLVIMENTO DO AGENTE

O próximo passo após a elaboração da MIB é escrever um esboço do agente *SNMP* em linguagem *C*. Para isso, seguimos os passos indicados na *URL* http://www.net-snmp.org/tutorial-5/commands/mib-options.html.

O utilitário de código aberto do projeto net-snmp que tem a finalidade de gerar linhas de código em C para uma *MIB*, chama-se "*mib2c*".

Para preparar o ambiente para o desenvolvimento, recomenda-se uma distribuição *Linux* mais atualizada, com todo pacote de desenvolvimento instalado, pois o *NET-SNMP* e seus módulo dependem bastante das bibliotecas de desenvolvimento para o *linux* e compilador da linguagem C instalado no sistema.

A distribuição *linux* utilizada para o desenvolvimento foi o *Red Hat 9*, e a versão do *NET-SNMP* foi a 5.2.3 cujo código fonte foi baixado do site http://www.net-snmp.org.

Vale ressaltar que a distribuição linux instalada no microcomputador conectado aos elementos que fazem parte do protótipo do conversor, era a princípio Conectiva 10 e posteriormente fora instalado a distribuição Fedora. Entretanto os passos aqui descritos estão preparados para funcionar em qualquer distribuição.

Primeiramente foi criado um diretório temporário para acomodar o arquivo de instalação do produto:

```
[root@localhost tmp]# mkdir /tmp/install
```

Em seguida copiou-se o arquivo de instalação para este diretório, onde as propriedades do mesmo ficaram desta forma:

```
[root@localhost install]# ll
total 3924
-rw-r--r--  1 schimidt schimidt 4006389 Jul 29 17:16 net-snmp-5.2.3.tar.gz
```

Como o pacote está compactado, o primeiro passo é descompactar o pacote com o comando gunzip, como se segue:

```
[root@localhost install]# gunzip net-snmp-5.2.3.tar.gz
```

Após a descompactação, listando o conteúdo do diretório de instalação temos:

```
[root@localhost install]# ll
total 21024
-rw-r--r--  1 schimidt schimidt 21493760 Jul 29 17:16 net-snmp-5.2.3.tar
```

Para abrir o pacote e extrair todos arquivos contidos nele, utiliza-se o utilitário tar conforme sintaxe abaixo:

```
[root@localhost install]# tar -xvf net-snmp-5.2.3.tar
```

Após a extração do pacote, se listarmos o diretório de instalação percebe-se a existência de um subdiretório:

```
[root@localhost install]# ll
total 21032
drwxr-xr-x  14      555 users        4096 Jul 13 20:33 net-snmp-5.2.3
-rw-r--r--   1 schimidt schimidt 21493760 Jul 29 17:16 net-snmp-5.2.3.tar
```

Deve-se então entrar no subdiretório:

```
[root@localhost install]# cd net-snmp-5.2.3
```

Iniciou-se então o processo de configuração que antecede a instalação. É necessário configurar o *NET-SNMP* para instalar com capacidade e suporte para os módulos escritos na linguagem *PERL*, bastando para isso executar conforme sintaxe abaixo:

```
[root@localhost net-snmp-5.2.3]# ./configure --with-perl-modules
```

No processo interativo deve-se informar os dados solicitados, e após a exibição do sumário da configuração, pode-se proceder para a compilação do produto net-snmp, utilizando o comando *make* conforme abaixo:

```
[root@localhost net-snmp-5.2.3]# make
```

Depois de executar o comando *make*, executar novamente o comando com o argumento "*install*":

```
[root@localhost net-snmp-5.2.3]# make install
```

Após a instalação dos pacotes adicionais do *NET-SNMP*, os passos para converter uma *MIB* para um esqueleto em linguagem C, são descritos nos próximos parágrafos.

Deve-se primeiramente copiar a *MIB* construída para o conversor para o diretório onde estão todas as *MIB´s* que o produto reconhece, ou seja, o diretório */usr/local/share/snmp/mibs*:

```
[root@lab1 mibs]# cp PUCCAMP-CONV-MIB.txt /usr/local/share/snmp/mibs
```

Após a cópia da *MIB* do Conversor de Comprimento de Ondas para o respectivo diretório, deve-se configurar uma variável de ambiente de nome "*MIBS*" para que o utilitário *mib2c* possa ler os *OID´s* das *MIB´s* contidas no diretório. Essa definição da variável de ambiente pode ser feita de duas formas, quais sejam:

Na primeira esta variável de ambiente fica com o valor "*ALL*", sendo que o utilitário poderá tratar qualquer *OID* de qualquer *MIB* que esteja sob o diretório */usr/share/snmp/mibs*:

```
[root@lab1 mibs]# export MIBS=ALL
```

Na segunda forma, declara-se de forma explícita a *MIB* com a qual se deseja gerar código C para os atributos:

```
[root@lab1 mibs]# export MIBS=+PUCCAMP-CONV-MIB
```

O sinal de "+" significa que esta *MIB* será carregada pelo utilitário *mib2c* além das *MIB´s* principais necessárias que servem como base para atender os padrões snmp.

Outra alternativa para não ter que copiar a *MIB* para o diretório onde ficam alojadas todas as outras *MIB´s* reconhecidas pelo *NET-SNMP* seria criar

uma sub-árvore de diretórios *(.snmp/mib*) sob o "diretório casa" do usuário no sistema operacional *linux*. Desta forma os passos seriam:

```
[root@lab1 mibs]# mkdir $HOME/.snmp
[root@lab1 mibs]# mkdir $HOME/.snmp/mibs
[root@lab1 mibs]# cp PUCCAMP-CONV-MIB.txt $HOME/.snmp/mibs
[root@lab1 mibs]# export MIBS=+PUCCAMP-CONV-MIB
```

Deve-se criar um arquivo de configuração do snmp através da execução do seguinte comando:

```
[root@lab root]# snmpconf -r none -g basic_setup
```

As respostas às questões resultantes do comando acima foram informadas como se segue:

```
  The location of the system: PUC-Campinas LAB
  The contact information: Carlos Roberto Schimidt
  does this host offer physical services (eg, like a repeater) [answer 0 or
1]: 1
  does this host offer datalink/subnetwork services (eg, like a bridge): 0
  does this host offer internet services (eg, supports IP): 1
  does this host offer end-to-end services (eg, supports TCP): 1
  does this host offer application services (eg, supports SMTP): 1
  Do you want to configure the agent's access control? (default = y):
  Do you want to allow SNMPv3 read-write user based access (default = y): n
  Do you want to allow SNMPv3 read-only user based access (default = y): n
  Do you want to allow SNMPv1/v2c read-write community access (default =
y):
  Enter the community name to add read-write access for: priv_puccamp
  The community name to add read-only access for: pucpublic
  The hostname or network address to accept this community name from
[RETURN for all]:
  The OID that this community should be restricted to [RETURN for no-
restriction]:
  Finished Output: rocommunity  pucpublic
  Do another rocommunity line? (default = y): n
  Do you want to configure where and if the agent will send traps? (default
= y): y
  Do you want to configure the agent's ability to monitor various aspects
of your system? (default = y):
  Do you want to configure the agents ability to monitor processes?
(default = y): n
  Do you want to configure the agents ability to monitor disk space?
(default = y) : n
  Do you want to configure the agents ability to monitor load average?
(default =  y): n
  Do you want to configure the agents ability to monitor file sizes?
(default = y) : n
```

Após a conclusão deste processo, um arquivo denominado *snmpd.conf* é gerado, que trata-se de um arquivo de configuração que instrui o

programa *snmpd* como funcionar. Este arquivo deve ser copiado para o diretório */etc*:

```
[root@lab root]# cp -p snmpd.conf /etc/
```

