DANIELA VANASSI DE OLIVEIRA

MOBILIDADE EM GERÊNCIA DE REDES SNMP

Florianópolis, Outubro de 2000

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

DANIELA VANASSI DE OLIVEIRA

MOBILIDADE EM GERÊNCIA DE REDES SNMP

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

Prof. Dr. João Bosco Mangueira Sobral

Florianópolis, Outubro de 2000

MOBILIDADE EM GERÊNCIA DE REDES SNMP

Daniela Vanassi de Oliveira

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

João Bosco Mangueira Sobral, Dr.
(Orientador)

Fernando A. O. Gauthier, Dr.
(Coordenador CPGCC)

Banca Examinadora

Aloysio de Castro Pinto Pedrozo, Dr. Ing.
COPPE/UFRJ

Ricardo Felipe Custódio, Dr. UFSC

Bernardo Gonçalves Riso, Dr. UFSC

Fernando Augusto da Silva Cruz, Mr. UFSC

"A melhor forma de prever o futuro é criá-lo"

Peter Ducker

AGRADECIMENTOS

Agradeço a Deus por mais esta oportunidade de vida, de trabalho, de descobertas.

A meus pais por toda paciência, carinho, confiança e dedicação. Obrigada por mais essa prova de amor. A meus familiares por sempre acreditarem em mim.

Aos professores do Departamento de Informática da UFSC por terem compartilhado de forma intensa e contagiante todo seu conhecimento. Em especial a meu orientador e amigo João Bosco Mangueira Sobral, por todos os ensinamentos e pela dedicação. Obrigada por tudo.

A colega Kathia Juca por toda sua disponibilidade.

Além destes, este trabalho só foi possível com a colaboração de vários agentes externos. Aos colegas de curso, amigos conquistados, pessoas que sempre irei levar em minha vida.

As Bands de Caxias, por todo companheirismo.

E finalmente ao Clube da Luluzinha, pelos tantos momentos que compartilhamos, vocês foram fundamentais para eu chegar aqui.

A todos que contribuíram direta ou indiretamente para minha formação e para a realização deste trabalho, meus sinceros agradecimentos!

SUMÁRIO

LISTA DE FIGURAS

ABREVIATURAS E SIGLAS

ATM	:	*Assynchronus Transfer Mode.*
ATP	:	*Agent Transfer Protocol*
CCITT	:	*ConsultativeCommittee for International Telegraph and Telephone*
CMIP	:	*Common Management Information Protocol*
CORBA	:	*Common Object Request Broker Adapter*
FTP	:	*File Transfer Protocol.*
HTTP	:	*HyperText Transfer Protocol.*
ICMP	:	*Internet Control Message Protocol.*
IEEE	:	*Institute of Eletrical and Eletronics Engineers.*
IETF	:	*Internet Engineering Task Force.*
IP	:	*Internet Protocol.*
IPv4	:	Protocolo da Internet versão 4.
JVM	:	*Java Virtual Machine.*
LAN	:	*Local Area Network*
MASIF	:	*Mobile Agent System Interoperability Facility.*
MIB	:	*Management Information Base.*
MIT	:	*Massachusetts Institute of Technology.*
NS	:	*Network Simulator*
OMG	:	*Object Management Group*
OSI	:	*Open Systems Interconection*
PDU	:	*Protocol Data Unit*
RFC	:	*Request For Comments*
RMON	:	*Remote Network Monitoring Management Information Base.*
RNP	:	Rede Nacional de Pesquisa.
SNMP	:	*Simple Network Management Protocol.*
TCP	:	*Transmission Control Protocol.*
TCP/IP	:	Conjunto de protocolos da arquitetura de rede da Internet.
Telnet	:	Serviço de acesso remoto.

UDP	:	*User Datagram Protocol.*
UFSC	:	Universidade Federal de Santa Catarina.
URL	:	*Uniform Resource Locator.*
WWW	:	*World Wide Web.*

RESUMO

Com a explosão da Internet, ocorreu a expansão de produtos, serviços, número de usuários, profissionais e fabricantes envolvidos. As redes cresceram e muitas organizações passaram a depender da funcionalidade integral da rede.

Este trabalho aborda uma experiência relacionada ao uso de agentes móveis em gerência de redes SNMP utilizando a MIB RMON (*Remote Network Monitoring Management Information Base*), [STA99]. O contexto do trabalho envolve as áreas de gerência de redes e computação distribuída considerada aqui através do paradigma de agentes móveis.

A área de gerenciamento de redes passou então por processos evolutivos juntamente com o SNMP (*Simple Network Management Protocol*), o protocolo mais utilizado em gerenciamento de redes TCP/IP. Foram criadas MIBs (*Management Information Base*) adicionais que acompanham tecnologias emergentes como ATM-MIB, RMON, RMON II e outras.

A área de computação distribuída tem buscado a otimização do desempenho de aplicações em rede. Uma das novas tecnologias que vem sendo inserida neste contexto é a de agentes móveis, que são pequenos programas com habilidades de comunicação, inteligência e mobilidade.

Como os recursos de rede estão geograficamente distribuídos, as informações para o gerenciamento também encontram-se distantes. Assim, pode-se integrar as funcionalidade da mobilidade dos agentes móveis na gerência com o protocolo SNMP.

Este trabalho demonstra a viabilidade de integrar essas tecnologias utilizando recursos como a ferramenta *Aglets* da IBM para agentes móveis e a API *AdventNetSNMPv3* para gerência SNMP . Será ilustrada uma experiência prática, desenvolvida para o gerenciamento de um *Switch*.

Como resultado desta experiência, foi concebido um modelo de gerência móvel extensível para uma variedade de MIBs e variáveis monitoradas.

ABSTRACT

With the explosion of the Internet, happened the expansion of products, services, number of users, profissionals and manufacturer involved. The networks grow and many organizations began to depend of the integral functioning of the network.

This work approach a experience relationated with the use of mobile agents in SNMP management network using the MIB RMON (*Remote Network Monitoring Management Information Base*), [STA99]. The context of the work envolved the management network area and distribuited computation, here considerated through mobile agents paradigm.

Then the network management area evolved together with the SNMP (*Simple Network Management Protocol*), the most utilized protocol in TCP/IP network management. Were create aditionals MIBs (*Management Information Base*) which accompany emerges tecnologies as ATM-MIB, RMON, RMON II and others.

The distribuited computation area has tried the otimization of the performance of aplications in network. One of the most recent tecnologies inserted int this context is mobile agents. These are small programs with ability of communication, intelligence and mobility.

How the netwok resources are geographicaly distribuited, the information to management are far too. Thus, is possible to integrate the mobility function of the mobile agents in management with the SNMP protocol.

This work demonstrate the viability to integrate this tecnologies using resources as IBM *Aglets* tool to mobile agents and the *AdventNetSNMPv3* API to management SNMP. Will be ilustre a practice experience, to management of a Switch.

As with results of this experience was implemented a mobile management model extensible for a variety of MIBs and monitoring variables.

1. INTRODUÇÃO

Este trabalho é implementado na plataforma de desenvolvimento e execução de agentes móveis *Aglets* da IBM[AGL00]. O objetivo na utilização de uma plataforma, é concentrar todas as funcionalidades dos agentes em uma única ferramenta, flexível para implementação das funções de gerência determinadas para o agente. Instituições de padronização e pesquisa como DARPA e OMG estão empenhadas em criar um padrão para o desenvolvimento de agentes. A utilização de plataformas como *Aglets*, determina funções para os agentes, quem são inerentes às funções disponíveis nas plataformas.

O objetivo deste trabalho é apresentar uma experiência prática de utilização da MIB RMON em gerência de redes com mobilidade. Será apresentado o desenvolvimento de um agente com funções de gerência atuando sobre a MIB RMON.

Outro aspecto a ser abordado é o conceito de gerência por delegação no modelo de agentes implementado. O agente móvel irá efetuar funções de gerência, coleta e avaliação dos dados sob demanda e diretamente no recurso gerenciado, o que atribui a ele a função de gerente remoto.

1.1. Justificativa da Escolha do Tema

A escolha do tema gerência de redes SNMP com a MIB RMON utilizando mobilidade, surgiu da necessidade imposta por um dos maiores problemas em gerência de redes: o tráfego intenso que as duas formas de comunicação entre as entidades (*polling* e registro de eventos)[STA99] geram sobre a rede. Em gerência, a prioridade no tráfego sempre é dos pacotes de informações de controle (gerência) sobre os pacotes de dados do usuário, uma vez que as informações de controle podem ser responsáveis pelo funcionamento da rede.

Vários estudos têm sido realizados, com o intuito de solucionar ou amenizar este problema, como redes ativas [TEN96], gerência por delegação[SIL00] e outros. Além de apresentar uma solução para este problema, o trabalho também objetiva demonstrar a viabilidade de se ter gerência de redes através da mobilidade de agentes. Com a imensa quantidade de recursos distribuídos que a Internet oferece, a tendência é que se crie mecanismos que usufruam a tecnologia existente.

1.2. Histórico da Pesquisa

A pesquisa recente para otimizar a gerência de redes, contém temas relacionados na utilização de agentes móveis no gerenciamento de redes.

Em [COS99] é apresentada uma avaliação analítica do uso de agentes móveis em redes. O trabalho aborda uma dedução matemática para analisar o uso de agentes móveis e do protocolo SNMP em redes, avaliando efeitos como a variação da latência e da banda passante, o tamanho inicial do agente móvel e o tamanho de três diferentes tarefas de gerenciamento no tempo de resposta. Este trabalho não apresenta funções de gerência de redes, apenas considera um cenário de gerenciamento, ou seja, a rede que está sendo gerenciada.

O trabalho desenvolvido por [KOC97] apresenta o estudo e desenvolvimento de uma arquitetura de agentes autônomos integrada ao gerenciamento de redes SNMP. O trabalho aborda aspectos de inteligência artificial e utiliza a linguagem KQML para comunicação entre os agentes de gerenciamento definidos.

Outra abordagem é sobre a arquitetura de redes ativas em gerência de redes e no conceito de delegação por autoridade apresentada em [SIL00]. Neste trabalho é implementada uma aplicação de agentes móveis utilizando a plataforma *Concórdia*[CON00]. A abordagem de gerência de redes apresentada é bastante simplificada, tentando descobrir a topologia da rede através da operação de *ping* nas estações.

Em [DUA99] é realizada uma comparação de performance entre agentes móveis e SNMP em simulações de tarefas de gerenciamento utilizando o *Network Simulator* (NS). O trabalho avalia o efeito da latência, do tamanho inicial do agente, da tarefa a ser executada e do retorno à estação de gerenciamento.

1.3. Formulação do Problema

O foco deste trabalho será apresentar o modelo tradicional de gerência, abordando o problema de tráfego intenso gerado pelas informações sobre a rede, que poderá ser otimizado com a utilização de agentes móveis.

O maior problema da aplicação de agentes móveis em gerência de redes, é a inexistência de capacidade de processamento de código em recursos de rede como roteadores, *hubs* e *switches*. Acredita-se que este tem sido o fator que mais limitou o desenvolvimento nesta área, pois impede o recurso de hospedar um agente móvel de gerenciamento.

Outro fator abordado, é o retorno da informação de gerência para a estação central de gerenciamento. O retorno pode ser feito periodicamente, para equilibrar o tamanho do agente móvel. Assim, manter sua principal funcionalidade de fluxo adequado na rede. Como alternativa, pode ser feito apenas no final do itinerário percorrido pelo agente. O problema nesta abordagem é que a informação de gerência continua a percorrer a rede e necessita de um processamento auxiliar na estação de gerenciamento para avaliação dos dados após a coleta.

1.4. Delimitação do Problema

Para ilustrar o modelo de gerência proposto, será efetuada a gerência de tráfego. O cenário escolhido será o de uma rede TCP/IP, para facilitar a prática do sistema

proposto em ambientes comerciais, nos quais são comuns problemas de tráfego em gerência de redes. Dessa forma, o protocolo SNMP será abordado e assim seus recursos como MIBs e as operações de gerência por ele determinadas.

A implementação será realizada no recurso *Switch*, que possui a MIB RMON habilitada, gerenciando a variável *etherHistoryUndersizePkts* Esta limitação é necessária para ilustrar o modelo proposto e para garantir a eficiência do código móvel, que é também determinada pelo seu tamanho em bytes.

Na plataforma de agentes móveis utilizada, serão implementadas somente as funções necessárias para um agente móvel de gerência. Não serão abordados segurança, inteligência e outros fatores adicionais, sendo esta uma proposta para trabalhos futuros.

1.5. Importância do Tema Pesquisando

O tema pesquisado é de importância significativa na comunidade de gerência de redes, por ser um problema comum às instituições que possuem grandes redes, com um número elevado de usuários, recursos e alto fluxo de informações. Na área de computação distribuída, uma nova abordagem de aplicação do modelo de agentes móveis é considerada em contraste ao modelo cliente/servidor.

