GUSTAVO CARVALHO GOMES

HABILITAÇÃO DE INFORMAÇÕES DE GERENCIAMENTO SNMP PARA O SISTEMA WIRELESS DA UFLA

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado II, para a obtenção do título de Bacharel em Ciência da Computação.

Orientador

Prof. Ricardo Martins de Abreu Silva

LAVRAS
MINAS GERAIS - BRASIL
2002

GUSTAVO CARVALHO GOMES

HABILITAÇÃO DE INFORMAÇÕES DE GERENCIAMENTO SNMP PARA O SISTEMA WIRELESS DA UFLA

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado II, para a obtenção do título de Bacharel em Ciência da Computação.

APROVADA em _____ de _____ de _____.

_____
Prof. Rêmulo Maia Alves

_____
Prof. Anderson Bernardo dos Santos

_____
Prof. Ricardo Martins de Abreu Silva
UFLA
(Orientador)

LAVRAS
MINAS GERAIS - BRASIL

Ao meu pai, pelo incentivo e apoio

À minha mãe, pelo carinho e amizade

A Marcela, pelo amor e paciência,

Dedico.

Agradecimentos

A todos os meus familiares que sempre me incentivaram a lutar por meus sonhos.

Aos colegas e amigos que, de uma forma ou de outra, colaboraram com este trabalho. Em especial ao Rodolfo por ter dividido o seu tempo comigo e me ajudado.

Aos professores Ricardo Martins de Abreu Silva, Rêmulo Maia Alves e Anderson Bernardo dos Santos pela orientação, colaboração e boa vontade.

E principalmente a Deus, pois sem Ele nada seria possível.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

A importância de gerência de redes nos dias atuais é incontestável. Redes de computadores estão crescendo em tamanho e complexidade e se torna necessário o desenvolvimento de ferramentas para auxiliar profissionais na tarefa de gerenciar redes de computadores.

A UFLA possui uma rede *Wireless* que tem como ponto de acesso uma *bridge* modelo AP-1000 da marca Lucent. O equipamento vem com um software de gerenciamento SNMP apenas para o sistema operacional Windows. Nosso projeto faz o estudo do protocolo SNMP e posteriormente o desenvolvimento de uma ferramenta de gerenciamento SNMP, utilizando a linguagem de programação Perl, para o sistema operacional Linux.

Abstract

The importance of management of nets in the current days is unanswerable. Nets of computers are growing in size and complexity and if it turns necessary the development of tools to aid professionals in the task of managing nets of computers.

The UFLA possesses a net Wireless that has about access point a bridge model AP-1000 of the mark Lucent. The equipment just comes with administration software SNMP for the operating system Windows. Our project makes the study of the protocol SNMP and later the development of an administration tool SNMP, using the programming language Perl, for the operating system Linux.

1. Introdução

Nos últimos anos, as redes de computadores têm tido especial destaque, quer pela proliferação das redes locais (LAN[1]), interligando computadores pessoais, estações de trabalhos, computadores de grande porte (*mainframes*), quer pelo aumento do acesso a redes de longa distância (WAN[2]) através da Internet, por exemplo, facilitando o processo de comunicação. Elas têm crescido em tamanho, complexidade e heterogeneidade exigindo um grande investimento na área de gerência de redes.

Com o intuito de padronizar ferramentas para gerência de redes que possam ser usadas em diversos tipos de produtos (independentes de fabricante), ações para a padronização vêm sendo tomadas através do uso de protocolos abertos de gerência e da definição de bases de dados que modelam componentes da rede.

1.1. Descrição do Trabalho Proposto

O presente trabalho tem como objetivo a implementação para o ambiente (sistema operacional) Linux de uma extensão para obtenção de novas informações de gerenciamento, através do protocolo SNMP, para a rede *Wireless* – UFLA. O sistema operacional a ser utilizado será o Linux, uma vez que já existe uma ferramenta para o gerenciamento/monitoramento da Rede *Wireless* – UFLA para o sistema operacional Windows.

---

[1] LAN – *Local Area Network*

[2] WAN - *Wide Area Network*

1.2. Motivação

1.2.1. A Necessidade de uma Ferramenta de Gerência

O crescimento exponencial do número de equipamentos interconectados através de redes de computadores tem resultado em enormes dificuldades para os administradores das redes. Para [Bri93] o contínuo crescimento em número e diversidade dos componentes das redes tem tornado a atividade de gerenciamento de rede cada vez mais complexa. Além disso, temos uma quantidade enorme de novos equipamentos sendo lançados pelos fabricantes a cada dia que passa. Segundo [Tan97] existem muitos fabricantes e fornecedores de rede, cada qual com sua própria concepção. Em suma, haveria um caos completo, e os usuários nada conseguiriam.

1.2.2. Necessidades da Instituição

A UFLA - Universidade Federal de Lavras, prove acesso a Internet para a população de Lavras e região através de uma Rede *Wireless*. Portanto, há necessidade para a instituição de se fazer o gerenciamento dos equipamentos. Atualmente a UFLA dispõe de um *software* para tal gerenciamento, entretanto ele se restringe apenas ao ambiente (sistema operacional) Windows, e seu conjunto de estatísticas é limitado.

1.2.3. Mercado para *Wireless*

A transmissão de dados *Wireless* é uma das soluções tecnológicas mais avançadas na área de telecomunicações. Ela encurta distâncias de forma impressionante, permitindo aos usuários implantarem uma rede de comunicação

completa, sem o emprego de cabos. Segundo [Tan97], essa tecnologia está dando origem a um mercado bastante promissor. De acordo com dados do *The Strategis Group*, nos EUA em 2007 cerca de 60% da população estará fazendo uso de dispositivos móveis conectados à Internet. Este crescimento será alavancado pela popularização dos serviços de banda larga no país. Na União Européia, em 2010, a previsão é de que 75% das pessoas entre 15 e 50 anos carreguem consigo um dispositivo móvel. De acordo com dados da *Frost & Sullivan*, por exemplo, em 2000 o Brasil movimentou mais de US$ 15 bilhões com seu mercado *Wireless* [Ibi01].

1.2.4. Crescimento do Sistema Operacional Linux

O Linux vem ganhando espaço no mercado principalmente pela questão da segurança, estabilidade, performance e por não haver necessidade de se comprar a licença para poder usá-lo. Por todos esses pontos o Linux vem sendo utilizado em muitas empresas, instituições de ensino e órgãos governamentais, diminuindo muito os gastos com licenças de software.

1.2.5. SNMP (*Simple Network Management Protocol*)

O protocolo SNMP (*Simple Network Management Protocol*), vem sendo, ao longo dos últimos anos, reconhecido como o padrão de fato para o gerenciamento de equipamentos de rede. E este protocolo tem obtido tanto sucesso em suas implementações que sua utilização não se restringe mais somente para gerência dos equipamentos tradicionais de rede.

Atualmente, qualquer dispositivo que possua uma interface de rede e que pretenda estar condizente com as perspectivas do mercado, deve possuir a capacidade de ser administrado e/ou gerenciado via SNMP. E esta "tendência SNMP" tem levado a que fabricantes dos mais diversos equipamentos, tais como fotocopiadoras e impressoras, implementem interfaces de rede para que seus equipamentos possam estar alinhados com esta realidade científica que hoje também virou uma realidade de mercado.

O modelo de gerência do SNMP possui a palavra "simples" no próprio nome e ele realmente o é, se comparado com o modelo *Reference Model – Open System Interconnection* - RM-OSI, modelo este que foi cuidadosa e demoradamente desenvolvido enquanto o SNMP interinamente cumpria suas tarefas. Mas o SNMP obteve tanto sucesso que, quando o OSI ficou pronto, em 1993, já existiam centenas de produtos com o SNMP implementado [Vie97].

Dentre suas simplicidades, uma que encoraja muito os fabricantes dos equipamentos é a facilidade de implementação em hardware [Kor94]. Esta característica parece ter sido fundamental para a difusão do SNMP.

2. Referencial Teórico

2.1. Gerência de redes

A gerência de redes está associada com as atividades necessárias para assegurar o bom funcionamento de uma rede. Existem cinco áreas de gerência de redes.

- A **gerência de configuração** é responsável pela descoberta, manutenção e monitoração de mudanças à estrutura física e lógica da rede. As funções básicas desta área de gerência são: coleta de informações sobre a configuração, geração de eventos, atribuição de valores iniciais aos parâmetros dos elementos gerenciados, registro de informações, alteração de configuração dos elementos gerenciados, início e encerramento de operação dos elementos gerenciados. [Oda94]

- A **gerência de faltas** é responsável pela detecção, isolamento e conserto de falhas na rede. As funções básicas são: detecção, isolamento e antecipação de falhas, supervisão de alarmes, restabelecimento dos elementos com problema, testes, registro de ocorrência e emissão de relatórios para análise. [Oda94] e [Cis99]

- A seguir vem a **gerência de desempenho** que é responsável pela monitoração e análise do desempenho e planejamento da capacidade. Esta área de gerência deve selecionar os indicadores de desempenho, monitorar e analisar o desempenho, planejar a capacidade e alterar o modo de operação. [Bri93] e [Her94]

- A **gerência de segurança** é responsável pela proteção dos elementos da rede, monitorando e detectando violações da política de segurança estabelecida. Esta área de gerência preocupa-se com a criação, monitoração e manutenção de serviços de segurança e com a manutenção de *logs* de segurança. [Cis99].

- Finalmente, a **gerência de contabilidade** é responsável pela contabilização e verificação de limites da utilização de recursos da rede, com a divisão de contas feita por usuários ou grupos de usuários. As funções básicas são: a coleta de informações sobre a utilização, o estabelecimento de quotas de utilização e escalas de tarifação, e a aplicação de tarifas e faturamento.

2.2. SNMP (*Simple Network Management Protocol*)

Na complexa rede de roteadores, *switches* e servidores dos dias atuais, talvez pareça aterrorizante gerenciar todos os dispositivos existentes em sua rede e certificar-se de que estejam não somente em execução, como também funcionando perfeitamente. É exatamente nesse momento que o *Simple Network Management Protocol* (SNMP) pode entrar em ação. O SNMP foi lançado em 1988 para atender à necessidade cada vez maior de um padrão para gerenciar os dispositivos de IP (*Internet Protocol*). O SNMP oferece aos usuários um conjunto "simples" de operações que permitem o gerenciamento remoto desses dispositivos. [Mau01]

2.2.1. Versões de RFCs e SNMP

A *Internet Engineering Task Force* (IETF) é responsável pela definição dos protocolos que controlam o tráfego na Internet, incluindo o SNMP. A IETF

publica *Requests for Comments* (RFCs), que são especificações para os diversos protocolos existentes no mundo do IP. Primeiramente, os documentos se submetem ao rastreamento de padrões como padrões *sugeridos*, depois passam para o status de *draft*. Quando um *draft* final é ocasionalmente aprovado, a RFC recebe o status de *standard* (*padrão*).

Duas outras designações de rastreamento de padrões, *histórico* e *experimental*, definem, respectivamente, um documento substituído por uma RFC mais recente e um documento que ainda não está pronto para se tornar um padrão. A seguir temos uma lista que engloba todas as versões atuais do SNMP e o status emitido pela IETF pra cada uma.

- *SNMP Version 1* (SNMPv1) – é a versão padrão atual do protocolo SNMP, definida na RFC 1157 e é um padrão completo da IETF. A segurança do SNMPv1 baseia-se em *comunidades*, que não são nada mais do que senhas: *strings* de texto puro que permitem que qualquer aplicativo baseado em SNMP, que reconheça a string, tenha acesso a informações de gerenciamento de um dispositivo. Geralmente, existem três comunidades no SNMPv1: *read-only, read-write* e *trap.*

- *SNMP Version 2* (SNMPv2) – é freqüentemente citado como SNMPv2 baseado em *strings* de comunidade. Essa versão tem a denominação técnica de SNMPv2c. Ela está definida na RFC 1905, RFC 1906 e RFC 1907, é uma IETF com status experimental. Embora seja experimental, alguns fornecedores já começaram a aceitá-la na prática.

- *SNMP Version 3* (SNMPv3) – será a próxima versão do protocolo a alcançar o status completo da IETF. No momento, é um padrão

sugerido, definido na RFC 1905, RFC 1906, RFC 1907, RFC 2571, RFC 2572, RFC 2573, RFC 2574 e RFC 2575, que inclui suporte para autenticação rigorosa e comunicação privativa entre as entidades gerenciadas.

2.2.2. Gerentes e agentes

No mundo do SNMP, existem dois tipos de entidades: gerentes e agentes. Um gerente é um servidor executando algum tipo de sistema de software que pode lidar com tarefas de gerenciamento de uma rede. Os gerentes costumam ser chamados NMS (*Network Management Stations*). Uma NMS é responsável pela operação de *polling* e por receber *traps* de agentes na rede.

Um *poll*, no contexto de gerenciamento de rede, é a operação de consultar informações em um agente (roteador, comutador, etc). Essas informações podem ser utilizadas posteriormente para detectar se ocorreu algum tipo de evento desastroso. Uma *trap* é um método utilizado por um agente para informar à NMS que algo aconteceu.