Pode-se testar o reconhecimento da *MIB* através de outro utilitário denominado *snmptranslate*. Este utilitário visa traduzir o nome de um determinado *OID* para seu respectivo número. Exemplos:

```
[root@lab root]# snmptranslate -IR systemIdleCPU
PUCCAMP-CONV-MIB::systemIdleCPU
[root@localhost etc]# snmptranslate -On PUCCAMP-CONV-MIB::systemIdleCPU
.1.3.6.1.4.1.23004.1.1.5
```

Para iniciar o programa deamon *snmpd* informando qual arquivo que contém as configurações de funcionamento, deve-se executar o comando abaixo:

```
[root@localhost ~]#  /usr/local/sbin/snmpd -c /etc/snmpd.conf
```

Deve-se instalar os módulos perl que vem com o software *NET-SNMP* para que o utilitário *mib2c* funcione sem problemas. Os passos para tal são os seguintes:

```
# cd net-snmp-5.2.3/perl
# perl Makefile.PL
# make
# make install
```

Partiu-se para a geração do esqueleto do código em linguagem de programação C, utilizando-se o utilitário *mib2c*.

Entretanto se a variável de ambiente *MIBS* não tiver valor e a *MIB* não foi copiada para o diretórios onde são alojadas as demais *MIB´s*, deve-se ter um retorno de erro similar à este:

```
[root@xmit mibs]# mib2c PUC-CONV-MIB.txt

You didn't give mib2c a valid OID to start with.  IE, I could not find
any information about the mib node "PUC-CONV-MIB.txt".  This could be
caused
because you supplied an incorrectly node, or by the MIB that you're
trying to generate code from isn't loaded.  To make sure your mib is
loaded, run mib2c using this as an example:
   env MIBS="+MY-PERSONAL-MIB" mib2c PUC-CONV-MIB.txt

You might wish to start by reading the MIB loading tutorial at:
```

```
    http://www.net-snmp.org/tutorial-5/commands/mib-options.html

  And making sure you can get snmptranslate to display information about
  your MIB node.  Once snmptranslate works, then come back and try mib2c
  again.
```

Contudo o comando *snmptranslate* possui uma opção "-m" que instrui o mesmo a carregar a *MIB* informada como argumento para esta opção. Exemplo:

```
  [root@lab1 mibs]# snmptranslate -m +PUCCAMP-CONV-MIB -IR systemIdleCPU
```

Mas o melhor método com certeza é copiar o arquivo da *MIB* do conversor para o diretório */usr/localshare/snmp/mibs* e configurar a variável "*MIBS=all*".

Após ter certeza de que os *OID´s* pudessem ser lidos pelo pacote de programas *NET-SNMP*, utilizou-se o utilitário *mib2c* para criar esqueleto de código em C a partir do *OID* puccamp. Para facilitar o processo, executou-se o utilitário *mib2c* primeiramente para criar código para as tabelas sob a árvore *MIB* embaixo do atributo *convCompWave*:

```
  [root@lab1 mibs]# mib2c -c mib2c.create-dataset.conf PUCCAMP-CONV-
MIB::convCompWave
  writing to convCompWave.h
  writing to convCompWave.c
  running indent on convCompWave.c
  running indent on convCompWave.h
```

Em seguida gerou-se o código para os demais atributos do conversor com esta linha de comando, entretanto em outro diretório, pois o utilitário *mib2c* sobregrava os arquivos *convCompWave.h* e *convCompWave.c*, porém com o código para os demais atributos:

```
  [root@lab1 mibs]#  mib2c -c mib2c.int_watch.conf PUCCAMP-CONV-MIB::
convCompWave
  writing to convCompWave.h
  writing to convCompWave.c
  running indent on convCompWave.c
  running indent on convCompWave.h
```

Os demais atributos dependem fundamentalmente do protótipo ficar completo em suas funcionalidades, mas os atributos pertinentes às tabelas, que é foco principal deste trabalho tiveram o código em C compilado para manipulação dos atributos e exemplificação de como a *MIB* irá auxiliar no gerenciamento de

uma rede totalmente óptica que se vale de um conversor de comprimento de ondas.

Contudo, para configurar o sub-agente por completo, os códigos gerados para as tabelas e para os demais atributos foram concatenados em um único código fonte denominado *aowcagent.c*, seu respectivo arquivo de cabeçalho *aowcagent.h*, cujo os mesmos podem ser examinado mais adiante neste apêndice.

Para terminar a instrumentação é necessário conhecer como executar os comandos que possam retornar as informações desejadas do equipamento ou carregar alguns atributos na inicialização do sub-agente.

Embora seja possível modificar o *SNMP* para gerenciar os atributos definidos para o conversor de comprimento de ondas, optou-se pela extensibilidade do *SNMP* criando um sub-agente a ser executado após a carga da deamon *snmpd*, que se conecta ao agente *master* e registra autoridade sobre determinados atributos conforme definição.

Optar pela extensibilidade do agente *SNMP* flexibiliza atividades de manutenção dos módulos programáticos. Para que o *SNMP* permita conexão de sub-agentes, deve-se incluir na linha do arquivo de configuração */etc/snmpd.conf* a seguinte linha:

```
master agentx
```

Para compilar o código fonte *aowcagent.c* como um sub-agente do *SNMP*, executou-se o seguinte comando no diretório onde for depositado os arquivos *aowcagent.c* e *aowcagent.h*, ressaltando que a variável *LD_LIBRARY_PATH* deve ser carregada antes da compilação,:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib

[root@lab 2]# net-snmp-config --compile-subagent aowcagent aowcagent.c
generating the tmporary code file: netsnmptmp.30341.c
checking for init_aowcagent in aowcagent.c
init_aowcagent(void)
running: gcc  -g -O2 -Dlinux  -I. -I/usr/local/include -o aowcagent
netsnmptmp.30341.c  aowcagent.c  -L/usr/local/lib -lnetsnmpmibs -
lnetsnmpagent -lnetsnmphelpers -lnetsnmp -ldl -lcrypto -lm
```

```
removing the tmporary code file: netsnmptmp.30341.c
subagent program aowcagent created
```

Após a compilação foi gerado um arquivo binário, que ao ser executado como um processo em *background*, conectou-se com o agente master *SNMP*, como se segue:

```
[root@localhost snmp]# ./aowcagent &
[1] 3250
[root@localhost snmp]# NET-SNMP version 5.2.3 AgentX subagent connected
```

A mensagem mostrada informa que o sub-agente conseguiu se conectar com o agente master *snmpd*, e que agora poderá tratar os atributos por ele registrado.

A partir deste instante é possível executar comando de manipulação dos atributos definidos para o conversor de comprimento de ondas, conforme exemplos:

```
  Consultando quanto de memória física existe no equipamento:

  [root@lab   2]#   snmpget   -v   2c   -c   priv_puccamp   192.168.0.178
systemPhysMemory.0
  PUCCAMP-CONV-MIB::systemPhysMemory.0 = INTEGER: 100

  Consultando informações relatives ao canal 21 na tabela channelTable:

  [root@lab   ctable]#    snmpget   -v   2c   -c   priv_puccamp   192.168.0.178
channelFreqIn.21 channelFreqOut.21  channelFreqPump.21  channelPotOut.21
  PUCCAMP-CONV-MIB::channelFreqIn.21 = STRING: "192,1"
  PUCCAMP-CONV-MIB::channelFreqOut.21 = STRING: "192,1"
  PUCCAMP-CONV-MIB::channelFreqPump.21 = STRING: "0"
  PUCCAMP-CONV-MIB::channelPotOut.21 = STRING: "0"

