Universidade Federal de Santa Catarina
Curso de Pós-Graduação em Ciência da Computação

Uma Aplicação Java-SNMP Para Monitoração de Redes Sem Fio

Marcos André Pisching

Florianópolis – SC
2001

Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Ciência da Computação

Marcos André Pisching

Uma Aplicação Java-SNMP Para Monitoração de Redes Sem Fio

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

Prof. Dr. Carlos Becker Westphall
Orientador

Florianópolis, junho de 2001.

Uma Aplicação Java-SNMP para Monitoração de Redes Sem Fio

Marcos André Pisching

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

Prof. Dr. Carlos Becker Westphall
Orientador

Prof. Dr. Fernando A. Ostuni Gauthier
Coordenador do Programa de Pós-Graduação
em Ciência da Computação

Banca Examinadora

Prof. Dr. Carlos Becker Westphall
Presidente

Prof. Dr. Luiz Carlos Zancanella
UFSC

Carla Merkle Westphall
Prof. Dra. Carla Merkle Westphall
UFSC

Prof. Dr. Elias Procopio Duarte Junior
UFPR

"Quem é corajoso
não foge da batalha da vida.
Todos temos nossas lutas,
mas só quem sabe suportá-las
pode ser classificado de herói,
de Homem em toda a extensão do termo.
Saiba merecer o título de Homem,
saiba ser herói,
não desanime diante das dificuldades.
Enfrente a vida, tal qual se apresenta,
com suas alegrias e dores, e
jamais pense em fugir covardemente."
(C. Torres Pastorino)

"TOUS LES JOURS ON VA DE
MIEUX EN MIEUX
A TOUS LES POINTS DE VUE"
(Segundo Prof. Westphall)

DEDICATÓRIA

Aos meus irmãos e amigos,

aos meus amados pais Erfriedo Bruno Pisching e Sirlei Pisching,

que tanto me incentivaram e me apoiaram para alcançar este mérito.

Dedico ainda a Deus, pois sem Ele nada poderia ser realizado.

AGRADECIMENTOS

É com grande estima que agradeço a todas as pessoas que auxiliaram direta ou indiretamente no desenvolvimento deste trabalho.

Agradeço aos meus amados pais Erfriedo Bruno Pisching e Sirlei Pisching pela vida, pelo constante apoio e incentivo, e pela educação que me deram. Aos meus queridos irmãos, cunhados, cunhadas, sobrinhos e familiares, que tanto incentivaram e apoiaram-me por todo o caminho que percorri até aqui.

Agradeço também ao Prof. Carlos Becker Westphall, que me orientou neste trabalho, e sempre esteve disposto a passar materiais atualizados, que foram de fundamental importância para o desenvolvimento deste.

Agradeço a todos os colegas e amigos, que souberam estar juntos nas horas de festas, de trabalho, de estudo e de apoio.

Agradeço aos meus colegas de serviço da UNIPLAC e SENAC-Lages, que me auxiliaram na passagem tanto de conhecimento como de pensamentos positivos, para o desenvolvimento deste trabalho.

À UFSC também vai o meu agradecimento pelo suporte de material, infra-estrutura e professores.

Ao grupo ISCC, em especial ao Paulo Roberto Sá, Ney Kassiano Ramos e Rita (Coordenadora), que permitiram o acesso aos equipamentos, colaborando na realização dos devidos testes e também para um conhecimento mais amplo do tema a ser desenvolvido.

Agradeço em especial aos amigos Silvio Costa Sampaio e Rodrigo Martins Pagliares, que me auxiliaram em muito na passagem de conhecimento, idéias e material para o desenvolvimento deste trabalho.

As amigas Vera e Valdete (CPGCC), pelo apoio e ótimo atendimento aos mestrandos.

Em especial a minha namorada Patrícia pelo apoio e compreensão durante o desenvolvimento deste trabalho.

Ao corpo discente representante da Câmara de Ensino junto a Pró-Reitoria de Ensino Gilmar Rodrigues, Alexandre Luis Giehl e Giuliano Saneh, pelo apoio para defesa desta dissertação.

Também agradeço pelas preciosas contribuições dos professores membro da banca de defesa desta dissertação.

Por fim, agradeço ao Criador Maior nosso Deus, sem Ele nada seria possível.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

AM	*Amplitude Modulation*
AP	*Access Point*
ARPA	*Advanced Research Projects Agency*
ATM	*Asynchronous Transfer Mode*
BER	*Basic Encoding Rules*
CGI	*Common Gateway Interface*
CMIP	*Common Management Information Protocol*
CSMA/CD	*Carrier Sense Multiple Access with Collision Avoidance*
DHCP	*Dynamic Host Control Protocol*
DSSS	*Direct Sequence Spread Spectrum*
FCS	*Frame Check Sequence*
FDDI	*Fiber Distributed Data Interface*
FHSS	*Frequency Hopping Spread Spectrum*
FM	*Frequency Modulation*
HTML	*Hipertext Mark-up Language*
HTTP	*Hipertext Transfer Protocol*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
ISDN	*Integrated Service Digital Network*
ISM	*Industrial, Scientific and Medical*
ISO	*International Organization for Standardization*
LAN	*Local Area Network*
MAC	*Medium Access Control*
MAN	*Metropolitan Area Network*
MIB	*Management Information Base*
MIT	*Management Information Tree*
NAT	*Network Address Translation*
NMS	*Network Management Systems*
ODBC	*Open Database Connectivity*
OID	*Object Identification*
OSI	*Open System Interconnection*
PA	*Ponto de Acesso*
PDU	*Protocol Data Unit*
POO	*Programação Orientada a Objetos*
RAS	*Remote Access Server*
RDSI	*Redes Digital de Serviços Integrados*
RF	*Rádio Freqüência*
RFC	*Request For Comments*
SMI	*Structure Management Information*
SNMP	*Simple Network Management Protocol*
SNR	*Signal-to-Noise Rate*
SS	*Spread Spectrum*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
UDP	*User Datagram Protocol*
UHF	*Ultra High Frequency*

VHF	*Very High Frequency*
WAN	*Wide Area Network*
WEP	*Wired Equivalent Privacy*
WLAN	*Wireless Local Area Network*
WWW	*World Wide Web*

RESUMO

As redes de computadores sem fio vêm sendo utilizadas expressivamente na implementação de LANs, MANs e WANs, de forma a coexistir entre as redes tradicionais que são interligadas através de cabos e, também, para permitir a interligação de computadores que estejam em locais de difícil acesso para a instalação de cabos. Essa tecnologia aparece também como solução para provimento de serviços da Internet, fazendo com que o usuário tenha uma alternativa a outros métodos para acessar a rede. Considerando o grande interesse pelo uso desta tecnologia, surge a necessidade do monitoramento e controle, desde a transmissão de pacotes até a identificação de ruídos da mesma. O objetivo deste trabalho é apresentar um aplicativo para monitorar e controlar o *Wireless Bridge(WavePOINT-II)*, aplicado ao acesso sem fio em provedores de serviços Internet. Este é o principal elemento para a comunicação sem fio, permitindo a ligação entre a base de transmissão e as estações remotas. O sistema permite o acesso remoto, através da Internet, possibilitando a verificação da qualidade de sinal e da quantidade de dados que chegam com sucesso ou falha. É possível verificar o estado dos enlaces em qualquer ambiente computacional e independente do local que se deseja verificar. Para tanto, a implementação do aplicativo é feita com a linguagem de programação Java envolvendo o protocolo SNMP e o serviço WWW da Internet.

ABSTRACT

The wireless computer networks have been widely used to implement LANs, MANs and WANs, with the purpose to coexist with the traditional cable networks, and also to make possible to link computers in places where the access is difficult to install the cables. That technology also appears as a solution for the Internet Service Providers (ISP), resulting that the user can choose other methods to access the network. Considering the great interest for the use of this technology appears the necessity of monitoring and controlling it, from the transmission of packages to the identification of its noise. The objective of this work is to present an application to monitor and control the Wireless Bridge (WavePOINT-II), applied to wireless access for ISPs. This is the main component for the wireless communication, allowing the connection between the transmission base and the remote stations. The system allows the remote access, through the Internet, making possible to verify the sign quality and the amount of data that arrive successfully or wrongly. With it is possible to verify the state of the connection in any computational environment and it is independent of the location were one needs to verify. For that, this work also presents the implementation of an application using the Java programming language and SNMP (Simple Network Management Protocol) and the WWW service.

1. INTRODUÇÃO

Inicialmente, as redes de computadores foram projetadas como um mecanismo para permitir o compartilhamento de recursos, tais como, impressoras, modems, discos rígidos, arquivos e outros. Os elementos de redes, geralmente homogêneos e fechados, existiam apenas em ambientes restritos (acadêmicos, governamentais e empresas de grande porte) e eram desenvolvidos apenas para atender tipos de serviços específicos [SOA97].

Com sua evoluação, as redes de computadores passaram a ser constituídos de equipamentos heterogêneos e abertos, permitindo a interconexão de diferentes tecnologias de redes de maneira transparente para o usuário, passando a proporcionar altas taxas de transmissão, baixas taxas de erro, interligação de computadores independente de localidade e, oferecendo ainda, maior confiabilidade na transmissão de dados. O serviço de compartilhamento de recursos foi o objetivo inicial das redes, abrindo espaço para os mais variados tipos de serviços, como o ensino, o comércio, a pesquisa e o entretenimento, os quais, podem ser oferecidos para qualquer tipo de usuário, através de uma rede chamada Internet.

A Internet se destacou devido a sua arquitetura baseada no TCP/IP (*Transmission Control Protocol/Internet Protocol*), que permite a interconexão de diferentes topologias e tecnologias de redes e, para mais tarde oferecer aplicativos gráficos, interativos e de fácil manuseio para a troca de informação em qualquer parte do mundo, aumentando mais ainda o seu número de usuários. Dentre os aplicativos o que mais se destaca é o *browser* Web da plataforma WWW (*World Wide Web*).

A popularização da Internet e os benefícios oferecidos pelos seus serviços, fez com que o número de usuários e de provedores de acesso a Internet crescesse de forma abrupta, privilegiando tanto o usuário doméstico como grandes corporações, tornando-se um dos meios de comunicação mais explorados na atualidade.

Com a expansão das redes de computadores e o crescente número de usuários, novas tecnologias para desenvolvimento de aplicativos baseados no modelo cliente/servidor foram criadas. A tecnologia que se destaca nesta área é a linguagem de

programação Java, que permite o desenvolvimento de aplicativos para qualquer plataforma de computador, tornando possível a execução de um aplicativo independente de sistema operacional e arquitetura de máquina [DEI98][SUN99].

O *hardware* também sempre teve o seu avanço tecnológico, de modo a melhorar o desempenho das redes de computadores, bem como, facilitar a instalação das mesmas. Entre os mais variados tipos de *hardware*, o que se destaca neste trabalho é o meio físico de transmissão baseado em Freqüência de Rádio (RF), o qual permite a instalação de uma rede de computador sem fazer uso de cabos óticos ou metálicos para a transmissão de dados. Essas redes são também chamadas de redes sem fio (*Wireless Networks*). Algumas empresas provedoras de serviços Internet estão adotando esta tecnologia para fornecer seus serviços ao usuário final, substituindo o sistema privado de telecomunicação, que geralmente opera com baixas taxas de transmissão de dados e altas taxas de ruídos, para formar então uma rede metropolitana de alta velocidade [WLA99]. A transmissão sem fio se dá a partir do elemento de rede *Wireless Bridge (WavePOINT-II)*, sendo este o foco principal de pesquisa para a realização deste trabalho. Este o principal elemento para a comunicação sem fio, sendo ele, responsável pela ligação entre a base de transmissão e as estações remotas. O monitoramento da rede acontece através deste elemento.

Considerando os parâmetros confiabilidade, rapidez e o crescente número de usuários que acessam a rede, independente da tecnologia utilizada para a transmissão de dados, faz-se necessária a implementação de protocolos de gerenciamento da rede. Neste caso, o protocolo que se destaca é o SNMP (*Simple Network Management Protocol*), o qual está implementado na arquitetura da Internet e viabiliza, através de suas primitivas de operações, o desenvolvimento de aplicativos gráficos para gerenciar e a rede.

Com o monitoramento da rede, que se faz através da comunicação entre agentes e gerentes, é possível obter informações como, por exemplo, as condições de tráfego entre ligações TCP/IP, o nível de ruídos, os pacotes que foram enviados com sucesso e que não obtiveram sucesso, identificar a existência de erros ou falhas na transmissão de dados, configurar a rede remotamente e aplicar critérios de segurança inibindo a vulnerabilidade de acesso a rede.

Considerando o provimento de serviços Internet através de redes sem fio, a finalidade deste trabalho é o desenvolvimento de um aplicativo que permita monitoramento destas redes. Para tanto, são utilizadas as tecnologias Java, SNMP e WWW, permitindo assim, que o administrador de uma rede obtenha informações através da Internet, independente de localização.

Para garantir um melhor funcionamento das redes sem fio aplicadas a provedores de serviços Internet, é que surge a idéia principal deste trabalho, o desenvolvimento de um aplicativo para monitoramento de redes sem fio, baseado no elemento de rede *Wireless Bridge (WavePOINT-II)* da família ORINOCO (produto desenvolvido pela Lucent).

1.1. OBJETIVOS

O objetivo geral deste trabalho é analisar a viabilidade e a necessidade de implementação de um aplicativo para o monitoramento de redes sem-fio. O sistema pode ser acessado em qualquer sistema operacional, independente de localização, para garantir o amplo funcionamento dos serviços oferecidos através de ondas de rádio. É utilizado o protocolo de gerência SNMP para obtenção de dados sobre as condições da rede, a linguagem de programação Java para o desenvolvimento do aplicativo, e o serviço WWW para permitir a interface com o usuário final (administrador de rede), independente de localização e plataforma.

Para atingir esta meta, são definidos os seguintes objetivos específicos:

- Estudo sobre a Internet e WWW;

- Estudo da linguagem de programação Java no contexto da implementação para gerência de redes;

- Gerência de redes e o protocolo SNMP com ênfase em redes sem fio;

- Redes sem fio (*wireless network*) aplicada ao provimento de serviços da internet;

- Necessidades e viabilidade do gerenciamento das redes sem fio;

- Ferramentas utilizadas para coleta de dados SNMP;

- Estudo do ambiente de desenvolvimento – estudo de caso – local da aplicação;

- Implementação e teste do aplicativo.

1.2. JUSTIFICATIVAS

O uso da linguagem Java tem demonstrado sua eficácia para o desenvolvimento de aplicativos rápidos, robustos e eficientes [DEI98]. Sendo uma linguagem orientada a objeto ela facilita o processo de desenvolvimento de aplicativos. A sua vasta biblioteca oferece um conjunto de classes que facilitam o desenvolvimento de aplicativos para gerenciamento de redes. Aplicativos desenvolvidos em Java podem ser executados em qualquer sistema, ou seja, são independentes de plataforma. Por isso, foi feita a adoção desta ferramenta para desenvolver um aplicativo de gerenciamento [LOR98][GON98].

A tecnologia WWW oferece o acesso remoto a informações independente de equipamento e localização, desde que os equipamentos de rede estejam instalados e configurados de forma adequada, de modo a permitir o acesso à rede. Com Java é possível desenvolver aplicativos que possam ser executados na Web. Assim, estas duas tecnologias em conjunto, permitem o desenvolvimento de um aplicativo para monitoramento e controle da rede de modo remoto, considerando ainda, a potencialidade das classes Java para efetuar o acesso a MIB (*Management Information Base*), através do SNMP [BAR98] [KLA00].

A tecnologia sem fio está em expansão e as ferramentas para monitorar e controlar este tipo de rede ainda são em número reduzido.

O desenvolvimento de um aplicativo que trate o gerenciamento de redes sem fio é importante, devido ao crescimento de usuários que necessitam de qualidade, segurança e robustez dos serviços oferecidos pela Internet. Com o aplicativo será possível monitorar a rede, controlar o tráfego de informações e detectar possíveis falhas nos elementos sem fio, o que facilitará o administrador da rede no controle e monitoramento da rede, visando fornecer ao usuário a qualidade de serviço requerida.

1.3. ORGANIZAÇÃO DO TRABALHO

Este trabalho está estruturado da seguinte maneira.

No capítulo 2 é apresentada uma revisão bibliográfica, contemplando os conceitos fundamentais para o desenvolvimento deste trabalho. Os estudos realizados nesta etapa referem-se aos serviços da Internet, à tecnologia Web, à linguagem de programação Java e a gerência de redes de computadores baseada no protocolo SNMP.

O capítulo 3 apresenta a descrição da tecnologia de redes sem fio (*wireless network*), bem como, os principais conceitos, a tecnologia de transmissão sem fio, as técnicas de espalhamento espectral, o provimento de serviços Internet usando redes sem fio, a aplicabilidade e viabilidade destas redes.

Na seqüência, o capítulo 4 apresenta o cenário para desenvolvimento do aplicativo para monitoramento e controle de redes sem fio.

Logo após, o capítulo 5 contempla a implementação do aplicativo, usando toda a tecnologia estudada e sendo aplicado ao cenário apresentado no capítulo 4.

Por fim, no capítulo 6 é realizada uma discussão do trabalho realizado, bem como apresentadas conclusões, dificuldades encontradas para o desenvolvimento do trabalho e perspectivas futuras.

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo, é apresentada uma revisão bibliográfica sobre os principais assuntos abordados neste trabalho, dentre eles, o provimento de serviços Internet, o serviço WWW, a linguagem de programação Java, gerência de redes, protocolos de gerência de redes e o protocolo SNMP (*Simple Network Management Protocol*).

2.1. O PROVIMENTO DE SERVIÇOS INTERNET

A Internet permite que qualquer organização ou pessoa, possa conectar seu(s) computador(es) na rede por intermédio de um provedor de acesso. Este, por sua vez, interliga sua rede às demais através de um serviço privado de comunicação, por exemplo, uma companhia telefônica [ARM99].

O que vale destacar nesta interligação, é o modo como o usuário final acessa a Internet. Para tal, é necessário, muitas vezes, uma linha telefônica dedicada, para a comunicação com o provedor de acesso. Com o uso de linhas telefônicas pode ocorrer a monopolização da mesma e, podendo ter um custo elevado, principalmente, quando a prestação de serviços é dedicada.

Uma solução alternativa está sendo utilizada para provimento de serviços Internet, que é a interconexão sem fio através de ondas de rádio. Com esta tecnologia é possível dispensar o sistema telefônico para acessar a rede, livrando-se assim, das altas taxas cobradas pelas empresas de telecomunicações e atingindo altas taxas de transmissão de dados [LUC99].

Para que qualquer pessoa possa usar a Internet, é necessário o uso da arquitetura TCP/IP, que dá uma ênfase toda especial à interligação de diferentes tecnologias de redes. A idéia básica da arquitetura TCP/IP é a de dividir as funções de troca de mensagens em dois grupos: funções de *controle de fluxo* e de *conectividade entre os equipamentos*. As do primeiro grupo pertencem ao protocolo TCP e, as do segundo, ao protocolo IP [SOA97].

O que torna a Internet tão popular, são os protocolos e serviços que ela oferece. Dentre estes serviços podemos destacar o correio eletrônico, comunicação remota, transferência de arquivo e o WWW, que será visto com mais detalhes na próxima seção.

2.2. O SERVIÇO WWW (*WORLD WIDE WEB*)

O WWW (*World Wide Web*) é um serviço baseado em hipertextos que permite ao usuário buscar e recuperar informações distribuídas por diversos computadores da rede. A seleção de informações é feita com base no conceito de hipertexto, um texto cujas palavras contêm ligações subjacentes com outros textos, o que torna possível leituras diversas, não lineares [ARM99][BAR98].

A tecnologia Web envolve vários componentes. Entre eles a linguagem HTML (*Hipertext Mark-up Language*), o protocolo HTTP (*Hipertext Transfer Protocol*), a interface CGI (*Common Gateway Interface*) entre outras.

O HTML é uma linguagem padronizada que permite a criação de páginas Web através da definição de textos, figuras, *tags*, *frames*, entre outros recursos. Esta linguagem também auxilia na criação de *links* hipertexto (ponteiros), conhecidos como *Hyperlinks* e que servem como caminho de um documento para outro, permitindo assim, a navegação nos *browsers Web*, como por exemplo Netscape ou Internet Explorer. Toda a sintaxe de HTML foi baseada em SGML (*Standard Generalized Markup Language*), desenvolvida pela IBM para fazer documentos multiplataforma, podendo assim, serem lidos em qualquer sistema operacional [ARM99].

O desenvolvimento para Web baseia-se em uma arquitetura cliente/servidor. Desta forma, um programa cliente é executado na máquina do usuário (*browser Web*), fazendo solicitações de páginas para um servidor HTTP, que tem a função de receber requisições provenientes de clientes e, quando possível, retornar as informações requisitadas, utilizando o protocolo HTTP.

O protocolo HTTP é orientado à conexão, e utiliza o TCP como mecanismo de transporte, tendo como função básica, requisitar informações ao servidor HTTP e

repassar as respostas obtidas ao solicitante do serviço. Este protocolo segue o simples modelo de pedido/resposta, sendo que é o responsável por estabelecer a conexão com o servidor Web (HTTP) especificado na URL (*Uniform Resource Location*), obter o documento pedido e só então fechar a conexão. Quando um usuário "clicar" em algum *Hiperlink*, ele está abrindo uma nova sessão de comunicação com o protocolo HTTP [TAN97].

Em um *browser Web* é possível apresentar diferentes tipos de documentos, podendo ser somente texto, formulários, imagens, som, gráficos e animações, o que é chamado de hipermídia. Os documentos hipermídia podem ser apresentados na forma de CGI (*Common Gateway Interface*), de aplicativos em scripts ou, ainda, por meio de *applets*.

CGIs são programas que podem ser escritos em diversas linguagens, como PERL ou C, e que se comunicam com servidores Web através de uma interface padronizada chamada de CGI [DEI98]. Então, de uma forma geral, o CGI é um padrão que permite a execução de programas ou *scripts* em um servidor. Com ele é possível criar programas capazes de realizar consultas a uma base de dados relacional, realizar transações comerciais, entre outras tarefas. No entanto, o CGI apresenta um comportamento estático, no sentido de que o usuário preenche os parâmetros desejados (como o código de um produto para um CGI destinado à consulta do estoque) e obtém os resultados.

Os *applets* realizam tarefas, apresentado um comportamento dinâmico, ao contrário do CGI, ou seja, é possível mudar os estados de uma página de acordo com as ações dos usuários ou de outros eventos. Para o desenvolvimento de applets é necessário o uso da linguagem de programação Java, que será vista na próxima seção [SUN99].

Considerando o fato de que é possível acessar qualquer servidor HTTP em qualquer computador que esteja conectado a Internet e o tratamento gráfico de fácil manipulação, os *browsers Web*, darão suporte ao desenvolvimento do trabalho.

2.3. A LINGUAGEM DE PROGRAMAÇÃO JAVA

A Linguagem Java é ao mesmo tempo um ambiente e uma linguagem de programação, produzido pela *Sun Microsystems, Inc.* Trata-se de mais um representante da geração de linguagens orientadas a objetos e foi projetado para resolver os problemas da área de programação cliente servidor. Java foi criada como parte de um grande projeto da Sun, cuja missão era desenvolver aplicativos complexos e avançados para aplicação em pequenos dispositivos eletrônicos. Esses dispositivos são sistemas portáteis, distribuídos, confiáveis e incorporados em tempo real, dirigidos aos consumidores em geral.

Java tem a capacidade de montar documentos HTML dinâmicos. Os aplicativos em Java são compilados em um código de bytes (*byte code*) independente de arquitetura. Esse código de bytes pode então ser executado em qualquer plataforma que suporte um interpretador Java [SUN99], conforme mostra a figura 1.

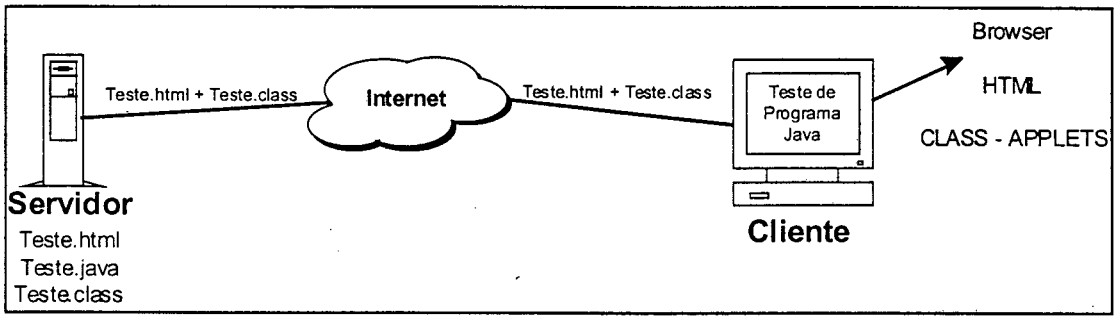


FIGURA 1 – Modelo browser/html/applets.