A segunda entidade, o agente, é a peça de software executada nos dispositivos da rede gerenciados. Pode ser um programa separado ou pode ser incorporado ao sistema operacional. Atualmente, a maioria dos dispositivos de IP é fornecida com alguma modalidade de agente SNMP interno. [Mau01]

2.2.3. Estrutura de Informações de gerenciamento e MIB

A SMI (*Structure of Management Information*) é um método para definir objetos gerenciados e os respectivos comportamentos. No jargão utilizado pelo

SNMP, essas variáveis são chamadas de objetos, não confundir com objetos no sentido de um sistema orientado a objetos. Essas variáveis têm apenas estados e não dispõem de métodos (além da leitura e da escrita de seus valores). Um agente possui uma lista dos objetos por ele rastreados. Esse tipo de objeto é o status operacional de uma interface. Essa lista define coletivamente as informações que a NMS pode utilizar para detectar o funcionamento geral do dispositivo em que o agente reside.

A MIB (*Management Information Base*) pode ser considerado um banco de dados de objetos gerenciados que o agente rastreia. Todo tipo de informação sobre status e estatísticas acessados pela NMS é definida em uma MIB. A SMI é um método para definir objetos gerenciados, enquanto a MIB é a definição (por meio da sintaxe da SMI) dos próprios objetos.

Um agente pode implementar várias MIBs, mas todos os agentes implementam uma MIB específica denominada MIB-II (RFC 1213). Esse padrão define variáveis para elementos como dados estatísticos de uma interface, assim como alguns outros aspectos relacionados ao próprio sistema. O principal objetivo da MIB-II é fornecer informações gerais sobre gerenciamento de TCP/IP e não engloba todo item possível que um fornecedor gerenciaria em um dispositivo específico. [Mau01]

Em resumo, a gerência de uma rede utilizando o protocolo SNMP, consiste em quatro componentes principais: [Tan97]

1. Nós Gerenciados;
2. Estações de Gerenciamento;
3. Informações de Gerenciamento;
4. Um protocolo de Gerenciamento;

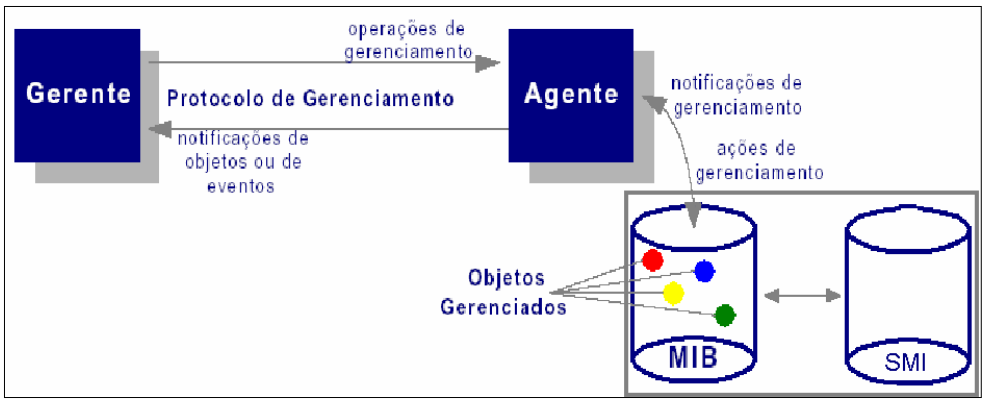

Figura 1 – Principais componentes do modelo SNMP

2.2.4. SNMP e UDP

O SNMP usa o *User Datagram Protocol* (UDP) como protocolo de transporte para passagem de dados entre gerentes e agentes. O UDP, definido na RFC 768, foi escolhido dentre os dois protocolos da camada de transporte da Internet (TCP e UDP) por não ter conexão; isto é, nenhuma conexão ponto-a-ponto é estabelecida entre o agente e a NMS quando os datagramas (pacotes) são transferidos de um lado para o outro. Esse aspecto do UDP torna-o não confiável, uma vez que não existe confirmação de datagramas perdidos no nível do protocolo. O próprio aplicativo do SNMP se encarregar de detectar se ocorreu perda de datagramas e retransmite-os se necessário. Em geral, isso é feito com um simples *timeout* (tempo de espera). A NMS envia uma solicitação de UDP para um agente e aguarda uma resposta. O intervalo de tempo em que a NMS espera depende da configuração na NMS. Se o *timeout* for decorrido e a

NMS não receber resposta do agente, será presumida a perda do pacote e a NMS retransmite a solicitação. O número de retransmissões de pacotes pela NMS também é configurável.

Pelo menos no que diz respeito às solicitações regulares de informações, a natureza não confiável do UDP não é um grande problema. Na pior das hipóteses, a estação de gerenciamento emite uma solicitação e nunca recebe uma resposta. Para as *traps*, a situação é um pouco diferente. Se um agente envia uma *trap* e ela nunca chega, a NMS não sabe se essas *trap* sequer foram enviada. Nem o próprio agente detecta a necessidade de reenvio da *trap*, porque a NMS não é obrigada a enviar uma resposta para o agente, confirmando o recebimento da *trap*.

O reverso da natureza incerta do UDP é o baixo *overhead*, de modo que o impacto sobre o desempenho da rede é menor. O SNMP foi implementado através da arquitetura de referência TCP/IP (Modelo de Referência TCP/IP); entretanto, essa implementação está voltada para as situações de casos especiais em que alguém está desenvolvendo um agente para um equipamento proprietário. Em uma rede gerenciada e muito congestionada, o SNMP por meio do TCP é uma péssima idéia. É importante entender que o TCP não é um mágico e que SNMP foi elaborado para trabalhar com redes enfrentando problemas - se sua rede nunca falhasse, não seria necessário monitorá-la. Quando uma rede está falhando, um protocolo que tenta obter os dados, mas desiste quando não consegue, é certamente uma opção de design melhor do que um protocolo que inunda a rede com retransmissões, na tentativa de obter credibilidade.

O SNMP usa a porta 161 do UDP para enviar e receber solicitações e a porta 162 para receber *traps* de dispositivos gerenciados. Todo dispositivo que implementa o SNMP deve utilizar esses números de porta como *defaults*, mas

alguns fornecedores permitem modificar as portas *defaults* na configuração do agente. Se esses *defaults* forem modificados, a NMS deve ser informada sobre essas alterações para consultar o dispositivo por meio das portas corretas.

A Figura 2 mostra o conjunto de protocolos TCP/IP, que é a base de toda a comunicação por TCP/IP. Atualmente, todo dispositivo para comunicação na Internet (como os sistemas Windows NT, servidores Unix, roteadores Cisco etc), deve utilizar esse conjunto de protocolos. Esse modelo é geralmente citado como uma pilha de protocolos, porque cada camada usa as informações da camada posicionada imediatamente abaixo.

Quando uma NMS ou um agente precisa executar uma função de SNMP (como uma solicitação ou uma *trap*), ocorrem os seguintes eventos na pilha de protocolos:

- **Aplicativo:** Primeiramente, o aplicativo de SNMP (NMS ou agente) determina o que fará. Por exemplo, ele pode enviar uma solicitação de SNMP para um agente, pode enviar uma resposta a uma solicitação de SNMP (enviada a partir do agente) ou enviar uma *trap* para uma NMS. A camada do aplicativo fornece serviços para um usuário final, como um operador solicitando informações de *status* de uma porta em um comutador de Ethernet.

- **UDP -** A camada seguinte, UDP, permite a comunicação entre dois hosts. O cabeçalho do UDP contém, entre outras informações, a porta de destino do dispositivo para o qual ela está enviando uma solicitação ou uma *trap*. Essa porta de destino será a 161 (consulta) ou 162 (*trap*).

- **IP -** A camada IP tenta fornecer o pacote de SNMP ao destino almejado, conforme especificado pelo respectivo endereço IP.

- **Medium Acess Control (MAC) -** O último evento que deve ocorrer para um pacote de SNMP alcançar seu destino é ser controlado na rede física, onde pode ser direcionado para o destino final.

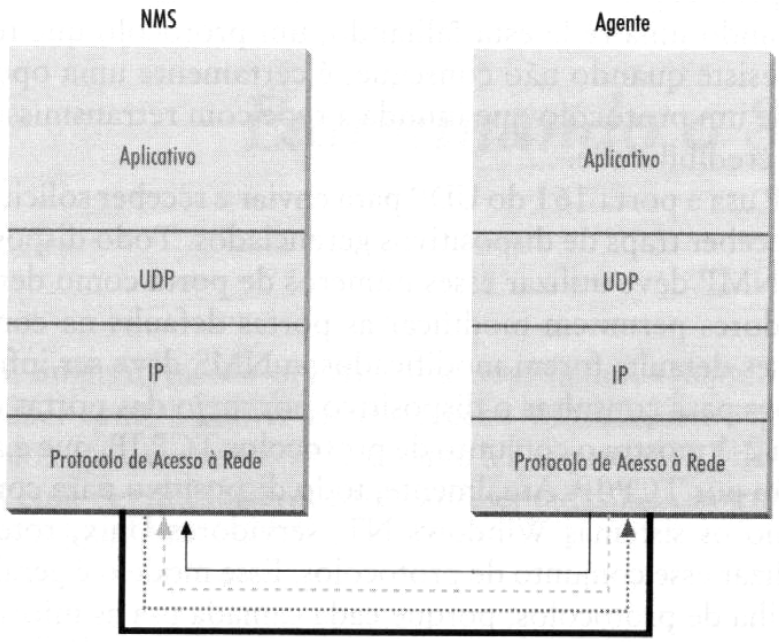

Figura 2 – Modelo de comunicação por TCP/IP e o SNMP

A camada MAC é formada pelos verdadeiros *drivers* de hardware e de dispositivo que colocam seus dados em um cabeamento físico, como um cartão Ethernet. Essa camada também é responsável por receber os pacotes da rede

física e reenviá-los para a pilha de protocolos, para que sejam processados pela camada do aplicativo (SNMP, nesse caso). [Mau01]

2.2.5. Comunidades de SNMP

O SNMPv1 e SNMPv2 usam o conceito de comunidades para definir uma confiabilidade entre gerenciadores e agentes. Um agente é configurado com três nomes de comunidade: *read-only*, *read-write* e *trap*. Os nomes de comunidades são basicamente senhas; não existe diferença entre uma *string* de comunidade e a senha que você usa para acessar sua conta no computador. As três *strings* de comunidade controlam tipos diferentes de atividades. Como o próprio nome sugere, a *string* de comunidade *read-only* permite ler valores de dados, sem modificá-los. A comunidade *read-write* pode ler e modificar valores de dados; com a *string* de comunidade *read-write*, é possível ler os contadores, redefinir seus valores e até as interfaces ou executar outras ações que modifiquem a configuração do roteador. Por último, a string de comunidade *trap* permite receber *traps* (notificações assíncrona) do agente.

A maioria dos fornecedores entrega seus equipamentos com *strings* de comunidade *defaults*, geralmente *public* para a comunidade *read-only* e *private* para a comunidade *read-write*. É importante modificar esses *defaults* antes de instalar o dispositivo definitivamente na rede. Ao configurar um agente da SNMP, convém configurar o destino de sua *trap*, que é o endereço para o qual o agente enviará todas as *traps* por eles geradas. Além disso, como as *strings* de comunidade do SNMP são enviadas em texto comum, é possível configurar um agente para enviar uma *trap* de falha de autenticação de SNMP quando alguém tentar consultar um dispositivo com uma *string* de comunidade incorreta. Entre

outros aspectos, as *traps* de falha de autenticação podem ser muito úteis ao determinar quando um invasor estiver tentando acessar sua rede.

Como as *strings* de comunidade são basicamente senhas, para selecioná-las use as mesmas regras aplicadas às senhas de usuário do Unix ou NT: nenhuma palavra do dicionário, nomes de esposas etc. Geralmente, é uma boa idéia usar uma *string* alfanumérica com combinação de maiúsculas/minúsculas. Conforme mencionado anteriormente, o problema com a autenticação do SNMP é que as *strings* de comunidade são enviadas em texto simples, o que facilita a interceptação dessas *strings* e o uso indevido contra você. O SNMPv3 trata dessas questões ao permitir entre outros aspectos, autenticação e comunicação segura entre os dispositivos do SNMP.

Existem meios de reduzir os riscos de ataque. Os *firewalls* ou filtros de IP minimizam a chance de alguém prejudicar um dispositivo gerenciado em uma rede, atacando-o por meio de SNMP. Você pode configurar o *firewall* para aceitar o tráfego do UDP somente de uma lista de *hosts* conhecidos. Por exemplo é possível o tráfego do UDP na porta 161 (solicitações do SNMP) em sua rede somente quando procedente de uma de suas estações de gerenciamento da rede.

O mesmo é válido para as *traps*; você pode configurar o roteador para aceitar o tráfego de UDP pela porta 162 para sua NMS somente se procedente de um dos *hosts* sendo monitorados. Os *firewalls* não são totalmente eficientes mas precauções simples, como estas, reduzem bastante seus riscos.