Além disso, este é um modelo prático e flexível que poderá ser implantado em diversos ambientes com necessidades de gerência.

A importância maior deste trabalho, é apresentar um exemplo prático e viável de gerência de redes distribuída.

1.6. Estrutura do Trabalho

No que segue, este trabalho está estruturado em capítulos, como descritos abaixo:

No segundo capítulo são apresentadas definições e a conceituação básica para compreensão de alguns termos relacionados a gerência de redes e a computação distribuída utilizados neste trabalho.

No terceiro capítulo será abordada a área de gerência de redes, a necessidade de se gerenciar uma rede, o modelo de gerência existente, seus componentes como: gerentes, agentes, base de informação de gerenciamento e protocolos. O modelo de gerência utilizado pelo protocolo SNMP também será detalhado.

O quarto capítulo aborda informações sobre a MIB RMON, características básicas, configurações usuais e os grupos nos quais os objetos da MIB são distribuídos. Também será apresentada a MIB RMON2.

No quinto capítulo é apresentada a definição e caracterização dos agentes, benefícios que a mobilidade adiciona, a forma de utilização de agentes através das plataformas, abordagens sobre as linguagens de desenvolvimento e alguns produtos que têm obtido destaque na área.

O sexto capítulo apresenta a plataforma de criação e execução de agentes móveis *Aglets* e os principais elementos envolvidos para o desenvolvimento de um agente móvel.

No sétimo capítulo é apresentada a implementação prática de uma experiência de gerência com mobilidade. Será apresentada a estrutura e a execução da implementação realizada.

Conclusões, resultados obtidos e perspectivas futuras estão contidas no oitavo capítulo.

2. DEFINIÇÕES E CONCEITUAÇÃO BÁSICA

Neste capítulo serão abordados conceitos utilizados no contexto da pesquisa sobre gerência de redes e computação distribuída.

Conceitos Relacionados ao Sistema de Gerenciamento

Recurso

É uma visão local/restrita de um equipamento de rede.

Gerente e Agente

No modelo de gerenciamento Gerente-Agente, um processo pode assumir qualquer um dos papéis. Os gerentes gerenciam recursos mapeados em objetos tornados visíveis pelos agentes. Estes objetos são objetos gerenciados. Cada recurso gerenciado é representado por pelo menos um objeto. Neste modelo, um gerente envia operações de gerenciamento a um agente e recebe notificações dele. Os agentes podem executar operações de gerência sobre um ou mais objetos gerenciados. Um gerente pode administrar vários agentes e estes podem ser gerenciados por vários gerentes.

Operações de gerenciamento

As operações de gerenciamento são primitivas executadas na fronteira entre um recurso e o objeto gerenciado que o representa [BRI93]. Podem ser operações orientadas a atributos e operações sobre esses objetos gerenciados como um todo.

A ISO especifica as seguintes operações orientadas a atributos: GET ATTRIBUTE VALUE (obtenção do valor do atributo), REPLACE ATTRIBUTE VALUE (substituição do valor do atributo), SET WITH DEFAULT VALUE

(substituição do valor do atributo pelo valor *default*), ADD MEMBER (inclusão de valores) e REMOVE MEMBER (remoção de valores).

As operações sobre objetos gerenciados como um todo são: CREATE, DELETE e ACTION.

Evento

Um evento é uma mudança no estado de um objeto. Constitui um comportamento autônomo de um objeto. O evento pode ser sinalizado pela emissão de uma notificação.

Notificações

Um sinal ou mensagem indicando que ocorreu um evento. Tal mensagem é sempre enviada por um agente para um gerente.

Objetos gerenciados

São os objetos definidos para o desenvolvimento do sistema gerenciado. Um objeto gerenciado é a representação de um recurso real que pode ser manipulado pelas aplicações de gerenciamento. Dessa forma, cada recurso que se deseja manipular e controlar deve ser descrito na forma de um objeto gerenciado[BRI93].

Os objetos gerenciados são definidos em termos de seus atributos ou propriedades, as operações a que podem ser submetidos; as notificações que podem emitir para informar sobre a ocorrência de eventos de gerenciamento e suas relações com outros objetos gerenciados.

Base de Informação de Gerenciamento(MIB)

No modelo de gerenciamento gerente-agente, o conjunto de objetos gerenciados refletindo os recursos a serem gerenciados, formam a Base de Informações de Gerenciamento.

3. GERÊNCIA DE REDES

Neste capítulo é realizado um apanhado geral sobre gerência de redes, a necessidade de se gerenciar uma rede, o modelo de gerência existente, seus componentes como: gerentes, agentes, base de informação de gerenciamento e protocolos. O modelo de gerência utilizado pelo protocolo SNMP também será detalhado.

Outro conteúdo abordado é a gerência por delegação, que objetiva descentralizar o gerente através da delegação das tarefas de gerenciamento para processos localizados no próprio recurso gerenciado ou localizados em outras máquinas que desempenham o papel de gerentes intermediários.

3.1. Necessidades de Gerenciamento

A evolução das redes e a conseqüente disseminação de informações, têm sido viável devido ao crescimento contínuo em número e diversidade dos componentes de rede, aplicações, número de usuários e a variedade de fornecedores envolvidos. O bom desempenho de uma rede depende da interoperabilidade desses aspectos, tanto durante o funcionamento, quanto na ocorrência de situações críticas que coloquem em risco a funcionalidade da rede.

Devido as dificuldades de isolamento e testes dos problemas das redes, diversidade dos níveis do pessoal envolvido (técnicos, administradores, gerentes), diversidade de formas de controle e monitoração, desenvolveu-se a área de gerência de redes.

A gerência de redes propõem soluções práticas, simplificadas e homogêneas de monitoramento e controle de uma rede, já que na grande maioria dos casos, os gerentes adaptam componentes de hardware e/ou software individualmente para gerenciar a rede.

O gerenciamento de uma rede objetiva registrar a ocorrência de eventos, conhecer e alterar a configuração, garantir a segurança, contabilizar a utilização de recursos, estabelecer limites para o disparo de alarmes, detectar, diagnosticar e prevenir a ocorrência de falhas. Neste contexto, a ISO criou uma divisão funcional, adotada pela maioria dos fornecedores de sistemas de gerenciamento de redes para descrever as necessidades de gerenciamento. Foram então definidas cinco áreas funcionais: Falha, Desempenho, Configuração, Contabilização e Segurança, [BRI93].

3.2. Como Gerenciar

A solução para o gerenciamento, consiste em utilizar um computador que interaja com os diversos componentes da rede, extraindo informações necessárias para seu gerenciamento. Uma base de dados, neste computador gerente da rede, contém informações adequadas/necessárias, para apoiar o diagnóstico e a busca de soluções para os problemas da rede.

O modelo clássico de gerência define uma arquitetura com os seguintes componentes: gerente, agente, base de informação de gerenciamento e o protocolo de gerência.

A interação sempre ocorre entre o gerente e o agente, que consulta a base de informação de gerenciamento e se comunica através do protocolo de gerência. O protocolo de gerência é quem permite a comunicação entre o gerente e o agente e quem define as operações executadas por ambos.

Pode-se citar dois protocolos de gerência mais conhecidos: o CMIP (*Common Management Information Protocol*) para gerência de redes OSI e o SNMP (*Simple Network Management Protocol*) para gerência de redes Internet, mais amplamente utilizado atualmente.

A seguir veremos com mais detalhes o protocolo SNMP e utilizaremos seu modelo para apresentar os componentes de gerência de redes Internet.

3.3. O Gerenciamento SNMP (*Simple Network Management Protocol)*

O SNMP é um protocolo da camada de aplicação da arquitetura de rede da Internet, como ilustrado na figura 3.1, utilizado para monitorar e controlar tanto os recursos como os serviços proporcionados por esses recursos em uma rede de computadores. O SNMP é especificado pela RFC 1157, editada pelo órgão normativo da Internet IETF (*Internet Engineering Task Force*).

FIGURA 3.1: CAMADAS DA ARQUITETURA DE REDES INTERNET

O protocolo SNMP visa minimizar o número e a complexidade das funções realizadas no gerenciamento, em comparação ao gerenciamento proposto pela OSI. O gerenciamento definido pelo SNMP é constituído por quatro componentes principais : agentes gerenciáveis, estações de gerenciamento, base de informação de gerenciamento e o protocolo de gerenciamento. Estes componentes são explicados a seguir:

3.3.1. *Agentes Gerenciáveis*

Agentes gerenciáveis podem ser computadores, roteadores, comutadores, impressoras ou qualquer outro dispositivo que suporte uma interface de comunicação SNMP.

Para ser gerenciável, a interface deve ser capaz de executar um processo de gerenciamento SNMP chamado de agente SNMP. O agente SNMP mantém um banco de dados local de variáveis que descreve o seu estado e histórico, além de permitir a manipulação desses dados.

3.3.2. Estações de Gerenciamento

O controle das informações da rede é feita a partir de uma estação de gerenciamento que contém um ou mais processos que se comunicam com os agentes através da rede. Os gerentes enviam comandos e recebem respostas ou, então, recebem notificações assíncronas dos agentes de gerenciamento.

A estação de gerenciamento funciona como interface de comunicação entre o gerente humano e os agentes de gerenciamento.

3.3.3. Base de Informações de Gerenciamento

Para poderem ser gerenciáveis, os agentes devem possuir estrutura de dados onde são armazenadas informações sobre os recursos sendo gerenciados. No SNMP esta de base de informação de gerenciamento é chamada de MIB (*Management Information Base*).

A estação de gerenciamento realiza o monitoramento dos recursos através dos agentes, a partir das informações armazenadas nas MIBs.

A MIB-II especificada na RFC 1213, define a segunda versão da base de informação de gerenciamento padrão, mais amplamente utilizada. A primeira versão,

MIB-I (RFC 1156) é um subconjunto da MIB-II, porém esta possui alguns objetos e grupos adicionais.

A mais importante extensão da MIB-II, é a MIB RMON (*Remote Network Monitoring Management Information Base*) para monitoramento remoto. A MIB RMON, utilizada neste trabalho, é apresentada no próximo capítulo.

Uma característica relevante na organização dos objetos dentro da MIB é a árvore de nomenclatura da MIB. As informações dentro da MIB estão estruturadas na forma de uma árvore de nomenclatura, conforme recomendação da ISO e ITU-T, designa um identificador para qualquer objeto. As árvores estão definidas em grupos e cada grupo possui um identificador representado na árvore de nomenclatura, tendo objeto ligados sob este identificador.

Todos os nodos possuem um rótulo, que consiste de um identificador numérico e um texto descritivo. O identificador é uma seqüência de números inteiros que marcam o caminho a partir da raiz da árvore até o objeto. Este identificador é chamado de Identificador de Objeto (*OBJECT IDENTIFIER*). Por exemplo, o caminho para se chegar a *etherStatsCollisions* (variável da MIB RMON que determina o número total de colisões no segmento *Ethernet*), possui o identificador de objeto com o formato:

1.3.6.1.2.1.16.1.1.1.13

A descrição textual serve para facilitar a memorização humana. Dessa forma a representação para esta mesma variável seria:

iso.org.dod.internet.mgmt.mib-2. rmon. statistics. EtherStatsTable. etherStatsEntry.etherStatsCollisions

uma forma mais amigável, correspondente a:

iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).rmon(16).statistics(1).etherStatsTable(1).etherStatsEntry(1).etherStatsCollisions(13)

3.3.4. Protocolo de Gerenciamento

O protocolo de gerenciamento é responsável pela comunicação entre os gerentes e os agentes envolvidos no processo de gerenciamento. O protocolo da arquitetura de rede Internet que realiza essa tarefa é o SNMP (*Simple Network Management Protocol*) que atua na camada de aplicação.

O SNMPv1 permite que a estação-gerente manipule os agentes gerenciáveis através das seguintes primitivas :

- *get*: indica uma operação de consulta dos valores da MIB;
- *set*: indica uma operação de modificação de valores da MIB;
- *trap*: informação de notificação enviada pelo agente ao gerente.
- *response*: informação com resultado da operação.

A figura 3.2 ilustra o modelo de gerência SNMP. Este modelo apresenta uma estação gerente SNMP que envia comandos para consulta (*Get*) e alteração de valores (*Set*) em dois dispositivos gerenciáveis: uma estação e uma impressora com agentes SNMP instalados. Os agentes SNMP podem enviar dois tipos de mensagens: a de resposta com o valor consultado (*Response*) ou uma mensagem de notificação (*Trap*) para a estação gerente da rede.