Utilizando Java é possível a criação de programas do tipo *applets* ou *applications*. O primeiro trata de aplicações destinadas a serem executadas em *browsers* Web. Neste caso, a classe principal da aplicação deve ser derivada da classe *Applet*. O segundo tipo de programa é destinado a execução a partir de um interpretador [SUN99] [DEI98].

Programas em Java são plenamente integráveis com o sistema de interface Web e oferecem um nível de interatividade e robustez muito maior que uma interface baseada unicamente em HTML. Para exemplificar estas características, pode-se citar o caso de um usuário que necessite enviar informações de um formulário a um servidor,

utilizando um *browser* Web. Para realizar esta operação sem a utilização do Java, o usuário teria que digitar os dados, para então, submetê-los para um servidor utilizando um CGI e na ocorrência de algum erro, seria retornada uma mensagem indicativa. Neste caso, o usuário teria que corrigir os dados e realizar a submissão novamente. Utilizando Java, os dados podem ser analisados antes de serem enviados para o servidor. Isto torna a interface mais amigável e diminui a carga de processamento [DEI98].

Dentro das expectativas deste trabalho, a linguagem Java oferece as características necessárias para o desenvolvimento do aplicativo e para monitoramento e gerenciamento de rede. Possui dinamismo, é independente de plataforma, e pode ser executada em rede (como a Internet). Com a utilização de applets como página dinâmica, é possível criar gráficos sofisticados, baseado em informações do servidor, o que será importante para o desenvolvimento deste trabalho.

2.3.1. Características da Linguagem Java

Dentre as principais características da linguagem de programação Java, destacam-se:

- **Orientação a Objetos**: Os objetos encapsulam dados e funções relacionados em unidades coesas, é fácil localizar dependências de dados, isolar efeitos de alterações e realizar outras atividades de manutenção, e talvez o mais importante, as linguagens OO facilitam a reutilização e manutenção de código [DEI98] [SUN99].

  A linguagem Java inclui um conjunto de bibliotecas de classes que fornecem tipos de dados básicos, características de entrada e saída de sistemas e outras funções de utilitário. Essas classes básicas fazem parte do conjunto de desenvolvimento da Java, que também possui classes que suportam o sistema de rede, protocolos comuns da Internet e funções do conjunto de ferramentas de interface com o usurário. Como essas bibliotecas de classe

são escritas em Java, são portáveis nas plataformas, da mesma forma como todos os aplicativos Java.

- **Distribuição** de informações para compartilhamento e trabalho conjunto, com a distribuição de carga de trabalho do processamento, é uma característica essencial dos aplicativos cliente/servidor. Há uma biblioteca de procedimentos TCP/IP incluída nos códigos fonte e de distribuição binária do Java. Isso facilita aos programadores o acesso remoto às informações, usando protocolos como HTTP e FTP. Estas características foram consideradas no momento da escolha da linguagem de programação Java para a implementação deste projeto [DEI98] [ARM99].

  O termo "distribuído" descreve relacionamentos entre objetos de sistemas, estejam eles em sistemas remotos ou locais. Uma das grandes vantagens dos applets e aplicativos é que eles podem abrir e acessar objetos na Web através de URLs com a mesma facilidade com que podem acessar um sistema de arquivos local [ARM99].

- **Interpretação Independente de Plataforma**: Os programas escritos em Java são compilados em formato binário de código de bytes (*bytecodes*), que então são interpretados por um ambiente de execução do Java específico. Portanto, a linguagem Java é ao mesmo tempo compilada e interpretada [SUN99].

- **Multitarefas**: Os objetos binários de códigos de bytes do Java são formados por seqüências de execução múltiplas e simultâneas. Essas seqüências são conhecidas como contextos de execução ou processos leves. O Java oferece suporte no nível de linguagem para multitarefa, resultando em uma abordagem de programação mais poderosa e de múltiplas facetas [SUN99].

- **Robustez e Consistência**: Quanto mais robusto um aplicativo, mais confiável ele será. Isso é desejável tanto para os desenvolvedores de software quanto aos usuários. A robustez dará a garantia que o aplicativo

continuará executando mesmo que aconteça algum erro por parte do usuário, sistema operacional ou até mesmo do aplicativo [DEI98] [SUN99].

Uma linguagem consistente é uma linguagem confiável, ou seja, possui uma mínima probabilidade de apresentar *bugs* e fazer com que os programas escritos nela não "congelem". Um programa Java não pode congelar o sistema porque não tem permissão para acessar toda a memória do computador.

- **Segurança**: Como a linguagem Java foi criada para ambientes de rede, os recursos de segurança receberam muita atenção. Por exemplo, se ao executar um binário transferido por *download* da rede, o mesmo pode estar infectado por vírus. Os aplicativos Java apresentam garantia de resistência contra vírus e de que não são infectados por eles. No Java, a autenticação do usuário é implementada com um método de chave pública de criptografia. Isso impede, de certa forma que *hackers* e *crackers* examinem informações protegidas como nomes e senhas de contas [DEI98] [ARM99].

  Alguns dos outros recursos, como a consistência e o fato de que o Java é interpretado e também compilado, auxiliam a segurança. Por exemplo, o fato dos programas em Java não poderem acessar a memória significa que eles podem ser executados com segurança. Além disso, as instruções em *bytecodes* contêm informações extras para verificar a legalidade do programa e possíveis violações de segurança.

  Os mecanismos de segurança do Java atuam em quatro níveis diferentes da arquitetura do sistema. Primeiro: a própria linguagem Java foi projetada para ser segura, e o compilador Java garante que o código fonte não viola essas regras seguras. Segundo: todos os códigos de máquina executados pelo programa em tempo de execução são protegidos para ter certeza de que eles também obedecem a regras. Terceiro: o carregador da classe assegura que as classes não violam o espaço do nome ou restrições de acesso quando eles são carregados para dentro do sistema. Finalmente: segurança específica da

API limita o escopo de execução ao *applets*. Essa camada final depende da segurança e integridade garantidas a partir das outras três camadas.

- **Simplicidade**: Um dos principais objetivos do projeto do Java foi criar uma linguagem o mais próxima possível do C++, para garantir sua rápida aceitação no mundo do desenvolvimento OO. Outro objetivo do seu projeto foi eliminar os recursos obscuros e danosos do C++, que fugiam à compreensão e aumentavam a confusão que poderia ocorrer durante as fases de desenvolvimento, implementação e manutenção do software. O Java é simples porque é pequeno. O interpretador básico do Java ocupa aproximadamente 40K de RAM, excluindo-se o suporte a multitarefa e as bibliotecas padrão, que ocupam outros 175K. Mesmo a memória combinada de todos esses elementos é insignificante, se comparada a outras linguagens e ambientes de programação [DEI98].

  Uma das maiores dificuldades dos programadores de C e C++ é a de gerenciamento de armazenamento, alocação e liberação de memória. Geralmente, um programador destas linguagens deve se preocupar com a quantidade de memória que está usando, com a liberação de espaços que não estão mais sendo utilizados. Esta é uma das grandes causas de erros de memória. É por este motivo que os desenvolvedores do Java criaram o *coletor automático de lixo,* que libera os endereços de memória por ele reservado automaticamente.

- **Alto Desempenho**: Há muitas situações em que a interpretação de objetos de códigos de bytes proporciona desempenho aceitável. Mas outras circunstâncias exigem desempenho mais alto. O Java concilia tudo isso oferecendo a tradução dos códigos de bytes para o código de máquina nativo em tempo de execução. O alto desempenho permite a implementação de aplicativos Web em Java, na forma e programas pequenos e velozes, que podem ampliar significativamente os recursos tanto do cliente quanto do servidor [SUN99].

- **Multiencadeamento**: Multiencadeamento é a capacidade de manipular várias tarefas simultaneamente. Os computadores conseguem realizar multiencadeamento através de *multithreading*. O C e C++ se encaminham para um programa de *thread*, ou encadeamento, único, que inibe a capacidade de desenvolver programas multiencadeados [ARM99].

  Com Java pode-se usar *threads* múltiplos, que permite a execução de diversos *threads* ao mesmo tempo, dentro do mesmo programa, em paralelo, sem interferência entre eles. O uso de *threads* será essencial na implementação dos aplicativos de coleta de dados do tráfego na rede [ARM99].

- **Conectividade Com Banco de Dados**: Java tem a especificação JDBC (*Java DataBase Connectivity*), que permite que aplicações Java acessem quaisquer bases de dados, sendo inclusive possível trocar o banco de dados sem modificar a aplicação.

2.4. GERÊNCIA DE REDES

Na Internet os mais variados tipos de computadores, dos pessoais aos de grande porte, estão conectados por meio de diferentes tipos de redes físicas, como por exemplo, *Ethernet*, *token-ring*, *FDDI*, *Frame-Relay*, *ATM*, *ISDN*, linhas *dial-up* e redes sem fio, usando *hubs*, roteadores, *gateways*, *modens* e placas de redes de múltiplos fornecedores, caracterizando-se assim, os sistemas heterogêneos. O uso de todos estes elementos de redes, quando mal configurados ou instalados podem resultar em: aumento de manutenção, baixo desempenho, ineficiência e insegurança [ZAC00] [KLA00].

A gerência de redes pode ser entendida como o processo de monitorar e controlar uma rede de computadores de tal modo que seja possível maximizar sua eficiência e produtividade [BAR98]. Tal processo compreende um conjunto de funções integradas que podem estar centralizadas em uma máquina ou distribuídas por milhares de quilômetros, em diferentes organizações e residindo em máquinas distintas. Com

estas funções é possível controlar uma rede de computadores e seus serviços, provendo mecanismos de monitoração, análise e controle dos dispositivos e recursos da rede [MEN98].

2.4.1. Etapas no Processo de Gerência

Uma aboradagem para estruturar um sistema de gerência consiste em utilizar um processo gerente centralizado que interage com os diversos componentes da rede a serem gerenciados para deles extrair as informações necessárias para a monitoração e controle [COM99][GON98].

A gerência de redes está intimamente relacionada com o controle de atividades e do uso de recursos da rede. Como tarefas básicas da gerência de redes, pode-se obter informações sobre o funcionamento da rede e tratá-las, com o objetivo de obter diagnósticos e encaminhá-los para soluções de problemas que possam ser detectados ou escolher uma forma de melhor utilizar os recursos da rede.

Assim, podemos dizer, que são três as etapas no processo de gerência de redes [COM99]. A primeira trata da *coleta de dados,* que é um processo, em geral automático, e consiste de monitoração sobre os recursos gerenciados. A segunda é o *diagnóstico,* que consiste no tratamento e análise realizadas a partir dos dados coletados. O processo de gerenciamento executa uma série de procedimentos com o intuito de determinar a causa do problema representado no recurso gerenciado. Por fim, a *ação,* onde, uma vez diagnosticado o problema, cabe uma ação, ou controle, sobre o recurso, caso o evento não tenha sido passageiro (incidente operacional) [BAR98][ARM99].

2.4.2. Modelo Genérico de Gerenciamento de Redes

Um modelo genérico de gerenciamento de redes apresenta basicamente quatro elementos [KLA00][CHI99]:

1. **Objetos Gerenciados**: Um objeto gerenciado representa um recurso sujeito ao gerenciamento, isto é, que pode ser gerenciado. Ele é definido em termos de seus atributos, das operações a que pode ser submetido, das notificações que pode emitir e de seus relacionamentos com outros objetos gerenciados. O conjunto de objetos gerenciado juntamente com seus atributos, operações e notificações, constitui a MIB (*Management Information Base*). A MIB, ou Base de Informação Gerencial é o nome conceitual para a informação de gerenciamento. Como exemplo, podemos citar os roteadores, *bridges*, *gateways*, impressoras, *hubs*, *modems* e outros.

2. **Agentes**: são os processos que informam a ocorrência de eventos e mantém variáveis que refletem o estado dos elementos gerenciados, além de também poderem alterar os valores destas variáveis. Os agentes são vistos como entidades que fazem a interface com os dispositivos a serem gerenciados. Eles incluem sistemas finais que suportam aplicações de usuários bem como os nós que oferecem um serviço de comunicação, tais como processadores de *front-end*, controladores de *clusters*, pontes e roteadores.

3. **Gerentes**: têm a responsabilidade de comandar as variáveis, tomar decisões e executar as aplicações de gerenciamento e o protocolo de gerenciamento. O gerente é um agente que possui um conjunto de NMA's (*Network Management Application*). Um NMA pode ser entendido como uma aplicação que inclui uma interface de operador para permitir a um usuário autorizado gerenciar a rede.

4. **Protocolo de Gerenciamento**: possibilita a monitoração e controle dos elementos de rede e promove a troca de informações de gerenciamento entre *agente* e *gerente*.

A figura 2 ilustra uma arquitetura típica do gerenciamento de redes, com as interações entre agente/gerente.

2.4.3. SISTEMAS DE GERENCIAMENTO DE REDES

Um sistema de gerenciamento de redes é uma coleção de ferramentas de monitoração e controle que é integrado no sentido de possuir dois elementos básicos. O primeiro trata uma única interface de operação com um conjunto de comandos potentes, mas amigável, para realizar a maioria das tarefas de gerenciamento de redes. O segundo é uma quantidade mínima de equipamentos separados [COM99][KLA00]. Isto é, a maioria dos elementos de *hardware* e *software* requeridos para o gerenciamento de redes, está incorporado dentro do equipamento do usuário.

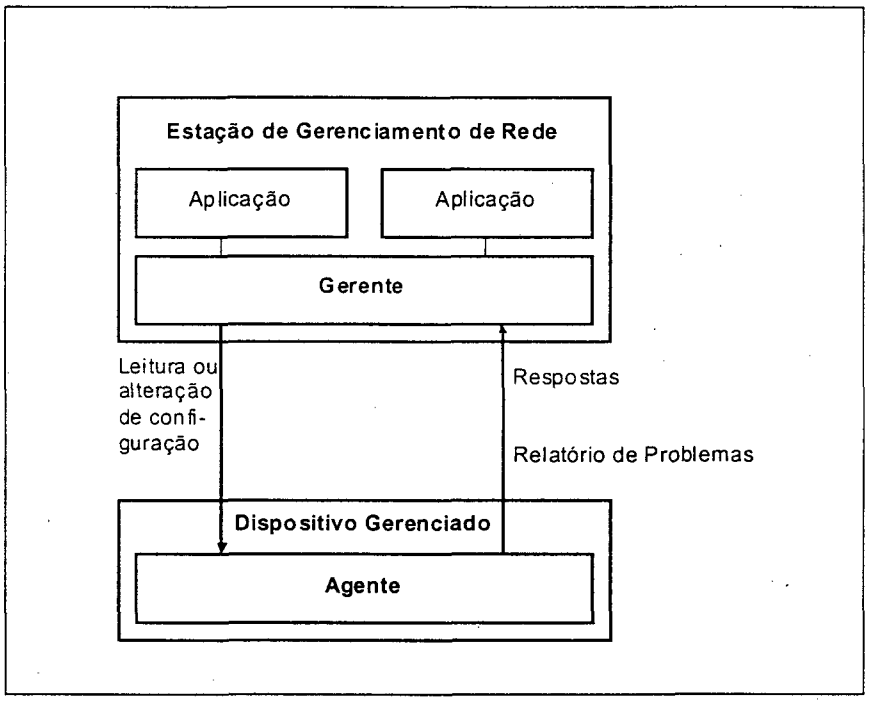


FIGURA 2 – Arquitetura do Gerenciamento de Redes – Interações Agente/Gerente

De forma simplificada, pode-se dizer que um sistema de gerenciamento de redes contém dois elementos: um gerente e vários agentes.

2.4.4. ARQUITETURA DO SOFTWARE DE GERENCIAMENTO

A arquitetura do software de gerenciamento residente no gerente e nos agentes varia de acordo com a funcionalidade da plataforma adotada. Genericamente, o software pode ser dividido em três grandes grupos, como mostra a figura 3:

- **Software de Apresentação:** é a interface de usuário, que permite à ele o monitoramento e o controle da rede. Normalmente ela está localizada num sistema hospedeiro gerente. Em alguns casos é comum existir uma interface em alguns agentes a fim de permitir a execução de testes e também a visualização ou alteração de alguns parâmetros localmente.

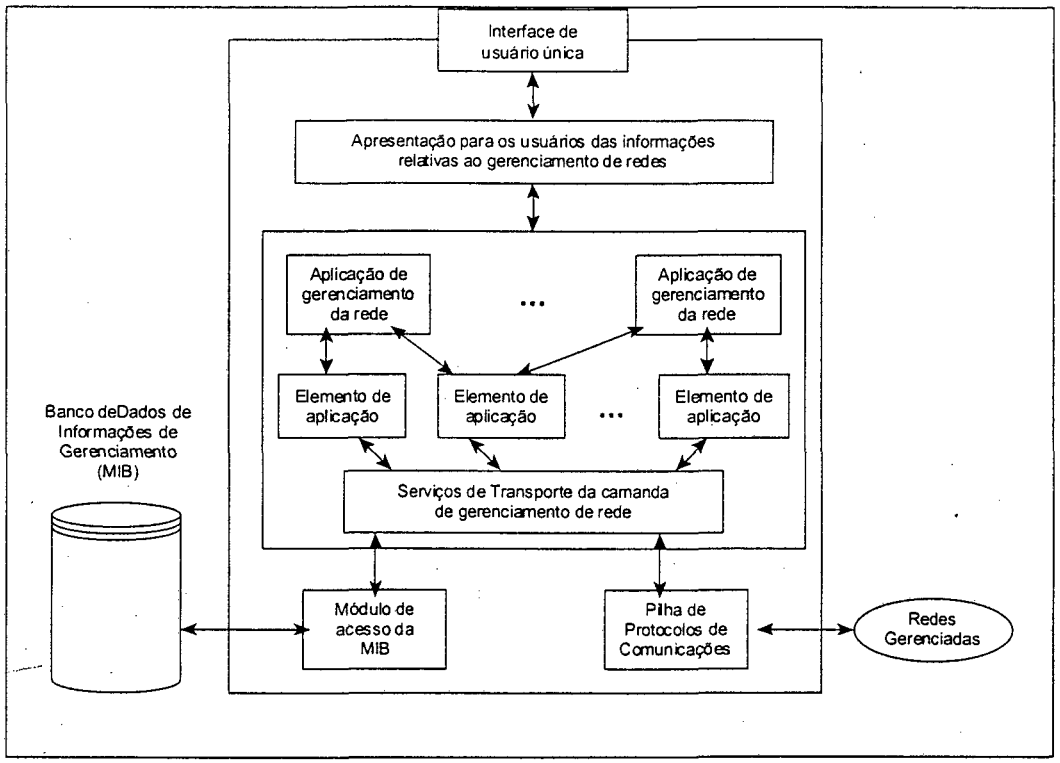


FIGURA 3 - Visão genérica para uma arquitetura de gerenciamento de redes [SUA99].

- **Software de Gerenciamento:** conforme a figura 3, podemos observar que uma estrutura genérica de um software de gerenciamento está organizada em três níveis. O primeiro é a aplicação de gerenciamento de rede que provê os serviços de interesse do usuário como, por exemplo, gerenciamento de falhas, de configuração, de segurança, etc. O segundo é composto pelos

elementos de serviço da aplicação, que implementam funções de propósito gerais servindo de suporte a diversas aplicações, tais como, alarmes genéricos ou sumarização de dados. O terceiro nível é o serviço de transporte de dados de gerenciamento que consiste de um protocolo usado para a troca de informações entre gerentes e agentes e de uma interface de serviço para os elementos de serviço de aplicação.

- **Software de Suporte ao Gerenciamento:** para executar suas tarefas, o software de gerenciamento necessita acessar uma base de informações de gerenciamento (MIB) e agentes e gerentes remotos.

2.4.5. A RELAÇÃO ENTRE GERENTE E AGENTE

As atividades de gerência de redes são controladas por processos, conforme mostra a figura 2. Estes processos podem ser classificados como processo *gerente* e processo *agente* [KLA00].

O processo gerente é a parte de uma aplicação distribuída associada ao usuário da gerência. Ele tem a responsabilidade de realizar operações de gerência, sobre os objetos gerenciados, enviando comandos através dos processos agentes. O gerente receberá notificações enviadas pelo agente a fim de obter algumas informações sob o estado da rede.

O processo agente é a parte de uma aplicação distribuída que irá executar sobre o objeto gerenciado os comandos enviados pelo processo gerente. Assim, ele passará para o gerente uma visão dos objetos sendo gerenciados e refletirá o comportamento destes objetos, emitindo notificações sobre os mesmos.

Uma aplicação de gerência pode exercer o papel de gerente, de agente ou ambos. Um processo de gerência no papel de agente atua sobre objetos em seu ambiente local, executando ações de gerência sobre estes objetos como conseqüência de operações enviadas pelo gerente.

Os processos gerentes monitoram ou modificam as instâncias remotas de objetos gerenciados, bastando para tanto se comunicarem com os agentes, controlando desta forma os recursos de uma rede. Para que haja a comunicação entre gerentes e agentes há a necessidade de um protocolo de gerência. Para desenvolvimento do trabalho o protocolo utilizado será o SNMP, que será descrito com maiores detalhes na seção 2.6.

2.5. PROTOCOLOS DE GERÊNCIA

O protocolo de gerenciamento de rede é o método pelo qual, agentes e gerentes de rede trocam informações. O protocolo define quais mecanismos de transporte podem ser usados, qual informação há em um pacote e em qual formato essa informação está organizada [ZAC00] [KLA00].

Os protocolos de gerência foram desenvolvidos devido a necessidade de estabelecer monitoramento e controle sobre todos os componentes da rede, de forma a garantir que esta esteja sempre em funcionamento e que os problemas seja identificados, isolados e solucionados o mais rápido possível. Entretanto, esta não é uma tarefa fácil. As redes têm assumido grandes proporções, com um grande número de computadores, além da constante adição de novos componentes, oferecendo integração de dados e voz, multiplexadores e roteadores, além de tantos outros, o que tem adicionado mais complexidade a esse ambiente [BAR98] [COM99].

A principal meta de um protocolo de gerenciamento é permitir aos gerentes de rede realizar tarefas como obter dados sobre desempenho e tráfego da rede em tempo real, diagnosticar problemas de comunicação e reconfigurar a rede atendendo às mudanças nas necessidades dos usuários e do ambiente.

Ciente destas dificuldades, a ISO (*International Organization for Standardization*) desenvolveu e padronizou o protocolo CMIP (*Common Management Information Protocol*) para gerenciamento de redes OSI (*Open Systems Interconnection*). Por outro lado, existe o IETF (*Internet Engineering Task Force*) com um conjunto de padrões para o gerenciamento de redes TCP/IP, ou seja, gerenciamento

de redes Internet. O protocolo definido pelo IETF é o SNMP (*Simple Network Management Protocol*) [TAN97][RFC1448]. Considerando o aspecto funcional deste trabalho, como gerenciamento para Internet, será abordado aqui, somente o protocolo SNMP.

2.6. O PROTOCOLO SNMP

O protocolo SNMP (descrito nos RFCs 1155, 1157, 1212, 1213) foi projetado, em meados dos anos 80, como uma resposta aos problemas de comunicação entre diversos tipos de redes. A idéia básica por trás do SNMP é oferecer uma maneira facilmente implementável e com baixo *overhead* para o gerenciamento de roteadores, servidores, estações de trabalhos e outros recursos de redes heterogêneas. No momento de sua concepção, a meta foi que ele representasse apenas uma solução provisória até que surgisse um projeto melhor de protocolo para gerência de redes. Entretanto nenhuma solução melhor tornou-se disponível.

O SNMP é um protocolo simples, como o nome diz (*"Simple" Network Management Protocol*). Na arquitetura Internet TCP/IP, ele está no nível de aplicação, operando sobre o UDP (*User Datagram Protocol*), como mostra a figura 4 . Ele é considerado "simples" porque os agentes requerem um software mínimo. Muito do poder de processamento e de armazenamento de dados reside no sistema de gerenciamento, enquanto um subconjunto complementar dessas funções reside no sistema gerenciado. Assim, o funcionamento do SNMP é baseado no paradigma chamado de '*fetch-store*', que se caracteriza por poucos comandos atuando sobre uma ou mais variáveis, de forma que a supervisão do sistema é feita simplesmente via operações '*get*', e o controle ou atuação sobre o funcionamento de um equipamento é feita usando também operações do tipo '*set*', sobre determinadas variáveis desse equipamento [CHI99] [COM99].

Além das operações *get* e *set*, que permitem ao administrador buscar o valor de um dado ou armazenar um valor em um dado, respectivamente, o SNMP oferece mais algumas que são implementadas a partir dessas, como mostra a tabela 1.