É importante saber que se alguém tiver acesso de leitura-gravação a qualquer um de seus dispositivos do SNMP, poderá controlá-los por meio do

SNMP (por exemplo, poderá definir as interfaces do roteador, desabilitar portas ou até modificar as tabelas de direcionamento). Uma maneira de proteger as *strings* de comunidade é usar uma *Virtual Private Newwork* (VPN) para garantir que o tráfego da rede seja codificado. Outro método é mudar freqüentemente as *strings* de comunidade. Em uma rede pequena, não é difícil modificar essas *strings*, mas em uma rede que ocupa quarteirões da cidade ou muito mais, e possui dezenas (ou centenas ou milhares) de *hosts* gerenciados, pode ser problemático mudar as *strings* de comunidade. Uma solução fácil é escrever um script simples em *Perl* que use o SNMP para alterar essas *strings* em seus dispositivos. [Mau01]

2.2.6. *Struture of Management Information*

O primeiro passo para entender que tipo de informação um dispositivo pode fornecer é conhecer como esses dados são representados no contexto do SNMP. A *Structure of Management Information Version 1* (SMIv1, RFC 1155) faz exatamente isso: define com exatidão como os objetos gerenciados são nomeados e especifica os respectivos tipos de dados associados. A *Structure of Management Information Version* 2 (SMIv2, RFC 2578) fornece otimizações para o SNPMv2. Começaremos com a SM1v1 e abordaremos a SMIv2 em seguida.

A definição de objetos gerenciados pode ser fragmentada em três atributos:

- **Nome:** O nome ou identificador de objeto (OID - *Object Identifier*) define com exclusividade um objeto gerenciado. Geralmente, os nomes são exibidos em dois formatos: numérico e o "legível pelo ser

humano". Em ambos os casos, os nomes são longos e inconvenientes.

- **Tipo e sintaxe:** O tipo de dado de um objeto gerenciado é definido por meio de um sub-conjunto da *Abstract Syntax Notation One* (ASN.1). A ASN.1 é um meio de especificar o modo como os dados são representados e transmitidos entre gerenciados e agentes, no contexto do SNMP. A vantagem da ASN.1 em relação aos tipos de dados de outras linguagens, é o fato de que a notação independe da máquina. Isso significa que um PC executando o Windows NT pode se comunicar com uma máquina do Sun SPARC sem considerar aspectos como o seqüenciamento de bytes.

- **Codificação:** Uma única instância de um objeto gerenciado é codificada em uma *string* de octetos por meio do método *Basic Encoding Rules* (BER). O método BER define o modo de codificação e decodificação dos objetos para que sejam transmitidos através de um meio de transporte, como a Ethernet.

2.2.6.1. Nomeando OIDs

Os objetos gerenciados são organizados em uma hierarquia em árvore. Essa estrutura é base do esquema de atribuição de nomes do SNMP. Um ID de objetos é formado por uma seqüência de inteiros baseada nos nós da árvore, separada por pontos. Embora exista uma forma legível ao ser humano mais amistosa do que uma *string* de números, essa forma não é nada mais do que uma seqüência de nomes separados por pontos, cada qual representando um nó da árvore. Assim, é possível utilizar os próprios números ou uma seqüência de

nomes que representam os números. A figura 3 mostra os primeiros níveis dessa árvore.

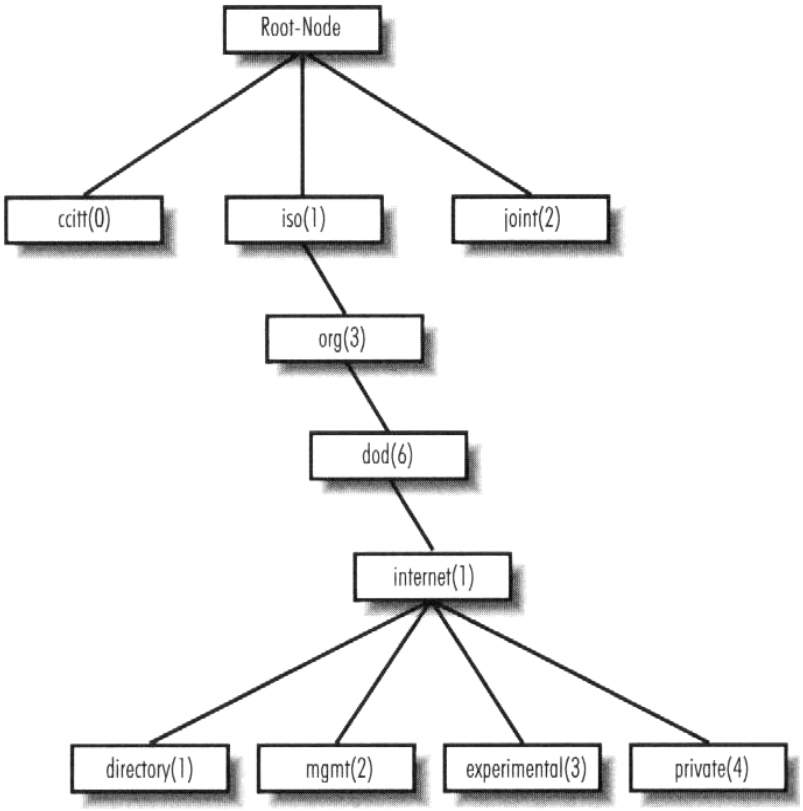

Figura 3 – Árvore de objetos da SMI

Na árvore de objetos, o nó posicionado no início da árvore é denominado raiz, tudo o que tiver filhos será uma subárvore e tudo o que não tiver filhos será chamado nó de folha. Por exemplo, a raiz na Figura 3, o ponto inicial da árvore, é denominada "Root-Node" (Nó-raiz). A respectiva subárvore é formada por ccitt(0), iso(1) e joint(2). Nessa ilustração, iso(1) é o único nó que contém uma subárvore; os outros dois nós são nós de folha, ccitt(0) e joint(2) não pertencem ao SNMP e não serão discutidos aqui.

Abordaremos neste estudo a subárvore iso(1).org(3).dod(6).internet(1), representada na forma de OID numérica e nome em texto associado. A notação decimal de ponto é o modo de representação interna de um objeto gerenciado dentro de um agente; o nome em texto, como um nome de domínio IP, evita a necessidade de memorizar *strings* de inteiros longas e cansativas.

A ramificação directory não é usada no momento. A ramificação management, ou mgmt, define um conjunto padrão de objetos de gerenciamento da Internet. A ramificação experimental é reservada para fins de teste e pesquisa.

Os objetos na ramificação private são definitos unilateralmente, o que significa que os indivíduos e as organizações são responsáveis pela definição dos objetos dessa ramificação. Examine a definição da subárvore internet e de quatro de suas subárvores:

internet	OBJETCT IDENTIFIER ::= (iso org(3) dod (6) 1)
directory	OBJETCT IDENTIFIER ::= (internet 1)
mgmt	OBJETCT IDENTIFIER ::= (internet 2)
experimental	OBJETCT IDENTIFIER ::= (internet 3)
private	OBJETCT IDENTIFIER ::= (internet 4)

A primeira linha declara internet como a OID 1.3.6.1. que está definida como uma subárvore de iso.org.dod ou 1.3.6. (::= é um operador de definição). As quatro últimas declarações são semelhantes, mas definem as outras ramificações que pertencem à ramificação Internet. Para a ramificação directory, a notação (internet 1) informa que ela faz parte da ramificação internet e que a OID é 1.3.6.1.1. A OID de mgmt é 1.3.6.1.2, e assim por diante.

No momento, existe uma única ramificação na subárvore private, usada para permitir que os fornecedores de hardware e software definam seus objetos privados para qualquer tipo de hardware ou software que o SNMP deva gerenciar. A respectiva definição da SMI é:

enterprises OBJECT IDENTIFIER ::= {private 1}

A *Internet Assigned Numbers Authority* (IANA) gerencia atualmente todas as atribuições de números de empresa privada para indivíduos, instituições organizações, empresas etc.

2.2.7. SMI na Versão 2

A SMIv2 estende a árvore de objetos da SMI ao adicionar a ramificação *snmpV2* à subárvore *internet*. A Figura 4 mostra como os objetos de *snmpV2* se enquadram no cenário maior; a OID dessa nova ramificação *é 1.3.6.1.6.3.1.1* ou *iso.org.dod.internet.snmpV2.snmpModules.snmpMIB.snmpMJBObjects*.
[Mau01]

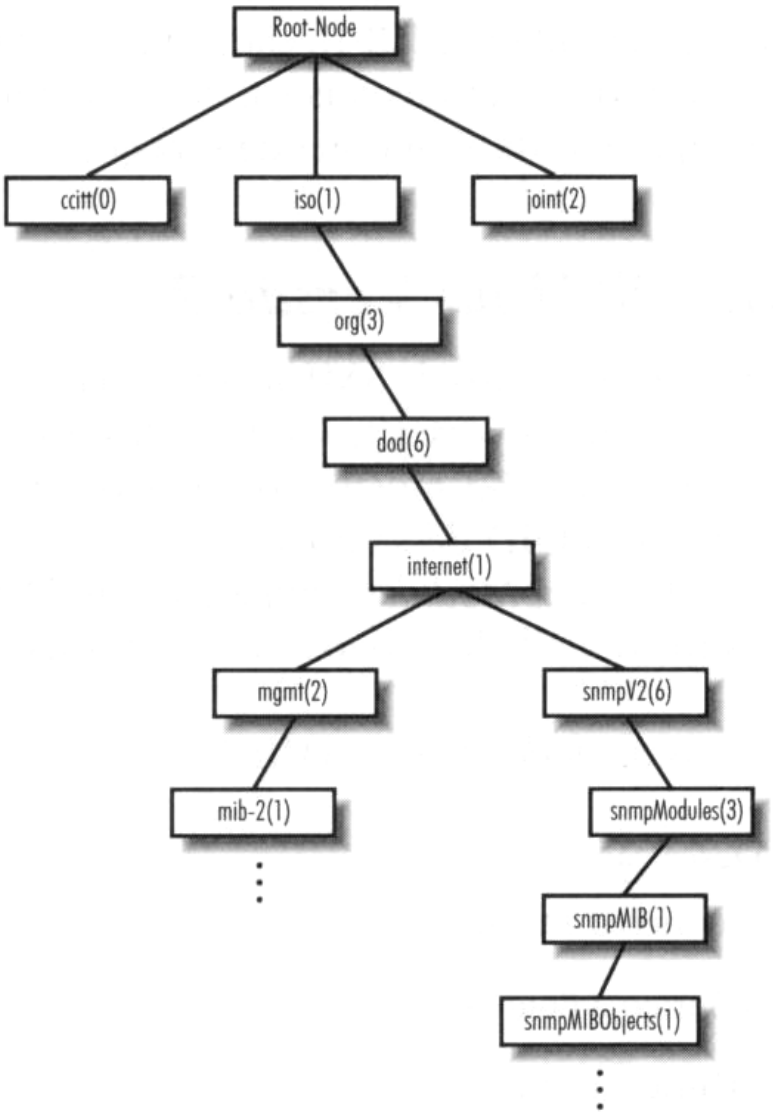

Figura 4 – Árvore de registros da SMIv2 para o SNMPv2

2.2.8. MIB-II

A MIB-II é um grupo de gerenciamento muito importante, porque cada dispositivo com suporte para o SNMP também deve aceitar a MIB-1I. Não descreveremos com detalhes cada objeto na MIB; definiremos apenas as subárvores. A seção da *RFC1213-MIB* que define as OIDs básicas da sub-árvore *mib-2* é parecida com esta:

```
mib-2          OBJECT IDENTIFIER ::=      { mgmt 1 }
system         OBJECT IDENTIFIER ::=      { mib-2 1 }
interfaces     OBJECT IDENTIFIER ::=      { mib-2 2 }
at             OBJECT IDENTIFIER ::=      { mib-2 3 }
ip             OBJECT IDENTIFIER ::=      { mib-2 4 }
icmp           OBJECT IDENTIFIER ::=      { mib-2 5 }
tcp            OBJECT IDENTIFIER ::=      { mib-2 6 }
udp            OBJECT IDENTIFIER ::=      { mib-2 7 }
egp            OBJECT IDENTIFIER ::=      { mib-2 8 }
transmission   OBJECT IDENTIFIER ::=      { mib-2 10 }
snmp           OBJECT IDENTIFIER ::=      { mib-2 11 }
```

A mib-2 é definida como iso.org.dod.internet.mgmt.l ou 1.3.6.1.2.1. A partir daqui, é possível ver que o grupo system é mib-2 1 ou 1.3.6.1.2.1.1, e assim por diante. A Figura 5 mostra a subárvore MIB-II da ramificação mgmt.