  Modificando informações relatives ao canal 21 na tabela channelTable:

  [root@lab   ctable]#   snmpset   -v   2c   -c   priv_puccamp   192.168.0.178
channelFreqIn.21 = 192,1  channelFreqOut.21 = 192,3 channelFreqPump.21 =
192,2 channelPotOut.21 = 1
  PUCCAMP-CONV-MIB::channelFreqIn.21 = STRING: "192,1"
  PUCCAMP-CONV-MIB::channelFreqOut.21 = STRING: "192,3"
  PUCCAMP-CONV-MIB::channelFreqPump.21 = STRING: "192,2"
  PUCCAMP-CONV-MIB::channelPotOut.21 = STRING: "1"
```

A.3 CÓDIGO FONTE EM LINGUAGEM C DO SUB-AGENTE DO CONVERSOR

O código fonte completo para tratar todos atributos especificados na *MIB* do conversor de comprimento de ondas recebe o nome de *aowcagent.c* e seu arquivo de cabeçalho recebe o nome de *aowcagent.h.*

Contudo para facilitar o desenvolvimento, o código fonte foi gerado em duas partes, sendo que a primeira parte foi construída para tratar os atributos que fazem parte de alguma tabela da *MIB* e a segunda parte trata dos demais atributos.

O código a seguir foi gerado para dar tratamento aos atributos que não fazem parte de tabelas:

```
/*
 * Note: this file originally auto-generated by mib2c using
 *        : mib2c.int_watch.conf,v 1.2.6.2 2005/12/20 12:23:08 tanders Exp $
 */

#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>
#include "convCompWave.h"

/*
 * The variables we want to tie the relevant OIDs to.
 * The agent will handle all GET and (if applicable) SET requests
 * to these variables automatically, changing the values as needed.
 */

u_long    systemUpTime = 0;  /* XXX: set default value */
u_long    systemFreeMemory = 0;  /* XXX: set default value */
long    systemPhysMemory = 0;  /* XXX: set default value */
U64    systemIdleCPU = 0;  /* XXX: set default value */
long    systemRestart = 0;  /* XXX: set default value */
long    convStatusFan1 = 0;  /* XXX: set default value */
long    convStatusPower = 0;  /* XXX: set default value */
long    convStatusCPU = 0;  /* XXX: set default value */
long    convStatusMem = 0;  /* XXX: set default value */
long    convAttrPhysMem = 0;  /* XXX: set default value */
long    convAttrClockCPU = 0;  /* XXX: set default value */
long    convAttrNrChannel = 0;  /* XXX: set default value */

/*
 * Our initialization routine, called automatically by the agent
 * (Note that the function name must match init_FILENAME())
 */
void
init_convCompWave(void)
{
  netsnmp_handler_registration *reg;
  netsnmp_watcher_info         *winfo;

    static oid systemUpTime_oid[] = { 1,3,6,1,4,1,23955,1,1,1 };
    static oid systemFreeMemory_oid[] = { 1,3,6,1,4,1,23955,1,1,2 };
    static oid systemPhysMemory_oid[] = { 1,3,6,1,4,1,23955,1,1,3 };
    static oid systemIdleCPU_oid[] = { 1,3,6,1,4,1,23955,1,1,4 };
    static oid systemRestart_oid[] = { 1,3,6,1,4,1,23955,1,1,5 };
    static oid convStatusFan1_oid[] = { 1,3,6,1,4,1,23955,1,4,1 };
    static oid convStatusPower_oid[] = { 1,3,6,1,4,1,23955,1,4,2 };
    static oid convStatusCPU_oid[] = { 1,3,6,1,4,1,23955,1,4,3 };
    static oid convStatusMem_oid[] = { 1,3,6,1,4,1,23955,1,4,4 };
    static oid convAttrPhysMem_oid[] = { 1,3,6,1,4,1,23955,1,5,1 };
    static oid convAttrClockCPU_oid[] = { 1,3,6,1,4,1,23955,1,5,2 };
    static oid convAttrNrChannel_oid[] = { 1,3,6,1,4,1,23955,1,5,3 };

  /*
```

```
   * a debugging statement.  Run the agent with -DconvCompWave to see
   * the output of this debugging statement.
   */
  DEBUGMSGTL(("convCompWave", "Initializing the convCompWave module\n"));


    /*
     * Register scalar watchers for each of the MIB objects.
     * The ASN type and RO/RW status are taken from the MIB definition,
     * but can be adjusted if needed.
     *
     * In most circumstances, the scalar watcher will handle all
     * of the necessary processing.  But the NULL parameter in the
     * netsnmp_create_handler_registration() call can be used to
     * supply a user-provided handler if necessary.
     *
     * This approach can also be used to handle Counter64, string-
     * and OID-based watched scalars (although variable-sized writeable
     * objects will need some more specialised initialisation).
     */
    DEBUGMSGTL(("convCompWave",
                "Initializing  systemUpTime  scalar  integer.    Default  value  =
%d\n",
                systemUpTime));
    reg = netsnmp_create_handler_registration(
              "systemUpTime", NULL,
               systemUpTime_oid, OID_LENGTH(systemUpTime_oid),
               HANDLER_CAN_RONLY);
    winfo = netsnmp_create_watcher_info(
                &systemUpTime, sizeof(u_long),
                 ASN_TIMETICKS, WATCHER_FIXED_SIZE);
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched systemUpTime" );
    }

    DEBUGMSGTL(("convCompWave",
                "Initializing systemFreeMemory scalar integer.  Default value =
%d\n",
                systemFreeMemory));
    reg = netsnmp_create_handler_registration(
             "systemFreeMemory", NULL,
              systemFreeMemory_oid, OID_LENGTH(systemFreeMemory_oid),
              HANDLER_CAN_RONLY);
    winfo = netsnmp_create_watcher_info(
                &systemFreeMemory, sizeof(u_long),
                 ASN_GAUGE, WATCHER_FIXED_SIZE);
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched systemFreeMemory" );
    }

    DEBUGMSGTL(("convCompWave",
                "Initializing systemPhysMemory scalar integer.  Default value =
%d\n",
                systemPhysMemory));
    reg = netsnmp_create_handler_registration(
             "systemPhysMemory", NULL,
              systemPhysMemory_oid, OID_LENGTH(systemPhysMemory_oid),
              HANDLER_CAN_RONLY);
    winfo = netsnmp_create_watcher_info(
                &systemPhysMemory, sizeof(long),
                 ASN_INTEGER, WATCHER_FIXED_SIZE);
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched systemPhysMemory" );
    }

    DEBUGMSGTL(("convCompWave",
                "Initializing  systemIdleCPU  scalar  integer.    Default  value  =
%d\n",
```

```
                systemIdleCPU));
    reg = netsnmp_create_handler_registration(
              "systemIdleCPU", NULL,
               systemIdleCPU_oid, OID_LENGTH(systemIdleCPU_oid),
               HANDLER_CAN_RONLY);
    winfo = netsnmp_create_watcher_info(
                &systemIdleCPU, sizeof(U64),
                 ASN_COUNTER64, WATCHER_FIXED_SIZE);
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched systemIdleCPU" );
    }

    DEBUGMSGTL(("convCompWave",
                "Initializing systemRestart scalar integer.   Default value =
%d\n",
                systemRestart));
    reg = netsnmp_create_handler_registration(
              "systemRestart", NULL,
               systemRestart_oid, OID_LENGTH(systemRestart_oid),
               HANDLER_CAN_RWRITE);
    winfo = netsnmp_create_watcher_info(
                &systemRestart, sizeof(long),
                 ASN_INTEGER, WATCHER_FIXED_SIZE);
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched systemRestart" );
    }