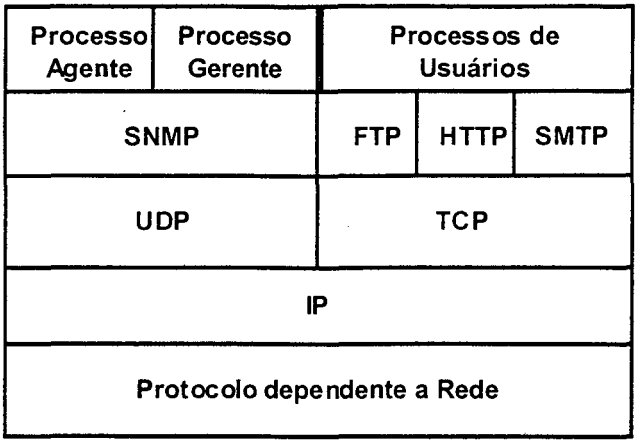


FIGURA 4 - Protocolo SNMP sobre as Camadas TCP/IP.

Operação	Relação	Função
get-request	Gerente → Agente	Permite recuperar o valor de uma ou mais variáveis.
get-next-request	Gerente → Agente	Permite recuperar o valor de uma variável sem se saber o nome exato.
get-response	Agente → Gerente	Resposta do servidor a um dos comandos acima.
set-response	Gerente → Agente	Permite atribuir valores às variáveis.
Trap	Agente → Gerente	Permite o relato de eventos a estações de gerenciamento de forma assíncrona.

Tabela 1 - Operações de Gerenciamento do SNMPv1 sobre a MIB.

2.6.1. A ARQUITETURA SNMP

O modelo arquitetural SNMP, conforme a figura 5, é composto de uma coleção de estações de gerenciamento de rede e elementos de rede. As estações de gerenciamento de rede executam aplicações de gerenciamento, as quais, permitem o monitoramento e controle dos elementos de rede. Para ser diretamente gerenciado através do SNMP, um nó deve ser capaz de executar um processo de gerenciamento SNMP, que é um agente. Os elementos de rede são dispositivos como estações hospedeiras (*hosts*), *gateways*, servidores de terminais (pontes, *hubs*, *modems*, multiplexadores), e outros similares, que possuem agentes de gerenciamento responsáveis pela execução das funções de gerenciamento de rede, solicitadas pelas estações de gerenciamento. Para fazer a comunicação das informações de

gerenciamento entre estações de gerenciamento e os agentes existentes nos elementos de rede é utilizado o protocolo SNMP [RFC 1157] propriamente dito.

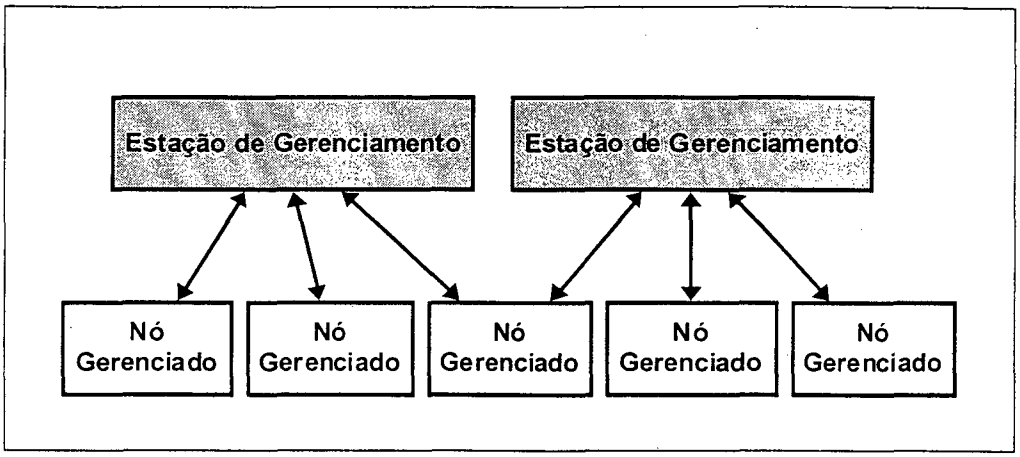


FIGURA 5 - A arquitetura do modelo SNMP [SUA99].

O protocolo SNMP foi desenvolvido com base no modelo cliente/servidor, em que o lado cliente é constituído pelo equipamento que executa o gerenciamento da Internet e o lado servidor, pelos equipamentos a serem gerenciados [CHI99]. Os equipamentos do lado servidor normalmente são conhecidos como agentes.

2.6.2. COMUNICAÇÃO ENTRE GERENTE E AGENTE

Como mostra a figura 6, no modelo de gerenciamento SNMP existem basicamente processos agentes (servidores) e processos gerentes (clientes). Os agentes são processos cuja função é receber solicitações do gerente, logo, eles realizam o processamento necessário e enviam as respostas obtidas ao gerente. Um processo agente pode ainda, enviar notificações de um evento qualquer, como por exemplo a mudança de *status* na interface de um roteador. As notificações são enviadas através de *traps*[1]. O processo agente pode estar instalado em qualquer equipamento de rede, podendo este ser um *hub*, *switch*, roteador, estação de trabalho, entre outros. O processo gerente é

[1] Traps é uma operação utilizada por um agente SNMP para notificar de forma assíncrona a um gerente algum evento extraordinário que tenha ocorrido no objeto gerenciado.

responsável por enviar solicitações para os agentes, obter as respostas e realizar o tratamento de *traps*.

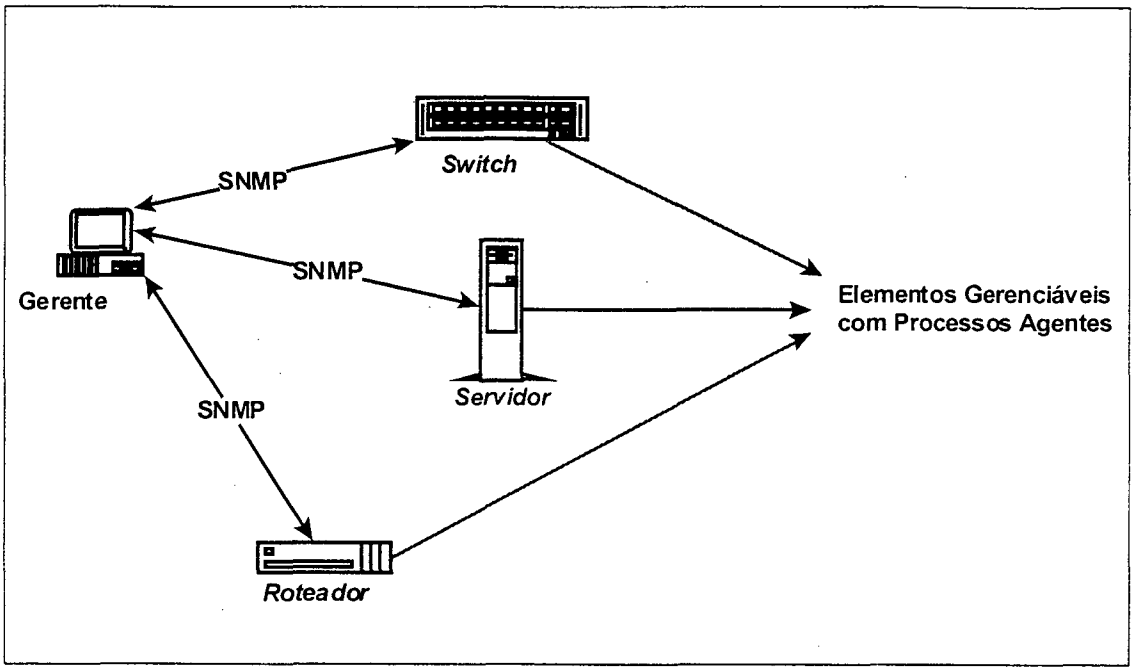


FIGURA 6 – Comunicação entre Gerentes e Agentes.

O funcionamento dos processos agentes se dá através da leitura da MIB (*Management Information Base*) que o agente possui. A MIB possui todas as informações de gerência. O agente ao receber uma mensagem SNMP do gerente identifica que operação está sendo requisitada e quais variáveis deverão ser tratadas. Em seguida ele executa a operação sobre a MIB, para então, enviar a mensagem de resposta ao gerente.

Os gerentes servem de interface entre as aplicações de gerência e os agentes. Cabe a ele enviar comandos aos agentes, solicitando informações sobre variáveis de um objeto gerenciado, ou solicitar a modificação de valor de determinadas variáveis.

Os gerentes processam estas informações colhidas pelos agentes, e as repassam à aplicação que as requisitou. A comunicação entre o gerente e as aplicações é possível através da utilização das API's (*Application Program Interface*) do gerente SNMP pelo sistema.

A API é um conjunto de funções que intermediam a execução de comandos entre um programa e outro, de forma a simplificar a um programa o acesso a funções do outro programa, e que no caso do SNMP intermediam as execuções entre uma aplicação de gerência e o gerente SNMP [TAN97].

2.6.2.1. INFORMAÇÕES DE GERENCIAMENTO – MIB

Um equipamento gerenciado deve manter em seu agente as informações de controle e estado que o administrador possa acessar. Em um roteador, por exemplo, são mantidas várias informações estatísticas como tráfego de entrada e saída, as quais podem, mais tarde, serem recuperadas pelo administrador da rede. O protocolo SNMP não especifica exatamente quais dados podem ser acessados [KLA00][COM99].

As informações de gerenciamento do modelo SNMP estão relacionadas com o conceito de objetos gerenciados, que corresponde a recursos físicos ou lógicos passíveis de serem monitorados e/ou controlados [SUA99].

Neste contexto um objeto gerenciado é uma variável de dados. Cada objeto gerenciado tem um valor e é definido por meio de um tipo de objeto. As informações sobre objetos gerenciados são armazenadas na MIB. Os objetos gerenciados (variáveis) da MIB são definidos através de uma organização hierárquica (MIT – *Management Information Tree*), também conhecida como espaço de nome do identificador do objeto (*object identifier namespace*), administrado pelos comitês ISO e CCITT, conforme descrito na subseção posterior. A MIB para Internet classifica as informações de gerenciamento em oito categorias, conforme a tabela 2, e cada uma delas possui um identificador numérico. Assim, a representação da MIB pode ser literal ou numérica.

A MIB padrão fornece um conjunto de objetos gerenciados (variáveis) que estão relacionados conforme cada uma das categorias citadas acima. Alguns exemplos de variáveis são apresentados na tabela 3.

Algumas variáveis da MIB podem armazenar tipos de dados inteiros, strings ou até mesmas estruturas mais complexas, como por exemplo, uma tabela de registros.

Categoria da MIB	Tipo de informação
Sistema	Informações sobre o equipamento – Sistema Operacional do roteador ou do host.
Interfaces	Informações sobre Interfaces de rede específicas.
At	Conversões de endereços – por exemplo, mapeamento ARP.
Ip	Informações sobre o Protocolo IP.
Icmp	Informações sobre o Protocolo de Controle de Mensagens Internet.
Tcp	Informações sobre o Protocolo TCP
Udp	Informações sobre o Protocolo UDP
Egp	Informações sobre o Protocolo EGP (Protocolo de Gateway Externo).

Tabela 2 – Categorias de Informação na MIB.

Variáveis da MIB	Categoria	Função
SysUpTime	Sistema	Tempo desde a última reinicialização
IfNumber	Interfaces	Número de interfaces de rede
IpInReceives	Ip	Número de datagramas recebidos
IpForwDatagrams	Ip	Número de datagramas encaminhados
IcmpInEchos	Icmp	Número de solicitações Echo ICMP recebidas
TcpMaxConn	Tcp	Conexões máximas de TCP permitidas
UdpInDatagrams	Udp	Número de datagramas do UDP recebidos
EgpInMsgs	Egp	Número de mensagens EGP recebidas

Tabela 3 – Exemplo de variáveis MIB [COM99].

2.6.2.2. Estrutura da MIB e Esquema de Nomeação

Conforme citado anteriormente, a MIB usada pelo protocolo SNMP é organizada através de uma representação hierárquica, como mostra a figura 7. Cada nó da árvore tem associado um nome e um valor inteiro, para identificar o objeto tanto na forma literal quanto numérica.

As tabelas 1 e 2 mostrados anteriormente apresentam as categorias da MIB e alguns objetos referentes a essas categorias, respectivamente. A figura 8 mostra as 8 (oito) categorias pertencentes a MIB em sua representação hierárquica, bem como alguns objetos de cada categoria.

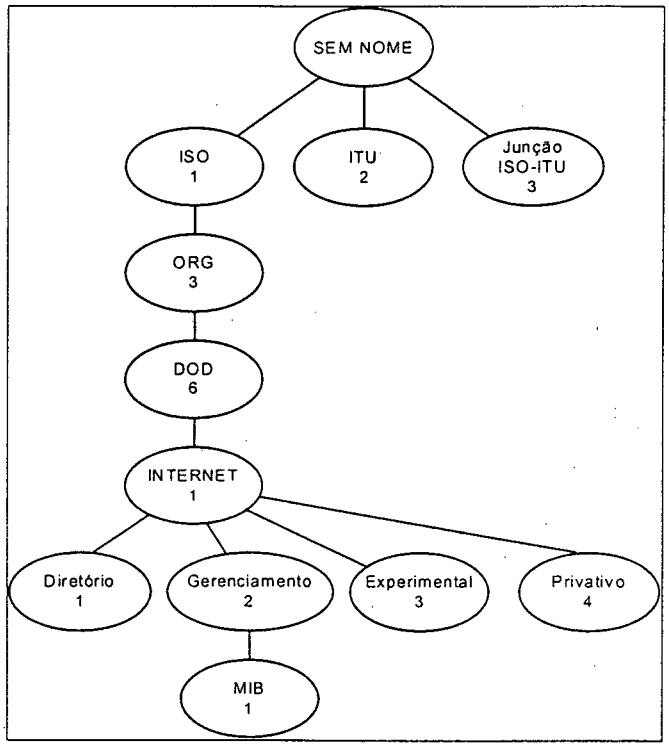


FIGURA 7 – Organização hierárquica de nomes acima da MIB.

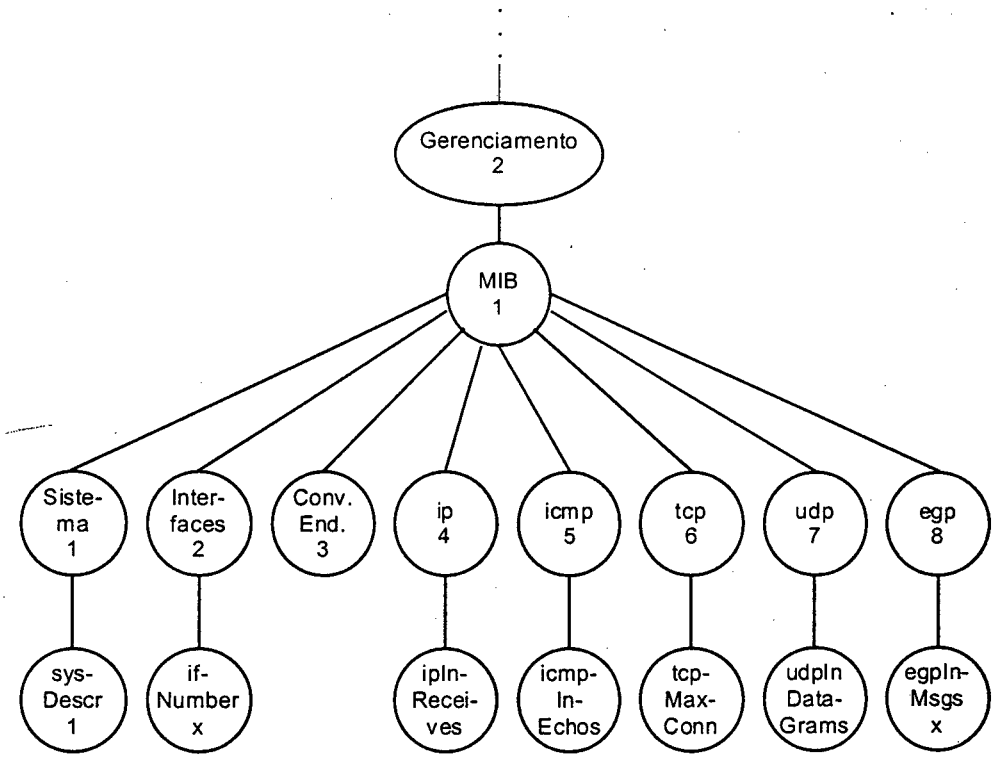


FIGURA 8 - Categorias da MIB – Espaço do Nome Identificador do Objeto.

Como visto nas figuras anteriores, as variáveis SNMP são constituídas pela concatenação do identificador do objeto gerenciado (OID) e o identificador de uma instância deste objeto. Um objeto gerenciado é formalmente descrito pela linguagem ASN.1 (*Abstract Syntax Notation One*), que define o tipo de dado (inteiro, *string*, estrutura, etc.), as restrições de acesso, a localização do objeto dentro de uma MIB, entre outras informações. Um identificador de um objeto gerenciado é obtido pela concatenação dos identificadores de cada sub-ramo da árvore da raiz, até alcançar o objeto desejado. Então, a identificação do objeto *sysDescr* na representação literal seria:

iso.org.dod.internet.gerenciamento.mib.sistema.sysDescr

e na representação numérica:

1.3.6.1.2.1.1.1

A linguagem ASN.1 é usada tanto para representação de variáveis como para a representação e codificação das mensagens SNMP.

2.6.2.3. FORMATO DE MENSAGENS SNMP

As mensagens do SNMP usam o padrão de codificação ASN.1. A mensagem consiste em três partes principais, conforme mostra a figura 9. A primeira parte informa a versão do protocolo, a segunda um identificador de comunidade do SNMP e a terceira uma área de dados.

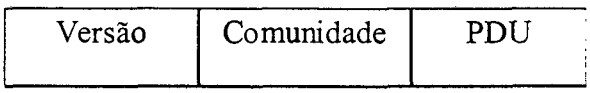


FIGURA 9 – Formato de Mensagem do SNMP.

A área de dados é dividida em *unidades de dados do protocolo (PDU's – Protocol Data Unit)*, como pode ser visto na figura 10. Cada PDU consiste em uma solicitação (enviada pelo cliente) ou uma resposta (enviada pelo servidor). O campo PDU pode conter qualquer um dos cinco tipos de operações utilizadas pelo SNMP (SNMP PDU). As PDU's GetRequest PDU, GetNextRequest PDU e SetRequest PDU

são definidas no SNMP com o mesmo formato da GetResponse PDU com os campos *error-status e error-index* com o valor zero.

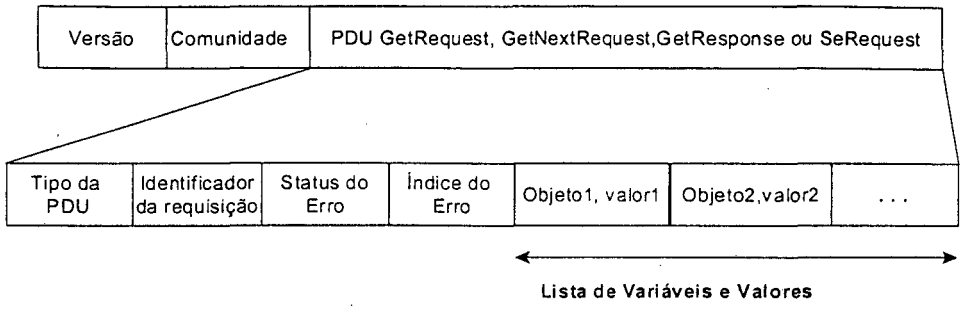


FIGURA 10 – Formato das PDUs GetRequest, GetNextRequest, GetResponse e SetRequest.

Para suprir as deficiências apresentadas pelo SNMP original, uma nova versão foi apresentada, chamada de SNMP versão 2 ou SNMPv2, que provê mecanismos de recuperação de grande quantidade de informações, de gerenciar a gerência e também proporcionar um nível de segurança maior [COM99] [SUA99].

3. REDES SEM FIO (*WIRELESS NETWORKS*)

Neste capítulo é apresentada a tecnologia de comunicação de dados sem fio para a interligação de redes de computadores. O meio físico de transmissão sem fio apresenta diferentes tipos de tecnologias, configurações e benefícios, podendo substituir e coexistir com outros os meios de comunicação. Nos últimos anos os avanços da comunicação, possibilitaram o surgimento de várias tecnologias, que desde então, vem atendendo as necessidades reais daqueles que necessitam os meios de comunicação, com a maior qualidade possível. Desde o início dos sistemas computacionais até os sofisticados e atuais sistemas de computadores, muitas evoluções aconteceram, dentre elas a comunicação sem fio (*wireless data communications*).

3.1. O QUE É UMA REDE SEM FIO

As redes sem fio, por meio da radiodifusão ou raios infravermelhos, são uma alternativa viável para lugares onde é difícil, ou mesmo impossível instalar cabos metálicos e/ou de fibra ótica, podendo substituir ou ampliar as redes implementadas com cabos já existentes. Nas ligações entre redes locais ou metropolitanas, o uso da transmissão sem fio tem papel relevante, principalmente quando as redes estão distantes e o tráfego inter-rede é elevado [SOA97]. Neste caso, o meio privado de comunicação (sistema telefônico) pode ser inadequado e a radiodifusão pode fornecer uma largura de banda mais adequada.

Ao contrário das transmissões por meios de cabos (meios guiados), seja eles pares trançado, coaxiais ou fibra ótica, a transmissão sem fio (meio não guiado) tem como principal característica o uso do ar para transmitir dados, podendo ser por meio de raios infravermelhos, laser, microondas ou rádio-freqüência (RF).

As redes sem fio podem suportar velocidades de acima de 11 Mbps [LUC99], ultrapassando distâncias de 16 km, usando protocolos de segurança, além de permitirem

a conexão com as redes de outras tecnologias e poderem ser independentes de modo a formar uma rede totalmente sem fio.

Esta tecnologia pode ser usada tanto para ambientes externos (*outdoor*) como para internos (*indoor*). Em ambientes externos e abertos podem ser utilizadas para interligar empresas, condomínios, residências entre outros (um prédio a outro). Já em ambiente internos ou fechados elas podem ser utilizadas para a comunicação com prédios de escritórios, hospitais, universidades, fábricas, armazéns e outros (um setor à outro).

Para o desenvolvimento deste trabalho será levado em consideração o meio de transmissão através de rádio-freqüência *outdoor*, o qual pode ser utilizado para a implementação de redes LAN ou MAN.

3.2. MOTIVAÇÕES PARA O USO DE REDES SEM FIO

De modo geral, as redes sem fio *outdoor* e *indoor* apresentam algumas considerações que favorecem o seu uso e o crescimento do número de usuários dessa tecnologia comparado com as tecnologias cabeadas, entre as quais destacam-se [WLA00]:

- *Alcance e Cobertura*: As distâncias que as ondas de rádios podem se propagar devem-se ao projeto do produto, incluindo a potência do transmissor e o tipo do receptor, e o caminho em que as ondas se propagam, especialmente em recintos fechados (*indoor*). Interações com objetos como construções, incluindo paredes, metais, e até mesmo pessoas, podem afetar a propagação, e assim, será definido o alcance e a cobertura que um sinal de um sistema particular alcança.

- *Vazão*: Assim como em redes implementadas através de cabos, a vazão (*throughput*[2]) depende de como a rede está configurada e o tipo do produto

[2] *Throughput*: Resultados que um computador pode produzir durante um certo tempo.

que está sendo utilizado. O número de usuários, fatores de propagação tais como o alcance e o tipo do equipamento e do sistema utilizado, são fatores que podem afetar a vazão da rede. A taxa de transmissão, dependendo da configuração da rede e do tipo do equipamento utilizado, fica entre 1 e 11 Mbps.

- *Integridade e confiabilidade*: a tecnologia de transmissão de dados sem fio tem sido provada há mais de 50 anos em sistemas comerciais e militares. Projetos robustos sob a tecnologia sem fio, mostram que este sistema pode ser mais robusto que sistemas telefônicos (como o celular) e fornecem um desempenho de integridade de dados igual ou melhor que sistemas de rede implementadas com cabos.

- *Interoperabilidade com Infra-estrutura Implementada Através de Cabos*: a maioria dos sistemas de redes sem fio fornecem a interconexão com padrões industriais de redes cabeadas, incluindo o padrão *Ethernet* (802.3) e o *Token Ring* (802.5). A interoperabilidade baseada em padrões permite que redes constituídas por cabos e sem fio possam ser interconectadas de forma transparente para quem vai usá-las.