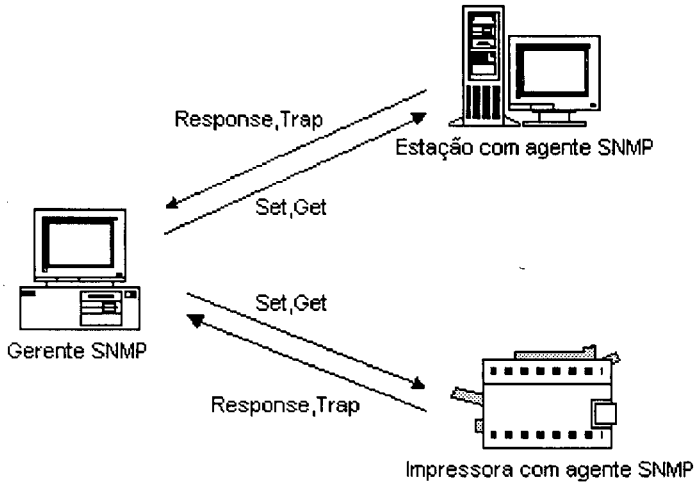

FIGURA 3.2: MODELO DE GERÊNCIA SNMP

O SNMP é projetado para operar sobre o protocolo UDP (*User Datagram Protocol*) da camada de transporte da arquitetura de rede da Internet (figura 3.1), devido ao fato de não poder haver interrupções na comunicação das mensagens do protocolo, dessa forma, sendo não confirmado. Ele utiliza as portas 161 e 162 para comunicação dentro da arquitetura de redes Internet.

Uma questão importante entre o agente e o gerente, no contexto deste trabalho, é a forma de comunicação. O SNMP apresenta duas formas: *polling* e registro de eventos.

O polling é uma interação freqüente entre o gerente e o agente em que o gerente verifica se o agente está ativo e pode, opcionalmente, requisitar informações de sua MIB.

No registro de eventos, a iniciativa da comunicação parte do agente. O agente é configurado anteriormente para enviar informações periódicas sobre o seu estado de funcionamento para o gerente. A informação enviada pode dizer respeito à ocorrência

de algum evento significativo como troca de estado, falhas ou mesmo dados da MIB para atualizar algum documento dinâmico.

O maior problema em gerência de redes é o tráfego intenso que as duas formas de comunicação das informações de gerência (*polling* e registro de eventos) geram sobre a rede. Uma das principais características de gerência é que a prioridade no tráfego sempre seja dos pacotes de informações de controle (gerência) sobre os pacotes de dados do usuário, uma vez que as informações de controle podem ser responsáveis pelo funcionamento da rede. Este trabalho propõem uma solução para este problema.

3.4. Gerenciamento Centralizado *versus* Distribuído

A gerência centralizada tem sérios problemas com o tráfego excessivo gerado entre a estação-gerente e os dispositivos monitorados, podendo gerar congestionamentos nas redes.

Alguns autores, apresentam paradigmas como soluções alternativas para o gerenciamento centralizado: gerenciamento hierárquico fortemente distribuído e fracamente distribuído e o gerenciamento cooperativo.

Como modelos e propostas relacionadas ao gerenciamento distribuído pode-se citar: gerência por delegação[SIL00], agentes móveis[BRE98] e redes ativas [TEN96].

A gerência por delegação (*Management by Delegation*) foi especificada em 1995 e possibilita a construção de sistemas distribuídos em gerência de redes. Ao invés de codificar todo o conhecimento de gerência dentro de uma única entidade como o gerente, neste modelo propõe que certas responsabilidades existentes possam ser delegadas para outros processos distribuídos na rede. Algumas tarefas podem ainda ser controladas pela autoridade central, mas os gerentes locais podem executar várias tarefas de forma autônoma.

Em gerência por delegação o objetivo é a descentralização do gerente através da delegação das tarefas de gerenciamento para processos localizados no próprio recurso gerenciado ou localizados em outras máquinas que desempenham o papel de gerentes intermediários.

A arquitetura de gerenciamento deste modelo inclui um protocolo de gerenciamento, agentes e ambientes de suporte para processos que devam ser executados e que devem estar presentes em cada dispositivo da rede.

Diferentemente do modelo de gerenciamento tradicional, a estação de gerência faz o empacotamento de uma tarefa (código e dados) e a envia, como agentes, para ser executada nos dispositivos da rede delegando para eles a execução das tarefas. As execuções podem ser assíncronas, liberando a estação central para executar outras tarefas, gerando consequentemente um grau de paralelismo na arquitetura de gerenciamento.

O modelo de gerência por delegação em contraste com a gerência centralizada, proporciona as seguintes vantagens:

- automação das funções de gerenciamento através das aplicações delegadas que reduzem a carga da estação central de gerência;
- redução da necessidade de constantes *polling* na rede entre o gerente e os agentes, aumentando o desempenho da rede;

Um problema que pode ocorrer com a existência de vários gerentes delegados é de autoridade, pois cada gerente deve saber o que pode controlar e a quem se reportar de forma que o sistema fique coerente e não haja conflito de instruções entre gerentes diferentes.

3.5. Conclusão

Atualmente quando ocorre a implantação de uma rede em uma organização é fundamental que se introduza uma gerência de redes efetiva. Tanto para o controle e

monitoramento da rede para a manutenção de sua funcionalidade, quanto para projeções futuras, estabelecimentos de métricas e necessidades da rede.

O modelo de gerenciamento proposto pelo SNMP é amplamente utilizado. MIBs proprietárias são desenvolvidas por muitos fabricantes e os agentes instalados nos dispositivos de rede adicionam um custo considerável ao valor final do equipamento. Outro item a ser destacado em gerência, é a interoperabilidade entre os dispositivos gerenciáveis que as ferramentas de gerência procuram abordar.

Avaliando-se o problema do tráfego intenso que as informações de gerência geram sobre a rede, o modelo de gerência por delegação em contraste com a gerência centralizada, proporciona vantagens significativas que solucionam ou amenizam este problema, conforme a estrutura e a forma de implantação na rede.

4. MONITORAMENTO REMOTO

Este capítulo aborda informações sobre a MIB RMON. É apresentada uma visão geral, tendo como objetivo mostrar sua funcionalidade, características básicas, configurações usuais e os grupos nos quais os objetos da MIB são distribuídos.

Também será apresentada a segunda versão da MIB RMON, a RMON2 que constitui uma evolução significativa no tipo de monitoramento que pode ser efetuado nas subredes.

4.1. A MIB RMON

Um dos elementos de maior importância no padrão SNMP é a especificação de monitoramento de rede remoto (RMON - *Remote Network Monitoring*) lançada em 1991. A RMON define uma MIB SNMP descrita na RFC 1757 (anteriormente 1271), complementando a MIB-II.

Considerando uma LAN com ***n*** dispositivos, cada um com um agente SNMP, um gerente SNMP pode obter informações de tráfego de entrada e saída de cada dispositivo individualmente com a MIB-II, mas não pode obter as informações da rede como um todo [STA99].

A RMON define objetos de controle e estatística com informações históricas e atuais. Fornece informações em tempo real da rede como um todo ao invés dos recursos gerenciáveis individualmente.

Um dispositivo de monitoramento de rede remoto, chamado de monitor, agente RMON ou *probe*, é um software que reside dentro de um elemento de rede (roteador, estação de trabalho, *switch, hub*). Neste trabalho utilizaremos o termo agente RMON para referenciarmos este elemento.

4.2. Como a RMON Trabalha

O agente RMON monitora o tráfego de pacotes e analisa seus cabeçalhos, oferecendo com isso informações sobre os links, conexões entre as estações, padrão de tráfego e estado dos nós da rede. Como os pacotes trafegam pela rede, o agente RMON coleta e analisa os dados continuamente em tempo real em um segmento de LAN remoto e armazena os dados localmente de acordo com a especificação da MIB. A RMON é considerada um abordagem orientada ao tráfego, uma vez que as conclusões são obtidas dos pacotes que trafegam na rede e não da inspeção de cada recurso.

Uma rede de grande porte, normalmente apresenta vários agentes RMON ou pelo menos um por segmento, para o gerenciamento da rede. Estes dispositivos podem ser utilizados para acessar um cliente da rede, geograficamente remoto.

Uma configuração usual de RMON, consiste de uma estação de gerenciamento de rede e de um agente RMON. A estação de gerenciamento de rede pode estar sendo executada em plataformas baseadas em Windows ou UNIX. Da estação de gerenciamento pode-se solicitar, através de comandos SNMP, informações ao agente RMON. O agente RMON envia a informação solicitada para a estação de gerenciamento, que processa e disponibiliza para o usuário. O agente RMON ainda pode realizar a compressão semântica de dados antes de enviá-los ao gerente, contribuindo para redução de tráfego na rede. Pode-se ter múltiplos agentes RMON executando em diferentes segmentos de rede, normalmente um por sub-rede.

A figura 4.1 apresenta um cenário de distribuição dos agentes RMON em segmentos de rede juntamente com a estação de gerenciamento [MAG92].

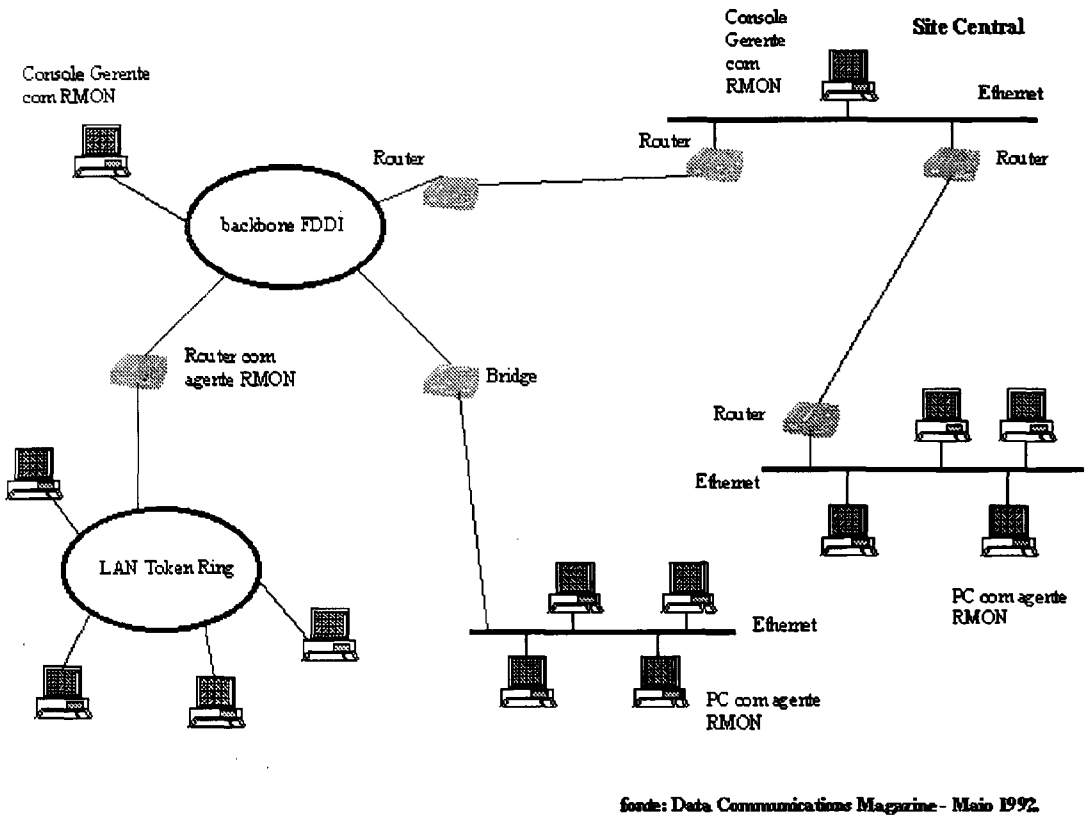

FIGURA 4.1: CENÁRIO EXEMPLO DE DISTRIBUIÇÃO DA MIB RMON

RMON permite monitoramento até a camada MAC. Este tipo de operação permite que se detecte, isole, efetue o diagnóstico e registre problemas da rede antes de ocorrerem situações críticas, ou seja, RMON permite que se faça gerência pró-ativa na rede.

4.3. Os Grupos RMON

Os objetos gerenciados constituem a interface entre um agente RMON e uma aplicação de gerenciamento RMON, não sendo esses objetos manipulados diretamente

pelo gerente humano. Normalmente os usuários não manipulam a complexidade dos objetos durante as operações de gerenciamento. Estas funções devem ser manipuladas pela aplicação de gerenciamento.

A MIB RMON contém objetos gerenciados divididos em dez grupos:

1. *Statistics*: Contém as estatísticas básicas para cada subrede monitorada. Consiste de uma tabela simples, com uma entrada para cada interface monitorada. As estatísticas são na forma de contadores que iniciam em 0 quando uma entrada válida é criada.

2. *History*: Registra amostras de estatísticas da informação disponível no grupo *Statistics*. É usado para definir funções de amostras de uma ou mais interfaces monitoradas. Consiste de duas tabelas: *historyControlTable*, que especifica a interface e os detalhes da função e *etherHistoryTable*, na qual gravamos os dados.