    DEBUGMSGTL(("convCompWave",
                "Initializing convStatusFan1 scalar integer.   Default value =
%d\n",
                convStatusFan1));
    reg = netsnmp_create_handler_registration(
              "convStatusFan1", NULL,
               convStatusFan1_oid, OID_LENGTH(convStatusFan1_oid),
               HANDLER_CAN_RONLY);
    winfo = netsnmp_create_watcher_info(
                &convStatusFan1, sizeof(long),
                 ASN_INTEGER, WATCHER_FIXED_SIZE);
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched convStatusFan1" );
    }

    DEBUGMSGTL(("convCompWave",
                "Initializing convStatusPower scalar integer.   Default value =
%d\n",
                convStatusPower));
    reg = netsnmp_create_handler_registration(
              "convStatusPower", NULL,
               convStatusPower_oid, OID_LENGTH(convStatusPower_oid),
               HANDLER_CAN_RONLY);
    winfo = netsnmp_create_watcher_info(
                &convStatusPower, sizeof(long),
                 ASN_INTEGER, WATCHER_FIXED_SIZE);
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched convStatusPower" );
    }

    DEBUGMSGTL(("convCompWave",
                "Initializing convStatusCPU scalar integer.   Default value =
%d\n",
                convStatusCPU));
    reg = netsnmp_create_handler_registration(
              "convStatusCPU", NULL,
               convStatusCPU_oid, OID_LENGTH(convStatusCPU_oid),
               HANDLER_CAN_RONLY);
    winfo = netsnmp_create_watcher_info(
                &convStatusCPU, sizeof(long),
                 ASN_INTEGER, WATCHER_FIXED_SIZE);
```

```
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched convStatusCPU" );
    }

    DEBUGMSGTL(("convCompWave",
                "Initializing convStatusMem scalar integer.   Default value =
%d\n",
                convStatusMem));
    reg = netsnmp_create_handler_registration(
              "convStatusMem", NULL,
               convStatusMem_oid, OID_LENGTH(convStatusMem_oid),
               HANDLER_CAN_RONLY);
    winfo = netsnmp_create_watcher_info(
                &convStatusMem, sizeof(long),
                 ASN_INTEGER, WATCHER_FIXED_SIZE);
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched convStatusMem" );
    }

    DEBUGMSGTL(("convCompWave",
                "Initializing convAttrPhysMem scalar integer.   Default value =
%d\n",
                convAttrPhysMem));
    reg = netsnmp_create_handler_registration(
              "convAttrPhysMem", NULL,
               convAttrPhysMem_oid, OID_LENGTH(convAttrPhysMem_oid),
               HANDLER_CAN_RONLY);
    winfo = netsnmp_create_watcher_info(
                &convAttrPhysMem, sizeof(long),
                 ASN_INTEGER, WATCHER_FIXED_SIZE);
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched convAttrPhysMem" );
    }

    DEBUGMSGTL(("convCompWave",
                "Initializing convAttrClockCPU scalar integer.  Default value =
%d\n",
                convAttrClockCPU));
    reg = netsnmp_create_handler_registration(
              "convAttrClockCPU", NULL,
               convAttrClockCPU_oid, OID_LENGTH(convAttrClockCPU_oid),
               HANDLER_CAN_RONLY);
    winfo = netsnmp_create_watcher_info(
                &convAttrClockCPU, sizeof(long),
                 ASN_INTEGER, WATCHER_FIXED_SIZE);
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched convAttrClockCPU" );
    }

    DEBUGMSGTL(("convCompWave",
                "Initializing convAttrNrChannel scalar integer.  Default value =
%d\n",
                convAttrNrChannel));
    reg = netsnmp_create_handler_registration(
              "convAttrNrChannel", NULL,
               convAttrNrChannel_oid, OID_LENGTH(convAttrNrChannel_oid),
               HANDLER_CAN_RONLY);
    winfo = netsnmp_create_watcher_info(
                &convAttrNrChannel, sizeof(long),
                 ASN_INTEGER, WATCHER_FIXED_SIZE);
    if (netsnmp_register_watched_scalar( reg, winfo ) < 0 ) {
        snmp_log( LOG_ERR, "Failed to register watched convAttrNrChannel" );
    }


  DEBUGMSGTL(("convCompWave",
              "Done initalizing convCompWave module\n"));
}
```

As linhas de código a seguir foram geradas para dar tratamento aos atributos de tabelas da *MIB* do conversor de comprimento de ondas.

```
          Este código fonte permite o tratamento dos atributos da tabela
channelTable.c

          /*
           * Note: this file originally auto-generated by mib2c using
           *        : mib2c.create-dataset.conf,v 5.4 2004/02/02 19:06:53 rstory
Exp $
           */

          #include <net-snmp/net-snmp-config.h>
          #include <net-snmp/net-snmp-includes.h>
          #include <net-snmp/agent/net-snmp-agent-includes.h>
          #include "channelTable.h"

          /** Initialize the channelTable table by defining its contents and how
it's structured */
          void
          initialize_table_channelTable(void)
          {
              static oid channelTable_oid[] = {1,3,6,1,4,1,23955,1,3};
              size_t channelTable_oid_len = OID_LENGTH(channelTable_oid);
              netsnmp_table_data_set *table_set;

              /* create the table structure itself */
              table_set = netsnmp_create_table_data_set("channelTable");

              /* comment this out or delete if you don't support creation of new
rows */
              table_set->allow_creation = 1;

              /**************************************************
               * Adding indexes
               */
              DEBUGMSGTL(("initialize_table_channelTable",
                          "adding indexes to table channelTable\n"));
              netsnmp_table_set_add_indexes(table_set,
                                    ASN_INTEGER,  /* index: channelID */
                                    0);

              DEBUGMSGTL(("initialize_table_channelTable",
                          "adding column types to table channelTable\n"));
              netsnmp_table_set_multi_add_default_row(table_set,
                                                      COLUMN_CHANNELID,
ASN_INTEGER, 0,
                                                      NULL, 0,
                                                      COLUMN_CHANNELFREQOUT,
ASN_OCTET_STR, 1,
                                                      NULL, 0,
                                                      COLUMN_CHANNELFREQBOMB,
ASN_OCTET_STR, 1,
                                                      NULL, 0,
                                                      COLUMN_CHANNELPOTOUT,
ASN_OCTET_STR, 1,
                                                      NULL, 0,
                                                      COLUMN_CHANNELFREQIN,
ASN_OCTET_STR, 1,
                                                      NULL, 0,
                                                      COLUMN_CHANNELSTATUS,
ASN_INTEGER, 1,
                                                      NULL, 0,
                                        0);

              /* registering the table with the master agent */
```

```
               /* note: if you don't need a subhandler to deal with any aspects
                  of the request, change channelTable_handler to "NULL" */

netsnmp_register_table_data_set(netsnmp_create_handler_registration("channelTable
", channelTable_handler,

channelTable_oid,

channelTable_oid_len,

HANDLER_CAN_RWRITE),
                                        table_set, NULL);
           }

           /** Initializes the channelTable module */
           void
           init_channelTable(void)
           {

             /* here we initialize all the tables we're planning on supporting */
               initialize_table_channelTable();
           }

           /** handles requests for the channelTable table, if anything else
needs to be done */
           int
           channelTable_handler(
               netsnmp_mib_handler               *handler,
               netsnmp_handler_registration      *reginfo,
               netsnmp_agent_request_info        *reqinfo,
               netsnmp_request_info              *requests) {
               /* perform anything here that you need to do.  The requests have
                  already been processed by the master table_dataset handler, but
                  this gives you chance to act on the request in some other way
                  if need be. */
               return SNMP_ERR_NOERROR;
           }
```

Estas linhas de código foram especialmente escritas para tratamento da tabela *freeChannelTable*, onde a tabela é inicializada com alguns canais pré-definidos com todas informações sobre freqüências tendo a atriução de um valor de “0”, que significa que para este canal não existe freqüência de entrada ou de saída associada.