- *Interoperabilidade com Infra-estrutura Sem Fio*: existem vários tipos de interoperabilidade possíveis entre redes sem fio. Tudo depende da tecnologia escolhida e da implementação do sistema. Produtos de fabricantes diferentes que empregam a mesma tecnologia e a mesma implementação permitem a interconexão de adaptadores e pontos de acesso. A meta da padronização industrial, tal como a especificação IEEE 802.11, é permitir a conformidade dos produtos para interoperar sem a colaboração explícita entre os fabricantes.

- *Interferência e Coexistência*: as redes sem fio baseadas em ondas de rádio podem sofrer interferência de outros produtos que transmitem energia no mesmo espectro de freqüência.

- *Simplicidade/Facilidade de Usar*: aos novos usuários desta tecnologia não é necessário muita informação nova para tirar proveito da rede, já que, o modo de transmissão de dados é transparente aos usuários. Geralmente os produtos para redes sem fio incorporam uma variedade de ferramentas para diagnosticar e coletar informações sobre os elementos sem fios do sistema. O processo de instalação e configuração das redes sem fio é mais simples quando comparado aos sistemas instalados com cabos para os administradores de rede [WLA00]. O único sistema que requer a instalação de cabos é o ponto de acesso, livrando o administrador de instalar cabos até os equipamentos da rede. Em uma rede implementada por meio de cabos torna-se mais complicado o processo de reconfiguração da rede, enquanto que, no sistema sem fio, a reconfiguração pode ser feita de forma mais simples e rápida.

- *Segurança*: devido ao fato da tecnologia sem fios ter suas raízes em aplicações militares, a segurança foi por muito tempo um critério relevante no projeto de dispositivos sem fios. É extremamente difícil que receptores intencionais venham interceptar o sistema sem fio. Técnicas de criptografia complexas dificultam o acesso indevido [WLA00].

- *Escalabilidade*: Redes sem fio podem ser projetadas para ser extremamente simples ou bastante complexas. Elas podem suportar uma grande quantidade de nodos e permitir o aumento da área física adicionando pontos de acesso para impulsionar ou estender a cobertura.

3.2.1. APLICAÇÕES DAS REDES SEM FIO

As redes sem fio fornecem toda a flexibilidade e a resposta rápida necessárias para atender às modificações de uma rede. Os produtos sem fio atendem a uma vasta gama de aplicações, sendo especialmente úteis para:

- Estender cabeamentos de rede local;

- Ambientes difíceis de instalar cabos;
- Ambientes em constante mudança;
- Redes locais pré-instaladas prontas para uso;
- Acesso a rede para computadores móveis;
- Redes temporárias para atender sobrecarga de trabalho;
- Substituição de linhas privativas de alta velocidade (conexão ponto-a-ponto).

3.3. A TECNOLOGIA DE TRANSMISSÃO SEM FIO

As redes sem fio usam ondas eletromagnéticas, de modo a permitir que dois ou mais equipamentos transmitam suas informações de um ponto a outro sem depender de qualquer conexão física, a não ser o ar.

Para efetuar a transmissão de informações sem fio, são utilizados o rádio, a microonda e o raio infravermelho. Para que isso possa ocorrer é necessário que a amplitude seja modulada, a freqüência ou a fase das ondas. Isto para que a mensagem possa ser reconhecida pelo transmissor e pelo receptor. Ainda é possível o uso da luz ultravioleta, o raio X e o raio gama, mas ambos apresentam alguns problemas: são difíceis de modular, não se propagam através de prédios e ainda, são perigosos aos seres vivos [LIM99].

Para o desenvolvimento desta dissertação o meio utilizado e avaliado será o rádio, devido as características apresentadas na seção 3.2.1.

3.3.1. ONDAS DE RÁDIO

As ondas de rádio são freqüentemente chamadas de portadoras de rádio, pois elas executam a função de entregar energia para um receptor remoto. O dados transmitidos são sobrepostos no portador de rádio de tal forma que possam ser extraídos

com precisão no receptor. Isso é chamado de modulação do portador pela informação que será transmitida. Uma vez que os dados são sobrepostos (modulação) sobre o portador de rádio, o sinal de rádio ocupa mais que uma única freqüência, desde que a freqüência ou taxa de bits da informação modulada aumenta o portador.

Se as ondas de rádio são transmitidas em freqüências de rádio diferentes, várias portadoras de rádio podem existir no mesmo local e ao mesmo tempo sem que um interfira no outro. Para extrair os dados, o receptor deve sintonizar ou selecionar a freqüência de rádio adequada, rejeitando todos os outros sinais de freqüências diferentes.

As ondas de rádio são produzidas através de circuitos eletrônicos, logo, elas percorrem longas distâncias podendo entrar facilmente em prédios, como acontece, por exemplo, quando se escuta uma rádio AM ou FM [LIM99].

As ondas de rádio apresentam algumas características, dentre elas podemos destacar [SOA97]:

- Utiliza-se freqüências altas – UHF e VHF – para diminuir a interferência devido à sua maior velocidade;
- As ondas de rádio são unidirecionais, permitindo que elas percorram todas as direções a partir da origem; por tanto, o transmissor e o receptor não precisam estar alinhados, isto é trabalhar na mesma freqüência;
- Devido à capacidade que as ondas de rádio tem para percorrer longas distâncias, a interferência entre os usuários é um problema, por essa razão todos os governos exercem um rígido controle sobre as faixas de freqüência utilizadas nos transmissores de rádios;
- Existem requisitos que devem ser respeitados para que a transmissão tenha êxito, tais como, potência de transmissão e mínima distorção da propagação do sinal.

3.3.2. PADRÃO PARA TRANSMISSÃO SEM FIO

O padrão para transmissão sem fio foi publicado no segundo semestre de 1997 pelo IEEE, o qual elaborou padrões para redes sem fio locais e metropolitanas. O IEEE definiu o grupo de trabalho 802.11, tendo como objetivo, definir uma especificação para conectividade sem fio entre estações de uma área local e/ou metropolitana [LIM99].

Esta padronização especifica a subcamada de controle de acesso ao meio (MAC – *Médium Access Control*) comum a todas as camadas físicas, e garante uma interoperabilidade entre os diversos fornecedores da tecnologia, ou seja, se todos os equipamentos de uma rede sem fio forem padronizadas conforme o IEEE 802.11, independente de marca ou modelo, eles poderão trocar dados entre si.

As principais características deste padrão e do seu uso são [TAN99]:

- Tecnologia de acesso ao meio CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*).
- WEP (*Wired Equivalent Privacy*) – chave de criptografia disponível em alguns cartões.
- Atualmente suporta taxas de transmissão entre 2 e 11 Mbps.
- 13 canais de transmissão.
- Usa os protocolo TCP/IP.

As técnicas de transmissão por ondas de rádio e raios infravermelhos, considerados para transmissão de dados sem fio, e definidos na camada físicos são: Espalhamento do Espectro por Saltos de Freqüência (FHSS – *Frequency Hopping Spread Spectrum*), Espalhamento do Espectro por Seqüência Direta (DSSS – *Direct Sequency Spread Spectrum*) e Infravermelho.

No caso da tecnologia de rádio, o padrão IEEE 802.11, baseia-se na utilização da faixa denominada ISM (*Industrial, Scientific and Medical*) a qual é alocada para a emissão eletromagnética de diversos tipos de aparelhos eletrônicos. Por esta razão, para

a transmissão nestas faixas não é exigido nenhum tipo de licença. Para operar nela é necessário utilizar técnicas de espalhamento espectral, que será detalhada na seção 3.2.3 [LIM99].

3.3.3. TÉCNICAS DO ESPALHAMENTO ESPECTRAL

Existem vários métodos de transmitir informações via sinais de rádio, e cada um deles possui suas próprias características, vantagens e desvantagens. O método por amplitude modulada (AM – *Amplitude Modulation*) é um exemplo muito usado e antigo.

Como cada método apresenta suas próprias características, deve-se escolher aquele que melhor condiz um determinado tipo de aplicação. Por exemplo, a modulação AM é simples e tanto o transmissor como o receptor AM tem um custo muito baixo; porém sofre muito com qualquer ruído e não transmite bem sinais de alta freqüência. Por outro lado, as rádios FM (*Frequency Modulation*) são um pouco mais sofisticadas e apresentam uma imunidade maior a ruído, transmitindo com um grau maior de fidelidade sinais de alta freqüência.

Será tratado nesta seção o espectro eletromagnético. A maioria das transmissões utiliza uma banda de freqüência estreita para obter a melhor recepção. No entanto, em alguns casos, o transmissor pula de freqüência em freqüência em um padrão regular, ou as transmissões são intencionalmente dispersadas através de uma banda de freqüência larga. Essa técnica é chamada de **espectro de dispersão – *spread spectrum*** [TAN97].

O espectro de dispersão é um outro método de transmissão de dados via rádio que surgiu a partir de uma necessidade militar: como transmitir informações via rádio sem que o inimigo possa interceptar a mensagem ou interferir na comunicação. Desse trabalho, que começou na II Guerra Mundial, surgiu um método de modulação muito sofisticado, apresentando as seguintes características:

- Alta imunidade a ruídos;

- Baixa interferência de outros sistemas;
- Difícil de ser interceptado ou monitorado – seguro.

A modulação espectro de dispersão nos dias atuais é utilizada em muitos sistemas, principalmente onde se exige confiabilidade. Por exemplo, sistemas de satélites, telefonia celular, sistema de localização global via satélite, transmissão de dados e outros. Além disso, devido a suas características, para os sistemas de rádios com espectro de dispersão foram reservadas algumas faixas de freqüência, as quais podem ser usadas sem a necessidade de registros. Dentre estas estão:

- 902 Mhz até 928 Mhz (914 Mhz);
- 2400 Mhz até 2483,5 Mhz (2,4 Ghz);
- 5752,5 Mhz até 5850 Mhz (5,8 Ghz).

A modulação espectro de dispersão via rádio pode ser feita de duas maneiras básicas. Uma delas chama-se Espalhamento do Espectro por Saltos de Freqüência (*Frequency Hopping Spread Spectrum* – FHSS) e a outra, Espalhamento do Espectro por Seqüência Direta (*Direct Sequence Spread Spectrum* – DSSS), descritas a seguir.

3.3.3.1. *FREQUENCY HOPPING SPREAD SPECTRUM* – FHSS

A modulação FHSS funciona através da troca de freqüência de trabalho de tempos em tempos. Uma rádio FM, por exemplo, trabalha transmitindo na mesma freqüência todo o tempo, já um sistema FHSS transmite um curto período de tempo e depois muda para outra freqüência próxima por outro período e assim por diante.

Dessa forma, se existir uma fonte de ruído em uma freqüência, a transmissão de dados só será afetada por um curto período de tempo. Se existe outro sistema transmitindo na mesma área ele só será afetado por um curto período.

Se na mesma área houver mais de um sistema FHSS trabalhando, a comunicação será pouco afetada, pois como ambos sistemas vão estar pulando de maneira diferente eles vão utilizar as mesmas freqüências apenas algumas vezes, o que irá garantir a continuidade da comunicação.

A única desvantagem do FHSS quando comparado com o DSSS é que ele perde algum tempo pulando de uma freqüência para outro, assim, este sistema, tem uma taxa de transferência um pouco menor. Dessa forma, os sistemas FHSS são ideais para transmissão de dados em ambientes abertos, onde existem ruídos e outros sistemas FHSS que podem entrar em operação na mesma área. Este tipo de sistema pode ser bem aplicado na implementação de redes metropolitanas, e pode ser chamado de transmissão *outdoor*.

3.3.3.2. *DIRECT SEQUENCE SPREAD SPECTRUM* – DSSS

Ao contrário do FHSS, o DSSS transmite em uma faixa de freqüência fixa, porém muito larga e distribuída por igual. Ao contrário, uma rádio FM, por exemplo, transmite a maior parte do seu sinal em uma única freqüência, assim, se um rádio receptor FM estiver um pouco fora de sintonia o sinal já se torna fraco e ruidoso. No caso do DSSS o sinal não é muito forte em nenhuma freqüência específica, ele tem a mesma força por uma grande faixa de freqüências.

Assim sendo, uma interferência em uma certa freqüência irá afetar apenas uma parte do sinal, sobrando ainda os sinais transmitidos em outras freqüências próximas. Além disso, se houver algum outro sistema de rádio utilizando uma freqüência dentro da faixa de freqüências utilizadas pelo DSSS ele será pouco afetado, pois o sinal gerado pelo DSSS é fraco.

O DSSS tem uma capacidade de transmissão muito maior que o FHSS, porém se dois sistemas idênticos DSSS forem instalados na mesma área (e operarem na mesma faixa de freqüências) eles irão se interferir muito fortemente e acabar afetando a troca de informações.

Por isso, os sistemas DSSS são ideais para operação dentro de escritórios, lojas, hotéis e qualquer outro lugar onde a existência de paredes atenue as interferências externas, o que pode ser chamado de transmissão *indoor*.

3.4. O PROVIMENTO DE SERVIÇOS INTERNET USANDO REDES SEM FIO– OUTDOOR

O método tradicional de provimento de serviços Internet se dá através do uso do sistema privado de telecomunicações, onde o usuário doméstico, empresa, indústria ou uma organização necessita de um provedor de acesso ou, alternativamente adquire uma linha telefônica dedicada para acessar a rede.

Quem usa o sistema privado de telecomunicações, geralmente fica dependente de baixas taxas de transmissão, devido a largura de banda (capacidade de transmissão) desses sistemas serem baixas. Consegue-se um diferencial significativo, quem adquire uma linha dedicada da empresa de telecomunicações, porém se paga mais caro por isso. Por outro lado, quem faz o acesso através de um provedor tem taxas de transmissão bem menores. O uso deste último, necessita o emprego de uma linha telefônica, que geralmente tem tarifas elevadas pelo seu uso, e as vezes, é difícil o acesso, já que, todas as linhas disponíveis no provedor podem estar ocupadas.

A exigência por parte dos usuários da rede de serviços precisos, rápidos e confiáveis nem sempre é atingida. Problemas podem acabar causando a impaciência do usuário (cliente) e o desinteresse pelo uso da rede. A tecnologia de rede sem fio aplicada ao provimento de serviços da Internet tem como principal objetivo superar esses desafios, e causar a satisfação para quem usa a rede.

Para que possa existir o provimento de acesso a Internet, o único que deverá usar o sistema privado de telecomunicação é o provedor de acesso. O usuário fará uso do ar como meio físico de transmissão, onde a comunicação ocorre através de antenas, as quais transferem e recebem informação por freqüência de rádio, conforme foi explicado na seção anterior.

A topologia de uma rede sem fio para provimento de acesso à Internet é mostrada na figura 11. Ela consiste de um provedor de acesso, o qual, como dito anteriormente, necessita de um enlace com o sistema privado de telecomunicação conectado a um roteador, o que permite a integração de redes locais à Internet através de uma ISDN (*Integrated Service Digital Network*). O provedor de acesso ainda possui os servidores de serviços da Internet e sistemas de segurança para garantir que usuários indevidos venham a violar o acesso ao provedor e aos dados dos usuários.

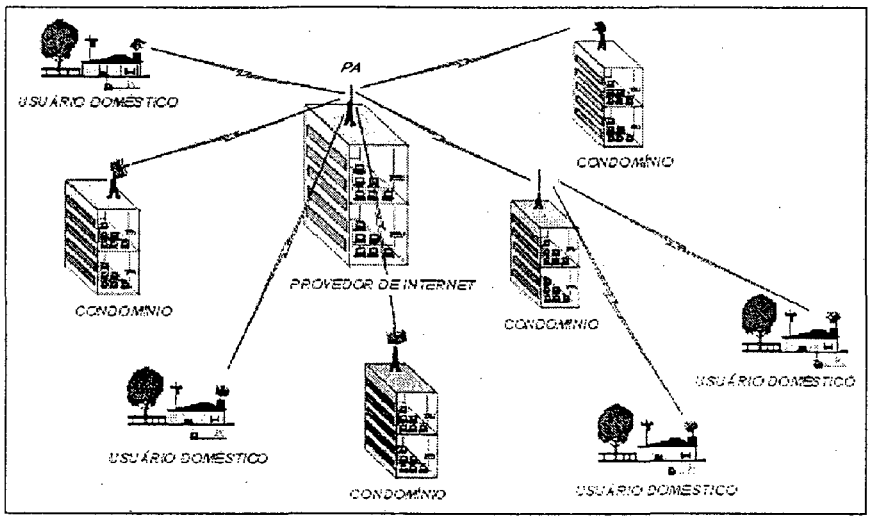


FIGURA 11 – Estrutura de um provedor de acesso à Internet sem fio [COL00].

Na figura 11, como pode ser observado, é possível identificar alguns condomínios e usuários domésticos ligados ao provedor de acesso através de antenas. Cada prédio ou casa consiste de uma *LAN*, conectada a um servidor, usando, geralmente, o padrão *Ethernet*. Assim, esta topologia é integrada, ou seja, redes *Ethernet* tradicionais e sem fio podem se comunicar. Para esta integração existem pontos de acesso – PA (*Access Points – AP*), que servem como pontes (*bridges*) para a integração das tecnologias de transmissão implementadas por meio de cabos e tecnologias sem fio. Estas pontes podem ser monitoradas através de funções de gerenciamento, usando o protocolo SNMP, considerando que elas possuem uma MIB. Desta forma, é possível detectar problemas oriundos da comunicação, e tratá-los através de um software de gerência.

No caso da figura 11, acontece uma interligação direta do ponto de acesso principal aos clientes (provedor de acesso aos condomínios e usuários domésticos), sem a ocorrência de obstáculos. No entanto existem regiões geográficas que apresentam uma série de obstáculos naturais, ou até mesmo barreiras como prédios. Como a transmissão *outdoor* acontece de forma visada (direcionada – antena à antena), é necessário a instalação de repetidores, que possibilitam quebrar as barreiras como obstáculos e até mesmo permitir a transmissão de dados em distâncias maiores. Um exemplo de rede sem fio com obstáculos é mostrado na figura 12.

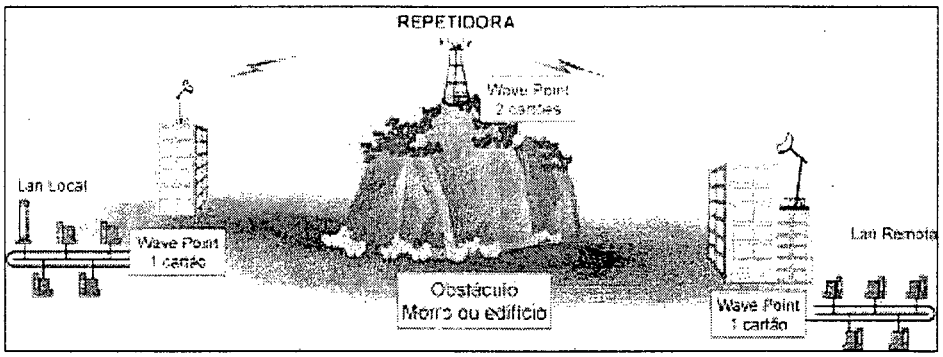


FIGURA 12 – Antena repetidora para comunicação entre prédios com obstáculos [COL00].

Para o cliente final, o uso da tecnologia sem fio tem seus benefícios. Entre eles destacam-se os fatos de que as taxas de transmissão são mais altas se comparadas ao sistema privado de telecomunicação, e o custo de uso é razoavelmente mais baixo. Por outro lado, esse tipo de rede apresenta problemas de segurança, pois equipamentos que operam na mesma freqüência podem captar as informações, problema este, que se resolve com o uso de criptografia.

3.5. CONCLUSÃO

De modo geral, a tecnologia sem fio vem sendo muito utilizada, coexistindo entre as atuais redes implementadas através de cabos, e os sistemas privados de telecomunicação para acesso a Internet e, também, para interconectar equipamentos onde é difícil se chegar com cabos.

Os produtos referentes a esta tecnologia em geral oferecem a possibilidade de gerenciamento via SNMP. Considerando o crescente uso desta tecnologia, é possível usar as potencialidades do SNMP para controlar e monitorar a rede, de modo a garantir aos usuários segurança e qualidade na transmissão de informações e evitar o seu desagrado quando estiver usando a rede, uma vez que, tudo o que interessa ao usuário, é poder usar a rede em qualquer lugar e em qualquer momento, sem que tenha que solucionar problemas antes de inicializar um equipamento da rede. Assim sendo, para garantir um bom serviço ao usuário da rede, faz-se necessário o desenvolvimento de um aplicativo para o monitoramento e controle da rede.

4. AMBIENTE DE DESENVOLVIMENTO - CENÁRIO

Neste capítulo é apresentado o cenário onde será desenvolvido o trabalho prático desta dissertação. Este cenário faz parte da arquitetura de um Provedor de Serviços Internet (*ISP – Internet Service Provider*), através de rádio freqüência.

O ISCC (Internet - Sistema Catarinense de Comunicação), é um *ISP* situado na cidade de Lages-SC. Além de fornecer serviços Internet via MODEM, agora disponibiliza, também, seus serviços através de ondas de rádio, o que é ponto relevante para o desenvolvimento deste trabalho.

Junto com os seus clientes, o ISCC forma uma rede metropolitana (MAN) sem fio. Ela fornece serviços 24 horas ininterruptas por dia, desde que não haja falhas em seus equipamentos e pontos de acesso, sem cobrar taxa adicional por tempo de uso.

Para o provimento de serviços Internet, a empresa conta com os seguintes elementos de rede:

- *Link* dedicado de 512 *Kbytes* com a Telesc (PPP);
- Um roteador da marca Lucent Portmaster II, equipado com 30 (trinta) portas para modem, ligado a uma linha privada (Telesc);
- Um servidor Unix Free BSD configurado com *firewall*, em uma máquina Pentium III 750MHz, 128 MB de memória RAM e 3 placas de rede 10/100 Mbps;
- Um servidor de páginas Web, configurado sob o Sistema Operacional Windows NT Server 4.0, instalado em uma máquina Pentium III 750MHz, também com 128 MB de RAM;
- Uma máquina Pentium III 600 MHz, configurada como servidor de E-mail, rodando sob o Sistema Operacional Windows NT, IIS 4.0;

- Um *Hub* de 16 portas 3Com Home Connect com taxas de 10/100 Mbps; 60 *modems* do tipo Port Master 3;

- Um sistema RAS (*Remote Access Server*), com *modem* ligado via HDSL a uma linha telefônica;

- Pontos de acesso WaveLAN (adaptadores PCI ou ISA) da família Orinoco [LUC99];

- Um ponto de acesso WavePOINT-II da família Orinoco com dois *slots* PCMCIA para cartões de rádio e uma porta Ethernet RJ-45, conectado, a uma antena Omnidirecional;

- Antenas parabólicas (do tipo vazada) instaladas nos condomínios e empresas para os quais fornece o serviço; conectadas à cartões adaptadores[3] *PCI* ou *ISA*, para transmissão via rádio, instaladas em servidores Linux, sobre máquinas Pentium 200MHz, 64 MB de RAM;

- Para a interligação interna dos elementos de rede nos condomínios, empresas e ISP é usado uma estrutura de rede *Ethernet*.

4.1. INTERLIGAÇÃO DOS ELEMENTOS DE REDE E PONTOS DE ACESSO

A maneira como os elementos de rede internos ao ISP estão interligados é mostrado, na figura 13. O acesso a Internet se dá através de um *link* dedicado com a TELESC de 512 Kbytes (PPP). Este *link* é feito através de um roteador (*Router* – Lucent) com 30 portas para modem *Port Master*. Este equipamento está conectado a um Hub (10/100 Mbps) que dá acesso aos servidores de página WWW, *E-mail*, *Firewall* entre outros.

[3] Adaptadores PCI ou ISA: são cartões genéricos que convertem um slot ISA ou PCI de um PC em um slot PCMCIA, permitindo assim que seja utilizado o cartão WaveLAN PCMCIA.

O *firewall* está instalado em um equipamento com sistema operacional *Free BSD* e, possui 3 (três) placas de rede (10/100 Mbps), das quais, uma é usada para o acesso aos demais servidores (através de um *Hub*), e outra para conectar o *Hub* (10/100 Mbps) da rede interna do ISCC (setor Contábil, Rádio Clube, Financeiro e Diretoria).