3. *Alarm*: Permite ao operador configurar um intervalo de amostragem e um alarme para qualquer contador registrado pelo agente. Usado para definir um conjunto de entradas para performance de rede. Se um limite é alcançado, um alarme é gerado e enviado para o console central. Por exemplo, um alarme pode ser gerado se houver mais que 500 erros CRC em certo período de tempo de amostragem.

4. *Host*: Contém contadores de vários tipos de tráfego para servidores e de servidores anexados à subrede. Usado para obter estatísticas dos servidores específicos da LAN. O monitor remoto aprende sobre novos *hosts* da LAN, observando a origem e o destino do endereço MAC dos pacotes. Um conjunto de estatísticas é mantido para cada *host* conhecido.

5. *HostTopN*: Contém estatísticas que relatam os servidores em uma lista baseada em alguns parâmetros no grupo *Host*. Usado para manter estatísticas

sobre o conjunto de computadores hospedeiros em uma subrede que encabeça uma lista baseada em algum parâmetro. Por exemplo, uma lista pode ser mantida para os 10 computadores hospedeiros que transmitem a maior parte dos dados durante um certo dia.

6. *Matrix:* Mostra erros e informações de utilização no formato de uma matriz, assim o operador pode reaver informação de qualquer par de endereço de rede. Usado para gravar informação sobre o tráfego entre pares de computadores hospedeiros em uma subrede. Este método é usual para reaver informações específicas, como quais dispositivos fazem mais uso de um servidor.

7. *Filter*: Permite ao monitor observar pacotes. Pode capturar todos os pacotes que passem pelo filtro ou apenas gravar estatísticas baseadas nos pacotes. O grupo *Filter* permite que uma estação-gerente instrua um monitor remoto a observar pacotes específicos em uma interface. A construção básica de blocos definidos neste grupo tem dois tipos de filtros: um de Dados e outro de *Status*.

8. *Package Capture*: Decide como os dados são enviados ao console-gerente. Pode ser usado para estabelecer o buffer para captura de pacote a partir de um canal do grupo *Filter*.

9. *Event*: Um evento é causado por uma condição existente na MIB e faz uma mensagem SNMP (*trap*) ser envianda do agente para o gerente como notificação do evento. Uma tabela de eventos é gerada pelo agente RMON.

10. *TokenRing:* Uma extensão da MIB RMON foi definida na RFC 1513 para gerenciamento de redes *TokenRing*, IEEE 802.5. Este grupo foi incluído pois a maioria dos objetos são para subredes *Ethernet*, mesmo com objetos importantes para todos os tipos de subredes.

Todos os grupos são opcionais, porém há algumas dependências:

- O grupo *Alarm* requer o grupo *Event*;
- O grupo *HostTopN* requer o grupo *Host*;
- O grupo *Package Capture* requer o grupo *Filter*.

4.4. A MIB RMON2

Em meados de 1994, uma extensão da especificação da MIB RMON foi lançada. A RMON2, que inclui monitoramento de tráfego acima do nível MAC. A RMON2 aborda três camadas, permitindo um monitoramento até o nível de aplicação (a primeira versão de RMON permite monitoramento somente até a camada MAC), possibilitando coletar informações, como a banda usada por uma determinada aplicação. Como características significativas da MIB RMON2, pode-se citar:

1. Um agente RMON2 pode monitorar tráfego de endereços e protocolos a nível de rede, incluindo o IP.
2. Monitoramento do tráfego a nível de aplicação, como e-mail, transferência de arquivo e protocolos de WWW.

A RMON pode obter os frames do nível MAC e ler os endereços fontes e destino à nível MAC destas unidades de informação da camada MAC. No entanto, se existir um roteador na LAN, o RMON só pode monitorar o tráfego total dentro e fora do roteador. Não existe uma maneira de determinar a última fonte de tráfego de entrada que chegou no roteador, ou o último destino do tráfego de saída deixado pelo roteador.

O agente RMON2 tem a capacidade de ler o cabeçalho do protocolo a nível de rede, que normalmente é o IP. Isso possibilita ao agente RMON2 analisar o tráfego que circula através do roteador para determinar a última fonte e destino. Com esta capacidade, o gerente da rede pode obter muitas respostas como:

1. Se há uma carga excessiva na LAN devido ao tráfego de entrada no roteador, qual é a rede ou quais computadores hospedeiros geram volume no tráfego de entrada?

2. Se o roteador está sobrecarregado por causa da alta quantidade de tráfego de saída, quais computadores hospedeiros locais geram volume naquele tráfego de saída e para qual destino, redes ou computadores hospedeiros, aquele tráfego está direcionado?

3. Se existe uma carga muito alta de tráfego chegando por um roteador e passando por outro, que redes ou computadores hospedeiros são responsáveis pelo volume deste tráfego?

A RMON2 certamente fornece características importantes para um gerenciamento de mais alto nível nas subredes. Os grupos adicionados são especificados a seguir.

4.5. Os Grupos RMON2

A MIB RMON2 é uma extensão da MIB RMON, sendo adicionados cinco novos grupos:

1. *Protocol Directory:* fornece uma maneira do gerente RMON2 saber quais protocolos um agente RMON2 interpreta. Esta informação é importante quando o gerente e o agente RMON são de diferentes fabricantes.

2. *Protocol Distribution*: este grupo resume como muitos octetos e pacotes são enviados de cada um dos protocolos suportados.

3. *Address Mapping:* combina cada endereço de rede com um endereço a nível MAC e com uma porta específica no dispositivo de rede. Isto é útil em aplicações de descoberta de um nodo e da topologia da rede descobrindo rotas específicas do tráfego da rede.

4. *Network Layer Host*: Habilita usuários a decodificar pacotes baseados em seus endereços a nível de rede. Isso permite ao gerente da rede ver além do roteador os computadores hospedeiros conectados. Esta tabela coleta dados como o grupo *Host* da MIB RMON1, porém o grupo *Host* coleta estatísticas baseadas no endereço MAC e o *Network Layer Host* coleta estatísticas baseado no endereço a nível de rede

5. *Network Layer Matrix:* Este grupo consiste de cinco tabelas: duas tabelas de controle e três tabelas de dados. Essas tabelas são associadas e tratam de coleções de estatísticas a nível de rede de pares de computadores hospedeiros.

Informações mais detalhadas sobre a RMON2 podem ser obtidas em [STA99].

5. AGENTES MÓVEIS

Para apresentarmos o modelo de gerência proposto, é necessário que se conheça as funcionalidades dos agentes e seu modo de operação. Neste capítulo é apresentada a definição e caracterização dos agentes, benefícios que a mobilidade adiciona, a forma de utilização de agentes através das plataformas, abordagens sobre as linguagens de desenvolvimento e alguns produtos que têm obtido destaque na área.

5.1. Definição e Caracterização dos Agentes

Desde os anos 80, o paradigma de agentes, tornou-se crescente na área de tecnologia de software. Tanto organizações de padronização e pesquisa (DARPA, OMG), como companhias comerciais (IBM, *ObjectSpace* entre outros), estão empenhadas em produzir e oferecer um número considerável de soluções práticas e teóricas sobre agentes. A tecnologia de agentes é uma área com uma grande quantidade de trabalhos, produtos, linguagens, empresas, instituições e áreas envolvidas. Contudo possui certas limitações.

Várias definições são encontradas para conceituar um agente. Baseando-se em [FRA 00], pode-se definir que os agentes são entidades de software com um conjunto de operações em nome de um usuário ou de outro programa com algum grau de independência ou autonomia, e assim dispõem de algum conhecimento ou representação das metas de usuários.

Os agentes são entidades inseridas em um ambiente, que procuram conhecer e sobre ele vão atuar, de forma a cumprir seus objetivos. Os agentes se caracterizam por

conceber sistemas sofisticados de forma mais próxima ao pensamento humano devido a melhor modelagem de objetos dinâmicos e autônomos.

A figura 5.1 a seguir, ilustra um exemplo da arquitetura de agentes.

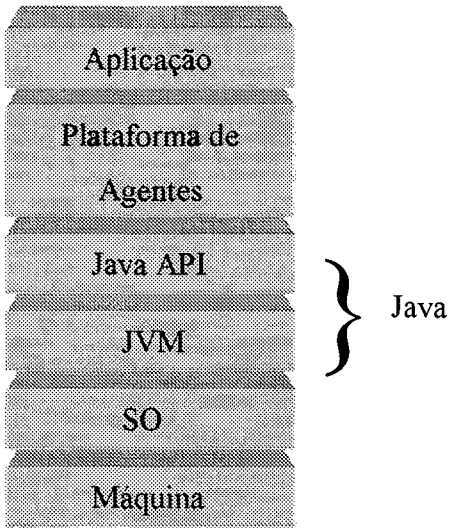

FIGURA 5.1: AMBIENTE DE EXECUÇÃO DE AGENTES

Conforme especificado em [BRE98], um agente possui as seguintes propriedades, que o diferenciem de um programa de software normal:

1) Autonomia

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

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

2) Reatividade

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

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

3) Pró – Atividade

A propriedade de pró-atividade está bastante relacionada a reatividade. Se um agente não só reage as trocas do seu ambiente, mas toma iniciativas baseado em circunstâncias específicas, seu comportamento é denominado pró - ativo.

A capacidade de um agente de tomar iniciativa, requer que ele tenha objetivos bem definidos e um modelo interno de raciocínio.

4) Cooperação e Comunicação

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

Agentes cooperativos, são usados quando um problema excede a capacidade de um agente individual ou agentes existentes, que já possuem uma solução e cujo conhecimento pode ser usado por outros agentes.

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

5) Capacidade de aprendizado

Para ser considerado um agente, ele deve possuir um certo grau de inteligência. Porém, uma ampla variedade de níveis de inteligência são apresentados pelos agentes.

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

6) Mobilidade

Mobilidade é a habilidade que o agente tem de percorrer redes de comunicação eletrônica. Agentes móveis são capazes de mover-se de um computador a outro na rede. Cada computador associado, deve ser capaz de empacotar o agente móvel e enviá-lo para outro computador e também receber, validar e executar agentes.

A autonomia combinada com a mobilidade fornece uma grande vantagem. Se o agente móvel atua autonomamente, o usuário não necessita manter conexão contínua com a rede. Ele envia o agente com a tarefa pela rede e remove sua conexão. Quando o agente tiver obtido os resultados, ele automaticamente retorna e reestabelece a conexão para seu usuário, ou aguarda o usuário conectar-se. Isso reduz o custo de conexão.

Os agentes estacionários são limitados a um computador específico. Embora eles sejam capazes de enviar mensagens usando uma rede existente, ou contactar outros agentes na rede, eles não podem mover-se.

A mobilidade torna aparente várias vantagens significativas do uso de agentes. Os benefícios mais significativos são.

1) Redução da Carga da Rede: Um agente móvel não necessita realizar suas tarefas enviando uma série de mensagens pela rede. Ele pode executar a maioria de suas tarefas localmente, e assim o volume de dados transferidos sobre a rede é reduzido ao mínimo. Além disso, somente a informação necessária será

transportada. O custo de comunicação será reduzido, porque sua medida é baseada no tempo de duração da conexão e no volume de dados transferidos.

2) Ocultam a latência da rede: Sistemas críticos necessitam de respostas em tempo real para mudanças no ambiente. O controle desses sistemas através de uma rede grande ocasiona uma latência inaceitável. Agentes móveis oferecem uma solução, pois podem realizar suas tarefas localmente.

3) Encapsulamento de protocolos: Cada máquina em um sistema distribuído, possui seu próprio código, necessário para implementar a transferência de dados. Novos requisitos de segurança e eficiência demandam mudanças no protocolo que podem ocasionar problemas na manutenção do código existente. Agentes móveis podem mover-se para máquinas remotas a fim de estabelecer canais de comunicação baseados em protocolos proprietários.

4) Executam de forma assíncrona e autônoma: Tarefas podem ser embutidas em agentes móveis que podem ser enviados pela rede. Após serem enviados, os agentes são autônomos e independentes da criação de processos, podendo executar assincronamente. Este recurso é útil porque um dispositivo móvel pode se reconectar na rede para coletar o agente mais tarde.

5) Capacidade de adaptação dinâmica: Agentes móveis possuem a habilidade de perceber mudanças no ambiente de execução e reagir autonomamente. Múltiplos agentes podem interagir entre si e se distribuir pela rede, de modo a manter uma configuração ótima para resolver um problema em particular.

6) Natureza heterogênea: Redes de computadores são geralmente heterogêneas, tanto na perspectiva de hardware como de software. Agentes móveis são independentes da máquina e também da rede, sendo dependentes somente de seu ambiente de execução, facilitando a integração de sistemas.

7) Robustos e tolerantes a falhas: A habilidade dos agentes de reagirem dinamicamente a situações e eventos desfavoráveis, torna fácil a construção de sistemas distribuídos robustos e tolerantes a falhas. Se uma estação pode ser desligada, todos os agentes em execução na máquina podem ser advertidos para que possam continuar suas tarefas em outra estação da rede.