```
  /*
   * Note: this file originally auto-generated by mib2c using
   *        : mib2c.create-dataset.conf,v 5.4 2004/02/02 19:06:53 rstory Exp $
   */

  #include <sys/types.h>
  #include <net-snmp/net-snmp-config.h>
  #include <net-snmp/net-snmp-includes.h>
  #include <net-snmp/agent/net-snmp-agent-includes.h>
  #include <net-snmp/agent/table.h>
  #include <net-snmp/agent/table_data.h>
  #include <net-snmp/agent/table_dataset.h>
  #include "puccamp.h"
  #include <stdlib.h> /* adicionado */
```

```
  #include <strings.h> /* adicionado */

  /** Initialize the freeChannelTable table by defining its contents and how it's
structured */
  void
  initialize_table_freeChannelTable(void)
  {
      int i , v , x ;
      static oid      freeChannelTable_oid[] =
          { 1, 3, 6, 1, 4, 1, 23955, 1, 2 };
      size_t          freeChannelTable_oid_len =
          OID_LENGTH(freeChannelTable_oid);
      netsnmp_table_data_set *table_set;
      netsnmp_table_row *row; /* adicionado */

      /*
       * create the table structure itself
       */
      table_set = netsnmp_create_table_data_set("freeChannelTable");

      /*
       * comment this out or delete if you don't support creation of new rows
       */
      table_set->allow_creation = 1;

      /**************************************************
       * Adding indexes
       */
      DEBUGMSGTL(("initialize_table_freeChannelTable",
                  "adding indexes to table freeChannelTable\n"));
      netsnmp_table_set_add_indexes(table_set,  ASN_INTEGER,            /*  index:
freeChannelID1 */
                                    0);

      DEBUGMSGTL(("initialize_table_freeChannelTable",
                  "adding column types to table freeChannelTable\n"));
      netsnmp_table_set_multi_add_default_row(table_set,
                                              COLUMN_FREECHANNELID1,
                                              ASN_INTEGER, 0, NULL, 0,
                                              COLUMN_FREECHANNELSTATUS,
                                              ASN_INTEGER, 1, NULL, 0,
                                              COLUMN_FREECHANNELNR,
                                              ASN_INTEGER, 0, NULL, 0,
                                              COLUMN_FREECHANNELFREE,
                                              ASN_INTEGER, 1, NULL, 0, 0);

      /*
       * registering the table with the master agent
       */
      /*
       * note: if you don't need a subhandler to deal with any aspects
       * of the request, change freeChannelTable_handler to "NULL"
       */
      netsnmp_register_table_data_set(netsnmp_create_handler_registration
                                      ("freeChannelTable",
                                       freeChannelTable_handler,
                                       freeChannelTable_oid,
                                       freeChannelTable_oid_len,
                                       HANDLER_CAN_RWRITE), table_set, NULL);

      /*
       * criando a linha para a tabela e adicionando dados
       */
      row = netsnmp_create_table_data_row();
      /*
       * set the index to the IETF WG name "snmpv3"
       */
      v = 0 ;
```

```
      i = 21 ;
      netsnmp_table_row_add_index(row, ASN_INTEGER, &i, sizeof(i));


      /*
       * Adicionando para freeChannelStatus o valor de 0
       *netsnmp_table_row_add_index(row, ASN_INTEGER, index, sizeof(int));
       */
      netsnmp_set_row_column(row,  COLUMN_FREECHANNELSTATUS,  ASN_INTEGER,  (char*)
&v, sizeof(v));
      netsnmp_mark_row_column_writable(row,     COLUMN_FREECHANNELSTATUS,     1);
/* make writable via SETs */

      /*
       * Configurando freeChannelNr na 3 coluna da tabela
       */
      netsnmp_set_row_column(row,  COLUMN_FREECHANNELNR,  ASN_INTEGER,  (char*)  &v,
sizeof(v));
      netsnmp_mark_row_column_writable(row, COLUMN_FREECHANNELNR, 1);


      /*
       * Configurando freeChannelfree na 4 coluna da tabela como canal livre
       */
      netsnmp_set_row_column(row,  COLUMN_FREECHANNELFREE,  ASN_INTEGER,  (char*)
&v, sizeof(v));
      netsnmp_mark_row_column_writable(row, COLUMN_FREECHANNELFREE, 1);        /*
make writable via SETs */

      /*
       * adicionando a linha para a tabela
       */
      netsnmp_table_dataset_add_row(table_set, row);

      /*
       * add the data, for the second row
       */
      x = 22 ;
      for ( x=22 ; x<=HIGH_CHANNEL ; x++ ) {
      i = x ;
      v = 0 ;
      row = netsnmp_create_table_data_row();
      netsnmp_table_row_add_index(row, ASN_INTEGER, &i, sizeof(i));

      netsnmp_set_row_column(row,  COLUMN_FREECHANNELSTATUS,  ASN_INTEGER,  (char*)
&v, sizeof(v));
      netsnmp_mark_row_column_writable(row, COLUMN_FREECHANNELSTATUS, 1);

      netsnmp_set_row_column(row,  COLUMN_FREECHANNELNR,  ASN_INTEGER,  (char*)  &v,
sizeof(v));
      netsnmp_mark_row_column_writable(row, COLUMN_FREECHANNELNR, 1);

      netsnmp_set_row_column(row,  COLUMN_FREECHANNELFREE,  ASN_INTEGER,  (char*)
&v, sizeof(v));
      netsnmp_mark_row_column_writable(row, COLUMN_FREECHANNELFREE, 1);        /*
make writable via SETs */
      netsnmp_table_dataset_add_row(table_set, row);
      }

      /*
       * Finally, this actually allows the "add_row" token it the
       * * snmpd.conf file to add rows to this table.
       * * Example snmpd.conf line:
       * *   add_row freeChannelTable eos "Glenn Waters" "Dale Francisco"
       */
      netsnmp_register_auto_data_table(table_set, NULL);
      DEBUGMSGTL(("example_puccamp", "Done initializing.\n"));
```

```
  }


  /** Initializes the puccamp module */
  void
  init_puccamp(void)
  {

      /*
       * here we initialize all the tables we're planning on supporting
       */
      initialize_table_freeChannelTable();
  }