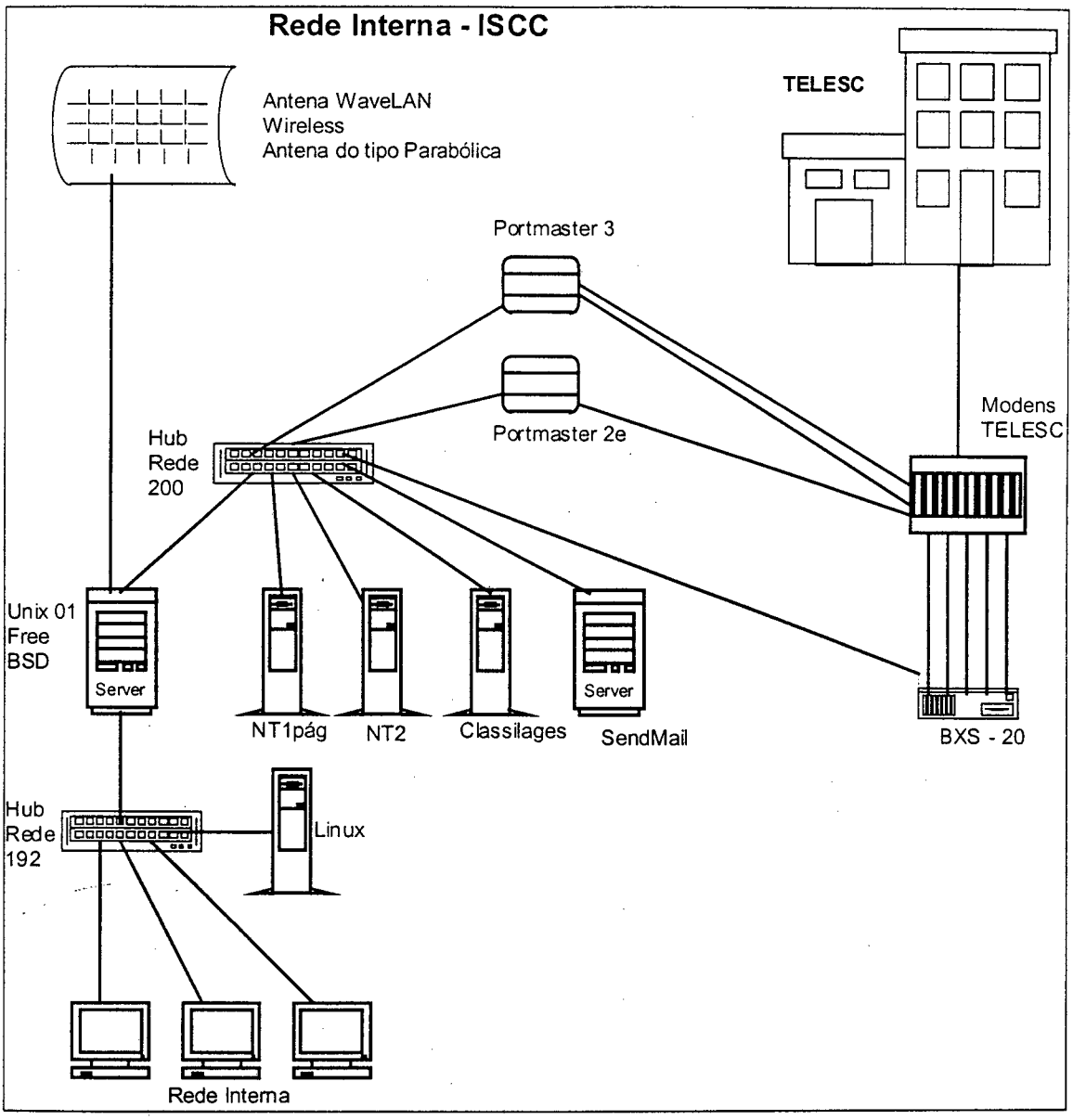


FIGURA 13 – Configuração interna do ISP.

Neste equipamento ainda, foi adaptado um cartão PCI com suporte para cartão PC *Card's* do tipo *WaveLAN* da ORINOCO para ponto de acesso sem fio. É este mesmo ponto de acesso que permite a formação da rede MAN sem fio pelo ISCC.

O adaptador WaveLAN está conectado a uma antena parabólica (transmissão ponto a ponto). Esta por sua vez, direcionada para uma antena do tipo Omnidirecional (multiponto), instalada numa torre de telecomunicações da própria empresa (situada na Cidade Alta – Bairro de Lages localizado num morro), conforme mostra a figura 14. A antena Omni, conectada ao WavePOINT-II serve de ponto de acesso para o restante da rede MAN, ou seja, condomínios e empresas que usam os serviços fornecidos pelo ISCC.

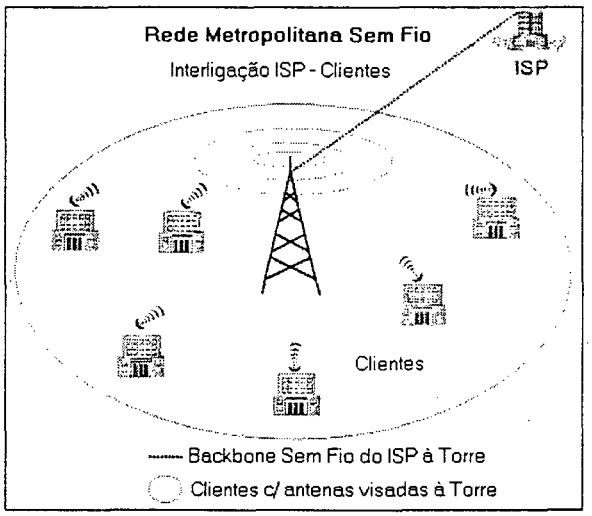


FIGURA 14 – Estrutura da MAN do ISP [WAV99].

A estrutura básica para condomínios e empresas, é composta por uma antena do tipo parabólica, que é direcionada à torre e conectada ao adaptador *WaveLan*. Este está adaptado em uma máquina servidora, que permitirá uma ligação *Ethernet* entre os elementos de rede internos (computadores pessoais e *hubs*), formando uma rede LAN. Esta máquina também tem as funções de NAT (*Network Address Tranlation*) e de DHCP (*Dynamic Host Control Protocol*).

Atualmente o ISCC fornece serviços através de rádio freqüência para 6 (seis condomínios e 4 (quatro) empresas privadas. A capacidade máxima de transmissão do

WavePOINT da Orinoco é de 11 Mbps, mas para conexão com vários enlaces WaveLAN, atinge-se taxas de 2Mbps por enlace [WAV99].

Considerando-se o sistema descrito acima, foi desenvolvida a ferramenta para o monitoramento e o controle da rede sem fio. Foram levadas em consideração as demandas de banda, taxas de ruído, e tentativas de acesso ao meio, visto que, o número de usuários cresce a cada momento, despontando, assim, cada vez mais, a necessidade de serviços que apresentam bom desempenho, eficiência e segurança.

4.2. ELEMENTOS GERENCIÁVEIS

No estudo realizado no cenário, pode-se destacar alguns elementos de redes passíveis à gerência através do protocolo SNMP. No contexto do trabalho, destacam-se o *WavePOINT-II ApManager* e os pontos de acesso com seus respectivos servidores.

Para o monitoramento e controle do WavePOINT-II e os demais elementos de rede, podem ser usados os obejtos gerenciaveis contidos nas MIBs, conforme descrição apresentadas nas RFCs de domínio público RFC1213, RFC1398, RFC1493 e, as proprietárias, *Kbridge-mib* (ver ANEXO 1), WaveLAN e WaveNET [LUC99].

Em particular, as MIBs WaveLAN e WaveNET tratam de objetos gerenciaveis para elementos de rede sem fio. Porém, os agentes implementados para os equipamentos da família Orinoco, mais precisamente, o ApManager WavePOINT-II, não registram seus dados nessas MIBs [APS99]. Para a implementação do trabalho, serão utilizadas variáveis implementadas na *Kbridge-mib*, e algumas apresentadas na RFC1213.

As variáveis descritas na *Kbridge-mib* não são documentadas publicamente, e por isso não está disponível. O documento apresentado no anexo foi cedido gentilmente por responsáveis da empresa *KarlNet Incorporated*, conforme contato mantido através do e-mail *info@karlnet.com*.

No capítulo 5 serão analisadas e apresentadas algumas variáveis, usadas para a implementação de uma applet, a qual, apresentará dados que permitirão o monitoramento e o controle do ISP sem fio.

5. IMPLEMENTAÇÃO DE APPLETS PARA MONITORAMENTO DO ISP

Neste capítulo é apresentado o processo de desenvolvimento de um sistema para monitoramento de redes sem fio, aplicadas ao provimento de serviços Internet. Inicialmente é feito um levantamento de dados (seção 5.1), referente as variáveis (objetos gerenciados) contidas nas MIBs (RFC1213, *Kbridge-mib*). São priorizadas aquelas que fornecem informações sobre as condições de tráfego, erros de transmissão, nível de ruídos, nível de desempenho, e outras, bem como o modo de acesso, tipo de dado, quais operações podem ser realizadas sobre elas, e também suas respectivas descrições. Essas são de fundamental importância para executar as operações de gerência, através do protocolo SNMP.

Identificadas as variáveis o passo seguinte foi o estudo da API codificada em Java para executar as operações *Get* e *Get-Next* sobre a MIB.

Com as variáveis e o conhecimento da API, o próximo passo para a implementação foi a codificação da interface do sistema. Para isso é usado o JDK 1.2.2 da *Sun Microsystems*, visto que, além da API SNMP apresentada neste, há uma vasta gama de bibliotecas de classes oferecidas para o gerenciamento SNMP [LOR98][SUN99].

Tendo o conhecimento das variáveis a serem processadas, e do ambiente para desenvolvimento de aplicativos, é possível desenvolver o sistema, ou melhor, fazer a coleta de dados das MIBs, executar o processamento e mostrar os resultados, para isso, uma interface amigável deve ser bem definida, e logo, ser integrada a API SNMP.

Para fazer o uso do sistema e avaliar sua aplicabilidade, deve ser definido o processo de configuração e o modo de acesso, bem como a definição de quem realmente pode usá-lo.

Após a instalação e configuração do sistema, o mesmo deve ser testado e validado, de modo a obter resultados que venham a ser importantes para o desenvolvimento deste trabalho.

Nas próximas seções deste capítulo, serão detalhados os passos relevantes para a implementação das applets. Estes vêm de encontro às necessidades de entendimento para implementação de sistemas de gerenciamento de redes.

5.1. ARQUITETURA DA IMPLEMENTAÇÃO

A figura 15, mostra a arquitetura da implementação do sistema para monitoramento através da Web. Pode-se observar três situações independentes de localização (**Adm1**, **Adm2** e **Adm3**), onde o administrador pode monitorar e obter resultados referentes ao desempenho da rede metropolitana (rede *MAN* do *ISP*).

Na primeira situação – **Adm1** – o administrador pode fazer o monitoramento em um equipamento local, ou seja, no próprio ISP, acessando a página Web com a interface desejada.

Logo, na segunda situação – **Adm2** – o administrador faz o monitoramento através de um *link* sem fio. Neste caso ele pode estar testando a implantação de uma rede sem fio no próprio local onde está sendo feita a instalação, ou até mesmo verificando a qualidade de *link* entre outros *hosts* remotos.

Por fim, na terceira situação – **Adm3** – o administrador da rede pode estar na sua residência, e através de uma conexão discada, ele pode estar verificando a qualidade do *link* entre a estação local e um *host* remoto, quando for chamado para resolver algum problema de um *link* sem fio. Assim, pode fazer uma analise antecipada e diagnosticar algum problema da rede antes de sair de casa para resolvê-lo.

5.2. IDENTIFICAÇÃO DOS OBJETOS GERENCIADOS - MIBS

Todos os elementos de redes passíveis de gerenciamento via SNMP, mantém seus valores agrupados em uma ou mais MIBs. Tradicionalmente, os equipamentos usam pelo menos a MIB descrita e documentada na RFC 1213, também conhecida por

MIB-II. Alguns elementos ainda, possuem MIBs proprietárias, que contém variáveis específicas para determinados tipos de serviços e equipamentos. Com elementos wireless não seria diferente, eles também tem suas MIBs para controle e monitoramento (gerenciamento), as quais podemos chamar de MIBs Wireless. Numa pesquisa inicial, foram encontradas as MIBs proprietárias WaveLAN e WaveNET, ambas documentadas [WNET97][WLAN97]. Após a realização de alguns testes identificou-se que o agente implementado para o elemento gerenciavel (WavePoint-II) não envia seus dados para essas MIBs.

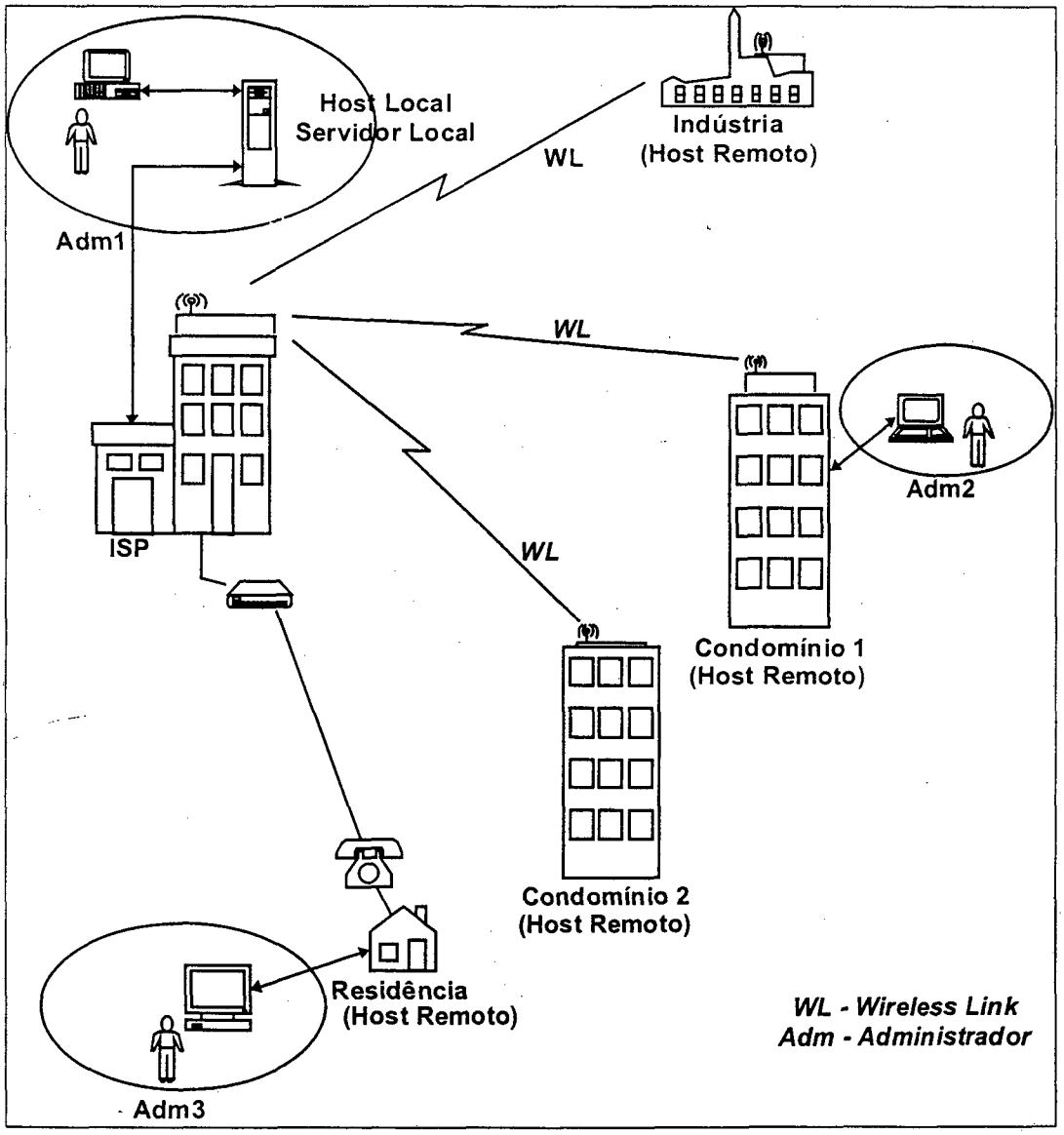


FIGURA 15 – Arquitetura da Implementação

Continuando a pesquisa sobre o elemento gerenciável, foi encontrada uma documentação a respeito da MIB voltada para *wireless*. Esta apresenta somente a identificação do objeto gerenciável (*Object Id – OID*) sem o nome das respectivas variáveis, o que torna difícil o reconhecimento do contexto do *OID*. As variáveis apresentadas por esta documentação são voltadas para teste sobre *link wireless* [MIB00].

A resposta para os nomes das variáveis foram encontradas por meio da *KarlNet Inc*, que disponibilizaram via e-mail um documento contendo os nomes e os respectivos *OID* das variáveis, apresentados como uma publicação inicial. Deste documento foram estudadas e analisadas algumas variáveis importantes para o monitoramento de redes sem fio. A organização hierárquica desta MIB, representada na sua forma literal é a seguinte:

iso.org.dod.internet.mgmt.enterprises.karlnet

e na sua forma numérica:

.1.3.6.1.4.1.762.

As variáveis encontradas nesta MIB possuem características particulares para o tratamento de redes sem fio, mais precisamente de *Wireless Bridges*. O ramo *762* apresenta dois grupos de variáveis, onde o primeiro representa as variáveis para controlar *(1)* o sistema e, o segundo é composto por variáveis para leitura *(2)* do elemento gerenciável. As variáveis para controle e monitoramento são apresentadas nos com maiores detalhes no ANEXO 1.

Baseado nas MIB-II e na *Kbridge-mib* é que o sistema foi implementado. De modo geral, as variáveis apresentam dados referente as estações que estão conectadas à rede sem fio. Assim, é possível monitorar a comunicação entre a ponte e uma estação em particular (AP). Os dados que se destacam em particular, para o desenvolvimento do trabalho, entre cada uma das interligações são aqueles referentes a:

- nome da estação;
- endereço IP;

- endereço físico;
- interface;
- tipo de tecnologia de rádio;
- qualidade do link;
- nível de sinal e de ruído;
- número de pacotes enviados e recebidos;
- número de pacotes perdidos
- fragmentos transmitidos;
- número de falhas e tentativas de conexões;
- número de fragmentos recebidos;
- número de erros FCS;
- número de pacotes recebidos e descartados.

Com esses dados é possível analisar a qualidade do link entre uma estação e a ponte *(WavePOINT-II)*, e ainda identificar os pontos de acessos da MAN, bem como os dados referentes a transmissão de pacotes, podendo-se analisar o número de pacotes enviados com sucesso e os que fracassaram, os que foram recebidos e aqueles que foram perdidos.

Além das variáveis desta MIB ainda são utilizadas algumas variáveis da MIB-II, descritas na RFC1213. Desta MIB pode-se obter informações referentes ao elemento gerenciável, tratado de forma específica. Os dados tratados dizem respeito aos seguintes ramos: *System, ICMP, Interface, IP/ARP, IP, TCP/UDP e SNMP* [STA96]. Com estes é possível analisar e monitorar o elemento gerenciável, de modo a identificar suas características, total de mensagens recebidas e enviadas, número de *octetos* com erro e

descartados (enviados e recebidos), protocolos desconhecidos, velocidade da interface, segmentos enviados e recebidos, datagramas enviados e recebidos, erros de cabeçalho e *timeout.*

Na seção 5.5 algumas das variáveis são discutidas com maiores detalhes.

5.3. A API SNMP

A comunicação entre gerente e agente se dá através da implementação de rotinas que têm o objetivo de enviar e receber dados entre uma estação gerenciável e um gerente, ou seja, executar as operações de gerência, além de apresentar métodos para estabelecer conexões seguras entre agente e gerente, rotinas para encontrar um agente (através de senha e endereço IP). Estas rotinas na sua grande maioria estão implementadas em APIs (Applications Programming Interface), também chamadas de bibliotecas de classes.

Observou-se que a API desenvolvida pela *AdventNet* é uma das mais utilizadas para o desenvolvimento de ambientes de gerência de rede, porém outras ainda existem. A API utilizada neste trabalho, é de código aberto e disponível gratuitamente. Esta foi desenvolvida por Jonathan Sevy [JON00].

5.4. FERRAMENTA PARA DESENVOLVIMENTO

A linguagem de programação Java é multiplataforma, ou seja, é uma ferramenta que gera código passível de ser executado em diferentes sistemas operacionais. Considerando o objetivo do trabalho, esta é utilizada para o desenvolvimento de *applets*, que podem ser apresentadas através de um *browser* por meio da Inernet. Entre as ferramentas Java disponíveis a que será utilizada para a implementação do sistema é o JDK (Java Development Kit) desenvolvido pela SunMicrosystems, visto que é uma

ferramenta padrão e preparada para rodar em qualquer browser Web, com primitivas de segurança nativas.

A API tratada na seção anterior foi desenvolvida em Java, o que facilita então a integração com a API padrão Java, usada para a implementação de interfaces amigáveis e dinâmicas. Estas interfaces podem facilitar o monitoramento (gerenciamento) de rede por meio da Internet.

5.5. DESENVOLVIMENTO DA INTERFACE

Considerando o estudo realizado no ambiente de transmissão sem fio, o elemento *Wireless Bridge (WavePOINT-II)* foi pesquisado, e neste foi identificado o suporte a gerência de redes via SNMP. Este equipamento, é o principal elemento para a comunicação sem fio, o qual, faz a ligação entre a base de transmissão e as estações remotas. Percebeu-se neste equipamento, a necessidade de monitoramento a fim de permitir controlar o seu funcionamento. As interfaces desenvolvidas neste trabalho permitem este monitoramento, através da Web.

As interfaces desenvolvidas neste trabalho estão divididas em 9 (nove) módulos. Cada um deles possui um objetivo específico e são apresentados nas próximas seções.

5.5.1. IDENTIFICAÇÃO DO SISTEMA

Conforme apresentado abaixo na figura 16, neste módulo são apresentadas, de um modo geral, informações sobre o sistema. Estas informações são essenciais ao administrador para a identificação do elemento que está sendo monitorado, ou seja, ele pode obter o conhecimento do equipamento monitorado.

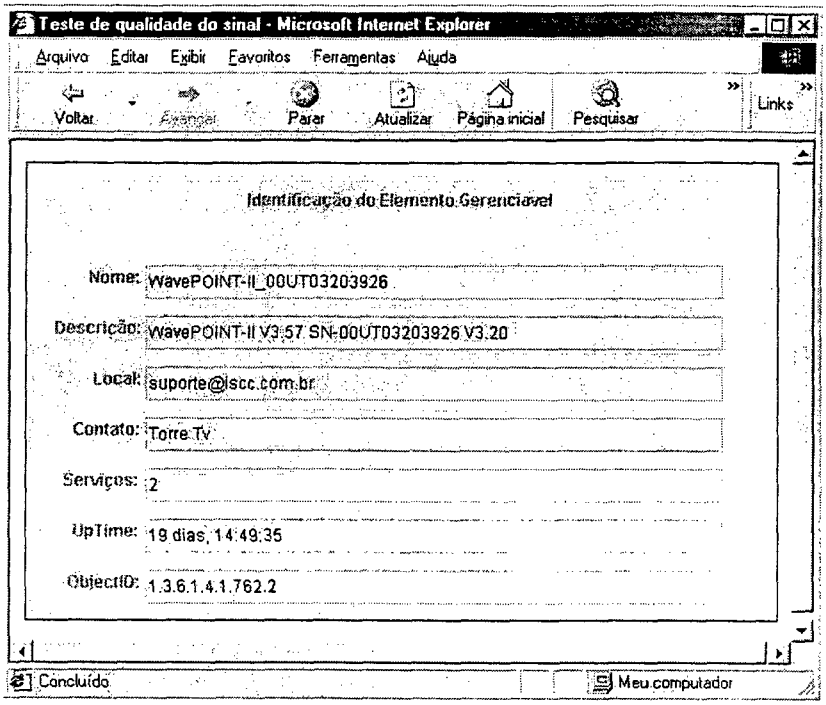


FIGURA 16 – Interface para Identificação do Elemento Gerenciável.

Para apresentar os dados referentes a identificação do sistema, foram tratadas algumas variáveis do grupo *System* da MIB II do agente monitorado: *SysName, SysDescr, SysLocal, SysContact, SysServices, SysUpTime* e *SysObjectID* [STA96].

5.5.2. DADOS DAS *INTERFACES*

A figura 17, mostra a interface do módulo "Dados sobre Interface", que apresenta ao administrador as informações sobre as interfaces da entidade gerenciada. Como pode-se observar, são listadas as interfaces identificadas e quando uma delas é selecionada em particular, são apresentadas diversas informações coletadas sobre a interface. Elas estão descritas na MIB II do agente monitorado [RFC1213].