5.2. Linguagens de Agentes

A escolha da linguagem para desenvolvimento de um agente, tem grande efeito na arquitetura do agente que é produzido. Cada linguagem de programação fornece um conjunto de funcionalidades adequadas para implementação de agentes [BRE 98]. As características relevantes a serem observadas na escolha de uma linguagem para agentes, segundo [BRE 98], são:

1) Orientação a objetos: agentes são objetos e a comunicação de agentes ocorre através da invocação de métodos que constituem sua interface pública.
2) Independência de plataforma: agentes são usados em computadores heterogêneos e em diferentes sistemas de agentes distribuídos. A linguagem deve oferecer facilidades que permitam essa interação.
3) Capacidade de comunicação: a linguagem deve possibilitar que se implemente componentes orientados à comunicação, através dos quais, agentes comunicam-se entre si e com recursos internos e externos em um ambiente de rede.
4) Segurança: a linguagem deve fornecer um alto grau de funcionalidade ou através de modelos de segurança específicos da linguagem ou através da integração de modelos externos (protocolos de criptografia, *firewalls*) .
5) Manipulação de código: algumas aplicações requerem que o código do programa de um agente seja manipulado em tempo de execução. Além disso, são necessários mecanismos de identificação do código do agente.

Além dessas características, a linguagem também deve ser reativa, multitarefa e deve permitir armazenamento de dados persistentes.

Dentre as linguagens mais utilizadas está Java, desenvolvida pela *Sun Microsystems* [JAV 00]. Para converter o programa de código fonte em código binário, o compilador Java o traduz para *byte codes* que são interpretados pela JVM (*Java Virtual Machine*) sem modificações nas plataformas que suportarem Java. Dessa forma, Java utiliza o compilador para criar o *byte code,* e um interpretador para executar o programa emitido nesse código.

5.3. Utilizando Agentes Móveis

Para utilizar agentes móveis, um sistema deve incorporar uma plataforma de mobilidade. A plataforma tem que fornecer facilidades que suportem todos os modelos de agentes, incluindo o modelo de navegação. Para o modelo de ciclo de vida, são necessários serviços para criar, destruir, iniciar, suspender e encerrar agentes. O modelo computacional refere-se as capacidades computacionais de um agente, que incluem manipulação de dados e primitivas de controle de *thread.* O modelo de segurança descreve a maneira na qual agentes podem acessar recursos de rede e a maneira de acesso interno dos agentes na rede. O modelo de comunicação define a comunicação entre agentes e entre um agente e outras entidades (isto é, a rede). Todas as questões referentes ao transporte de um agente (com ou sem seu estado) entre duas entidades computacionais residindo em diferentes locais são manipulados pelo modelo de navegação.

A plataforma acrescenta certos custos operacionais, incluindo, aumento de requisitos de memória e execução e atrasos de acesso em cada dispositivo participante do itinerário do agente. A evolução de tecnologias tem sido bastante rápida, a *Java Virtual Machine (JVM)*, que é a base para as plataformas de agentes móveis com desenvolvimento para Java, tem tido evoluções constantes. Acredita-se que o uso de componentes como *chips* Java nos dispositivos, será muito importante no futuro das

redes. Além disso, pacotes promissores como Jini, atenderão muitas das necessidades dos sistemas baseados em agentes.

O tamanho do agente móvel depende do que ele faz, se contém algoritmos complexos ou algumas poucas funções. Os agentes podem estender suas capacidades utilizando código da rede. Os agentes podem carregar somente sua funcionalidade mínima, que pode crescer dependendo do ambiente local e das necessidades. Essa capacidade é facilitada pela mobilidade de código.

Agentes móveis são criados para executar suas tarefas em diferentes computadores da rede. Um agente pode interromper sua execução e migrar de uma máquina para outra, carregando dados sobre seu estado, informações obtidas de execuções de tarefas anteriores. Como o número de nodos visitados é crescente, o tamanho dos agentes móveis também aumenta, tornando a migração difícil. Uma solução possível para este problema seria visitar um número fixo de nodos, retornar ou mandar todos os dados para a origem do agente (reduzindo o tamanho do agente) . O tamanho inicial do agente móvel também afeta a performance. Se um agente tiver um tamanho grande, a migração torna-se mais difícil. O tamanho inicial depende da tarefa a ser realizada e da linguagem usada para implementá-lo.

Algumas limitações podem ser identificadas, como a falta de padronização, a ausência de interoperabilidade entre plataformas e o estágio de desenvolvimento de Inteligência Artificial.

5.4. Produtos na Área

Diversas plataformas são utilizadas para desenvolvimento e execução de agentes móveis. Foi realizada uma abordagem mais específica sobre as que utilizam a linguagem Java como linguagem de desenvolvimento, pelas características adequadas que Java possui de portabilidade, orientação a objetos entre outras. As plataformas mais encontradas e citadas são: *Concordia*, da *Mitsubishi Electric's Horizon Systems Labs* [CON00]; *Voyager* desenvolvido pela *ObjectSpace Inc* [VOY00], combina a plataforma

de agentes móveis autônomos com suporte a invocação de método remoto e CORBA (*Common Object Request Broker Adapter*); *Grasshopper*, desenvolvido pela *GMD FOCUS* e *IKV++GmbH* [GRA00] e finalmente *Aglets*, da *IBM [AGL 00]*. A seguir serão expostas as propriedades da plataforma *Aglets*, utilizada no desenvolvimento deste trabalho.

5.5. Conclusão

Este capítulo apresentou informações relevantes para a compreensão de agentes móveis no modelo de gerência proposto neste trabalho. As informações apresentadas também servem como justificativa das escolhas relacionadas a utilização da plataforma e da linguagem Java neste trabalho.

As descrições referentes as características dos agentes diferenciam um programa de software normal de um agente, abrangendo o limite funcional que um software pode ter em uma rede, já que o agente móvel possui maior flexibilidade e interoperabilidade. Estas descrições, também são importantes para definir os requisitos que devem existir no dispositivo que hospedará um agente móvel.

6. *AGLETS*

Neste capítulo é feita a apresentação do estudo realizado sobre a plataforma de criação e execução de agentes móveis *Aglets*, mostrando suas características e os principais elementos envolvidos para o desenvolvimento de um agente móvel.

Será apresentado o protocolo ATP e a forma de comunicação entre dois *Aglets*, além do comportamento de um *Aglet*: criação, envio para um computador hospedeiro, recebimento e execução. Para tanto, será realizada uma descrição dos principais métodos encontrados na API e no ambiente de desenvolvimento fornecido pela IBM.

6.1. Conceito e Caracterização dos *Aglets*

Um *Aglet* (ou "*agile applet*") é um pequeno programa de aplicação ou *applet* com a capacidade de agente móvel em serviços de uma rede de computadores. São objetos que podem ser movidos de um local para outro. *Aglets* é uma plataforma para o desenvolvimento de agentes móveis implementada em Java, desenvolvida pelo Laboratório de Pesquisa da IBM em Tókio.

As principais características dos *Aglets* são [OSH98]:

1. Capacidade de passagem de objetos: o *Aglet* é um objeto com seus próprios métodos, dados de estado e itinerário de percurso, que pode ser enviado para

outro *Aglet* ou passado por si próprio através da rede como uma entidade autônoma. Um *Aglet* tem a habilidade de decidir por si próprio que ações tomar, onde e quando ir para algum lugar.

2. Interação com outros objetos de programa: ele pode interagir localmente com outros *Aglets* ou objetos estacionários. Quando necessário, ele pode se auto enviar ou enviar outro *Aglet* para locais remotos para interagir com outros objetos.

3. Operação de desconexão: Se o computador corrente está desconectado da rede, o *Aglet* pode decidir por si próprio mover-se quando o computador for reconectado.

4. Execução paralela: múltiplos *Aglets* podem ser enviados para rodar concorrentemente em diferentes computadores.

6.2. A Plataforma

A plataforma para desenvolvimento constitui o grande diferencial entre os agentes móveis, já que é inerente à plataforma todos os recursos que o agente móvel irá possuir para interagir com o ambiente.

Os recursos existentes para trabalhar-se com *Aglets* são: o *Aglets Workbench*, um ambiente de programação visual para criar *Aglets;* o ATP (*Agent Transfer Protocol*), um protocolo para transferência de agentes entre computadores da rede. Ambos estão sendo oferecidos pela OMG (*Object Management Group*), como uma proposta para o MASIF (*Mobile Agent System Interoperability Facility*). A IBM oferece o *Workbench* sem custos para os desenvolvedores.

A API *Java Aglet* (J-AAPI) constitui uma interface para construção de *Aglets*, definindo métodos para criação, manipulação de mensagens, envio, recebimento,

ativação/desativação, clonagem e destruição de um *Aglet*. A J-AAPI necessita do JDK1.1 ou versão superior para ser executada.

A seguir serão expostas características sobre os principais elementos: *Tahiti* e ATP.

6.2.1. O Tahiti

A plataforma *Aglets* possui um servidor, denominado *Tahiti*, o qual utiliza uma interface gráfica para monitorar e controlar a execução de *Aglets* num servidor. Pode-se iniciar novos *Aglets*, saber os *Aglets* que estão sendo executados no sistema em um determinado momento, disponibilizar, receber e encerrar *Aglets* através deste ambiente. O usuário pode ainda solicitar ao servidor para mostrar informações de uso de memória, estado de *threads* e mensagens de log. Para comunicação, o *Tahiti* é iniciado na porta padrão 434, mas pode ser modificado.

O *Tahiti* é um ambiente visual proprietário, o que facilita a manipulação dos agentes, mas dificulta a manipulação da interface gráfica da aplicação desenvolvida. Por isso a forma de manipulação dos métodos dos *Aglets* é efetuada através da API.

6.2.2. O Protocolo ATP

Para comunicação entre os agentes é utilizado o ATP (*Agent Transfer Protocol*), um protocolo independente de plataforma para transferência de agentes entre os computadores da rede. O ATP também permite manipular a mobilidade de agentes de forma genérica e uniforme, apesar da linguagem de programação e da plataforma ser específica.

O ATP é um protocolo a nível da aplicação para transmissão de agentes móveis, modelado no protocolo HTTP. Para habilitar a comunicação remota entre agentes, o

ATP suporta passagem de mensagens. Essa comunicação é derivada do padrão da OMG, MASIF (*Mobile Agent System Interoperability Facility*), o qual permite vários sistemas agentes interoperarem. Esta interface abstrai o nível de comunicação, definindo interfaces e fornecendo uma representação comum em Java que suporta a IDL definida no padrão MASIF.

Enquanto os agentes móveis podem ser escritos em diferentes linguagens e para uma variedade de sistemas agentes, o ATP fornece a oportunidade para manipular agentes de maneira genérica e uniforme. Por exemplo, qualquer máquina hospedeira de agente terá um único nome independente do conjunto de sistemas de agentes suportados. ATP também fornece um mecanismo de transporte uniforme e permite uma facilidade de consulta de agente padrão para ser usada através da rede.

Embora a interface MASIF seja para objetos CORBA, a interface atualmente definida nos *Aglets* não são baseadas em CORBA, elas são definidas como classes de interface normais Java ou classes abstratas, que atuam como uma capa para o protocolo atualmente sendo usado. Diferente de objetos Java normais, os *Aglets* nunca são coletados com o coletor de lixo automaticamente, porque um *Aglet* é ativado e possui sua própria *thread* de controle. O programador necessita explicitamente destruir o *Aglet*.

Quando um *Aglet* for desativado, enviado ou destruído, o objeto *AgletRef* é removido da tabela de referência. Além disso, a referência interna para o *Aglet* e os componentes associados como o *MessageManager* ou propriedades, são setadas para nulo para que o coletor de lixo possa varrer esses objetos, isto significa que se tiver uma referência para esse *Aglet* em outro lugar, ela não será coletada pelo coletor de lixo.

A plataforma *Aglet* não possui um mecanismo de comunicação para transferência de dados serializados do *Aglet* para um destino. Ao invés disso, ele usa a comunicação por API que abstrai a comunicação entre sistemas agentes. Esta API define métodos para criar e transferir agentes, gerenciamento de percurso do agente e do protocolo no sistema, independente do meio.

O ATP define quatro métodos padrão:

1. *Dispatch*: solicita a um sistema agente-destino para reconstruir um agente e iniciar sua execução. Se a requisição for bem sucedida, quem enviou deve terminar o agente e liberar os recursos que ele utilizava.

2. *Retract*: solicita ao sistema agente - destino para enviar um agente específico de volta para quem o enviou. O receptor é responsável por reconstruir e reiniciar o agente. Se o agente for transferido com sucesso, o receptor deve terminar o agente e liberar os recursos que ele utilizava.

3. *Fetch*: solicita ao receptor para recuperar e enviar alguma informação identificada (normalmente arquivos de classes).

4. *Message*: utilizado para enviar uma mensagem para um agente identificado por um *agent-id* (identificador do agente).