  /** handles requests for the freeChannelTable table, if anything else needs to
be done */
  int
  freeChannelTable_handler(netsnmp_mib_handler *handler,
                           netsnmp_handler_registration *reginfo,
                           netsnmp_agent_request_info *reqinfo,
                           netsnmp_request_info *requests)
  {
      /*
       * perform anything here that you need to do.  The requests have
       * already been processed by the master table_dataset handler, but
       * this gives you chance to act on the request in some other way
       * if need be.
       */
      return SNMP_ERR_NOERROR;
  }
```

A.4 Criação de Gráficos com o MRTG

Uma vez que a *MIB* do conversor de comprimento de ondas foi estruturada e o sub-agente responde aos comandos *SNMP*, utilizou-se outro software de código aberto para criar gráficos estatísticos sobre os atributos que forem necessários.

Tal produto denominado *MRTG,* foi concebido inicialmente para criação de gráficos estatísticos baseados na contagem dos pacotes trafegados em interfaces de redes gerenciadas pelo protocolo *SNMP*, como pode ser visto na Figura 29 que mostra um exemplo customizado para o conversor de comprimento de ondas.

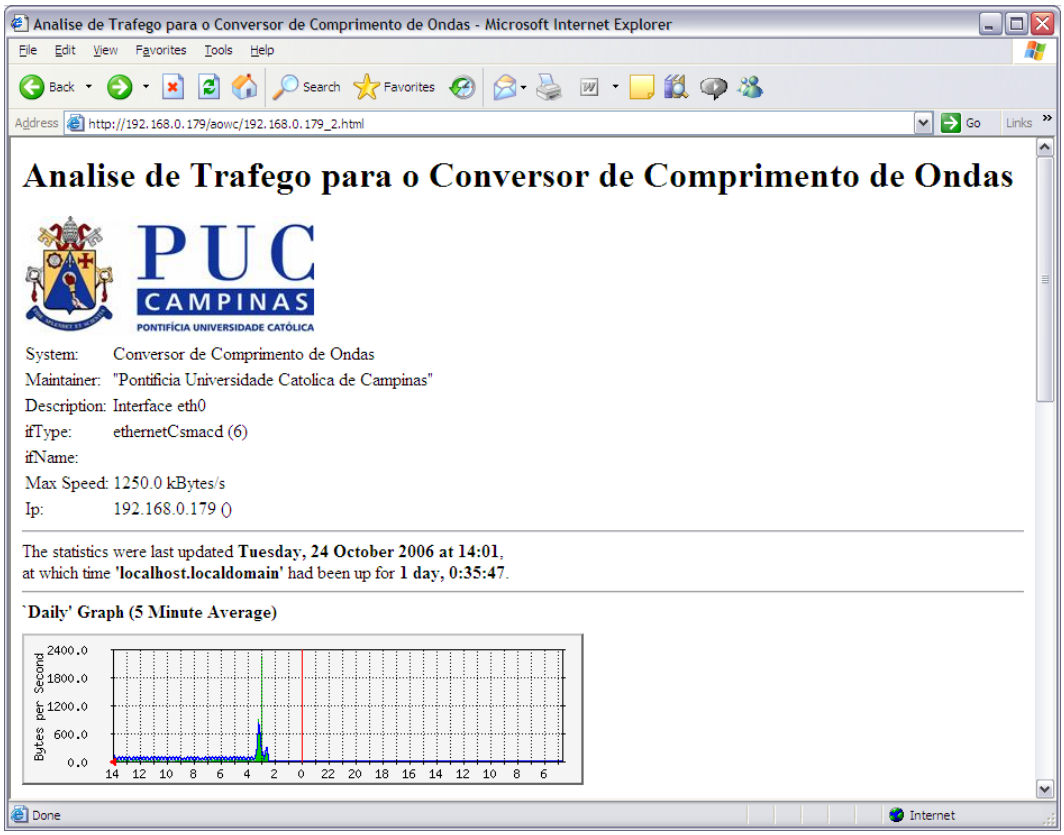

Figura 29. Estatísticas das interfaces ethernet do conversor.

Contudo, o produto possui uma abertura para que se carregue uma *MIB* específica e utilize os *OID´s* específicos desta *MIB* para a geração de gráficos.

Focado no conversor de comprimento de ondas, foi criado um arquivo de configuração para ser utilizado pelo *MRTG* para construir um gráfico de utilização de *CPU* do Conversor de Comprimento de Ondas, conforme apresentado na Figura 30.

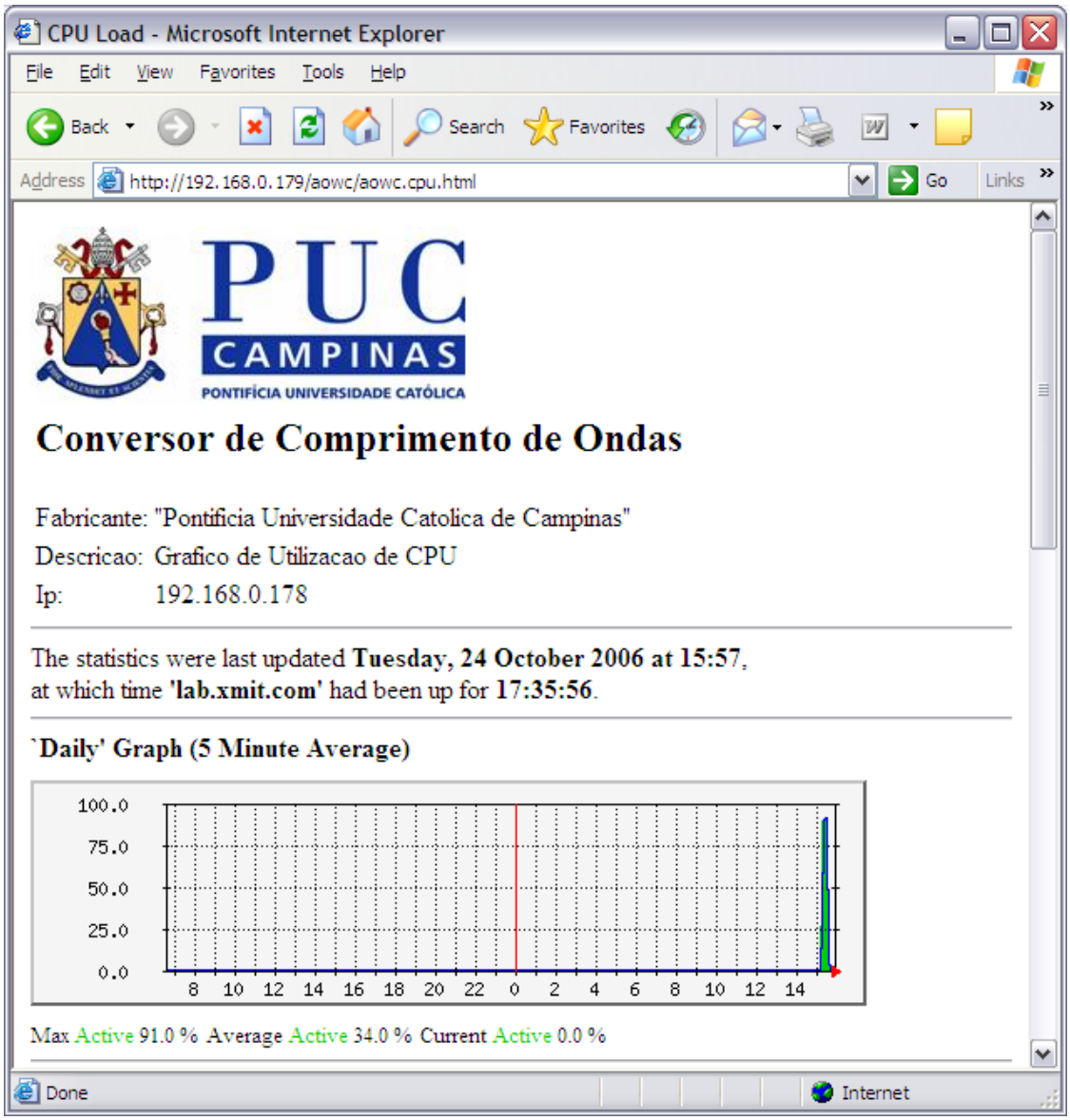

Figura 30. Gráfico de Utilização de *CPU* do Conversor.

O código para gerar a página que apresenta a utilização de *CPU* ficou da seguinte forma:

```
            Arquivo aowc_cpu.cfg
            ### Global Config Options

            #  for UNIX
            WorkDir: /var/www/html/aowc

            ### Global Defaults

            #  to get bits instead of bytes and graphs growing to the right
            # Options[_]: growright, bits

            EnableIPv6: no

            ###################################################################
            # System: localhost.localdomain
            # Description: Linux localhost.localdomain 2.6.9-42.0.3.ELsmp #1 SMP Fri Oct 6
06:21:39 CDT 2006 i686
            # Contact: "Carlos Schimidt"
```

```
            # Location: PUC-CAMP
            ##########################################################################

            Target[cpuidle]: ssCpuRawIdle.0&ssCpuRawIdle.0:pucpublic@192.168.0.178
            ##SetEnv[cpuidle]: MRTG_INT_IP="192.168.0.178" MRTG_INT_DESCR="cpu"
            MaxBytes[cpuidle]: 100
            Title[cpuidle]: Analise de CPU para o Conversor de Comprimento de Ondas
            PageTop[cpuidle]:  <H1>Analise  de  CPU  para  o  Conversor  de  Comprimento  de
Ondas</H1>
             <TABLE>
               <img src="./puc-camp5.jpg">
               <TR><TD>System:</TD>     <TD>Conversor de Comprimento de Ondas</TD></TR>
               <TR><TD>Maintainer:</TD>    <TD>"Pontificia    Universidade    Catolica    de
Campinas"</TD></TR>
               <TR><TD>Description:</TD><TD>Interface eth0  </TD></TR>
               <TR><TD>Ip:</TD>         <TD>192.168.0.178 </TD></TR>
             </TABLE>
            ShortLegend[cpuidle]: %
            YLegend[cpuidle]: CPU Utilization
            Legend1[cpuidle]: IDLE CPU in % (Load)
            Legend2[cpuidle]:
            Legend3[cpuidle]:
            Legend4[cpuidle]:
            LegendI[cpuidle]: idle:
            LegendO[cpuidle]:
            Options[cpuidle]:
```

Para explorar mais ainda a ferramenta *MRTG* como o que podemos extrair de informações do conversor de comprimento de ondas, a Figura 31 um exemplo criado para acompanhar a freqüência de saída de um determinado canal do conversor.

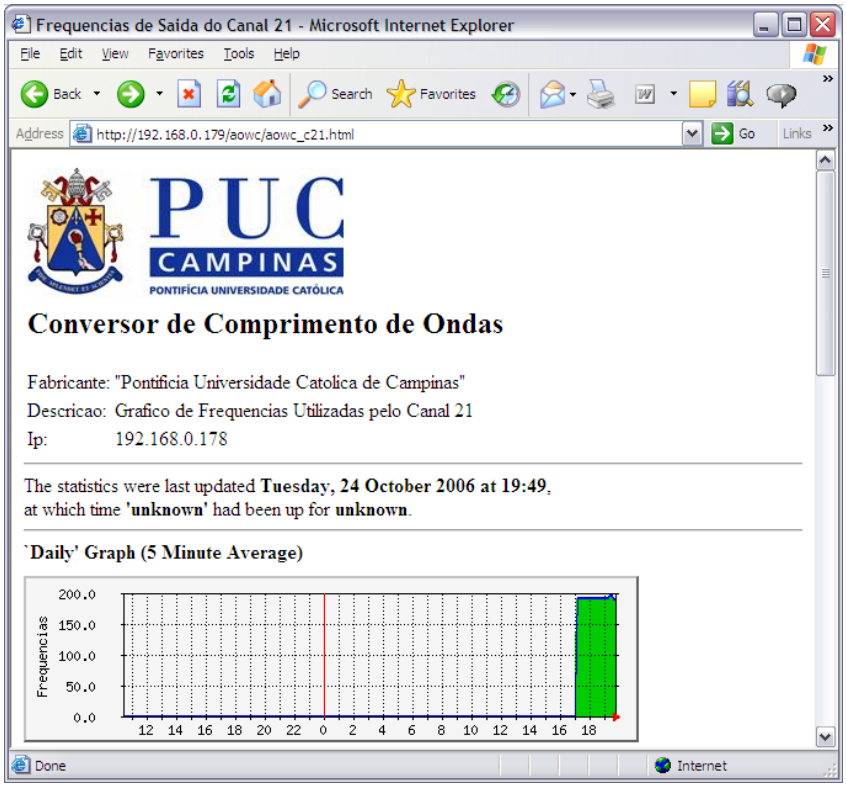

Figura 31. Gráfico de Frequêncuias associadas com o canal 21.

O arquivo de configuração para gerar o gráfico de atributos específicos do conversor, requer uma diretiva denominada *LoadMIBs* que informa o caminho absoluto do arquivo de *MIB* do conversor de comprimento de ondas:

```
            Arquivo aowc_c21.cfg

            WorkDir: /var/www/html/aowc
            LoadMIBs: /usr/local/share/snmp/mibs/PUCCAMP-CONV-MIB.txt
            Target[aowc_c21]: `/var/www/html/aowc/snmpget_c21.sh`
            MaxBytes[aowc_c21]: 200
            Title[aowc_c21]: Frequencias de Saida do Canal 21
            PageTop[aowc_c21]:
               <img src="./puc-camp5.jpg">
             <TABLE>
               <TR><TD><H2>Conversor de Comprimento de Ondas</H2></TD></TR>
             </TABLE>
             <TABLE>
               <TR><TD>Fabricante:</TD>  <TD>"Pontificia  Universidade  Catolica  de
Campinas"</TD></TR>
               <TR><TD>Descricao:</TD><TD>Grafico  de  Frequencias  Utilizadas  pelo
Canal 21 </TD></TR>
               <TR><TD>Ip:</TD>         <TD>192.168.0.178</TD></TR>
             </TABLE>
            Options[aowc_c21]: growright,unknaszero,nopercent,gauge
            LegendI[aowc_c21]: Frequencias
            LegendO[aowc_c21]:
            YLegend[aowc_c21]: Frequencias
```

A.5 Telas de Aplicativos Comerciais

Apenas com o intuito de demonstrar a interface das ferramentas comerciais analisadas para avaliar os recursos que facilitam o desenvolvimento de uma MIB e também do agente estendido, segue as telas principais dos produtos informados neste documento, considerando que os mesmos tratavam-se de versões em caráter demonstrativo que não permitiam o desenvolvimento total de uma MIB.

O software *Visual MIB Builder* pode ser visualizado na Figura 32 permite a elaboração, construção e compilação de uma MIB sendo desenvolvida, não tendo maiores complicações para sua utilização.

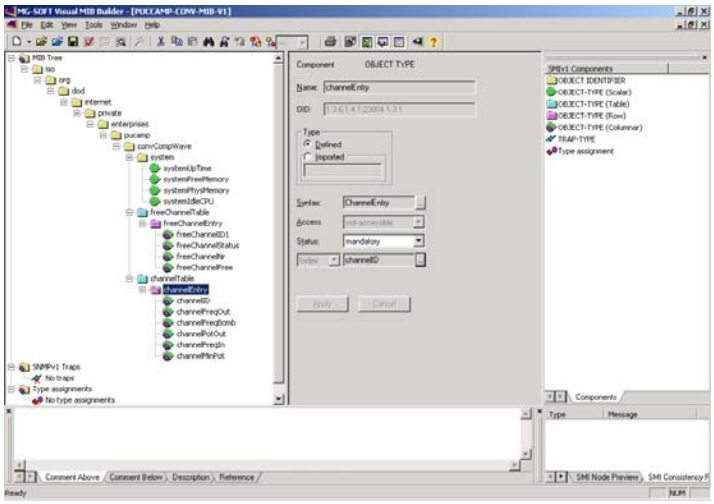

Figura 32. Construção de *MIB* pelo software *Visual MIB Builder*.

O software *NU-Design Visual MIBBuilder*, que é mostrado na Figura 33, foi o software que encontrei mais facilidade e interatividade na construção da MIB, pois possui campos que restringem a especificação incorreta de alguns atributos.

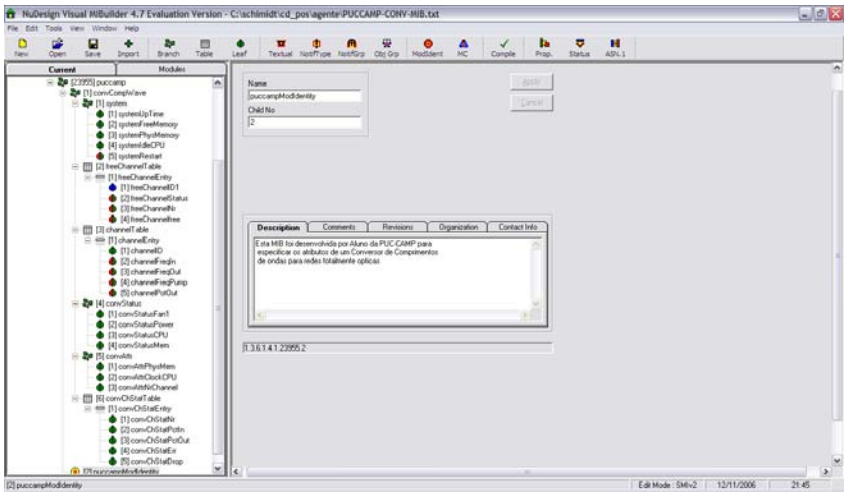

Figura 33. Construção de *MIB* pelo software *NUDesign*.

O software *Unbrowse SNMP*, que é mostrado na Figura 34 , foi o software que encontrei mais dificuldade para a construção da MIB, pois sua interface não é intuitiva o suficiente para utilização do mesmo sem um treinamento.

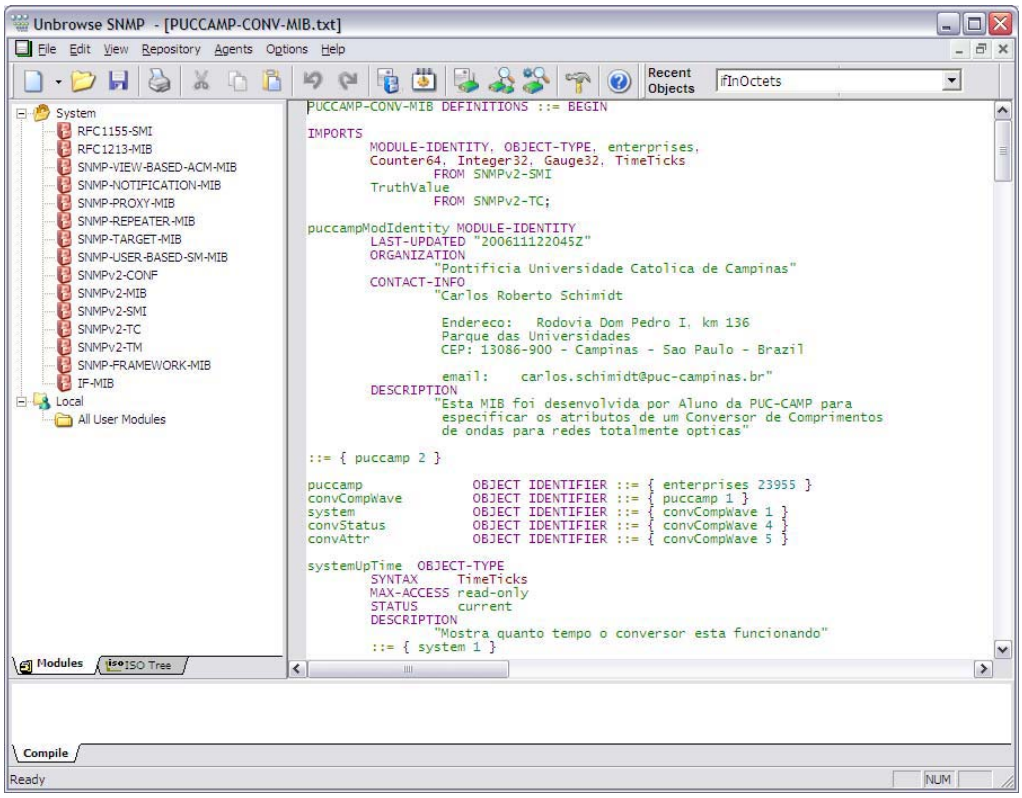

Figura 34. Construção da *MIB* pelo software *Unbrowse SNMP.*

APÊNDICE B - PROTÓTIPO DO CONVERSOR DA PUC-CAMP

Este apêndice vai apresentar os componentes utilizados na montagem experimental do conversor de comprimento de ondas do Laboratório de Meios de Transmissão da Pontifícia Universidade Católica de Campinas descrevendo suas respectivas funcionalidades.

O protótipo está descrito a seguir e ilustrado na Figura 35. De uma fonte laser saem duas fibras. A primeira emula o sinal de dados, modulado diretamente pelo gerador de bits indicado à esquerda da figura. A segunda fibra transporta o sinal de bombeio e está conectada a um filtro cujo objetivo é reduzir o ruído deste sinal. Ambas as fibras conectam-se à um acoplador 2x1. A saída deste aclopador está ligada a uma fibra altamente não-linear que promove a conversão de comprimentos de onda propriamente dita. Um filtro na saída desta fibra seleciona o canal convertido e o sinal é finalmente analisado em um analisador de espectros ópticos que tem como função medir se a potência do sinal está dentro do limiar aceitável. O software de controle instalado em um microcomputador com sistema operacional *Linux* será responsável pelas funções de automação em uma rede dinâmica..

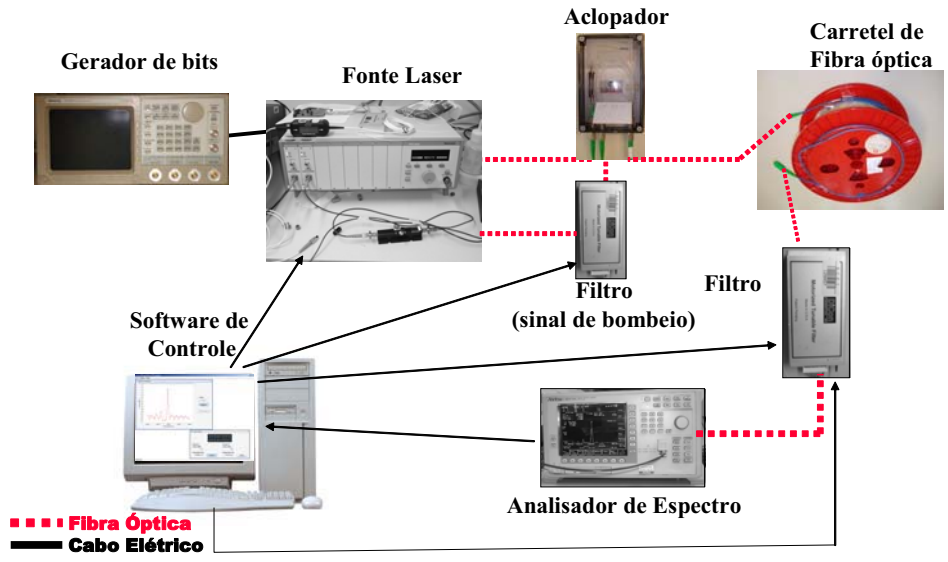

Figura 35. Esquema do protótipo do conversor de comprimento de ondas da PUC-Campinas.