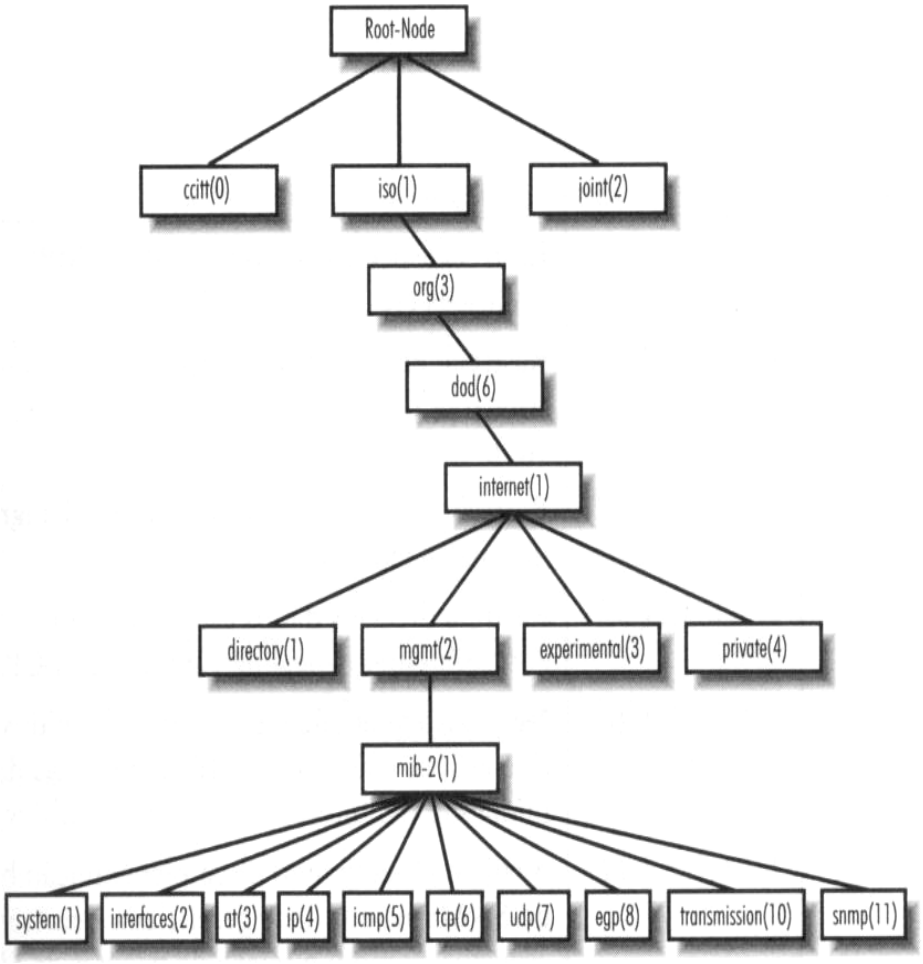

Figura 5 – Subárvore MIB-II

A Tabela 1 descreve resumidamente cada grupo de gerenciamento definido na MIB-II. [Mau01]. Todos os detalhes da MIB-II podem ser obtidos na RFC 1213.

Tabela 1- Descrição dos grupos da MIB-II

Nome da Subárvore	**OID**	**Descrição**
system	*1.3.6.1.2.1.1*	Define uma lista de objetos pertencentes à operação do sistema, como o tempo de funcionamento, contato e nome do sistema.
interfaces	*1.3.6.1.2.1.2*	Rastreia o status de cada interface em uma entidade gerenciada. O grupo *interfaces* monitora as interfaces em funcionamento ou inativas e rastreia aspectos, como octetos enviados e recebidos, erros e eliminações etc.
at	*1.3.6.1.2.1.3*	O grupo *address trans!ation (at) é* fornecido somente para manter a compatibilidade com versões anteriores e, provavelmente, será retirado da MIB-lII.
ip	*1.3.6.1.2.1.4*	Rastreia os diversos aspectos do IP, incluindo o roteamento do IP.
icmp	*1.3.6.1.2.1.5*	Rastreia aspectos como erros do ICMP, exclusões etc.
tcp	*1.3.6.1.2.1.6*	Rastreia, entre outros aspectos, o estado das conexões do TCP (como closed, listen, synSent etc).
udp	*1.3.6.1.2.1.7*	
egp	*1.3.6.1.2.1.8*	Rastreia diversos dados estatísticos sobre o EGP e mantém uma tabela de vizinhos do EGP
transmission	*1.3.6.1.2.1.10*	Não existem atualmente objetos definidos para este grupo, mas outras MIBs específicas de mídia são definidas por meio desta subárvore.
snmp	*1.3.6.1.2.1.11*	Avalie o desempenho da implementação básica do SNMP na entidade gerenciada e rastreia aspectos, como o número de pacotes de SNMP enviados e recebidos.

2.2.9. Operações do SNMP

Foi discutido até agora o modo como o SNMP organiza informações, mas ainda não abordamos como realmente podemos obter informações de gerenciamento.

Protocol Data Unit (PDU) é o formato de mensagem que os gerentes e agentes utilizam para enviar e receber informações. Existe um formato PDU padrão para cada uma das seguintes operações do SNMP: [Mau01]

- *get*
- *get-next*
- *get-bulk* (SNMPv2 e SNMPv3)
- *set*
- *get-response*
- *trap*
- *notification* (SNMPv2 e SNMPv3)
- *inform* (SNMPv2 e SNMPv3)
- *report* (SNMPv2 e SNMPv3)

2.2.9.1. Operação de get

A solicitação de *get* é iniciada pela NMS, que a envia para o agente. O agente recebe a solicitação e a processa para obter o máximo proveito. Alguns dispositivos com carga excessiva, como os roteadores, talvez não consigam responder a solicitação e precisarão excluí-la. Se o agente conseguir obter as informações solicitadas retornará um *get-response* para a NMS, onde é processado. Esse processo é ilustrado na Figura 6.

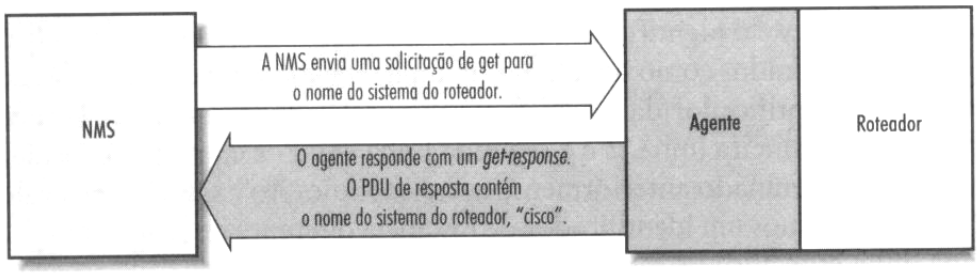

Figura 6 – Seqüência de solicitação de get

Como o agente sabe o que a NMS estava procurando? Um dos itens na solicitação do get é uma *vinculação de variáveis,* ou varbind, que é uma lista de objetos da MIB que permite que o receptor de uma solicitação veja o que o emissor deseja saber. As vinculações de variáveis podem ser consideradas como os pares *OID=valor* que facilitam para o emissor (a NMS, nesse caso) selecionar as informações necessárias quando o receptor preencher a solicitação e retornar a resposta.

O comando *get* serve para recuperar um único objeto da MIB de cada vez. Entretanto, tentar gerenciar dessa maneira pode ser uma perda de tempo. Nesse momento, o comando *get-next* entra em cena e permite recuperar mais de um objeto de um dispositivo em um intervalo de tempo.

2.2.9.2. Operação de get-next

A operação de *get-next* permite emitir uma seqüência de comandos para recuperar um grupo de valores de uma MIB. Em outras palavras, para cada objeto MIB a ser recuperado, são gerados separadamente uma solicitação de *get-*

next e um *get-response.*

O comando *get-next* atravessa uma sub-árvore em ordem lexicográfica. Como uma OID é uma seqüência de inteiros, é fácil para um agente iniciar na raiz da árvore de objetos da respectiva SMI e descer até encontrar a OID que está procurando. Quando a NMS receber uma resposta do agente ao comando *get-next* recém emitido, ela enviará outro comando *get-next* e continuará repetindo esse comando até que o agente retorne um erro, significando que o final da MIB foi alcançado e não há mais objetos a serem obtidos.

2.2.9.3. Operação de get-bulk

O SNMPv2 define a operação *de get-bulk,* que permite que um aplicativo de gerenciamento recupere uma grande seção de uma tabela, de uma só vez. A operação do *get* padrão pode tentar recuperar mais de um objeto MIB de uma vez, mas os tamanhos de mensagens são limitados pelos recursos do agente. Se o agente não puder retornar todas as respostas solicitadas, retornará uma mensagem de erro sem dados. A operação de *get-bulk,* por outro lado, instrui o agente a retornar o máximo da resposta possível. Isso significa que são possíveis respostas incompletas. Ao emitir um comando *get-bulk*, é necessário definir dois campos: "nonrepeaters" e "max-repetitions". O campo "nonrepeaters" informa ao comando *get-bulk* que é possível recuperar os primeiros N objetos com uma simples operação de *get-next*. O campo "max-repetitions" instrui o comando *get-bulk* a tentar até M operações *get-next* para recuperar os demais objetos. A Figura 7 mostra a seqüência do comando *get-bulk.*

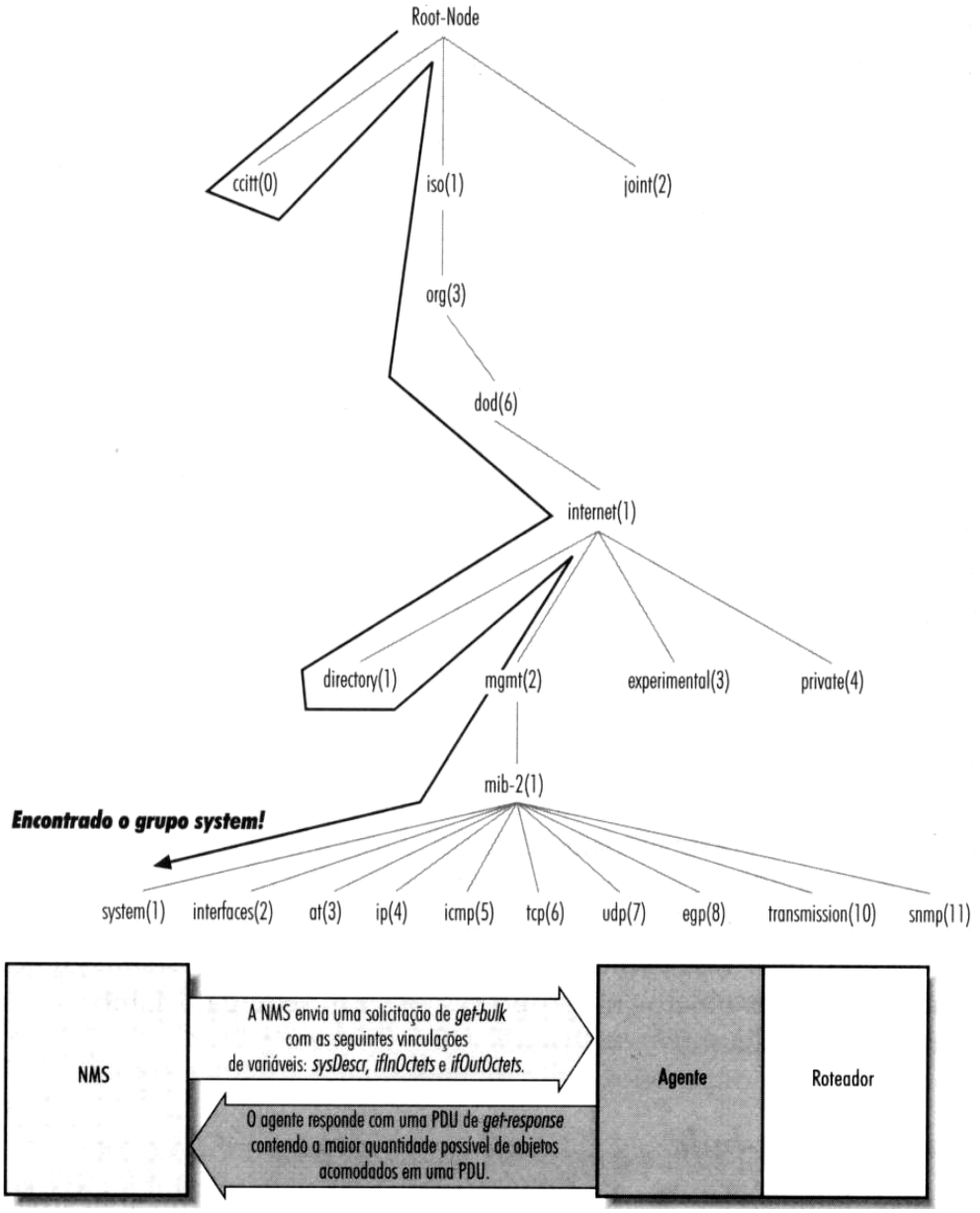

Figura 7 – Seqüência de solicitações de *get-bulk*

Na figura 7, estamos solicitando vinculações: sysDescr, ifInOctets e ifOutOctets. O número total de vinculações de variáveis solicitadas é determinado pela fórmula N + (M * R), onde N é o número de "nonrepeaters" (isto é, objetos escalares na solicitação – nesse caso 1, porque sysDescr é o único objeto escalar), M é o máximo de repetições (nesse caso, definimos aleatoriamente o numero 3) e R é o número de objetos não-escalares na solicitação (nesse caso 2, porque *iflnOctets e ifOutOctets são não-escalares)*. Usando os números deste exemplo temos 1 + (3 * 2) 7, que é o numero total de vinculações de variáveis que pode ser retornado por essa solicitação de *get-bulk.*

2.2.9.4. Operação de set

O comando *set* é utilizado para modificar o valor de um objeto gerenciado ou para criar uma nova linha em uma tabela. Os objetos definidos MIB como *read-write* ou *write-only* podem ser alterados ou criados com esse comando. Uma NMS pode definir mais de um objeto de cada vez.