A classe abstrata *Aglet* define métodos fundamentais para um agente móvel controlar sua mobilidade e ciclo de vida. Somente classes que herdem da classe *Aglet* podem ser movidas pela rede. Quando um novo *Aglet* for criado alguns atributos com informação sobre o criador e sobre o computador hospedeiro, onde ele é criado são armazenadas em uma variável *AgletInfo*. Normalmente quando se cria uma classe *Aglet*, ele é enviada para outro local. A criação do agente é sempre local. Quando um *Aglet* quer se comunicar com outro *Aglet*, ele tem primeiro que obter um *proxy* para o objeto. O *proxy* é uma interface que atua como manipulador de um *Aglet* e fornece uma maneira padrão de acessar o *Aglet* atrás dele. Métodos públicos de *Aglets* não podem ser acessados diretamente de outro *Aglet* por razões de segurança. Quando o *AgletProxy* é invocado, ele consulta o *SecurityManager* para determinar se o contexto da execução corrente é permitido para executar o método. O contexto de um *Aglet* é um *proxy* para um ambiente de execução que ele ocupa. Esse contexto é usado para obter informações locais como: endereço do computador hospedeiro, o *proxy* para *Aglets* no mesmo contexto e para criar um novo *Aglet* no contexto.

Objetos *Aglets* se comunicam pela troca de objetos da classe *Message*. Um objeto da classe *Message* possui um objeto *String*, para especificar o tipo de mensagem e um número variado de argumentos. Um *Aglet* que quer se comunicar com outro *Aglet*, primeiro tem que criar um objeto *Message* e então enviá-lo para o *Aglet* par. O *Aglet* receptor tem que definir seu método *handleMessage* para manipular mensagens.

Existem tipos diferentes de mensagens:

1. *Now-type*: é uma mensagem síncrona, onde o *Aglet* fica bloqueado até o receptor ter completado a manipulação da mensagem.

2. *Future-type*: é uma mensagem assíncrona, não bloqueia a execução corrente. O método retorna um objeto *FutureReply* que pode ser usado para obter o resultado ou aguardar por ele e obter posteriormente.

3. *Oneway-type*: é uma mensagem assíncrona, não bloqueia a execução corrente. Difere da mensagem *Future-type* na forma como é localizada no final da fila até se é mandada para o próprio *Aglet* e não retorna valor.

6.3. Comportamento de um *Aglet*

O comportamento suportado por um modelo de objeto *Aglet*, constitui de métodos que podem ser implementados através da API internamente ou podem ser manipulados através do *Tahiti*. Os métodos são: *creation*, *cloning*, *dispatching*, *retraction*, *deactivation*, *activation*, *disposal e messaging*. A seguir a explicação de cada método:

1. *creation*: o método *creation* ocorre em um contexto. Ao novo *Aglet* é destinado um identificador inserido no contexto e então inicializado.

2. *cloning*: o procedimento *cloning* de um *Aglet* produz uma cópia idêntica do *Aglet* original no mesmo contexto. A única diferença é o identificador e a execução reiniciada. *Threads* de execução não são clonadas.

3. *dispatching:* o método *dispatching* de um *Aglet* de um contexto para outro irá removê-lo de seu contexto corrente e inseri-lo no contexto de destino onde irá iniciar sua execução.

4. *retraction:* o método *retraction* irá removê-lo do seu contexto corrente e inserí-lo no contexto ao qual a *retraction* for solicitada.

5. *deactivation:* o método *deactivation* é a habilidade para temporariamente removê-lo do contexto corrente e armazená-lo em um armazenamento secundário. A ativação do *Aglet* será restabelecida no contexto.

6. *disposal:* o método *disposal* irá encerrar a execução corrente e irá removê-lo do contexto atual.

7. *Messaging:* o método *messaging* entre *Aglets* envolve envio, recebimento e manipulação de mensagens tanto síncrona quanto assincronamente.

6.4. Conclusão

Neste capítulo foi apresentada a plataforma de desenvolvimento e execução de agentes móveis *Aglets* da IBM. Foram apresentados os elementos relevantes da plataforma: ambiente de execução, protocolo e formas de comunicação. Foi feita uma analise técnica da forma de desenvolvimento de agentes com essa plataforma, através dos principais métodos disponibilizados na API.

As descrições dos métodos servem tanto para utilização através do ambiente visual *Tahiti*, quanto para manipulação através da API. Além dos métodos descritos, ainda existe outros que podem ser manipulados conforme a necessidade da aplicação.

A utilização de uma plataforma como *Aglets*, constitui uma forma bastante prática e direta para criação e manipulação de agentes móveis. Outras plataformas como *Voyager*, *Concordia*, *Grasshopper* foram analisadas e constatou-se que *Aglets* por ser uma plataforma somente de agentes móveis, não para desenvolvimento de aplicações distribuídas como CORBA, apresenta as funções de forma mais concentrada e direta, atendendo todas necessidades para o desenvolvimento deste trabalho.

7. IMPLEMENTAÇÃO

7.1. Considerações sobre a Implementação

A implementação foi realizada utilizando a plataforma de agentes móveis *Aglets* versão 1.0.3 e a API da ferramenta de gerenciamento de redes *AdventNetSNMPv3*[ADV00].

A API do *AdventNetSNMPv3* consiste de um pacote de desenvolvimento Java que permite que se trabalhe diretamente com detalhes do protocolo SNMP, reduzindo o tamanho da aplicação de gerenciamento desenvolvida. É considerada uma API de baixo nível e contém os seguintes módulos funcionais:

- Comunicação: consiste de classes Java usadas para comunicar com entidades SNMP pares. As classes de comunicação fornecem acesso a uma grande quantidade de detalhes nos parâmetros de comunicação utilizados.

- Suporte a Variáveis: fornece suporte a todas as estruturas de dados SNMP como inteiros, identificadores de objetos, contadores. Essa API é utilizada durante o envio e recebimento de requisições.

- Segurança: fornece funções para a implementação de mensagens de segurança, controle de acesso, gerenciamento de informação, configuração remota e administração de entidades específicas para cada aplicação.

No desenvolvimento, foi utilizado o JDK (*Java Development Kit*) versão JDK1.1.8, necessária para suporte à plataforma *Aglets*.

O código de gerência para interação com SNMP foi desenvolvido em um nível próximo a instruções de rede. As classes do *AdventNetNMPv3* explicitam o acesso a elementos internos, como ilustra o trecho de código a seguir:

```
// Abre uma sessão SNMP
SnmpSession session = new SnmpSession(api);
// seta o host remoto
session.setPeername( opt.remArgs[0] );
...
// Constroi Get request PDU
SnmpPDU pdu = new SnmpPDU();
pdu.setCommand( api.GET_REQ_MSG );
```

As instruções acima abrem uma sessão *SNMP*, relacionam ao equipamento remoto que será monitorado, constróem uma estrutura PDU (*Protocol Data Unit*) e determinam o comando *GET_REQ_MSG*. Este comando será efetuado na *MIB* do agente SNMP do recurso.

Outra funcionalidade é o acompanhamento do *Aglet* via *Web*. Cada vez que o *Aglet* for hospedado em um equipamento, é gerado um código *HTML* com suas informações. O envio do *Aglet* ao próximo equipamento a ser monitorado também pode ser feito via *browser*, ou através do ambiente *Tahiti*.

7.2. A Experiência Prática de Gerência com Mobilidade

Para a avaliação da viabilidade da utilização de mobilidade em gerência de redes, a função principal do *Aglet* é a coleta e tratamento das informações do recurso gerenciável.

Para ilustrar esse mecanismo, foi realizada a gerência de desempenho sobre um equipamento *Switch* fornecido pelo NPD (*Núcleo de Processamento de Dados)* da UFSC, com endereço 150.162.252.80.

A variável *etherHistoryUndersizePkts (OID* 1.3.6.1.2.1.16.2.2.1.10), a qual determina o número de pacotes recebidos menores do que 64 octetos de tamanho, foi a variável escolhida para ser monitorada. Essa variável está especificada no grupo *History* da MIB RMON. A cada iteração, o valor que é retornado pelo SNMP, apresenta-se adicionado ao valor anterior. Dessa forma, ao final da coleta, o *Aglet* calcula o número de pacotes recebidos e o tempo de duração da coleta de dados.

As figuras a seguir ilustram o acompanhamento do *Aglet* através do *Tahiti* e via *browser*.

Criando o *Aglet* através do *Tahiti*:

Considere o nome do *Aglet* como *WebAgent.*

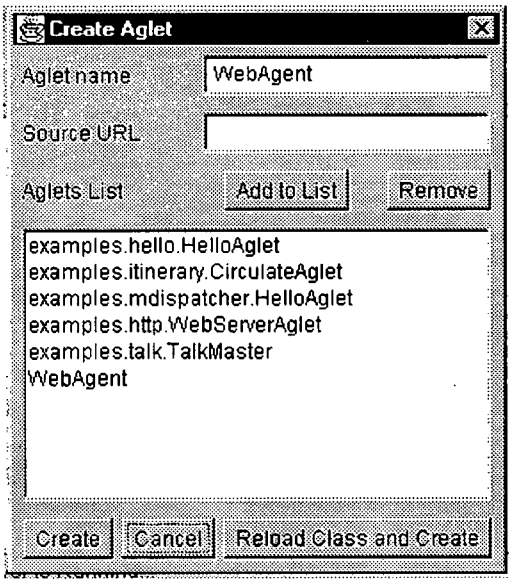

FIGURA 7.1: CRIAÇÃO DO *AGLET* NO AMBIENTE *TAHITI*

Visualização do *Aglet* carregado no *Tahiti.*

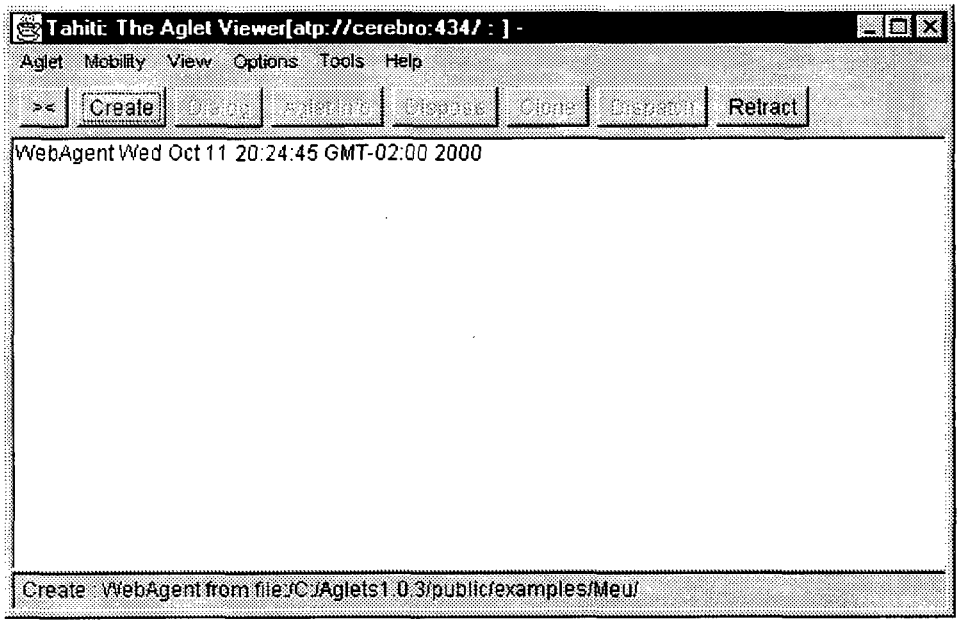

FIGURA 7.2: VISUALIZAÇÃO DO *AGLET* NO AMBIENTE *TAHITI*

A figura 7.3 ilustra o acompanhamento da execução do agente durante a coleta de dados.