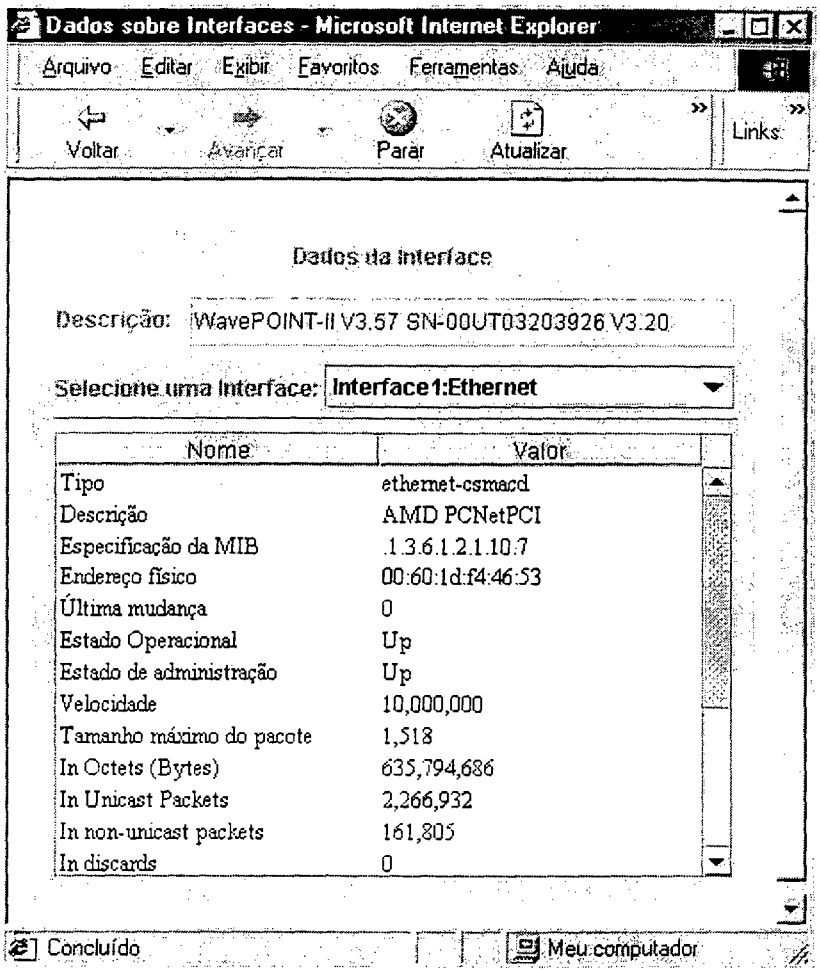


FIGURA 17 – Dados das Interfaces do PA.

5.5.3. DADOS SNMP

O módulo "Dados SNMP", mostrado na figura 18, tem como objetivo apresentar ao administrador as informações estatísticas sobre as operações de gerência SNMP (tráfego SNMP) entre agente e gerente, que ocorreram no equipamento gerenciado. Conforme pode-se observar na figura, algumas variáveis são utilizadas para contabilizar o número de operações. As variáveis apresentadas nesta interface são referente ao grupo SNMP da MIB II [STA96].

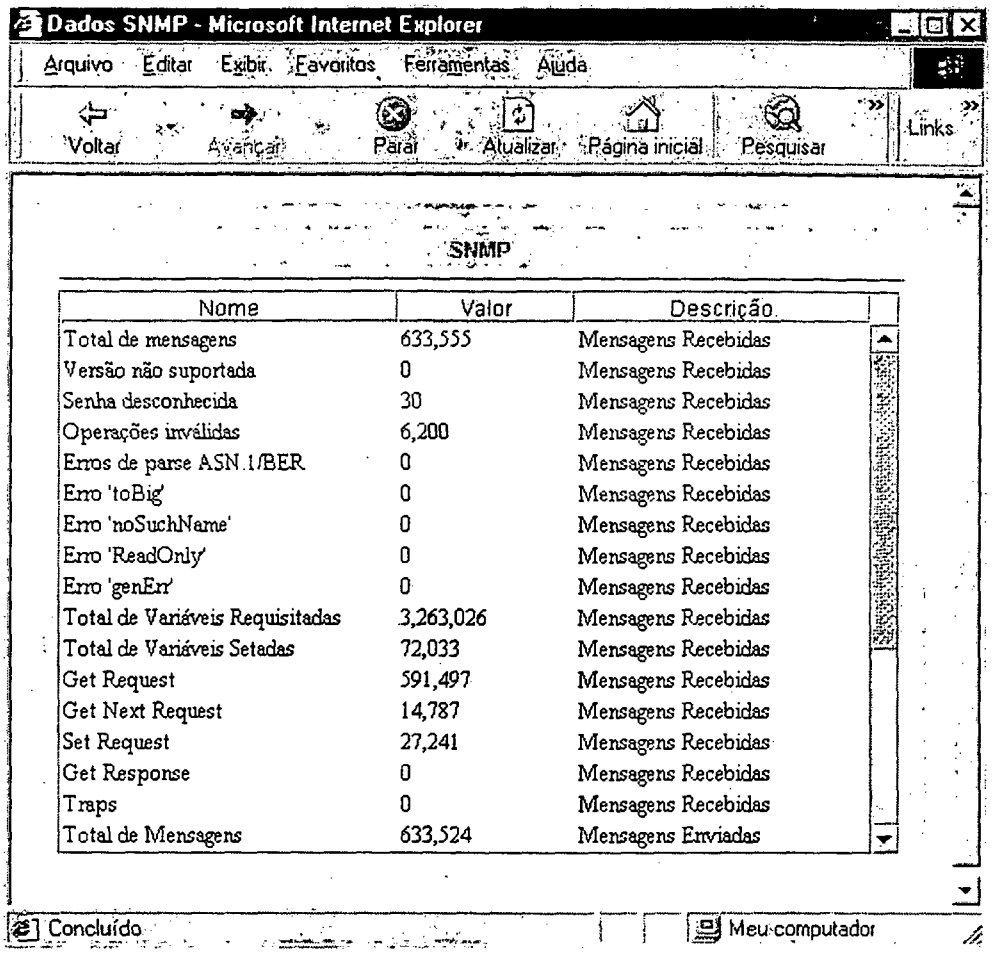


FIGURA 18 – Interface de dados sobre operações SNMP.

5.5.4. DADOS IP

O módulo "Dados IP" apresentado nesta seção tem como objetivo mostrar informações estatísticas de pacotes IP, os quais são coletados do grupo IP presentes na MIB II do agente monitorado [STA96]. A figura 19 apresenta a interface desenvolvida neste módulo.

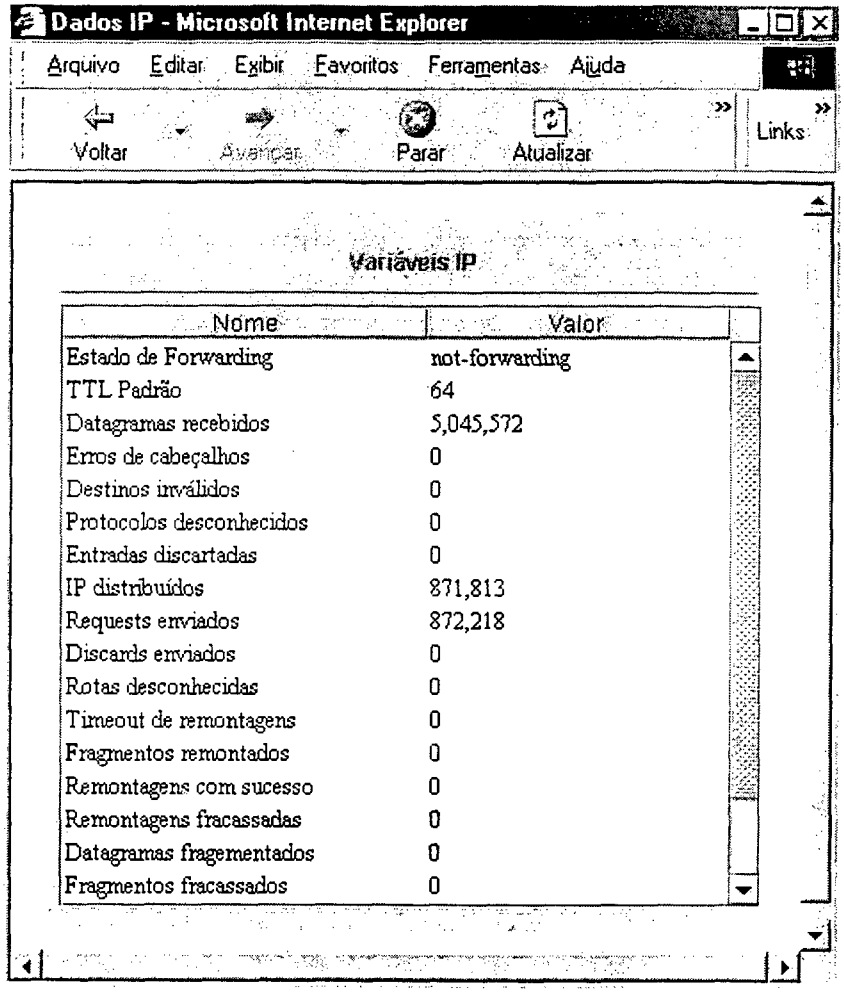


FIGURA 19 – Interface de dados IP.

5.5.5. DADOS TCP/UDP

A figura 20 apresenta dados referentes ao módulo "Dados TCP/UDP". Neste módulo será possível verificar dados estatísticos sobre segmentos TCP e datagramas UDP que foram enviados e recebidos.

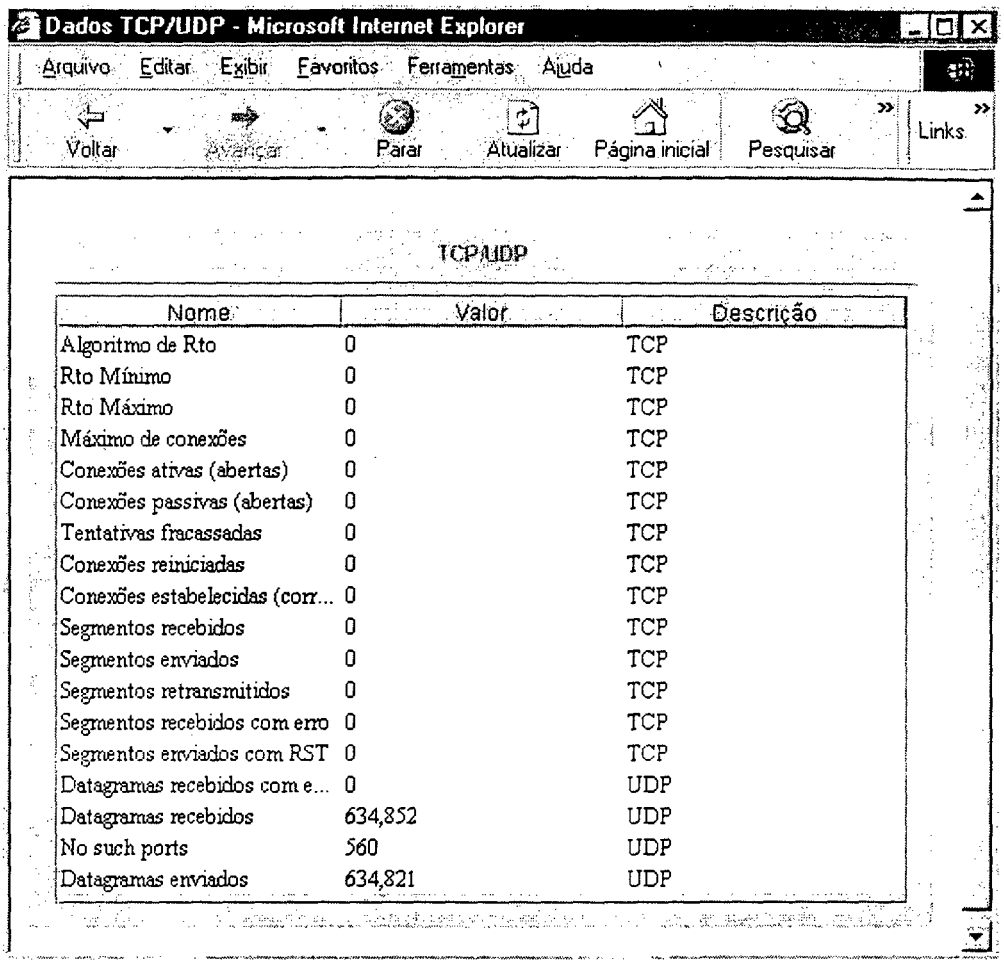


FIGURA 20 – Interface de dados sobre as camadas TCP e UDP.

5.5.6. DADOS DA TABELA IP ARP

A interface referente ao módulo "Dados da tabela IP ARP", conforme mostra a figura 21, apresenta uma simples tabela onde cada linha desta corresponde a uma interface física do sistema. As informações em cada linha fornecem um mapeamento do endereço de rede para endereço físico. As variáveis apresentadas nesta tabela são:

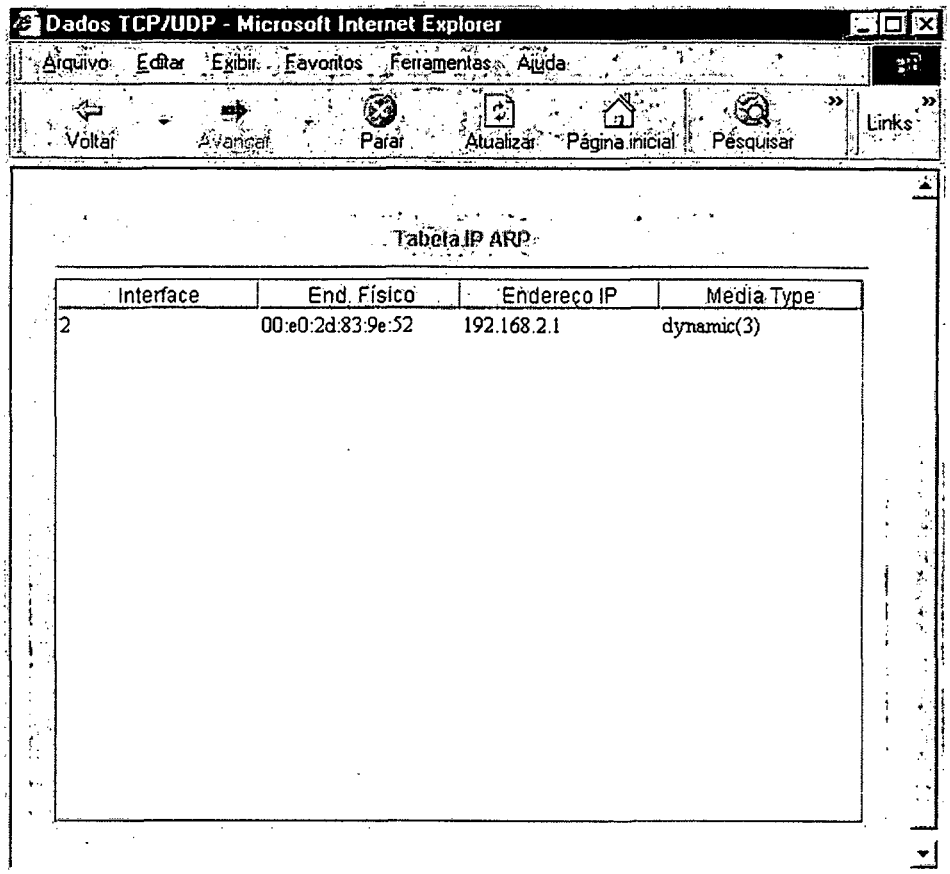


FIGURA 21 – Interface de dados sobre a tabela IP ARP.

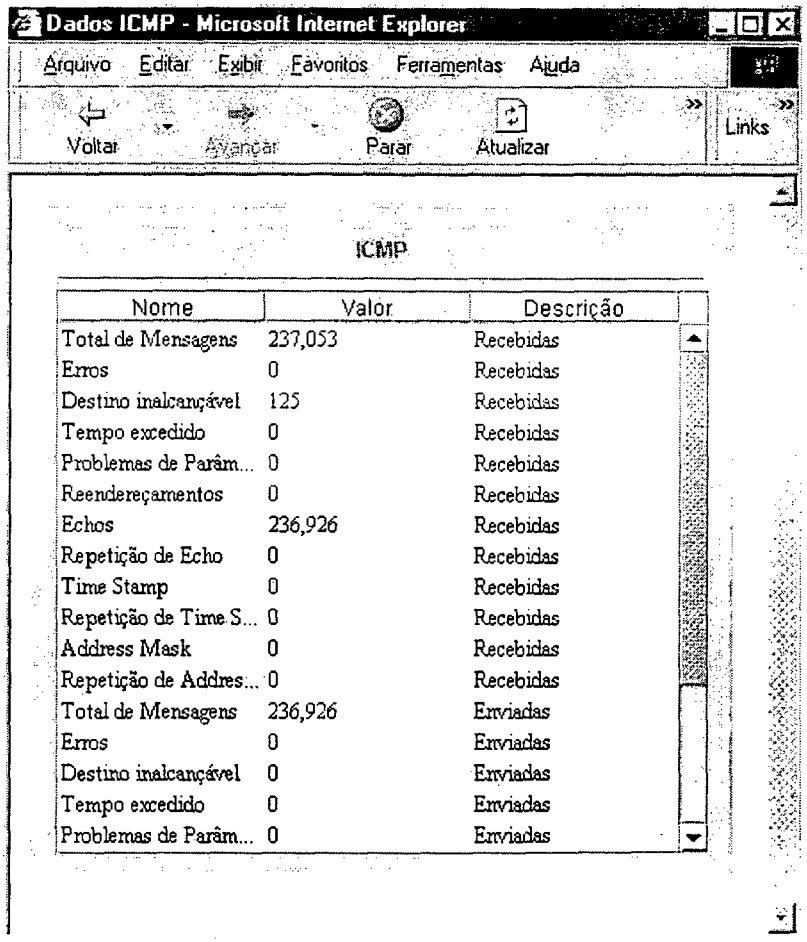
- Interface: interface na qual o elemento de rede é efetivo. Esta informação é lida do objeto *atIfIndex*, descrito na MIB II do agente monitorado. OID: 1.3.6.1.2.1.3.1.1.1.n;

- Endereço Físico: apresenta o endereço físico dependente do meio. Esta informação é lida do objeto *atPhysAddress*, descrito na MIB II do agente monitorado. OID: 1.3.6.1.2.1.3.1.1.2.n;

- Endereço IP: apresenta o endereço de rede (exemplo, endereço IP) correspondente ao endereço físico dependente do meio. Esta informação é lida do objeto *atNetAddress*, descrito na MIB II do agente monitorado. OID: 1.3.6.1.2.1.3.1.1.3.n;

5.5.7. DADOS ICMP

A figura 22 apresenta dados referentes ao módulo "Dados ICMP". Estes dados são referentes ao *Internet Control Message Protocol*, definido na RFC 792, que é uma parte integral da pilha de protocolo TCP/IP. Este protocolo acompanha obrigatoriamente o IP. Isto é, todos os sistemas que implementam IP devem fornecer o ICMP. Este protocolo fornece um significado para as mensagens transferidas de roteadores e outros *hosts* para um determinado *host*. Na sua essência, fornece retorno sobre problemas no ambiente de comunicação. Esta interface será útil para o administrador da rede obter informações sobre o total de mensagens recebidas e enviadas. As variáveis apresentadas na figura 21 são descritas no gurpo ICMP da MIB II do agente monitorado.



Nome	Valor	Descrição
Total de Mensagens	237,053	Recebidas
Erros	0	Recebidas
Destino inalcançável	125	Recebidas
Tempo excedido	0	Recebidas
Problemas de Parâm...	0	Recebidas
Reendereçamentos	0	Recebidas
Echos	236,926	Recebidas
Repetição de Echo	0	Recebidas
Time Stamp	0	Recebidas
Repetição de Time S...	0	Recebidas
Address Mask	0	Recebidas
Repetição de Addres...	0	Recebidas
Total de Mensagens	236,926	Enviadas
Erros	0	Enviadas
Destino inalcançável	0	Enviadas
Tempo excedido	0	Enviadas
Problemas de Parâm...	0	Enviadas

FIGURA 22 – Interface de dados ICMP.

5.5.8. DADOS DA BRIDGE (*BRIDGE LEARN TABLE*)

O módulo "*Bridge Learn Table*", tem como objetivo apresentar ao administrador, informações referentes ao estado corrente das interfaces de rede com seus respectivos host remotos.

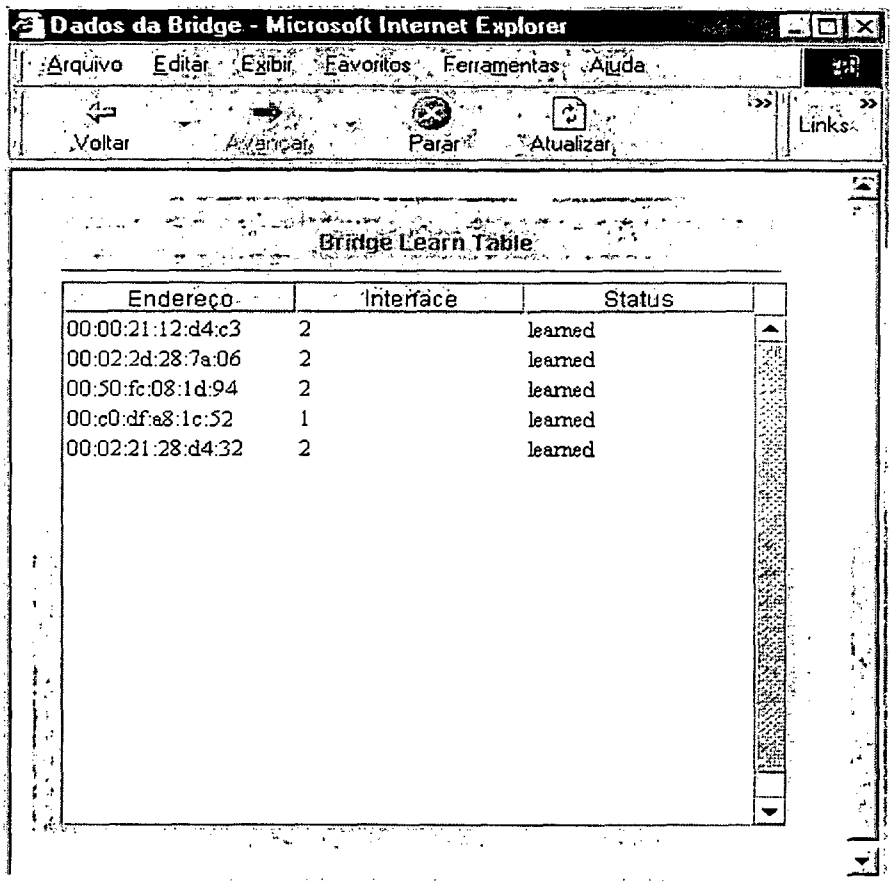


FIGURA 23 – Interface de dados da Bridge.

Os dados apresentados nesta interface estão descritos na *Kbridge-mib* do agente monitorado. São eles:

- *KbWirelessStationMACAddress:* endereço *Ethernet* (MAC) da interface wireless da estação remota que está conectada à Bridge. OID: 1.3.6.1.4.1.762.2.5;

- *kbWirelessInterfaceNumber*: o número da interface que corresponde à interface de rede wireless que esta entrada está ligada. Quando uma *Bridge* contém múltiplas interfaces de rede wireless, entradas diferentes podem conter valores diferentes para este campo. A numeração de interfaces inclui aquelas do tipo *non-wireless interface*, com a primeira interface de rede começando em 1. OID: 1.3.6.1.4.1.762.2.5

- *kbWirelessStationState*: indica o estado da conexão wireless remota. Neste caso é possível dois valores: *online (1)* ou *offline (2)*. É dito que a estação remota está *offline* quando ela pára a sua comunicação wireless. OID: 1.3.6.1.4.1.762.2.5.

Estes dados podem ser observados na figura 23.

5.5.9. TESTES DE QUALIDADE DO LINK

Para o desenvolvimento desta interface foram coletadas algumas informações junto ao administrador da rede do ISCC, o qual apresentou um *software* que acompanha o WavePOINT-II, denominado *AP-Manager*. O *AP-Manager* é um *software* de gerência local desenvolvido para o sistema operacional Windows. Deste, surgiu a idéia de se desenvolver um sistema para fazer o controle remoto de uma rede sem fio, praticamente com as mesmas características.

Num primeiro momento é identificado a MIB utilizada para o gerenciamento do WavePOINT-II. A documentação encontrada relata que este elemento de rede é compatível com as MIB's descritas nas RFC's 1213, 1398 e 1493, dando suporte ainda as WaveLAN e WaveNET MIB's (MIB's para elementos de rede sem fio) [LUC99]. Todas estas MIB's são de domínio público e padronizado para gerência de redes. A primeira parte da implementação deste módulo é baseada na MIB-II, resultando em valores concretos e esperados. Num segundo momento é feito os testes para o monitoramento baseado nas MIB's WaveLAN e WaveNET, estes, sem sucesso. Parte

do desenvolvimento do trabalho é lento devido aos insistentes testes realizados com as MIB's para redes sem fio.