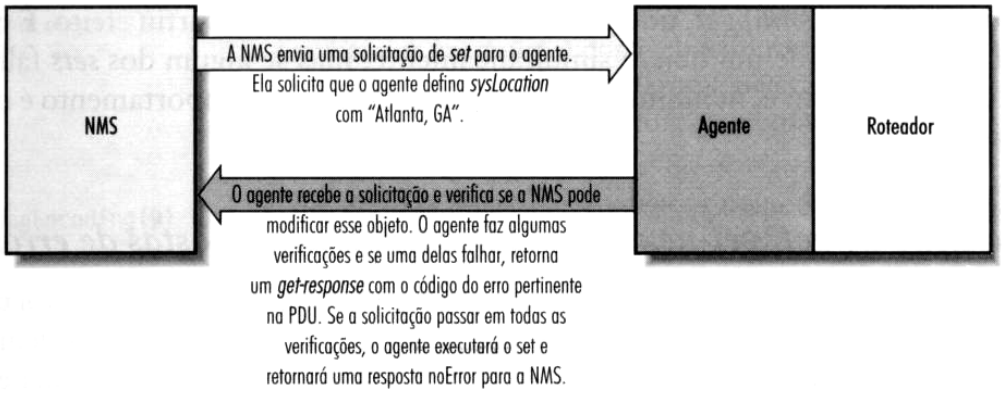

Figura 8 – Seqüência de solicitação de set

A Figura 8 mostra a seqüência de solicitações de set; esse comando é semelhante aos outros que vimos mas, na realidade, o comando está modificando algo na confirmação do dispositivo, em vez de apenas recuperar uma resposta a consulta.

2.2.9.5. Traps do SNMP

Uma *trap* é um meio de um agente informar à NMS que aconteceu algo errado. A Figura 9 mostra a seqüência da geração da *trap*.

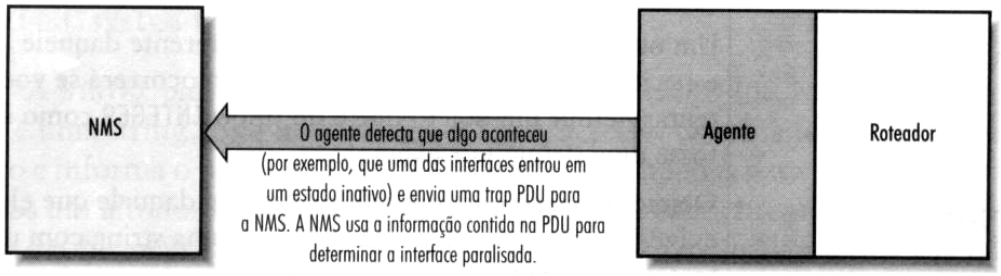

Figura 9 – Geração de trap

A *trap* se origina no agente e é enviada para o destino configurado no próprio agente. Geralmente, o destino da *trap* é o endereço IP da NMS. Nenhuma confirmação é enviada da NMS para o agente, de modo que o agente não sabe se a *trap* alcançou a NMS. Como o SNMP usa o UDP e as *traps* são elaboradas para informar os problemas ocorridos na rede, as *traps* estão propensas principalmente à perda e a não alcançar seus destinos.

Entretanto, o fato de que as *traps* podem desaparecer não as torna menos úteis; em um ambiente bem planejado, elas fazem parte do gerenciamento da rede. E mais recomendável que seu equipamento tente informar que algo está

errado, mesmo que a mensagem nunca alcance você, do que simplesmente desistir e deixar você adivinhando o que pode ter acontecido. Eis algumas situações que uma *trap* pode relatar:

- Uma interface de rede no dispositivo (onde o agente está em execução) foi paralisada.
- Uma interface de rede no dispositivo (onde o agente está em execução) foi reativada.
- Uma chamada recebida em um *rack* de modem não conseguiu estabelecer uma conexão com um modem.
- A ventoinha em um comutador ou roteador está com defeito.

2.2.9.6. Notificação do SNMP

Em um esforço de padronizar o formato PDU das *traps* do SNMPv1 (as traps do SNMPv1 têm um formato de PDU diferente de *get* e *set*), o SNMPv2 define NOTIFICATION-TYPE. O formato de PDU para o NOTIFICATION-TYPE é idêntico ao do *get* e *set*.

2.2.9.7. Mecanismo *inform* do SNMP

Finalmente o SNMPVv2 fornece um mecanismo *inform*, que permite a comunicação entre gerenciadores. Essa operação pode ser útil quando for necessárias mais de uma NMS na rede. Quando um *inform* é enviado de uma NMS para outra, o receptor envia uma resposta para o emissor confirmando o recebimento do evento. Esse comportamento é semelhante aos das solicitações de *get* e *set*. Observe que é possível utilizar um *inform* do SNMP para enviar *traps* do SNMPv2 para uma NMS. Se você usar um *inform* para esse objetivo, o

agente será notificado quando a NMS receber a *trap*.

2.2.9.8. Operação de *report* do SNMP

A operação de *report* foi definida na versão *draft* do SNMPv2, mas nunca implementada. Agora, faz parte da especificação do SNMPv3 e deve permitir que a comunicação entre os mecanismos do SNMP (principalmente para relatar os problemas relacionados ao processamento de mensagens do SNMP).

2.2.10. SNMPv3

A segurança tem sido o ponto fraco do SNMP desde o início A autenticação nas versões 1 e 2 do SNMP não significa mais do que uma senha (string de comunidade) enviada em texto explícito entre um gerenciador e um agente. Qualquer administrador de rede ou de sistema consciente da questão da segurança sabe as senhas em texto explícito não oferecem qualquer segurança. E comum a interceptação de string de comunidade por qualquer pessoa e, uma vez ocorrida, a senha pode ser utilizada para recuperar informações de dispositivos na rede, modificar as respectivas configurações e até derrubá-los.

O *Simple Network Management Protocol Version 3* (SNMPv3) lida com os problemas de segurança que infestaram o SNMPv1 e SNMPv2. Para todos os objetivos práticos, a segurança é a única questão que o SNMPv3 endereça; não ocorreram outras alterações no protocolo nem existem operações novas; o SNMPv3 tem suporte para todas as operações definidas nas Versões 1 e 2. Há várias convenções de texto novas que são apenas métodos mais precisos de interpretar os tipos de dados definidos em versões anteriores. [Mau01]

2.2.10.1. Modificações no SNMPv3

Embora o SNMPv3 não modifique o protocolo, exceto pela inclusão de segurança codificada, seus desenvolvedores tentaram torná-lo diferente ao introduzir novas convenções de texto, conceitos e terminologia. As alterações efetuadas na terminologia são tão radicais, que é difícil acreditar que os novos termos descrevam fundamentalmente o mesmo software como os antigos termos o faziam. Mas descrevem, sim. Entretanto, a diferença está no modo como se relacionam entre si e no fato de especificarem com muito mais precisão os componentes necessários a uma implementação do SNMP.

A alteração mais importante reside no fato de que a Versão 3 abandona a idéia de gerentes e agentes que a partir desta versão passam a ser denominados entidades do SNMP. Cada entidade consiste em um mecanismo do SNMP e em uma ou mais aplicações do SNMP. Esses novos conceitos são importantes porque definem uma arquitetura, em vez de definir tão somente um conjunto de mensagens; a arquitetura ajuda a isolar os diferentes componentes do sistema do SNMP para permitir uma implementação com segurança.

2.2.10.2. Mecanismo do SNMPv3

O mecanismo é formado por quatro componentes: o *Disparcher* (Escalonador) o *Message Processing Subsystem* (Subsistema de Processamento de Mensagens), o *Security Subsystem* (Subsistema de Segurança) e o *Access Control Subsystem* (Subsistema de Controle de Acesso). A Missão do *Dispatcher* é enviar e receber mensagens. Ele tenta detectar a versão de cada mensagem recebida (por exemplo, v1, v2 ou v3), e, se a versão for aceita, direciona a mensagem para o *Message Processing Sybsystem*. O *Dispatcher*

também envia mensagens do SNMP para outras entidades.

O *Message Processing Subsystem* prepara mensagens para serem enviadas e extrai dados das mensagens recebidas. Um sistema de processamento de mensagens pode conter diversos módulos de processamento de mensagens. Por exemplo, um subsistema pode conter módulos para processar solicitações do SNMPv1, SNMPv2 e SNMPv3, assim como um módulo para outros modelos de processamento que inicia serão definidos.

A *Security Subsystem* oferece recursos de autenticação e serviços de privacidade. A autenticação usa *strings* de comunidade (SNMP Versões 1 e 2) ou autenticação baseada em usuário do SNMPv3. A autenticação baseada em usuário utiliza os algoritmos MD5 ou SHA para autenticar usuários sem enviar uma senha explicitamente. Os serviços de privacidade usam o algoritmo DES para codificar e decodificar mensagens do SNMP. Atualmente o DES é o único algoritmo utilizado embora exista a possibilidade de incluir outros mais adiante.

O *Access Control Subsystem* responde pelo controle de acesso aos objetos da MIB. É possível controlar os objetos que o usuário pode acessar e as operações que executará nesses objetos. Por exemplo, convém limitar o acesso de leitura-gravação de um usuário a algumas partes da árvore *mib-2,* e permitir o acesso somente leitura à árvore inteira.

2.2.10.3. Aplicações do SNMPv3

A Versão 3 divide em algumas aplicações a maior parte daquilo que supomos como SNMP:

Command generator (Gerador de comandos)

Gera solicitações de *get, get-next, get-bulk* e *set* e processa as respostas. Essa aplicação é implementada por uma *Network Management Station* (NMS), que pode emitir consultar e solicitações de *set* para entidades em roteadores, comutadores, *hosts* do Unix, etc.

Command responder (Replicador de comandos)

Responde às solicitações dos comandos *get*, *get-next*, *get-bulk* e set. Essa aplicação é implementada por uma entidade em roteador da Cisco ou em um *host* do Unix. (Para as Versões 1 e 2, o replicador de comandos é implementado pelo agente do SNMP)

Notification originator (Gerador de notificações)

Gera *traps* e notificações do SNMP. Essa aplicação é implementada por uma entidade em um roteador ou um *host* do Unix. (Nas Versões 1 e 2, o gerador de notificações integra um agente do SNMP. Também existem utilitários gratuitos para gerar *traps*.)

Notification receiver (Receptor de notificações)

Recebe *traps* e mensagens informativas. Essa aplicação é implementada por uma NMS.

Proxy forwarder (Direciotiador proxy)

Facilita a transmissão de mensagens entre entidades.

A RFC 2571 permite a definição de aplicações adicionais no decorrer do tempo. Essa possibilidade de estender a estrutura do SNMPv3 é uma vantagem significativa em relação às versões anteriores do SNMP.

2.2.10.4. Como é uma entidade?

Até o momento discutimos sobre a entidade SNMPv3 em termos de definições abstratas. A Figura 10 (obtida na RFC 2571) mostra o conjunto de componentes que formam uma entidade. [Mau01]

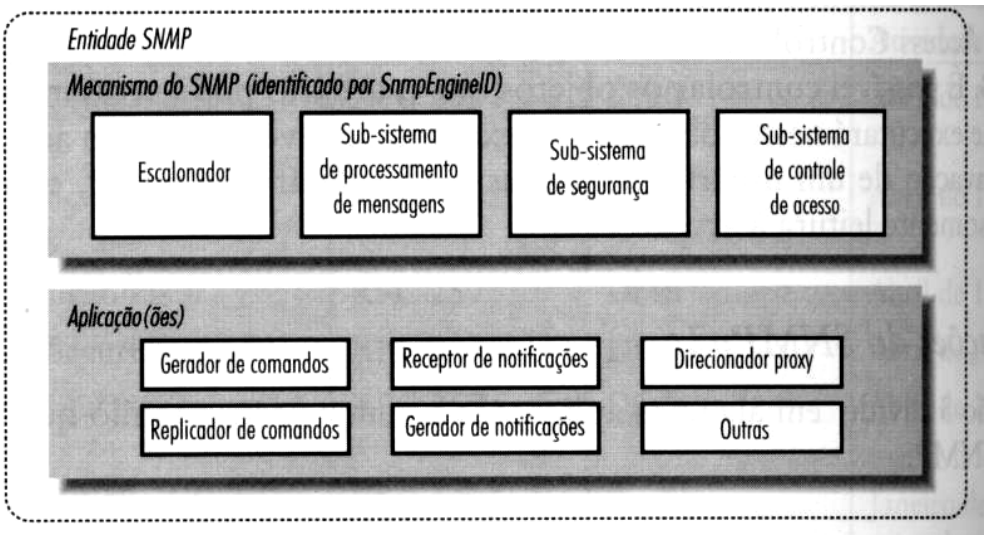

Figura 10 - Entidade SNMPv3

2.3. Wireless

O avanço da computação móvel tem obrigado as empresas a procurarem alternativas aos tradicionais meios para transmissão de dados.

A utilização de rádio para interligar redes locais (LANs) ou regionais (WANs) ganha mais espaço na medida em que a tecnologia avança e oferece mais recursos, e um dos principais resultados tem sido o avanço das soluções *Wireless*.

A tecnologia tem aparecido como a solução ideal para problemas distintos como a troca de informações entre redes baseadas em prédios distantes ou redes instaladas em um mesmo prédio, acaba com a confusão de fios na hora de instalar uma rede de micros, ou ainda, auxiliam na necessidade de manter uma equipe informada sobre mudanças na programação da empresa, conquistando a mobilidade oferecida pelo modem portátil sem fio. A tecnologia de comunicação *Wireless* está facilitando o dia-a-dia dos executivos e a troca de dados na organização, como se isto não bastasse, prometem qualidade, rapidez, segurança e, principalmente, custos reduzidos [Lam98].