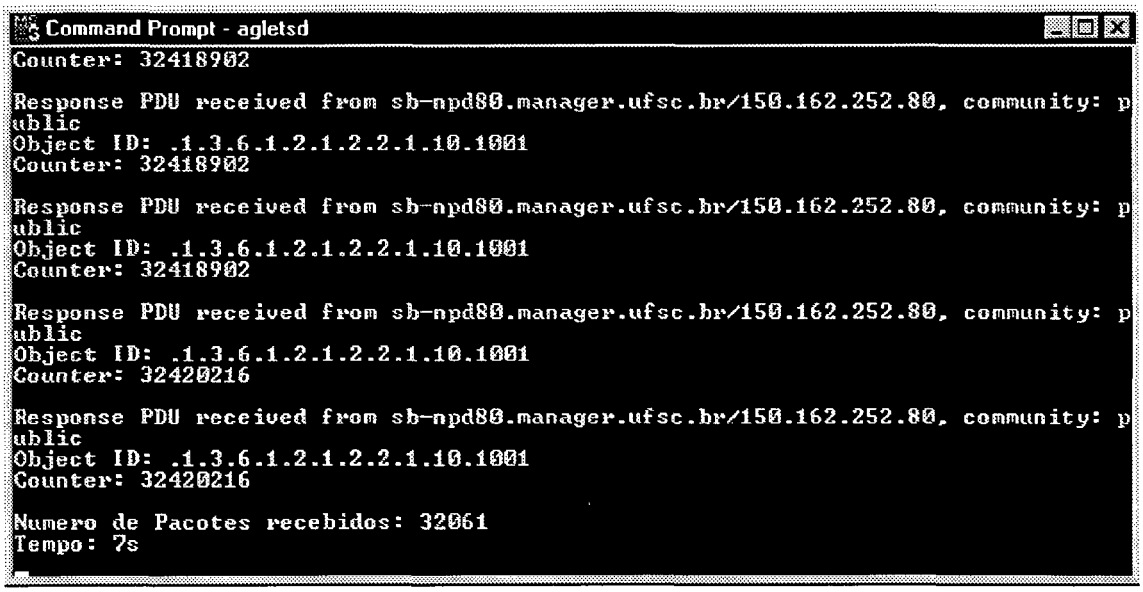

FIGURA 7.3: EXECUÇÃO DO *AGLET*

A figura 7.4 ilustra o acompanhamento do *Aglet* via browser:

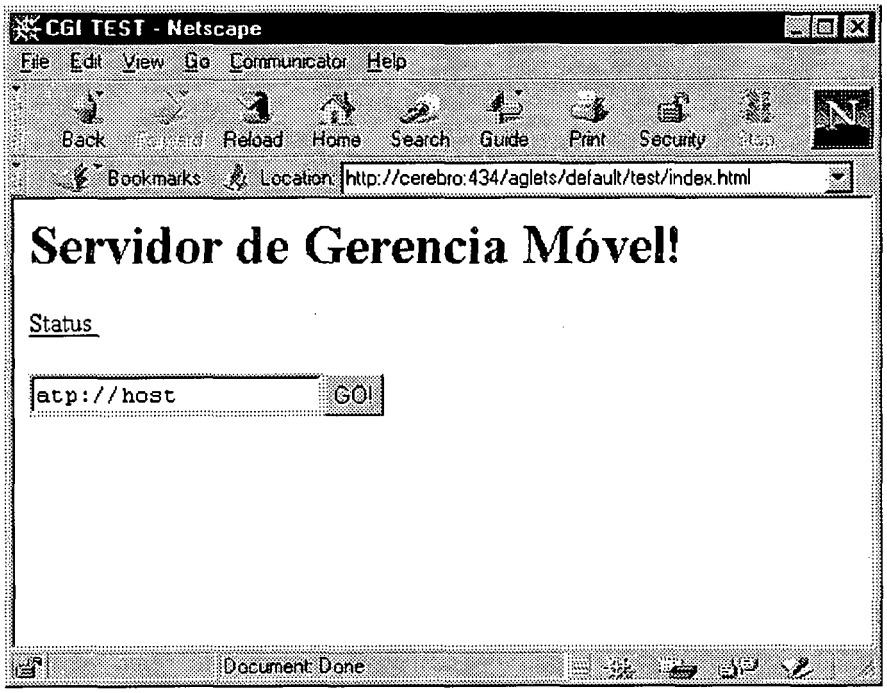

FIGURA 7.4: VISUALIZAÇÃO DO *AGLET* NO BROWSER

A figura 7.5 ilustra informações do *Aglet* exibidas no browser:

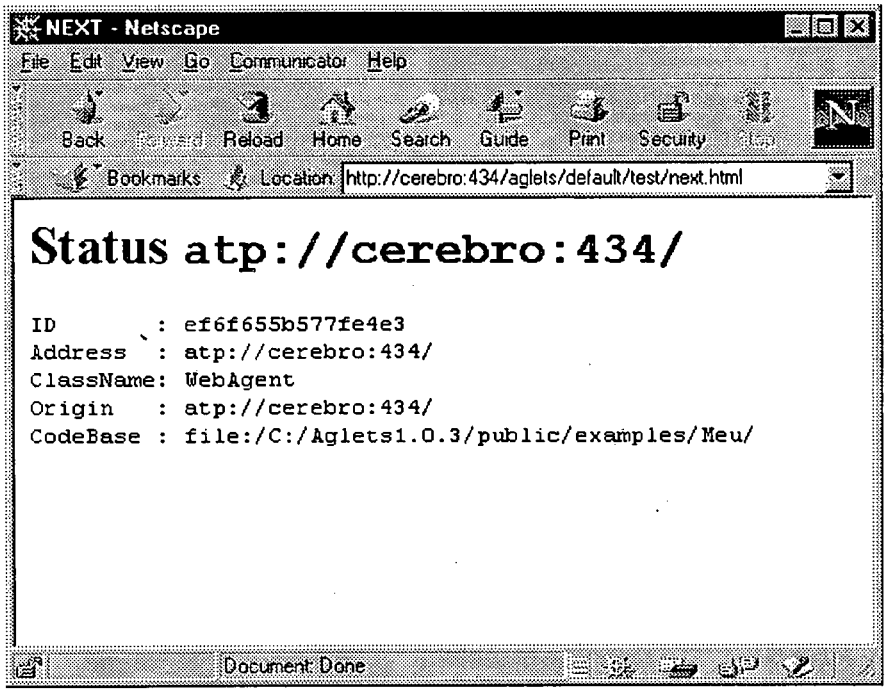

FIGURA 7.5: VISUALIZAÇÃO DO STATUS DO *AGLET* NO BROWSER

Especificando a transição do *Aglet* via browser:

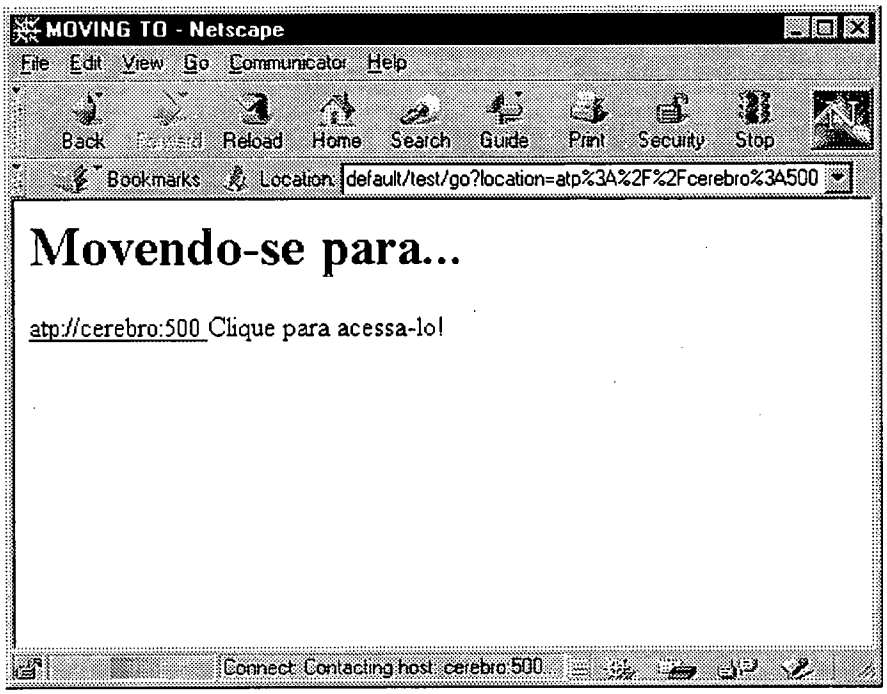

FIGURA 7.6: TRANSIÇÃO DO *AGLET* VIA BROWSER

Especificando o destino na área reservada no browser, neste caso o *Aglet* é movido através da ativação do método *dispatch.* Isso pode ser feito diretamente através da interface no ambiente *Tahiti.*

Visualização do *Aglet* na plataforma *Tahiti* após a transição:

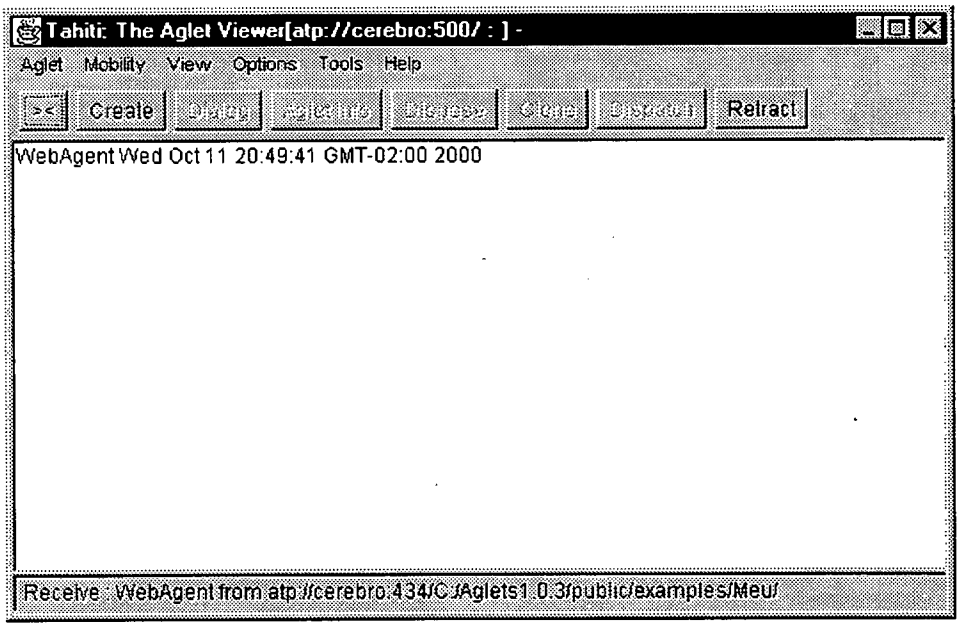

FIGURA 7.7: VISUALIZAÇÃO DO *AGLET* NO AMBIENTE *TAHITI*

Acompanhando a execução:

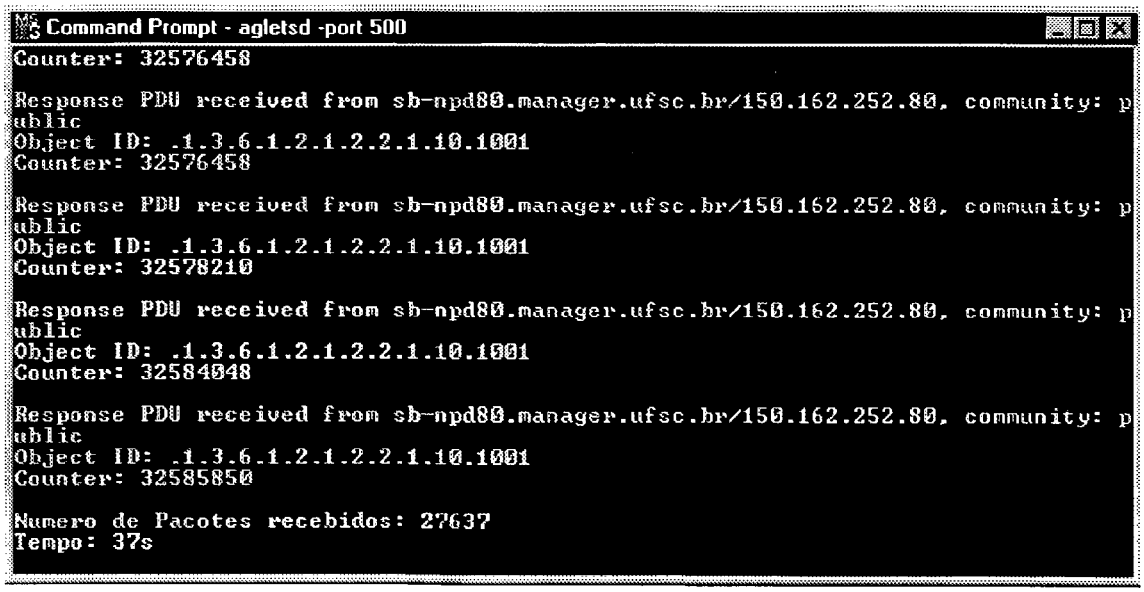

FIGURA 7.8: EXECUÇÃO DO *AGLET*

Por escassez de recursos, na ausência de segmentos de rede reais, foi considerado um mesmo elemento de rede gerenciável representado por um *Switch.* Os segmentos de rede foram representados por cada estação.

Um ambiente ideal para os testes seria o ilustrado na figura 7.9, onde o agente poderia mover-se entre as subredes hospedando-se nos recursos que possuem a MIB RMON habilitada.