Através de pesquisas na internet e troca de informações através de correio eletrônico, é identificada a empresa responsável pelo desenvolvimento do software para monitoramento *AP-Manager*, a KarlNet Incorporated. A esta empresa é solicitada a real MIB utilizada para o monitoramento do WavePOINT-II. Em gentil resposta, observa-se que as MIB's padrão para redes sem fio não são utilizadas para este elemento de rede, embora dê suporte. Juntamente com esta informação é repassada a MIB (ver ANEXO 1) para o monitoramento deste equipamento. Esta MIB é oficialmente não-documentada e de característica proprietária. A partir do conhecimento desta, um simples teste para verificar o nome do *host* é efetuado. De imediato a resposta é positiva. A partir daí então é desenvolvida a interface gráfica da aplicação. Esta é integrada com a API Java, bastando passar os valores das variáveis (objetos gerenciados) da MIB, também chamada de *KBridge-mib*. Os resultados recuperados são então tratados, resultando na interface apresentada na figura 24.

A figura "Teste de Qualidade do Link", apresenta uma interface para monitoramento do *link* entre uma estação remota e a WavePOINT-II (*AP-Bridge*). Com este módulo o administrador da rede pode verificar se a qualidade do *link* é satisfatória. Esta verificação acontece através da passagem de alguns parâmetros e posteriormente se faz a análise de resultados. A passagem de dados e a coleta de dados são destacados e numerados de 1 à 4, conforme mostra a figura 24.

Primeiramente, conforme o número 1 da figura 24, o administrador tem os dados da WavePOINT-II, obtidos através da passagem do número IP e de uma senha de acesso do elemento. Os dados apresentados são: nome do elemento, descrição, local e *Uptime* e ainda, uma lista para selecionar o nome do *host* remoto. Estes dados são retirados da MIB-II, conforme já visto nas seções anteriores, exceto o nome do host que é lido através da *Kbridge-mib*. O nome da estação remota é lida do objeto ***kbWirelessStationState***, descrito na *Kbridge-mib* do agente monitorado e o seu OID é 1.3.6.1.4.1.762.2.5.2.1.3.n.

Após informar o nome do *host* remoto os demais dados referentes a identificação do sistema são apresentados, sendo eles:

- Endereço Físico;
- Interface;
- Tipo de tecnologia de Rádio.

Na divisão número 3 da figura em questão é mostrado a taxa de sinal por ruído (*SNR – Signal-to-Noise Ratio*), a taxa de sinal e a taxa de ruído, tanto para o *host* local como para o *host* remoto, através de componentes *Gauge* para cada um dos *hosts*. Estes componentes são controlados através das variáveis descritas na *Kbridge-mib* e apresentados na tabela 4.

Nome	*OID*	*Descrição*
KbWirelessTestOurCurSNR	1.3.6.1.4.1.762.2.5.2.1.35.n	Retorna a taxa de sinal por ruído (SNR) de um link de rádio registrado pela Bridge.
KbWirelessTestOurCurSignalLevel	1.3.6.1.4.1.762.2.5.2.1.32.n	retorna o nível de sinal local corrente registrado pela Bridge.
KbWirelessTestOurCurNoiseLevel	1.3.6.1.4.1.762.2.5.2.1.33.n	retorna o nível de ruído local corrente registrado pela Bridge.
KbWirelessTestHisCurSNR	1.3.6.1.4.1.762.2.5.2.1.47.n	retorna a taxa de sinal por ruído (SNR) na estação remota de um link de rádio.
KbWirelessTestHisCurSignalLevel	1.3.6.1.4.1.762.2.5.2.1.44.n	retorna o nível de sinal corrente na estação remota.
KbWirelessTestHisCurNoiseLevel	1.3.6.1.4.1.762.2.5.2.1.45.n	retorna o nível de ruído corrente na estação remota.

Tabela 4 – Variáveis para verificação de Ruídos e Sinal – *Kbridge-mib*.

Por fim, os valores referentes ao número 4 da figura 24, apresentam o número de pacotes que foram recebidos e perdidos na estação local e na estação remota. Esses valores são obtidos através das variáveis mostradas na tabela 5.

Nome	*OID*	*Descrição*
KbWirelessTestHisTx	1.3.6.1.4.1.762.2.5.2.1.30.n	número de pacotes de testes transmitidos para a estação remota quando executando o teste de qualidade do sinal.
KbWirelessTestHisRx	1.3.6.1.4.1.762.2.5.2.1.31.n	número de pacotes de testes recebidos pela estação remota quando executando o teste de qualidade do sinal.
KbWirelessTestOurTx	1.3.6.1.4.1.762.2.5.2.1.28.n	número de pacotes de testes transmitidos pelo host local quando executando o teste de qualidade do sinal.
KbWirelessTestOurRx	1.3.6.1.4.1.762.2.5.2.1.29.n	número de pacotes de testes recebidos pelo host local quando executando o teste de qualidade do sinal.

Tabela 5 – Variáveis para verificação de pacotes recebidos e enviados – *Kbridge-mib*.

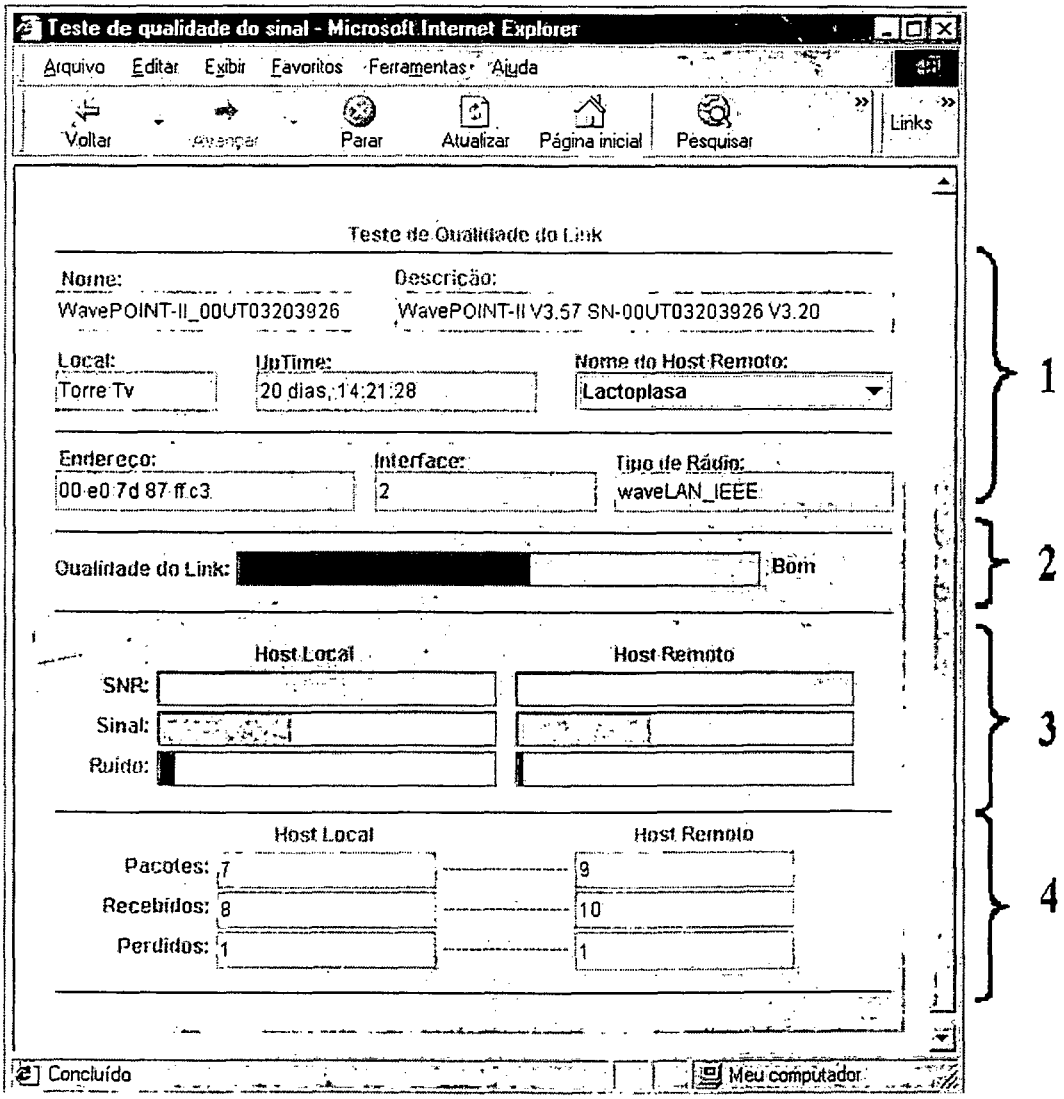


FIGURA 24 – Interface para verificar a Qualidade do Sinal.

O número de pacotes perdidos, tanto para o host local como para o host remoto, é calculado subtraindo-se o número de pacotes enviados do número de pacotes recebidos. Isso significa que, sendo o número de pacotes recebidos maior do que o número de pacotes enviados, houve retransmissões de pacotes perdidos.

6. CONCLUSÃO

A partir deste trabalho, pôde-se comprovar que as tecnologias Web e Java, em conjunto, permitem o desenvolvimento de aplicativos para serem acessados independente de localização e de plataforma. Esta característica é interessante para aplicações em gerência de redes, particularmente em ambiente sem fio (wireless), onde, por sua própria natureza, os elementos a serem gerenciados estão distribuídos geograficamente.

Através do levantamento de informações sobre o gerenciamento em redes *wireless*, percebeu-se que o protocolo SNMP encontra aplicação também para este tipo de ambiente, a partir da definição de novas MIBs, especialmente as proprietárias.

Considerando o crescente aumento do número de usuários que têm interesse pelo uso da tecnologia *wireless*, é clara a necessidade de monitoramento e controle destas redes sem fio. Isto para garantir ao usuário o desempenho e confiabilidade requisitado no uso de redes sem fio, visto que esta é uma tecnologia emergente e poucos são os softwares desenvolvidos com a finalidade de prover serviços Internet.

O estudo do ambiente de transmissão sem fio permitiu a identificação do elemento conhecido como *wireless bridge (WavePOINT-II)*, passível de ser gerenciado. Este elemento é a principal ligação entre a base de transmissão e as estações remotas. Assim, é óbvia a necessidade de monitoramento deste elemento a fim de permitir controlar o seu funcionamento. A ferramenta apresentada neste trabalho permite este monitoramento.

O aplicativo desenvolvido apresentou como resultados, dados em geral referentes as variáveis da MIB-II e também da MIB proprietária *Kbridge-mib*[4], apresentada em anexo. Com os valores coletados e apresentados através das interfaces implementadas chegou-se à conclusão de que o administrador de uma rede sem fio pode tomar decisões sobre seus equipamentos wireless quando estes apresentarem algum

[4] Esta MIB é oficialmente não-documentada, entretanto, após exaustiva busca por informações sobre a mesma, esta foi gentilmente cedida pelo corpo técnico da KarlNet Incorporated. Esta MIB, conhecida como KBRIDGE-MIB, é listada no Anexo 1.

resultado inadequado. Estas decisões poderão ser tomadas, por exemplo, analisando-se os dados referentes ao número de pacotes transmitidos e recebidos, tentativas de acesso ao meio, demanda de serviço, interfaces usadas, host remotos presentes na rede, locais em que estão instalados, taxa de transmissão, tipos de interfaces e elementos de rede, erros gerados na transmissão de dados e outros.

De modo geral, o administrador pode verificar a qualidade do sinal, a partir do nível de ruído e sinal, para então aperfeiçoar o seu equipamento, bem como identificar se a instalação de um equipamento, como uma antenta, é possível ou não.

Os equipamentos podem sofrer ruídos, e estes quando muito alto podem interferir na taxa de transmissão de dados, prejudicando o desempenho da rede. Com o aplicativo para monitoramento implementado, será possível identificar as taxas de ruídos entre um host local e um host remoto. Se o nível de ruído presente for muito significativo, o administrador pode tomar as medidas devidas para melhorar a transmissão de dados, de forma a reduzir as taxas de ruídos, que podem estar sendo gerados pelos elementos físicos da rede, como placas, cabos e outros; ou por interferências externas.

Durante a realização deste trabalho, e através dos resultados obtidos, surgiram algumas possibilidades de continuidade deste trabalho, dentre às quais pode-se identificar:

- Extensão ao aplicativo para que seja possível analisar o desempenho de uma rede sem fio;
- Implementação de alarmes que notifiquem os administradores quando o nível de sinal estiver muito baixo ou de ruídos muito alto, baseado em *thresholds*, permitindo que possam ser tomadas decisões à tempo de corrigir o problema antes da degradação da transmissão;
- Implementação de um módulo de gerência de falhas;
- Avaliação da segurança na transmissão de dados em redes sem fio.

O presente trabalho atingiu todos os objetivos apresentados no capítulo introdutório desta dissertação. Foi possível integrar as tecnologias Web, Internet, Java e SNMP para desenvolver um aplicativo de monitoramento de redes sem fio, baseado nas variáveis descritas na MIB II e, principalmente, as variáveis descritas na *Kbridge-mib*.

7. BIBLIOGRAFIA

[ANE98] ANEROUSIS, Nikolaos. ***Scalable Management Services Using Java and The World Wide Web***. AT&T Labs Research. Http://www.research.att.com/~nilkos/, 1998.

[ARM99] ARMANINI, Kátira Kowalski. ***Projeto I – WebVis: Uma Ferramenta para Análise de Tráfego em Links TCP/IP através de Browsers Web***. Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação – Departamento de Informática e Estatística – Universidade Federal de Santa Catarina. Florianópolis, fev. 1999.

[BAR98] BAROTTO, André Mello. ***Realização da Gerência Distribuída de Redes Utilizando SNMP, JAVA, WWW e CORBA***. Dissertação de Mestrado em Ciência da Computação – Departamento de Informática e Estatística – Universidade Federal de Santa Catarina. Florianópolis, abril 1998.

[CHI99] CHIOZZOTTO, M.; SILVA, L. A. P.. ***TCP/IP – Tecnologia e Implementação***. Editora Erica, São Paulo, 1999.

[COL00] COLETTO, Luis Henrique; ***Wireless – Redes Sem Fio.*** http://www.cirp.usp.br/cursos/curso_wireless/wireless.html. Seção Técnica de Manutenção. 2000.

[COM99] COMER, Douglas E., STEVENS, David L.. ***Interligação em Rede com TCP/IP – Princípios, Protocolos e Arquitetura***. Editora Campus Ltda, Rio de Janeiro, 1999.

[COM99] COMER, Douglas E., STEVENS, David L.. ***Interligação em Rede com TCP/IP – Projeto, Implementação e Detalhes Internos***. Editora Campus Ltda, Rio de Janeiro, 1999.

[DEI98] DEITEL, H. M.; DEITEL, P. J.; ***Java: How To Program***. 2 ed., Prentice Hall, New Jersey, 1998.

[FIT00] FITZEK, F. H. P.; MORICH R.; WOLISZ A.; ***Comparison of Multi-Code Link-Layer Transmission Strategies in 3Gwireless CDMA.*** Article, IEEE Communications Magazine, Pag 58-64. Technical University of Berlin, October, 2000.

[GON98] GONÇALVES, Paulo R. Riccioni. ***Gerenciamento de Serviços de Telecomunicações com CORBA e JAVA.*** Dissertação de Mestrado em Ciência da Computação – Departamento de Informática e Estatística – Universidade Federal de Santa Catarina. Florianópolis, Agosto 1998.

[HAG98] HAGGERTY, Paul and SEETHARAMAN, Krishnan. ***The Benefits of CORBA-Based Network Management.*** Communications of The ACM. Pág 73 a 79. October 1998.

[ISP98] ISPA. ***ISPA-430, Programação Java Básico.*** Cyclades, São Paulo, Janeiro 1998.

[JEP97] JEPSON, Brian. ***Java Database Programming.*** John Wiley & Sons, Inc. 1997.

[JOH99] JOHNSON, D. B.; MALTZ, D. A.; ***Dynamic Source Routing in Ad Hoc Wireless Networks.*** Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1999.

[JON01] JONATHAN, Sevy. ***SNMP API.*** Department of Mathematics and Computers Sciense, Drexel University.

http://edge.mcs.drexel.edu/GICL/people/sevy. May, 2000.

[KLA00] KLAUCK, Hugo André. ***Proposta de um Ambiente de Gerência de Redes de Alta Velocidade Utilizando CORBA, JAVA e HTML.*** Dissertação de Mestrado em Ciência da Computação – Departamento de Informática e Estatística – Universidade Federal de Santa Catarina. Florianópolis, abr. 2000.

[LIM00] LIMA, A.R.; SALES A. B.; WESPHALL C. B.. ***Wireless – Curso PCT – Motorola.*** LRG – INE – UFSC, Florianópolis, 2000.

[LIM99] LIMA, A. R.; SALES, A. B.; ***Transmissão de Dados em Redes de Computadores Sem Fio.*** Departamento de Informática e Estatística – Universidade Federal de Santa Catarina. Florianópolis, 1999.

[LOR98] LORENSET, Vera Lúcia. ***Gerenciamento Distribuído TMN: uma Experiência em Supervisão de Alarmes com CORBA.*** Dissertação de Mestrado em Ciência da Computação – Departamento de Informática e Estatística – Universidade Federal de Santa Catarina. Florianópolis, abril 1998.

[LUC95] LUCCA, José Eduardo de. ***Arquitetura de Segurança para Redes Aplicada a Sistemas de Gerência.*** Dissertação de Mestrado em Ciência da Computação – Departamento de Informática e Estatística – Universidade Federal de Santa Catarina. Florianópolis, junho 1995.

[LUC99] Lucent Tecnologies – ***WavePOINT-II Access Point Software, Version: 5.00.*** http://occupant.caltech.edu/software/drives/pc/lucent/wavelan/accesspoint/ap500/readme.txt. 1999.

[MEN98] MENEZES, E. S.; SILVA, P. L. L.; ***Gerenciamento de Redes: Estudos de Protocolos***. Departamento de Informática, UFPE, Setembro de 1998.

[MIB00] MIB Information – ***Wireless Link Test.*** http://edge.mcs.drexel.edu/GICL/people/sevy/airport/MIB.html, 2000

[ORF96] ORFALI, Robert; HARKEY, Dan; EDWARDS, Jerl. ***The Essential Distributed Objects – Survivel Guide.*** John Wiley & Sons, INC. 1996.

[PAL96] PALMA, Marcelo Reis; LAVINIKI, Selma Rita. ***Implementação de Objetos Gerenciados para Supervisão de Alarmes TMN Aplicados à Central EWSD***. Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação – Departamento de Informática e Estatística – Universidade Federal de Santa Catarina, Florianópolis, Dezembro 1996.

[RFC1213] RFC 1213 – ***Management Information Base for Network Management of TCP-IP-based internets: MIB-II.*** Network Working Group. Hughes LAN Systems, Mellon University. Abril de 1993.

[RFC1448] RFC 1448 – ***Protocol Operation for Version 2 of Simple Network Management Protocol (SNMPv2).*** Network Workin Group. Hughes LAN Systems; SNMP Research, Inc. April, 1993.

[SOA97] SOARES, L. F. G.; LEMOS G.; COLCHER, S.. ***Redes de Computadores – Das LANs, MANs e WANs às Redes ATM.*** Editora Campus Ltda, Rio de Janeiro, 1997.

[STA96] STALLINGS, William. ***SNMP, SNMPv2, and RMON: Pratical Network Management.*** Addison Wesley Longman, Inc. 1996.

[SUA99] SUAVÉ, J. P.; TEIXEIRA, J. H. Jr.; MOURA, J. A. B.; TEIXEIRA, S. Q. R.. ***Redes de Computadores – Serviços, Administração e Segurança.*** Editora Makron Books, São Paulo, 1999.

[SUN99] SUN MICROSYSTEMS INC. ***The Java 2 SDK – Standard Edition Version 1.2.2.*** Palo Alto – USA. Http://java.sun.com/. 1999.

[TAN97] TANEMBAUM, Andrew S.. ***Redes de Computadores.*** Tradução de PublicCare Serviços de Informática (3ª Edição). Editora Campus Ltda, Rio de Janeiro, 1997.

[WAD97] WADE, V.; LEWIS, D.; BRACHT, R.. ***The Development of Integrated Inter- and Intra-Domain.*** Paper. Department of Computer Science, Trinity College Dublin, Ireland; Computer Science Department, University College London, UK; IBM Heidlberg, Germany. 1997.

[WAV99] ***Wave Wireless Products.*** Aironet & BreezeCOM. http://www.breakfree-technologies.com/products_wave.html. 1999.

[WLA99] ***Introduction to Wireless LANs.*** WLANA – The Wireless LAN Alliance. http://www.wlana.com. 1999.

[WLAN97] WaveLAN.MIB – ***SNMP MIB for WaveLAN.*** Lucent Technologies Inc. Março de 1997.

[WNET97] WaveNET.MIB – ***SNMP MIB for Roaming for WaveLAN***. Lucent Technologies Inc. Março de 1997.

[WRI00] WRIGHT, M.; ***Wireless Networking: The Ties That Don't Bind***. www.ednmag.com/ednmag/reg/2000/05112000/10df2.htm. Maio, 2000.

[ZAC00] ZACKER, Craig e DOYLE, Paul. ***Redes de Computadores – Configuração, Manutenção e Expansão***. Editora Makron Books, São Paulo, 2000.

[ZHA00] ZHAO, D.; SHEN, X.; MARK, J. W.; ***Efficient Call Admission Control for Heterogeneous Services in Wireless Mobile ATM Networks.*** Article, IEEE Communications Magazine, Pag 72-78. University of Waterloo, Canada, October, 2000.