Desde a implementação da telefonia celular, o crescimento da comunicação sem fio vem aumentando de forma rápida. Vários sistemas de comunicação sem fio têm surgido, não só de telecomunicações, como o próprio telefone celular, mas também no segmento de redes corporativas para interligar LANs e WANs. A partir da introdução da comunicação *Wireless* no mercado, em meados de 1990, a popularidade desta tecnologia de transmissão de dados sem fio vem crescendo, desde então, dinamicamente.

Hoje, LANs sem fios são usadas em muitos campus universitários para interconectar edifícios distantes, em bancos, hospitais, lojas e etc. Nestes lugares, ela se tornou tão comum quanto aos telefones sem fio.

A comunicação *Wireless* possibilita o próximo passo na evolução dos ambientes computacionais, onde os recursos de informática poderão ser utilizados com mais flexibilidade, pois os usuários não necessitarão mais estar fisicamente conectados (através de um cabo) a uma rede [Sil97]. Sem sombra de dúvida, o *Wireless* tornou-se a saída para os problemas de comunicação ocasionados por cabos e fios.

Freqüentemente as empresas enfrentam dificuldades técnicas ao tentarem instalar suas redes de comunicação de dados via cabo. É o caso, por exemplo, de instalações nas quais há uma rua separando os edifícios ou então de edifícios distantes entre si. Em tais casos a empresa é obrigada a alugar uma linha telefônica do tipo LP ou a realizar um investimento muito grande em fios e cabos. Há ainda o caso daquelas que necessitam de uma velocidade de transferência de dados maior do que a oferecida pelas redes de comunicação convencionais [Sil97].

Wireless computing (computação sem fio), segundo [Kat94], refere-se a sistemas de informática conectados a um ambiente de trabalho via ligações sem fio, utilizando tecnologias como rádiofreqüência (RF), infravermelho, microondas ou laser.

[Sou96] define *Wireless* LAN (LANs sem fio), como redes locais que não utilizam cabos físicos para conexão das suas estações. Operam com placas de rede que possuem adaptador para antena, pela qual recebem os dados da rede

na forma de sinais de rádio. As placas com antenas são colocadas nos micros no lugar das placas de rede comuns.

De uma maneira geral, *Wireless* é o termo aplicado aos dispositivos de informática envolvidos em uma LAN sem fio ou até mesmo uma WAN que contenha elementos que utilizem conexões sem fio.

2.3.1. *Wireless* LANs

Sistemas de computação sem fio representam o próximo passo lógico na evolução dos sistemas de computação e na sua relação com o usuário, de acordo com considerações em [Kat94]. No decorrer dos anos, o usuário tem sido liberado da necessidade de estar presente no mesmo tempo e espaço que os sistemas de computação que utiliza [Sil97].

A comunicação sem fio vem se tornando cada vez mais comum e acessível nos últimos anos. Seguindo essa tendência, as *Wireless* LANs consolidaram-se como uma boa opção de rede local onde haja necessidade de mobilidade dos pontos da rede e/ou existem dificuldades de cabeamento.

Visto como uma parte da progressão da informática os sistemas de computação sem fio representam o próximo passo lógico na separação (libertação) do usuário dos ambientes computacionais. O usuário pode acessar os recursos do sistema (serviços, servidores, impressoras, etc.) a qualquer tempo, bastando estar localizado dentro dos limites de uma infra-estrutura de comunicação sem fio.

Muitos sistemas de comunicação fazem a transmissão dos dados utilizando fios de cobre (como par trançado, cabo coaxial) ou fibra ótica. As *Wireless*, entretanto, transmitem os dados pelo ar, não utilizando qualquer tipo de meio físico, como é o caso da transmissão por raios infravermelhos, lasers, microondas e *spead spectrum*. Cada uma destas técnicas é adequada a certas aplicações, que podem ser empregadas em LANs [Ufr96].

Na realidade, o ar (ou espaço livre) constitui-se de um meio natural para a propagação de sinais eletromagnéticos, podendo talvez, ser considerado o melhor suporte de transmissão, quando se fala em conectividade. Tal afirmação baseia-se no fato de que o ar provê uma interconexão completa, e permite uma grande flexibilidade na localização das estações.

Os sistemas *Wireless* estão sujeitos a interferências, como chuvas, que podem prejudicar ou até mesmo interromper a transmissão. A interferência eletromagnética (EMI) ocorre quando sinais eletromagnéticos indesejados afetam o sinal correto. Meios de transmissão podem suportar e guiar impulsos elétricos e ondas eletromagnéticas, porém nenhum meio consegue controlar todo o espectro magnético e nem eventuais "ruídos" que afetem esse meio. Em distâncias curtas a freqüência 2.4 Ghz utilizada pelo sistema *Wireless* é imune à atenuação por chuva. [Sod01]

As implementações *Wireless* LAN podem complementar o cabeamento tradicional (baseadas em par trançado, cabo coaxial e fibra ótica), usadas em combinação com LANs cabeadas, passam a ser uma solução bastante interessante para as organizações, desta forma os pontos que necessitam de mobilidade são conectados à rede pelo meio "*Wireless*" e as estações fixas são ligadas à rede via cabo.

A rede *Wireless* suporta os principais tipos de protocolos em *frames Ethernet* IEEE-802-3 (TCP/IP/IPX, NetBEUI). As redes sem fio têm o mesmo propósito que uma rede cabeada: dispor informações a todos os dispositivos ligados à rede. Contudo, sem o cabeamento físico para amarrar a localização de um nodo, a rede torna-se muito mais flexível: é fácil mover um nodo sem fio. As redes locais sem fio também são uma boa opção quando a arquitetura de um prédio torna difícil (ou impossível) a passagem de cabos de rede.

As *Wireless* se potencializam com o uso de computadores portáveis. Uma conexão sem fio permite que os computadores portáteis continuem sendo portáteis sem sacrificar as vantagens de estar conectado a uma rede.

2.3.2. O Padrão para *Wireless*

O surgimento da tecnologia de redes locais sem fio – *Wireless* LAN – vem preenchendo uma lacuna no mercado, com possibilidade de instalação de redes em ambientes em que o cabeamento seria impraticável devido a, por exemplo, problemas arquitetônicos/burocráticos; em ambientes em que a troca de localização das estações é constante; ou para permitir uma rápida e limpa montagem/desmontagem de uma rede local em ambientes de demonstração (feiras, visita a clientes, etc.).

No entanto, diversos fabricantes vêm lançando seus produtos, e a necessidade de interoperabilidade entre eles é cada vez maior. Para isto foi criado o padrão IEEE 802.11.

2.4. Perl

Perl (*Practical Extraction and Report Language*) é um ambiente de programação de uso geral, de alto nível e fácil aprendizagem, implementável nas principais plataformas (quase todas as distribuições de UNIX trazem o Perl como padrão do sistema. As versões para Windows e Mac também são perfeitas, aproveitando as características destas plataformas). Possui recursos poderosos para processamento de *strings*, interação com o sistema operacional e com a rede, além de facilidades como gerenciamento automático de memória.

Apesar de se apresentar como uma linguagem de scripts, seus executáveis funcionam compilados em tempo real, sendo rápidos e poderosos. Esta característica facilita muito o trabalho do programador, pois os programas podem ser desenvolvidos, modificados e testados muito facilmente e sem a perda de tempo inerente aos procedimentos de compilação de executáveis. Outra grande vantagem é o Perl possuir, embutido no compilador, um depurador de excelente funcionamento.

Estes motivos tornaram o Perl a linguagem preferida para a programação de aplicações que rodam em servidores da Internet. Os administradores de sistemas UNIX, em especial, precisam dominar o ambiente, já que muitos utilitários do sistema são baseados em Perl ou o usam ostensivamente.

Mas a característica mais importante do Perl é sua condição de "Software Livre". Os códigos, por definição, sendo uma linguagem de scripts, ficam "abertos". Desenvolvido inicialmente para implementar um ambiente de catalogação de erros, os códigos do primeiro Perl foram disponibilizados na Internet por seu criador, Larry Wall. Uma multidão de programadores (os Perl

Porters) passou a incorporar melhorias à linguagem, tornando-a completa. [Fil00]

3. Metodologia

O presente trabalho foi realizado inicialmente em minha residência em Lavras - MG com o estudo do protocolo SNMP, da linguagem Perl e do pacote com suporte para a linguagem Perl no SNMP (*SNMP Support for Perl*) e sua fase final de implantação no Centro de Informática da Universidade Federal de Lavras – CIN-UFLA, em Lavras – MG, onde se encontram os equipamentos e pessoal necessário para o projeto.

A principio foi feita uma grande pesquisa sobre o referencial teórico disponível atualmente para a aquisição e conseqüentemente o estudo do protocolo SNMP.

Após os estudo do protocolo SNMP passamos a analisar alguns pacotes de software disponível para fazermos o gerenciamento da rede utilizando o SNMP. Abaixo os pacotes estudados:

- SNMP Support for Perl
  http://www.switch.ch/misc/leinen/snmp/perl
- Net-SNMP C Library
  http://net-snmp.sourceforge.net
- Net-SNMP Perl Module
  http://www.cpan.org/authors/id/GSM
- SNMP++
  http://rosegarden.external.hp.com/snmp++

Feita a escolha do pacote, passamos ao estudo de sua documentação, tutoriais e exemplos, para uma melhor familiarização com o mesmo.

Cumpridas as etapas acima começamos o estudo dos equipamentos da Rede *Wireless* – UFLA e posteriormente a implantação da ferramenta para gerenciamento via SNMP.

4. Resultados e Discussão

4.1. Arquitetura

A rede *Wireless* – UFLA possui como ponto de acesso 2 (dois) modelos ORiNOCO AP-1000 da marca Lucent, que é uma *bridge* equipada com dois *slots* para cartões PCMCIA (*slot* A e *slot* B). O cartão PCMCIA *Wireless* utilizado nos AP-1000 da rede *Wireless* – UFLA é o modelo Silver da marca Orinoco que possui uma criptografia de 64 bits e se encontra no slot A. A *bridge* possui 11 canais com freqüências que variam de 2,412Ghz a 2,462 Ghz.

Um dos equipamentos se encontra instalado no alto do bairro Lavrinhas – Lavras-MG e este tem a função de fazer a ligação dos clientes *Wireless* da cidade com o servidor que está localizado no CIN-UFLA. A célula aberta pelo AP-1000 é de 60Km de raio.

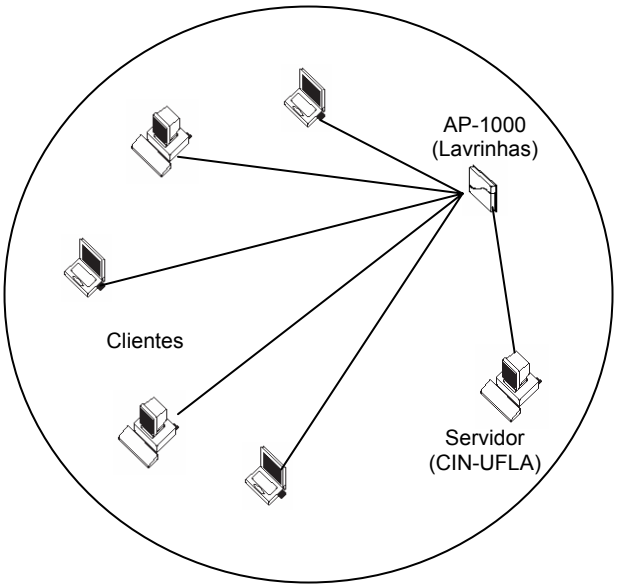

Figura 11: Infra-estrutura da Rede *Wireless* – UFLA

A segunda *bridge* AP-1000 se encontra nas dependências do CIN-UFLA, e é utilizada para fins de estudo e testes. Nossa escolha então foi trabalhar com está AP-1000 que conseqüentemente é o nosso agente.

Nossa NMS é um computador K6-II 550Mhz com 96MB de memória RAM e um disco rígido de 10GB que também está localizado nas dependências do CIN-UFLA. O micro possui *dual-boot*, com os sistemas operacionais Windows 98 e Red Hat Linux 7.3.

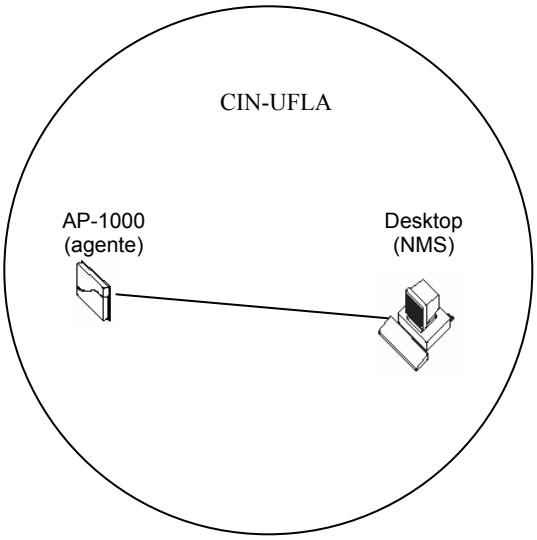

Figura 12: Infra-estrutura da Rede *Wireless* – Projeto

Para que o micro gerente (NMS) seja conectado a rede *Wireless* é necessário um cartão PCMCIA *Wireless* como o da AP-1000, ou seja, modelo Silver da marca Orinoco. Para que o cartão PCMCIA possa ser conectado ao desktop há necessidade de um adaptador para cartão PCMCIA. As figuras 13 e 14 mostram respectivamente o cartão PCMCIA e o adaptador já com um cartão conectado.