ANEXO 1 – KBRIDGE-MIB

```
KBRIDGE-MIB DEFINITIONS ::= BEGIN

        IMPORTS
                MODULE-IDENTITY, OBJECT-TYPE,
                Counter32, IpAddress, Gauge32,
                enterprises
                        FROM SNMPv2-SMI;
--
*************************************************************************
-- * MODULE IDENTITY
--
*************************************************************************

karlnet MODULE-IDENTITY
        LAST-UPDATED "200009181200"
        ORGANIZATION "KarlNet Incorporated"
        CONTACT-INFO
                "Postal: KarlNet Incorporated
                        5030 Postlewaite Road
                        Columbus, OH 43235 USA
                  Tel: +1 614 457 5275
                  Fax: +1 614 442 7599
                 E-mail: info@karlnet.com"
        DESCRIPTION
                "The MIB module for KarlNet entities.
                iso(1).org(3).dod(6).internet(1).mgmt(4).enterprises(1).karlnet(762)"
      REVISION      "200009181200"
        DESCRIPTION     "Changed kbWirelessStationSNR type from Gauge32 to
                        INTEGER"
      REVISION      "200007251200"
        DESCRIPTION     "Added description text on kbControlTemperature to
                        clarify that the temperature given is in half-degree
                        increments centigrade. That is, kbControlTemperature =
                        66 implies 33 degrees Celsius."
    REVISION        "200007111200"
        DESCRIPTION     "Initial Release."

        ::= { enterprises 762 }

kbridge-mib OBJECT IDENTIFIER ::= { karlnet 2 }

karlNetKBControl OBJECT IDENTIFIER ::= { kbridge-mib 1 }

--                              ****************************************

-- ** kbControl block
```

```
-- ****************************************
-- KBRIDGE control MIB.
-- 1 = bit-map for block transferred (future use).
-- 2 = check parity.
-- 3 = write block to flash ROM.
-- 4 = kbConfigReadEnable.
-- 5 = kbFlashProgEnable.
-- 6 = kbBoot.
-- 7 = kbStatus
-- 8 = kbEnvironment
-- 9 = kbDummy - used for testing write community
-- 10 = return rconsole buf
-- 11 = shutdown (1 = shutdown)
-- 12 = Temperature from Dallas chip in 1/2 degree centagrade increments)

kbControlBitMap OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS not-accessible
        STATUS obsolete
        DESCRIPTION
                "Not currently in use."
        ::= { karlNetKBControl 1 }

kbControlTestChecksum OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS write-only
        STATUS current
        DESCRIPTION
                "Send the KBridge config structure terminated with a two-byte checksum
                to this OID to check the value of the config structure checksum.
                The KarlBridge will return one of the following SNMP errors:
                        noError         -- the two-byte checksum was okay
                        badValue        -- the two-byte checksum failed"
        ::= { karlNetKBControl 2 }

kbControlWriteToFlash OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS write-only
        STATUS current
        DESCRIPTION
                "Write a config structure or entire BIN file to this OID in consecutive
                256-byte segments to reprogram the KarlBridge's Flash ROM module. In
                order for the KarlBridge to correctly reprogram the Flash ROM, the octet
                string MUST:
                        1. Be error-checked for correctness by the configuration program,
                        AND
                        2. Match several sanity checks made by the KarlBridge kernel,
                        AND
```

```
                3. Contain a valid checksum (see kbControlTestChecksum)"
        ::= { karlNetKBControl 3 }

kbControlConfigReadEnable OBJECT-TYPE
        SYNTAX INTEGER
                {
                        disable(0),
                        enable(1)
                }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "Whether or not the internal configuration structure is readable via
                SNMP. Possible values are:
                        disable(0)      -- the internal configuration structure IS NOT
                                readable
                        enable(1)       -- the internal configuration structure IS readable"
        ::= { karlNetKBControl 4 }

kbControlFlashProgEnable OBJECT-TYPE
        SYNTAX INTEGER
                {
                        disable(0),
                        enable(1)
                }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "Whether or not the Flash ROM is programmable via SNMP.
                Possible values are:
                        disable(0)      -- the Flash ROM IS NOT programmable
                        enable(1)       -- the Flash ROM IS programmable"
        ::= { karlNetKBControl 5 }

kbControlReboot OBJECT-TYPE
        SYNTAX INTEGER (1)
        MAX-ACCESS write-only
        STATUS current
        DESCRIPTION
                "Set this OID value to 1 to reboot the KarlBridge."
        ::= { karlNetKBControl 6 }

kbControlStatus OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS obsolete
        DESCRIPTION
                "This OID is not in use."
```

```
        ::= { karlNetKBControl 7 }

kbControlEnvironment OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This OID is not supported for general use."
        ::= { karlNetKBControl 8 }

kbControlTestSNMPWrite OBJECT-TYPE
        SYNTAX INTEGER (1)
        MAX-ACCESS write-only
        STATUS current
        DESCRIPTION
                "This OID is provided to test whether your SNMP software can
                successfully set an OID on the KarlBridge by setting its value to 1."
        ::= { karlNetKBControl 9 }

kbControlReturnConsoleBuf OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "Returns the internal Console Buffer record.
                Usage of this record varies depending on the type of KarlBridge queried.
                Values in the record are displayed on the system console of the
                KarlBridge where available."
        ::= { karlNetKBControl 10 }

kbControlShutdown OBJECT-TYPE
        SYNTAX INTEGER (1)
        MAX-ACCESS write-only
        STATUS current
        DESCRIPTION
                "Set this OID to 1 to shutdown the KarlBridge."
        ::= { karlNetKBControl 11 }

kbControlTemperature OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "For KarlBridges equipped with a Dallas temperature chip, returns the
                temperature of the Dallas chip in 1/2 degree centigrade increments (+/-
                0.5 degrees C).
                To determine the correct temperature in Centigrade, divide by 2.
```

```
                To determine the correct temperature in Fahrenheit, divide by 2, then
                multiply by 9/5 and add 32.
                KarlBridges that are not equipped with the Dallas chip return a
                'noSuchName' error."
        ::= { karlNetKBControl 12 }

-- ****************************************
-- ** Read Block
-- ****************************************

karlNetKBRdBlk OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "This OID is not supported for general use."
        ::= { kbridge-mib 2 }

karlNetKBWrtBlk OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "This OID is not supported for general use."
        ::= { kbridge-mib 3 }

karlNetKBHostAccess OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "This OID is not supported for general use."
        ::= { kbridge-mib 4 }

kbWireless OBJECT IDENTIFIER ::= { kbridge-mib 5 }

kbWirelessStationNumber OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of registered stations (i.e. valid entries) in the
                kbWirelessStationTable."
        ::= { kbWireless 1 }

kbWirelessStationTable OBJECT-TYPE
        SYNTAX SEQUENCE OF KbWirelessStationEntry
        MAX-ACCESS not-accessible
```

```
        STATUS current
        DESCRIPTION
                "List of wireless connections and their attributes.
                Each entry in the table corresponds to a particular
                wireless station (usually a satellite) that is
                attached to a particular wireless interface."
        ::= { kbWireless 2 }

kbWirelessStationEntry OBJECT-TYPE
        SYNTAX KbWirelessStationEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "An entry in the kbWirelessStationTable. Each entry
                corresponds to a particular wireless station connected
                to one of the KarlBridge's interfaces. The entries are
                indexed by kbWirelessStationIndex and also can be generally
                considered unique based upon the kbWirelessStationMACAddress
                field."
        INDEX { kbWirelessStationIndex }
        ::= { kbWirelessStationTable 1 }

KbWirelessStationEntry ::=
        SEQUENCE {
                        kbWirelessStationIndex
                                        Unsigned32,
          kbWirelessStationInterfaceNumber
                Unsigned32,
           kbWirelessStationName
                OCTET STRING,
           kbWirelessStationExclHellos
                Counter32,
           kbWirelessStationGoodHellos
                Counter32,
           kbWirelessStationLowHellos
                Counter32,
           kbWirelessStationSignalLevel
                Gauge32,
           kbWirelessStationNoiseLevel
                Gauge32,
           kbWirelessStationSignalQuality
                Gauge32,
           kbWirelessStationPktTransmits
                Counter32,
           kbWirelessStationMACAddress
                OCTET STRING,
           kbWirelessStationTransmits
                Counter32,
```

kbWirelessStationBadTransmits
Counter32,
kbWirelessStationReTransmits
Counter32,
kbWirelessStationIPAddress
IpAddress,
kbWirelessStationType
INTEGER,
kbWirelessStationSNR
INTEGER,
kbWirelessStationState
INTEGER,
kbWirelessPoll
Counter32,
kbWirelessPollData
Counter32,
kbWirelessPollNoData
Counter32,
kbWirelessPollMoreData
Counter32,
kbWirelessPollTimeouts
Counter32,
kbWirelessPollOfflines
Counter32,
kbWirelessTestTimeouts
Unsigned32,
kbWirelessTestInterval
Unsigned32,
kbWirelessTestPacketSize
Integer32,
kbWirelessTestOurTx
Counter32,
kbWirelessTestOurRx
Counter32,
kbWirelessTestHisTx
Counter32,
kbWirelessTestHisRx
Counter32,
kbWirelessTestOurCurSignalLevel
Gauge32,
kbWirelessTestOurCurNoiseLevel
Gauge32,
kbWirelessTestOurCurSignalQuality
Gauge32,
kbWirelessTestOurCurSNR
Gauge32,
kbWirelessTestOurMinSignalLevel
Gauge32,

kbWirelessTestOurMinNoiseLevel
Gauge32,
kbWirelessTestOurMinSignalQuality
Gauge32,
kbWirelessTestOurMinSNR
Gauge32,
kbWirelessTestOurMaxSignalLevel
Gauge32,
kbWirelessTestOurMaxNoiseLevel
Gauge32,
kbWirelessTestOurMaxSignalQuality
Gauge32,
kbWirelessTestOurMaxSNR
Gauge32,
kbWirelessTestHisCurSignalLevel
Gauge32,
kbWirelessTestHisCurNoiseLevel
Gauge32,
kbWirelessTestHisCurSignalQuality
Gauge32,
kbWirelessTestHisCurSNR
Gauge32,
kbWirelessTestHisMinSignalLevel
Gauge32,
kbWirelessTestHisMinNoiseLevel
Gauge32,
kbWirelessTestHisMinSignalQuality
Gauge32,
kbWirelessTestHisMinSNR
Gauge32,
kbWirelessTestHisMaxSignalLevel
Gauge32,
kbWirelessTestHisMaxNoiseLevel
Gauge32,
kbWirelessTestHisMaxSignalQuality
Gauge32,
kbWirelessTestHisMaxSNR
Gauge32,
kbWirelessTestLinkUp
INTEGER,
kbWirelessTestLostLink
INTEGER,
kbWirelessTestLostTestPkts
INTEGER,
kbWirelessStationRadioType
INTEGER,
kbWirelessRecordType
INTEGER,

```
                    kbWirelessStationPktReceives
                            Counter32,
                    kbWirelessStationReceives
                            Counter32,
                    kbWirelessStationBytesReceives
                            Counter32,
                    kbWirelessStationBytesTransmits
                            Counter32,
                    kbWirelessRegistrationRecord
                            OCTET STRING,
                    kbWirelessStationFragmentDiscards
                            Counter32,
                    kbWirelessStationFragmentMissings
                            Counter32,
                    kbWirelessStationFragmentLostFrames
                            Counter32,
                    kbWirelessStationFragmentErrors
                            Counter32
}

kbWirelessStationIndex OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The index of this entry in the kbWirelessStationTable.
                Each entry in the table has a unique kbWirelessStationIndex."
        ::= { kbWirelessStationEntry 1 }

kbWirelessStationInterfaceNumber OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The interface number of the wireless network interface
                that this entry is attached to. When a KarlBridge contains
                multiple wireless network interfaces, different entries
                may contain different values for this field. Interface
                numbering includes non-wireless interfaces, with the
                first network interface being number 1."
        ::= { kbWirelessStationEntry 2 }

kbWirelessStationName OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The name of the remote wireless station as reported
```

```
                to the KarlBridge."
        ::= { kbWirelessStationEntry 3 }

kbWirelessStationExclHellos OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of Hello packets from the KarlBridge that were
                received by the remote station with Excellent signal quality."
        ::= { kbWirelessStationEntry 4 }

kbWirelessStationGoodHellos OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of Hello packets from the KarlBridge that were
                received by the remote station with Good signal quality."
        ::= { kbWirelessStationEntry 5 }

kbWirelessStationLowHellos OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of Hello packets from the KarlBridge that were
                received by the remote station with Low signal quality."
        ::= { kbWirelessStationEntry 6 }

kbWirelessStationSignalLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The radio signal level in percentage."
        ::= { kbWirelessStationEntry 7 }

kbWirelessStationNoiseLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The radio noise level in percentage."
        ::= { kbWirelessStationEntry 8 }

kbWirelessStationSignalQuality OBJECT-TYPE
        SYNTAX Gauge32
```

```
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The relative signal quality of the radio transmissions."
        ::= { kbWirelessStationEntry 9 }

kbWirelessStationPktTransmits OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of Ethernet packets the remote station has
                transmitted through the wireless interface connected
                to the KarlBridge."
        ::= { kbWirelessStationEntry 10 }

kbWirelessStationMACAddress OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The Ethernet (MAC) address of the remote station's
                wireless interface that is connected to the KarlBridge."
        ::= { kbWirelessStationEntry 11 }

kbWirelessStationTransmits OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The total number of wireless transmissions the KarlBridge
                has made to the remote station through this interface.
                This number does not necessarily correspond to the number
                of Ethernet packets transmitted through this interface
                (see kbWirelessStationPktTransmits)."
        ::= { kbWirelessStationEntry 12 }

kbWirelessStationBadTransmits OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The total number of failed wireless transmissions the KarlBridge
                has made to the remote station through this interface. If a
                wireless transmission is not received after 10 attempts, this
                counter is incremented."
        ::= { kbWirelessStationEntry 13 }
```

```
kbWirelessStationReTransmits OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The total number of wireless re-transmissions the KarlBridge
                has attempted to the remote station through this interface."
        ::= { kbWirelessStationEntry 14 }

kbWirelessStationIPAddress OBJECT-TYPE
        SYNTAX IpAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The IP address of the remote station, if any."
        ::= { kbWirelessStationEntry 15 }

kbWirelessStationType OBJECT-TYPE
        SYNTAX INTEGER
                {
                        compatibility_Mode(1),
                        tc_Peer_to_Peer(2),
                        tc_Base_Station(3),
                        tc_Satellite_Station(4),
                        tc_Polling_Base_Station(5)
                }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The remote station's mode of operation on this radio interface. Possible
                values are:
                        compatibility_Mode(1) -- IEEE 802.11 mode or legacy Wireless(-
                        I) mode
                        tc_Peer_to_Peer(2) -- TurboCell Peer-to-Peer Station (no base
                        stations, no polling)
                        tc_Base_Station(3) -- TurboCell Base Station
                        tc_Satellite_Station(4) -- TurboCell Satellite Station
                        tc_Polling_Base_Station(5) -- TurboCell Polling Base Station"
        ::= { kbWirelessStationEntry 16 }

kbWirelessStationSNR OBJECT-TYPE
        SYNTAX INTEGER
                {
                unknown_SNR(1),
                low_SNR(2),
                good_SNR(3),
                excellent_SNR(4),
                }
```

```
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The relative Signal-to-Noise Ratio of the wireless connection
                between the remote station and the KarlBridge."
        ::= { kbWirelessStationEntry 17 }

kbWirelessStationState OBJECT-TYPE
        SYNTAX INTEGER
                {
                online(1),
                offline(2)
                }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The remote station's wireless connection status.
                The remote station is said to be offline when it appears
                to cease wireless communication, either because it no longer
                acknowledges poll packets or no longer sends poll packets"
        ::= { kbWirelessStationEntry 18 }

kbWirelessPoll OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of times the KarlBridge has polled the remote station."
        ::= { kbWirelessStationEntry 19 }

kbWirelessPollData OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of times the KarlBridge has polled the remote station
                that the remote station has responded with data, after which the
                remote station's transmit queue is empty."
        ::= { kbWirelessStationEntry 20 }

kbWirelessPollNoData OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of times the KarlBridge has polled the remote station
                that the remote station has not responded with data."
        ::= { kbWirelessStationEntry 21 }
```

```
kbWirelessPollMoreData OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS deprecated
        DESCRIPTION
                "The number of times the KarlBridge has polled the remote station
                that the remote station has had more data in its transmit queue
                than it could transmit in a single poll response."
        ::= { kbWirelessStationEntry 22 }

kbWirelessPollTimeouts OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "The number of times the KarlBridge has polled the remote station
                and the remote station has not responded before the timeout
                period expired."
        ::= { kbWirelessStationEntry 23 }

kbWirelessPollOfflines OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "The number of times the remote station has gone offline.
                The value is relevant only when the KarlBridge is in one of
                the Base Station modes (polling or non-polling)."
        ::= { kbWirelessStationEntry 24 }

kbWirelessTestTimeouts OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "The total number of radio link test packets for which the
                KarlBridge did not receive a response within the given timeout period."
        ::= { kbWirelessStationEntry 25 }
        kbWirelessTestInterval OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "The duration (in hundredths of seconds) of the radio link test
                performed between the KarlBridge and the remote station."
        ::= { kbWirelessStationEntry 26 }
```

```
kbWirelessTestPacketSize OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "The size of the radio link test packets
                sent between the KarlBridge and the remote station."
        ::= { kbWirelessStationEntry 27 }

kbWirelessTestOurTx OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The radio transmit rate of the KarlBridge's radio when
                performing the radio link test."
        ::= { kbWirelessStationEntry 28 }

kbWirelessTestOurRx OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The radio receive rate used by the KarlBridge
                when performing the radio link test."
        ::= { kbWirelessStationEntry 29 }

kbWirelessTestHisTx OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The radio transmit rate of the remote station's radio when
                performing the radio link test."
        ::= { kbWirelessStationEntry 30 }

kbWirelessTestHisRx OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The radio receive rate used by the remote station
                when performing the radio link test."
        ::= { kbWirelessStationEntry 31 }

kbWirelessTestOurCurSignalLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
```

```
        STATUS current
        DESCRIPTION
                "The current local signal level
                for the radio link test as recorded by the KarlBridge."
        ::= { kbWirelessStationEntry 32 }

kbWirelessTestOurCurNoiseLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The current local noise level
                for the radio link test as recorded by the KarlBridge."
        ::= { kbWirelessStationEntry 33 }

kbWirelessTestOurCurSignalQuality OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The current relative signal quality
                for the radio link test as recorded by the KarlBridge."
        ::= { kbWirelessStationEntry 34 }

kbWirelessTestOurCurSNR OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The current local Signal-to-Noise Ratio (SNR)
                for the radio link test as recorded by the KarlBridge."
        ::= { kbWirelessStationEntry 35 }

kbWirelessTestOurMinSignalLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The minimum local signal level
                recorded by the KarlBridge during the radio link test."
        ::= { kbWirelessStationEntry 36 }

kbWirelessTestOurMinNoiseLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The minimum local noise level
```

```
            recorded by the KarlBridge during the radio link test."
        ::= { kbWirelessStationEntry 37 }

kbWirelessTestOurMinSignalQuality OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The minimum local signal quality
            recorded by the KarlBridge during the radio link test."
        ::= { kbWirelessStationEntry 38 }

kbWirelessTestOurMinSNR OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The minimum local Signal-to-Noise Ratio (SNR)
            recorded by the KarlBridge during the radio link test."
        ::= { kbWirelessStationEntry 39 }

kbWirelessTestOurMaxSignalLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The maximum local signal level
            recorded by the KarlBridge during the radio link test."
        ::= { kbWirelessStationEntry 40 }

kbWirelessTestOurMaxNoiseLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The maximum local noise level
            recorded by the KarlBridge during the radio link test."
        ::= { kbWirelessStationEntry 41 }

kbWirelessTestOurMaxSignalQuality OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The maximum local signal quality
            recorded by the KarlBridge during the radio link test."
        ::= { kbWirelessStationEntry 42 }
```

```
kbWirelessTestOurMaxSNR OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The maximum local Signal-to-Noise Ratio (SNR)
                recorded by the KarlBridge during the radio link test."
        ::= { kbWirelessStationEntry 43 }

kbWirelessTestHisCurSignalLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The current signal level
                at the remote station for the radio link test."
        ::= { kbWirelessStationEntry 44 }

kbWirelessTestHisCurNoiseLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The current noise level
                at the remote station for the radio link test."
        ::= { kbWirelessStationEntry 45 }

kbWirelessTestHisCurSignalQuality OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The current signal quality
                at the remote station for the radio link test."
        ::= { kbWirelessStationEntry 46 }

kbWirelessTestHisCurSNR OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The current Signal-to-Noise Ratio (SNR)
                at the remote station for the radio link test."
        ::= { kbWirelessStationEntry 47 }

kbWirelessTestHisMinSignalLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
```

```
        STATUS current
        DESCRIPTION
                "The minimum signal level recorded
                at the remote station during the radio link test."
        ::= { kbWirelessStationEntry 48 }

kbWirelessTestHisMinNoiseLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The minimum noise level recorded
                at the remote station during the radio link test."
        ::= { kbWirelessStationEntry 49 }

kbWirelessTestHisMinSignalQuality OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The minimum signal quality recorded
                at the remote station during the radio link test."
        ::= { kbWirelessStationEntry 50 }

kbWirelessTestHisMinSNR OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The minimum signal level recorded
                at the remote station during the radio link test."
        ::= { kbWirelessStationEntry 51 }

kbWirelessTestHisMaxSignalLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The maximum signal level recorded
                at the remote station during the radio link test."
        ::= { kbWirelessStationEntry 52 }

kbWirelessTestHisMaxNoiseLevel OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The maximum remote noise level recorded
```

```
            at the remote station during the radio link test."
        ::= { kbWirelessStationEntry 53 }

kbWirelessTestHisMaxSignalQuality OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The maximum signal quality recorded
            at the remote station during the radio link test."
        ::= { kbWirelessStationEntry 54 }

kbWirelessTestHisMaxSNR OBJECT-TYPE
        SYNTAX Gauge32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The maximum Signal-to-Noise Ratio (SNR) recorded
            at the remote station during the radio link test."
        ::= { kbWirelessStationEntry 55 }

kbWirelessTestLinkUp OBJECT-TYPE
        SYNTAX INTEGER
            {
                down(0),
                up(1)
            }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The status of the radio link test. Possible values are:
                down(0)       -- the radio link test IS NOT currently running
                up(1)   -- the radio link test IS currently running"
        ::= { kbWirelessStationEntry 56 }

kbWirelessTestLostLink OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The number of time this link was lost while performing the radio link
            test."
        ::= { kbWirelessStationEntry 57 }

kbWirelessTestLostTestPkts OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS current
```

```
        DESCRIPTION
                "The total number of radio link test packets
                that were lost during the test interval."
        ::= { kbWirelessStationEntry 58 }

kbWirelessStationRadioType OBJECT-TYPE
        SYNTAX INTEGER
                {
                waveLAN_I(0),
                clarion_M10(1),
                waveLAN_IEEE(2),
                microwave(3),
                radioLAN(4)
                }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The type of radio the remote station is using on this interface.
                Possible values include:
                        waveLAN_I(0)        -- legacy Lucent Wireless-I radio (900 Hz)
                        clarion_M10(1)      -- Clarion M10 radio
                        waveLAN_IEEE(2)  -- Lucent ORiNOCO (WaveLAN) IEEE
                        802.11 radio (2.4 GHz)
                        microwave(3) -- some type of microwave radio
                        radioLAN(4)  -- some type of RadioLAN radio"
        ::= { kbWirelessStationEntry 59 }

kbWirelessRecordType OBJECT-TYPE
        SYNTAX INTEGER
                {
                        linktest(1),
                        turboCell(2),
                        combination(3),
                }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The type of this record entry. Possible values include:
                        linktest(1)     -- This record contains ONLY LinkTest results
                        turboCell(2)    -- This record contains ONLY a TurboCell station
                        entry
                        combination(3)      -- This record contains BOTH LinkTest
                        results AND a TurboCell station entry"
        ::= { kbWirelessStationEntry 60 }

kbWirelessStationPktReceives OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
```

```
        STATUS current
        DESCRIPTION
                "The number of packets the KarlBridge has received on this link."
        ::= { kbWirelessStationEntry 61 }

kbWirelessStationReceives OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of wireless transmissions the KarlBridge has received on
                this link."
        ::= { kbWirelessStationEntry 62 }

kbWirelessStationBytesReceives OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of bytes received by the KarlBridge on this link,
                including wireless packet headers."
        ::= { kbWirelessStationEntry 63 }

kbWirelessStationBytesTransmits OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of bytes transmitted by the KarlBridge on this link,
                including wireless packet headers."
        ::= { kbWirelessStationEntry 64 }

kbWirelessRegistrationRecord OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS read-only
        STATUS obsolete
        DESCRIPTION
                "For testing purposes only."
        ::= { kbWirelessStationEntry 65 }

kbWirelessStationFragmentDiscards OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of times on this connection that the KarlBridge
                has discarded a fragment. A fragment is discarded because not all
                fragments needed to reconstruct a packet are present or
```

```
                we need room for a more recent fragment."
        ::= { kbWirelessStationEntry 66 }

kbWirelessStationFragmentMissings OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of times on this connection that the KarlBridge
                has detected that a fragment needed to recontruct a packet is missing."
        ::= { kbWirelessStationEntry 67 }

kbWirelessStationFragmentLostFrames OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of times on this connection that the KarlBridge
                detected that a frame containing fragments needed to
                reconstruct a packet is lost."
        ::= { kbWirelessStationEntry 68 }

kbWirelessStationFragmentErrors OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The number of times on this connection the KarlBridge
                has encountered an error while reconstructing a packet from fragments."
        ::= { kbWirelessStationEntry 69 }
END
```

ANEXO 2 – TIPOS DE INTERFACE DE REDE

Número	Tipo
1	Other
2	Regular1822
3	Hdh1822
4	Ddn-
5	Rfc877 –x25
6	EthernetCsmacd
7	Iso88023Csmacd
8	Iso880024TokenBus
9	Iso88025TokenRing
10	Iso88026Man
11	StarLan
12	Proteon-10Mbit
13	Proteon-80Mbit
14	Hyperchannel
15	Fddi
16	Lapb
17	Sdlc
18	Dsl
19	El
20	BasicISDN
21	PrimaryISDN
22	PropPointToPointSerial
23	PPP

24	SoftwareLoopBack
25	Eon
26	Ethernet-3Mbit
27	Nsip
28	Slip
29	Ultra
30	Ds3
31	Sip
32	Frame-relay
33	Rs232
34	Para
35	Arcnet
36	ArcnetPlus
37	Atm
38	Miox25
39	Sonet
40	X25ple
41	Iso8802llc
42	LocalTalk
43	SmdsDxi
44	FrameRelayService
45	V35
46	Hssi
47	Hippi
48	Modem
49	Aal5

50	SonetPath
51	SonetVT
52	SmdsIcip
53	PropVirtual
54	PropMultiplexor

ANEXO 3 – A VARIÁVEL *SYSSERVICES*

Serviços: um valor que indica o conjunto de serviços que esta entidade gerenciada primariamente oferece. Este valor é uma soma. Esta soma toma inicialmente valor 0 (zero), então, para cada camada L, na faixa de 1 à 7, para qual este elemento executa transações, o valor 2^{L-1} é adicionado à soma. Para o contexto da pilha de protocolos para Internet, os valore devem ser calculados de acordo com a Tabela 6.

Camada	Funcionalidade
1	Física (exemplo, repetidores)
2	Enlace ou sub-rede (exemplo, *bridges*)
3	Internet (exemplo, IP *gateways*)
4	Fim-a-fim (exemplo, IP *hosts*)
7	Aplicações (exemplo, *mail relays*)

Tabela 6 – Valores para soma do objeto sysServices