Figura 13: Cartão PCMCIA *Wireless*

Figura 14: Adaptador para Cartão PCMCIA *Wireless*

4.2. Implementação

A final seguinte do projeto, iniciou-se com a escolha da linguagem que utilizamos para obtermos as informações da AP-1000 via SNMP. A linguagem escolhida foi a Perl, pela facilidade de implementação de seus scripts. Não houve a necessidade de se instalar o Perl, pois a versão do Red Hat Linux 7.3 já o traz como padrão de instalação. Foi necessário a instalação de uma biblioteca para SNMP da linguagem Perl (SNMP Support for Perl), que é um módulo chamado SNMP_util de autoria de Mike Mitchell. Esse módulo é distribuído com o módulo SNMP Perl de Simon Leinen. O módulo de Mike combinado ao módulo de Simon facilitam muito a programação do SNMP. É possível obter esses módulos a partir de um único arquivo - SNMP_Session-0.94.tar.gz é a versão mais recente do pacote que pode ser obtido através do endereço *http://www.switch.ch/misc/leinen/snmp/perl*. Abaixo se encontram os comandos utilizados para a instalação do pacote.

```
% tar xzf SNMP_Session-0.94.tar.gz
% cd SNMP_Session-0.94
% perl Makefile.PL
% make
% su root
Password:
# make install
# exit
% cd ..
```

Antes da implementação do script foi estudado o arquivo MIB-II, pois é importante sabermos ler e entender seu conteúdo, para a definição dos objetos a serem consultados. A seguir se encontra uma versão reduzida da MIB-II (tudo o que estiver precedido por " – " é um comentário).

```
RFCI1213-MIB DEFINITIONS ::= BEGIN
     IMPORTS
          mgmt, NetworkAddress, IpAddress, Counter,
          Gauge, TimeTicks
               FROM RFC1155-SMI
          OBJECT-TYPE
               FROM RFC 1212;
          mib-2      OBJECT IDENTIFIER ::= {mgmt}

- groups in MIB-II

system               OBJECT IDENTIFIER ::= {mib-2 1}
interfaces           OBJECT IDENTIFIER ::= {mib-2 2}
at                   OBJECT IDENTIFIER ::= {mib-2 3}
ip                   OBJECT IDENTIFIER ::= {mib-2 4}
icmp                 OBJECT IDENTIFIER ::= {mib-2 5}
tcp                  OBJECT IDENTIFIER ::= {mib-2 6}
udp                  OBJECT IDENTIFIER ::= {mib-2 7}
egp                  OBJECT IDENTIFIER ::= {mib-2 8}
transmission         OBJECT IDENTIFIER ::= {mib-2 10}
snmp                 OBJECT IDENTIFIER ::= {mib-2 11}

- Tabela Interfaces

- A tabela Interfaces contém informações sobre as
- interfaces da entidade. Cada interface é considerada
- associada a uma 'subnetwork' (sub-rede) Esse termo
- não deve ser confundido com 'subnet' (sub-rede), que
- se refere a um esquema de particionamento de
- endereços, usado no conjunto de protocolos de
- Internet

ifTable OBJECT-TYPE
     SYNTAX   SEQUENCE OF IfEntry
     ACCESS not-accessible
     STATUS mandatory

          DESCRIPTION
               "Uma   lista   de   entradas   da
               interface.   O  numero  de entradas é
               especificado pelo valor de ifNumber".
     ::= [interfaces 2]
```

```
ifEntry OBJECT-TYPE
      SYNTAX IfEntry
      ACCESS not-accessible
      STATUS mandatory
            DESCRIPTION
                  "Uma entrada de interface contendo
                  objetos existentes na camada   da
                  sub-rede   e    abaixo    de    uma
                  interface específica".
      INDEX [ifIndex]
      ::= [ifTable 1]

      IfEntry ::=
        SEQUENCE {
            IfIndex
                  INTERGER,
            ifDescr
                  DisplayString,
            ifType
                  INTEGER,
            ifMtu
                  INTEGER,
            ifSpeed
                  Gauge,
            ifPhysAddress
                  PhysAddress,
            ifAdminStatus
                  INTEGER,
            ifOperStatus
                  INTEGER,
            ifLastChange
                  TimeTicks,
            ifInOctets
                  Counter,
            ifInUcastPkts
                  Counter,
            ifInNucastPkts
                  Counter,
            ifInDiscards
                  Counter,
            ifInErrors
                  Counter,
            ifInUnknownProtos
```

```
                Counter,
            ifOutOctets
                Counter,
            ifOutUcastPkts
                Counter,
            ifOutNucastPkts
                Counter,
            ifOutDiscards
                Counter,
            ifOutErrons
                Counter,
            ifOutQLen
                Gauge,
            ifSpecifIc
                OBJECT IDENTIFIER
  }

ifIndex OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-only
     STATUS mandatory
     DESCRIPTION
          "Um valor único para cada interface, que
          varia entre 1 e o valor de ifNumber. O
          valor de cada interface deve permanecer
          constante pelo monos a partir de uma
          reinicialização do sistema de gerenciamento
          de  rede da entidade até a reinicialização
          seguinte."

     ::= {ifEntry}

ifDescr OBJECT-TYPE
     SYNTAX  DisplayString (SIZE (O.. .255))
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "Uma string de texto contendo informações
          sobre a interface. Essa string deve incluir
          o nome do fabricante, o nome do produto e a
          versão da interface do hardware."
     :: = {ifEntry 2 }
END
```

A primeira linha desse arquivo define o nome da MIB, nesse caso RFC1213-MIB. (A RFC 1213 é a que define a MIB-II). O formato dessa definição é sempre o mesmo. A seção IMPORTS da MIB é citada ocasionalmente como a seção de ligação, e permite importar tipos de dados e OIDs de outros arquivos de MIB, por meio da cláusula IMPORTS. Essa MIB importa os seguintes itens da RFC1155-SMI.

- `mgmt`
- `NetworkAddress`
- `IpAddress`
- `Counter`
- `Gauge`
- `TimeTicks`

Também é importante o OBJECT-TYPE da RFC 1212, a *Concise MIB Definition*, que define como os arquivos de MIB são escritos. Cada grupo de itens importados por meio da cláusula IMPORTS usa uma cláusula FROM para definir o arquivo de MIB em que os objetos são obtidos

As OIDs utilizadas no restante da MIB vêm depois da seção de ligação. Esse grupo de linhas configura o nível superior da subárvore mib-2, definida como mgmt seguida por .1. O objeto mgmt equivale a OID 1.3.6.1.2. Por conseguinte, a mib-2 é equivalente a 1.3.6.1.2.1. De modo semelhante, o grupo interfaces em mib-2 é definido como {mib-2 2} ou como 1.3.6.1.2.1.2.

Após a definição das OIDs, chegamos às definições dos objetos reais. Toda definição de objeto tem o seguinte formato:

```
<name> OBJECT-TYPE
      SYNTAX <datatype>
      ACCESS  <read-only, read-write, write-only ou
               not-accessible>
      STATUS <mandatory, optional ou obsolete>
      DESCRIPTION
           "Descrição em texto explicando esse objeto
           gerenciado específico.
      :: = {<OID exclusiva que define este objeto>}
```

O primeiro objeto gerenciado no subconjunto da definição da MIB-II é *ifTable,* que representa uma tabela de interfaces de rede em um dispositivo gerenciado (observe que os nomes dos objetos são definidos com maiúsculas/minúsculas, com a primeira letra minúscula). Eis a definição desse objeto com a notação do ASN.1:

```
ifTable OBJECT-TYPE
      SYNTAX  SEQUENCF OF IfEntry
      ACCESS  not-accessible
      STATUS  mandatory
      DESCRIPTION
           "Uma lista de entradas de interface. O
           número de entradas é determinado pelo valor
           de ifNumber."
      :: = {interfaces 2}
```

A SYNTAX de *ifTable* é SEQUENCE OF IfEntry. Isso significa que *ifTable* é uma tabela contendo as colunas definidas em *IfEntry.* O objeto está not-accessíble, o que indica que não é possível consultar em um agente o valor desse objeto. O status é mandatory, indicando que um agente deve implementar este objeto de acordo com a especificação da MIB-II. A DESCRIPTION descreve este objeto com exatidão. A OID única é *1.3.6.1.2.1.2.2* ou *iso.org.dod.internet.mgm.Interfaces.2.*

Examinemos agora a definição de SEQUENCE do arquivo de MIB apresentada anteriormente nesta seção, usada com a SEQUENCE OF tipo na definição de *ifTable:*

```
ifEntry ::=
      SEQUENCE {
            ifIndex
                  INTEGER,
            ifDescr
                  DisplayString,
            ifType
                  INTEGER,
            ifMtu
                  INTEGER,
            .
            .
            .
            ifSpecific
                  OBJECT IDENTIFIER
      }
```

O nome da seqüência *(IfEntry)* é formado por maiúsculas/minúsculas, mas a primeira letra *e* maiúscula, ao contrário da definição de objeto em *ifTable.* É assim que um nome de seqüência é definido. Uma seqüência é apenas uma lista de colunas de objeto e os respectivos tipos de dados da SMI, que define uma tabela de conceitos. Neste caso, esperamos encontrar variáveis definidas por *ifIndex, ifDescr, ifType* etc. Essa tabela pode conter qualquer número de linhas; o agente se encarrega de gerenciar as linhas residentes na tabela. A NMS pode adicionar linhas à tabela.

Uma vez que já existe a *IfEntry* para especificar o que encontraremos em qualquer linha da tabela, podemos reexaminar a definição da própria *IfEntry* (as linhas reais da tabela):

```
ifEntry OBJECT-TYPE
      SYNTAX  IfEntry
      ACCESS  not-accessible
      STATUS  mandatory
      DESCRIPTION
           "Uma entrada de interface contendo objetos
           da camada da sub-rede e abaixo para uma
           interface específica."
      INDEX { ifIndex }
      :: = { ifTable 1 }
```

ifEntry define uma linha específica em *ifTable.* A definição é quase idêntica à de *ifTable,* exceto pelo fato de termos inserido uma nova cláusula, INDEX. O índice é uma chave exclusiva, usada para definir uma linha individual em ifTable. O agente verifica se o índice é único no contexto da tabela. Se um roteador tiver seis interfaces, *ifTable* terá seis linhas. A OID de *ifEntry* é *1.3.6.1.2.1.2.2.1* ou *iso.org.dod.internet.mgmt.interfaces.ifTable .ifEntry.* O índice de ifEntry é ifIndex, definido como:

```
ifIndex OBJECT-TYPE
      SYNTAX  INTEGER
      ACCESS  read-only
      STATUS  mandatory
      DESCRIPTION
           "Um valor único para cada interface, que
           varia entre 1 e o valor de ifNumber. O valor
           de cada interface deve permanecer constante
           pelo menos de uma reinicialização do sistema
           de gerenciamento de rede da entidade até a
           reinicialização seguinte."
:: = { ifEntry 1 }
```

O objeto *IfIndex é* read-only, o que significa que podemos ver seu valor, mas não modificá-lo. O último objeto definido por nossa MIB *é ifDescr,* que *é* uma descrição textual da interface representada por essa linha específica em

if*Table*. O exemplo de nossa MIB termina com a cláusula END, que marca o final da MIB. Nos arquivos da verdadeira MIB-II, cada objeto listado na seqüência de *IfEntry* tem uma definição de objeto exclusiva. Nesta versão da MIB, listamos apenas duas delas, por questão de economia de espaço.

As MIBs utilizadas pelo AP-1000 são: RFC1213.mib, RFC1398.mib e RFC1493.mib.

4.2.1. Scripts

Foram criados dois scripts, um para consulta de objetos (*get*) e outro para modificar o valor de objetos (*set*). Segue os códigos dos dois scripts criados – snmpget.pl e snmpset.pl - e uma análise sobre a função de cada um.

Os scripts foram criados utilizando o editor "joe" e após sua criação devemos modificar suas permissões para podermos executá-los, utilizando os seguintes comandos.

```
% chmod +x snmpget.pl
% chmod +x snmpset.pl
```

4.2.1.1. Script snmpget.pl

Abaixo segue a listagem do script snmpget.pl:

```
#!/usr/bin/perl
#filename: /home/perl_script/smnpget.pl

use BER;
use SNMP_util;
use SNMP_Session;


$MIB1 = ".1.3.6.1.2.1.1.1.0";
$MIB2 = ".1.3.6.1.2.1.1.3.0";
$MIB3 = ".1.3.6.1.2.1.1.4.0";
$MIB4 = ".1.3.6.1.2.1.1.5.0";
$MIB5 = ".1.3.6.1.2.1.1.6.0";

$HOST = "192.168.1.10";

($value1) = &snmpget("public\@$HOST","$MIB1");
($value2) = &snmpget("public\@$HOST","$MIB2");
($value3) = &snmpget("public\@$HOST","$MIB3");
($value4) = &snmpget("public\@$HOST","$MIB4");
($value5) = &snmpget("public\@$HOST","$MIB5");


print "\n======================\n";
print " Host: $HOST\n";
print "======================\n";

if ($value1) { print "Descricao do Sistema: $value1\n"; }
else { warn "Não foi possível realizar a consulta do
objeto: $MIB1\n";}

if ($value2) { print "Tempo em funcionamento: $value2\n"; }
else { warn "Não foi possível realizar a consulta do
objeto: $MIB2\n";}

if ($value3) { print "Contato: $value3\n";}
else { warn "Não foi possível realizar a consulta do
objeto: $MIB3\n";}

if ($value4) { print "Nome: $value4\n";}
else { warn "Não foi possível realizar a consulta do
objeto: $MIB4\n";}

if ($value5) { print "Localizacao: $value5\n\n";}
else { warn "Não foi possível realizar a consulta do
objeto: $MIB5\n\n";}
```

As duas primeiras linhas informam respectivamente onde se encontra o compilador Perl e o script criado. As instruções *use* carregam módulo em Perl contendo funções e definições para trabalhar com o SNMP. Os três módulos utilizados são:

`BER`

Descreve como codificar os dados do gerenciamento em padrões de bits para transmissão. O *Basic Encoding Rules* (BER) é um padrão ISSO.

`SNMP_util`

Define um conjunto de funções que usa o módulo SNMP_Session para torná-lo amistoso ao programador. O próprio SNMP_util usa o padrão BER e o SNMP_Session, mas preferimos fazer uma referencia explícita a esses outros módulos. Há necessidade somente da inclusão do SNMP_util.

`SNMP_Session`

Fornece a linguagem Perl as funcionalidades básicas do SNMP

As linhas contendo as variáveis $MIB especificam os dados que desejamos obter, ou seja, as OIDs que estamos solicitando estão armazenadas em $MIB1, $MIB2, $MIB3, $MIB4, $MIB5.

Usamos a forma numérica desses objetos, mas também poderíamos ter usado a forma em texto da OID que são:

- *.org.dod.internet.mgmt.mib-2.system.sysDescr.0*
- *.org.dod.internet.mgmt.mib-2.system.sysUpTime.0*
- *.org.dod.internet.mgmt.mib-2.system.sysContact.0*

- *.org.dod.internet.mgmt.mib-2.system.sysName.0*
- *.org.dod.internet.mgmt.mib-2.system.sysLocation.0*

A primeira OID nos mostra uma descrição do agente, a segunda o tempo em funcionamento, a terceira o contato caso ocorra algum problema, a quarta o nome do agente e finalmente a quinta que nos mostra a localização do equipamento.

Observe que todas as OID tem um .0 (zero) inseridos no final. No SNMP, os objetos da MIB são definidos pela convenção x.y, onde x é a verdadeira OID do objeto gerenciado e y é o identificador da instância. Para objetos escalares (os objetos não definidos como uma linha em uma tabela) y e sempre 0. No caso de objetos definidos como linhas de tabelas, 1 é a primeira linha, 2 é a segunda e assim por diante.

A variável $HOST armazena o ip do agente a ser consultado. As linhas seguintes consultam o dispositivo. A função snmpget recupera os dados do dispositivo especificados pela variável $HOST. Observe os dois argumentos da função. O primeiro é o dispositivo a ser consultado, precedido pelo nome da comunidade public. O segundo argumento da função snmpget é a OID em que estamos interessados. Observe que as variáveis $value estão entre parênteses. Se omitirmos os parênteses, $value será definido com o número de itens no array retornado por snmpget.

Após a consulta do dispositivo, será impressa uma saída ou uma mensagem de erro caso não seje possível localizar o agente ou a OID.

Para executarmos o script na linha de comando devemos digitar:

```
% ./snmpget.pl
```

A resposta para nossa consulta (*polling*) será:

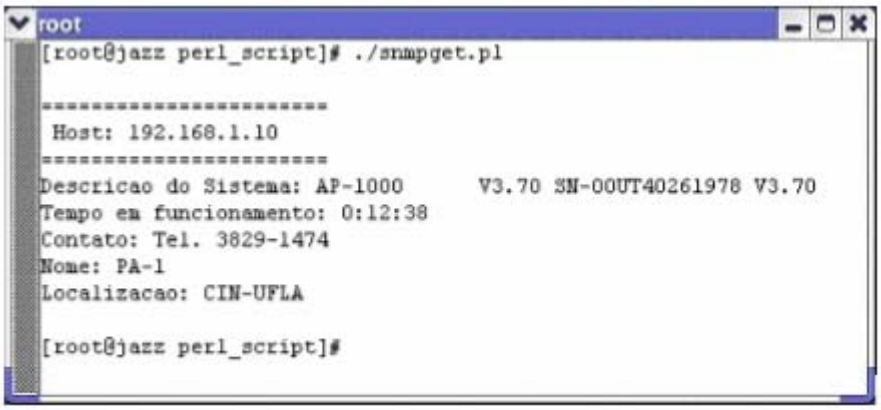

Figura 15: Tela de resposta para a consulta de *get*

4.2.1.2. Script snmpset.pl

Segue a listagem do script snmpset.pl:

```
#!/usr/bin/perl
#filename: /home/perl_script/smnpset.pl

use BER;
use SNMP_util;
use SNMP_Session;

$MIB1 = shift;
$HOST = "192.168.1.10";
$LOC = "@ARGV";

($value) = &snmpset("private\@$HOST","$MIB1", "string",
"$LOC");

if ($value) { print "Resultado: $value\n\n";}
else { warn "Nao foi possivel definir o novo valor para o
objeto: $MIB1\n\n";}
```

O conteúdo desse script e bem parecido com o do snmpget.pl. Analisaremos então somente as partes que diferem do script anterior.

A variável $MIB1 não tem um valor fixo, o valor da OID a ser modificado dever ser passado pela linha de comando. O argumento "shift" tem a função de buscar o valor definido na linha de comando. Uma nova variável ($LOC) irá armazenar a string que será passado também pela linha de comando que será o novo valor da OID. Observe que este script modifica somente as OID de texto. Caso haja necessidade de se modificar outro tipo, os argumentos que podem ser passados são:

`string`
: Representa o tipo string

`int`
: Representa o tipo inteiro de 32 bits

`ipaddr`
: Representa o tipo de endereço IP

`oid`
: Representa o tipo de identificador de objeto (OID)

A linha de comando abaixo modifica a OID sysLocation com o novo valor CIN-UFLA.

```
% ./snmpset.pl .1.3.6.1.2.1.1.6.0 CIN-UFLA
```

4.3. MIB-II

Segue abaixo a listagem do sub-conjunto da MIB-II utilizada nos nossos scritps (Grupo System):

```
-- the System group

-- Implementation of the System group is mandatory for all
-- systems.  If an agent is not configured to have a value
-- for any of these variables, a string of length 0 is
-- returned.

sysDescr OBJECT-TYPE
     SYNTAX  DisplayString (SIZE (0..255))
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "A textual description of the entity. This
          value should include the full name and
          version identification of the system's
          hardware type,  software operating-system,
          and networking software. It is mandatory
          that this only contain printable ASCII
          characters."
     ::= { system 1 }

sysObjectID OBJECT-TYPE
     SYNTAX  OBJECT IDENTIFIER
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "The vendor's authoritative identification
          of the network management subsystem
          contained in the entity.  This value is
          allocated within the SMI enterprises subtree
          (1.3.6.1.4.1) and provides an easy and
          unambiguous means for determining `what kind
          of box' is being managed.  For example, if
          vendor `Flintstones, Inc.' was assigned the
          subtree 1.3.6.1.4.1.4242, it could assign
```

```
          the identifier 1.3.6.1.4.1.4242.1.1 to its
          `Fred Router'."
     ::= { system 2 }

sysUpTime OBJECT-TYPE
     SYNTAX  TimeTicks
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "The time (in hundredths of a second) since
          the network management portion of the system
          was last re-initialized."
     ::= { system 3 }

sysContact OBJECT-TYPE
     SYNTAX  DisplayString (SIZE (0..255))
     ACCESS  read-write
     STATUS  mandatory
     DESCRIPTION
          "The textual identification of the contact
          person for this managed node, together with
          information on how to contact this person."
     ::= { system 4 }

sysName OBJECT-TYPE
     SYNTAX  DisplayString (SIZE (0..255))
     ACCESS  read-write
     STATUS  mandatory
     DESCRIPTION
          "An administratively-assigned name for this
          managed node. By convention, this is the
          node's fully-qualified domain name."
     ::= { system 5 }

sysLocation OBJECT-TYPE
     SYNTAX  DisplayString (SIZE (0..255))
     ACCESS  read-write
     STATUS  mandatory
     DESCRIPTION
          "The physical location of this node
          (e.g.,`telephone closet, 3rd floor')."
     ::= { system 6 }
```

```
sysServices OBJECT-TYPE
    SYNTAX  INTEGER (0..127)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "A value which indicates the set of services
        that this entity primarily offers. The value
        is a sum.  This sum initially takes the
        value zero, Then, for each layer, L, in the
        range 1 through 7, that this node performs
        transactions for, 2 raised to (L - 1) is
        added to the sum.  For example, a node which
        performs primarily routing functions would
        have a value of 4 (2^(3-1)).  In contrast, a
        node which is a host offering application
        services would have a value of 72 (2^(4-1) +
        2^(7-1)).  Note that in the context of the
        Internet suite of protocols, values should
        be calculated accordingly:

     layer  functionality
         1  physical (e.g., repeaters)
         2  datalink/subnetwork (e.g., bridges)
         3  internet (e.g., IP gateways)
         4  end-to-end  (e.g., IP hosts)
         7  applications (e.g., mail relays)

         For systems including OSI protocols, layers
         5 and 6 may also be counted."
    ::= { system 7 }
```

5. Conclusão e Trabalhos Futuros

Ao final do projeto podemos constatar a sua viabilidade, já que sua proposta foi totalmente cumprida, a implementação de uma ferramenta no sistema operacional Linux para o gerenciamento SNMP da Rede *Wireless* UFLA. Entretanto, observamos que a gestão prática do SNMP não está sendo amplamente utilizada pelos profissionais da área de informática, atribuímos essa deficiência a alguns fatores como a dificuldade de implementação, uma vez que ainda não existem ferramentas amplamente difundidas para este fim, além de haver pouca divulgação de estudos práticos (tutoriais, manuais, livros, estudos de caso, etc) sobre as formas de implementação e operacionalização do gerenciamento de rede via SNMP

Como conseqüência, o tempo gasto com a pesquisa do material e o estudo com o protocolo SNMP foi grande. Só foi possível completá-lo após a aquisição do livro SNMP Essencial - Editora Campus, 2001. Em contrapartida após sabermos com detalhes o protocolo SNMP a implementação da ferramenta utilizando a biblioteca SNMP_util para a linguagem Perl foi bem simples e o tempo gasto bem menor se compararmos com o tempo de estudo do protocolo.

Apesar deste trabalho cumprir o seu objetivo, há a possibilidade de se explorar ainda mais os recursos fornecidos pelo SNMP, como a consulta de outras OID, a implementação de uma interface, por exemplo, via Web e também a habilitação de *traps* pelos agentes.

6. Referências Bibliográficas

[Bri93] BRISA; **Gerenciamento de Redes**, **Uma Abordagem de Sistemas Abertos**. Editora Makron Books. São Paulo 1993.

[Cis99] CISCO SYSTEMS INC.: **Network Management Basic,** manuscrito, 1999.
[http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/nmbasic].

[Fil00] FILIPO, R. **Programação Básica em PERL,** manuscrito, 2000.
[http://www.cipsga.org.br].

[Her94] HEREGEING, H-G. & ABECK, S; **Integrated Network and System Management**. Addison - Wesley. USA 1994.

[Ibi01] INTERNET BUSINESS NETWORK.: **Estatísticas sobre Wireless,** manuscrito, 2001.
[http://www.ibiznet.com.br/inet/inet20010810130707.asp]

[Kat94] KATZ, R. H. **Adaptation and Mobility in Wireless Information Systems.** IEEE Personal Communications Magazine, 1994, volume 1, número 1.

[Kor94] KORZENIOWSKI, P. **Monitorando sua Rede**. BYTE Brasil, São Paulo, v.3, n. 12, p. 100-103, dez./1994.

[Lam98] LAMANNA, N. **Revista RNT Rede Nacional de Telecomunicações**. SP: Telepress Editora. N. 221, a. 19, 1998.

[Mau01] MAURO, D. R.; SCHMIDT, K. J. **SNMP Essencial**, Campus, 2001.

[Mei98] MEIRELLES, L. F. T. **Padronização para Redes de Computadores**. 1998.

[Oda94] ODA, C. S; Artigo: **Gerenciamento de Redes de Computadores – Noções Básicas**, 1994.

[Sil97] SILVA, G. A.; PEGORARO, M. W. ***Wireless* Computing: Alternativas para a CTMR**, 1997.

[Sod01] SODISA.NET. **Wireless Internet,** manuscrito**,** 2001
[http://www.sodisa.net/s_duvidas.html]

[Sou96] SOUSA, L. B**. Redes – Transmissão de Dados, Voz e Imagem**.
Editora Érica. São Paulo. 1996.

[Tan97] TANEMBAUM, A. S; **Redes de Computadores**, 3 ed., Campus, 1997.

[Ufr96] UFRGS – **Grupo de Redes. Transmissão de Dados Sem Fio**, manuscrito, 1996
[http://www.penta.ufrgs.be/redes.94-2/lisianeh/*Wireless*.html].

[Vie97] VIEIRA, A. T. **Implantação de Agente SNMP para Medição do Tráfego de Redes**.1997.