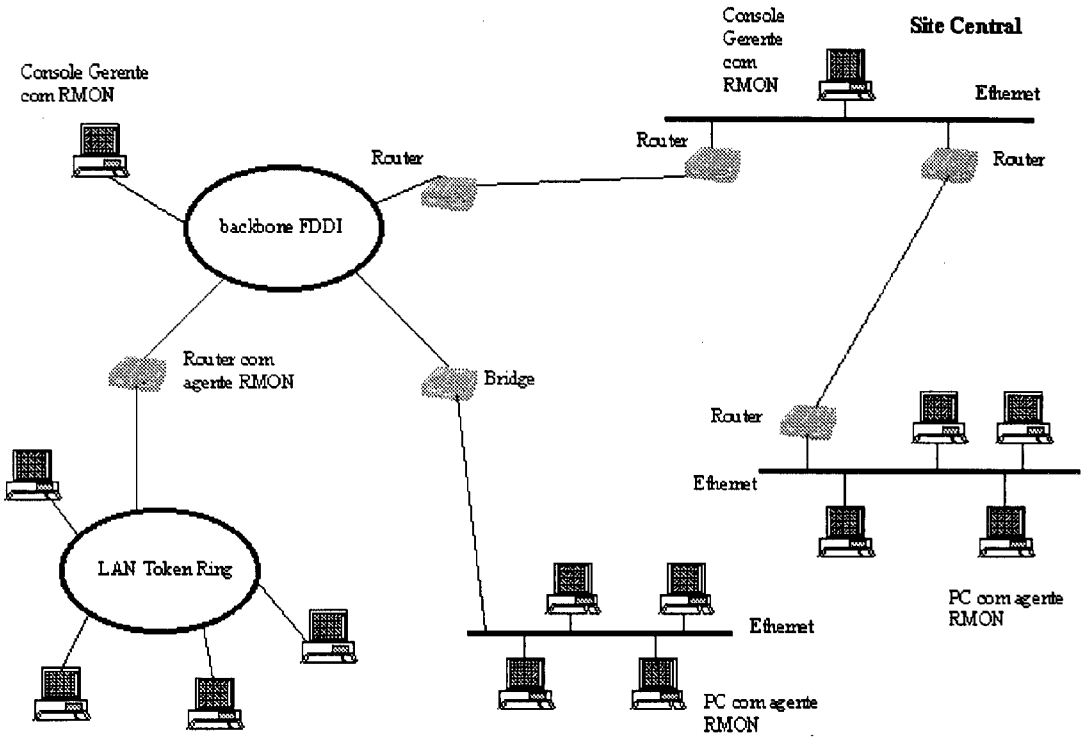

FIGURA 7.9: AMBIENTE DE REDE COM MONITORES RMON

7.3. A Implementação

A seguir, o principal trecho do código fonte, será abordado detalhadamente.

Os pacotes importados no código do *Aglet WebAgent*, incorporam algumas funcionalidades.

Dos *Aglets*:

```
import com.ibm.aglet.*;
import com.ibm.awb.util.Encoding;
```

Da API *AdventNetSNMPv3* de gerência:

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

Funções *Java:*

```
import java.net.URL;
import java.util.Enumeration;
import java.lang.*;
import java.util.*;
import java.io.*;
import java.net.*;
```

A classe principal, *WebAgent* herda da classe *Aglet* para incorporar todas as suas características:

```
public class WebAgent extends Aglet {
```

Desconsiderando a especificação de TAGs HTML, a seguir o método **run** de maior relevância. Este método é executado cada vez que o *Aglet* chega na estação destino.

```
public void run() {
```

Descrição do endereço IP do recurso e do identificador da variável a ser monitorada. Podem existir ***n*** variáveis, separadas por vírgula, caracterizando a expansão da gerencia que o agente pode efetuar. Não é necessário especificar o valor inicial que identifica a origem do objeto (.1.3.6.1.2.1). Assim para a variável de OID .1.3.6.1.2.1.2.2.1.10.1001, somente é necessário especificar 2.2.1.10.1001.

A seguir estão especificados: o IP do *Switch* (150.162.252.80) e o OID da variável *etherHistoryUndersizePkt* (2.2.1.10.1001).

```
String args[] = { "150.162.252.80", "2.2.1.10.1001"};
```

Inicializações das variáveis: **conteúdo** que armazena o resultado dos *pollings*; **total** que armazena o valor de pacotes; **primeiro, segundo**: que armazenam o primeiro e o último valores coletados.

```
String conteudo = "";
long total, primeiro = 0, segundo = 0;
```

Obtém a hora corrente e armazena na variável **tempoAnt**.

```
java.util.Date date = new java.util.Date();
Calendar calendar = Calendar.getInstance();
date = calendar.getTime();
calendar.setTime(date);
int tempoAnt = calendar.get(Calendar.SECOND);
```

Cria um *loop* para coleta de 100 valores (foi definido arbitrariamente a coleta de 100 valores).

```
for (int cont=0; cont<100; cont++)    {
```

Instancia um objeto **opt** com inicializações para o SNMP e verifica se foram especificados o IP e no mínimo uma variável.

```
ParseOptions opt = new ParseOptions(args,options,values,
usage);
if (opt.remArgs.length<2) opt.usage_error();
```

Instancia e inicia a API SNMP.

```
SnmpAPI api;
api = new SnmpAPI();
api.start();
```

Instancia uma sessão SNMP com o objeto **api** criado anteriormente.

```
SnmpSession session = new SnmpSession(api);
```

Seta o destino (IP do recurso gerenciável onde será efetuado o *polling*) para o valor de endereço IP anteriormente especificado.

```
session.setPeername( opt.remArgs[0] );
```

Instancia o objeto que constrói uma PDU. Define o comando *Get request*, para leitura do valor da variável.

```
SnmpPDU pdu = new SnmpPDU();
pdu.setCommand( api.GET_REQ_MSG );
```

Enquanto houver variáveis, insere os *OIDs* das variáveis em objetos *SnmpOID* e insere-os na **pdu**, validando-os.

```
for (int i=1;i<opt.remArgs.length;i++) {
SnmpOID oid = new SnmpOID(opt.remArgs[i]);
if (oid.toValue() == null)
        System.err.println("Invalid OID argument: " +
opt.remArgs[i]);
 else pdu.addNull(oid);
}
```

Abre uma sessão SNMP com tratamento de exceções, caso ocorra algum erro.

```
try {
        session.open();
} catch (SnmpException e) {
System.err.println("Erro abrindo sessão " + e.getMessage());
System.exit(1);
}
```

Envia **pdu** e recebe **pdu** de resposta. Verifica se ocorreu *timeout* (**pdu** retornou um valor nulo).

```
try {
        pdu = session.syncSend(pdu);
} catch (SnmpException e) {
System.err.println("Enviando PDU"+e.getMessage());
if (pdu == null) {
 System.out.println("Request timed out para: " +
opt.remArgs[0] );
System.exit(1);
}
System.exit(1);
}
```

Imprime os dados da **pdu** recebida: endereço e a comunidade.

```
System.out.println("Response PDU received from "
+pdu.getAddress()+ ", community: " +
pdu.getCommunity());
```

Verifica se há erro na **pdu** de resposta. Imprime a **pdu** recebida obtendo o resultado através do método *printVarBinds*.

```
if (pdu.getErrstat() != 0)
        System.err.println(pdu.getError());
else   System.out.println(pdu.printVarBinds());
```

Armazena o primeiro e o último valor coletado para posteriormente, calcular o número total de pacotes. A cada iteração, a **pdu** retorna a quantidade acumulada de pacotes recebidos.

```
if (cont == 0) {
Long aux = (Long)(pdu.getVariable(0)).getVarObject();
primeiro = aux.longValue();
}
if (cont == 99) {
Long aux = (Long)(pdu.getVariable(0)).getVarObject();
segundo = aux.longValue();
}
```

Encerra sessão e finaliza a **thread** da **api**.

```
session.close();
api.close();
```

Obtém a hora corrente e armazena na variável **tempoPos.**

```
java.util.Date dateP = new java.util.Date();
Calendar calendarP = Calendar.getInstance();
dateP = calendarP.getTime();
calendarP.setTime(dateP);
int tempoPos = calendarP.get(Calendar.SECOND);
```

Calcula tempo total para coleta dos 100 valores. Concatena o tempo total de coleta na string **conteudo** que armazena as *pdus* de resposta.

```
if (tempoPos > tempoAnt)
        total = tempoPos - tempoAnt;
else  total = 60 - tempoAnt + tempoPos;
conteudo += total;
conteudo += "\n";
```

Calcula o número total de pacotes e escreve juntamente com o tempo de coleta.

```
long pacotes = segundo - primeiro;
System.out.println("Numero de Pacotes recebidos: " +
pacotes);
System.out.println("Tempo: " + total + "s");
```

Grava a string **conteudo** no arquivo *DadosSNMP.txt.*

```
try {
BufferedWriter r = new BufferedWriter(new
FileWriter("DadosSNMP.txt"));
        r.write(conteudo);
        r.close();
}
catch(IOException e){}
```

8. CONCLUSÕES

8.1. Resultados Alcançados

Este trabalho proporcionou a consolidação prática de uma idéia um tanto teorizada. No amplo cenário distribuído que a Internet e seus recursos proporcionam, seria inaceitável manter a gerência de redes de forma estática. Os problemas no fluxo da informação de gerência que circula na rede, são considerados críticos em alguns casos. Soluções como a apresentada neste trabalho, diminuem ou eliminam significativamente este problema.

A integração de tecnologias demonstrou a flexibilidade que se tem em desenvolver trabalhos envolvendo agentes móveis, Web, gerência de redes SNMP e os vários recursos relacionados.

A viabilidade de utilização de agentes móveis sob demanda, também pode ser comprovada. A evolução e a provável padronização, irão disseminar a utilização de agentes móveis. É possível que num futuro bem próximo, o tempo de processamento seja consideravelmente reduzido e a busca de informações agilizada com a ampla distribuição de agentes móveis na Internet.

O acesso a MIB RMON também foi alcançado de forma satisfatória. Em ambientes maiores, o acesso certamente proporciona ganhos significativos quanto ao tempo de resposta. Em testes realizados com o agente interagindo dentro da subrede do recurso gerenciável, o tempo foi consideravelmente diminuído em comparação ao tempo de acesso a um recurso geograficamente distante.

Esta experiência também abrange MIBs diversas. O fator determinante são as MIBs habilitadas no recurso, proprietárias ou não. O *Aglet* implementado não possui limitações relacionadas às MIBs, sendo genérico para a estrutura de objetos do

protocolo SNMP. Esta é uma característica essencial nos atuais ambientes de rede com a imensa variedade de recursos. A quantidade de variáveis também é ilimitada, mas deve ser considerada para não expandir demasiadamente o tamanho do agente, atualmente com o tamanho de 11Kbytes de *bytecodes* Java.

8.2. Perspectivas Futuras

Durante a realização deste trabalho e através dos resultados obtidos, surgiram idéias que podem dar continuidade a este, tais como:

a) Em gerência de redes, podem ser explorados um número maior de variáveis e MIBs como a RMON II e a ATM-MIB.

b) Realizar uma análise de desempenho comparativo da gerência tradicional sobre gerência com mobilidade em um ambiente real.

c) Técnicas de inteligência artificial podem vir a compor a funcionalidade do *Aglet*, incluindo habilidades para o agente aprender e reagir baseado no conteúdo e no ambiente, podendo ser considerado "inteligente".

d) A mobilidade pode ser explorada na iteração entre agentes, entre plataformas de agentes móveis distintas e possivelmente na interoperabilidade com outras tecnologias como *CORBA, RMI* ou tecnologias voltadas exclusivamente para a *Web* como *Servlets, JSP* (*Java Server Pages*) [JAV00] e outras.

e) Outros trabalhos de desenvolvimento com plataformas de agentes móveis tem sido realizados na UFSC, relacionados a banco de dados, segurança e comércio eletrônico. Esses trabalhos podem compor um ambiente integrado de funcionalidades. O agente de gerenciamento móvel poderia previamente monitorar a subrede destino, analisando índices de tráfego para determinar o melhor momento de enviar outro agente, de comércio eletrônico, por exemplo.

9. REFERÊNCIAS BIBLIOGRÁFICAS

[AGL00] *Aglets* Plataform. Dezembro, 1999. Site http://www.trl.ibm.co.jp/aglets/

[ADV00] *AdventNetSNMPv3* API. Dezembro, 1999. Site http://www.adventnet.com

[BIE00] BIESZCZAD, A.; PAGUREK, B.; WHITE T.. Mobile Agents for Network Management. Laboratórios Bell. Carleton University. 2000.

[BRE98] BRENNER, W.; ZARNEKOW, R.; WITTIG, H.. Intelligent Software Agents. Springer. Berlim, Alemanha. 1998.

[BRI93] BRISA, Gerenciamento de Redes – Uma Abordagem de Sistemas Abertos. São Paulo, Makron Books, 1993..

[CON00] Plataforma Concórdia. Abril, 2000. Site http://www.mitsubishielectric.com

[COS99] COSTA, Tais Freire da Silva. Avaliação Analítica do Uso de Agentes Móveis na Gerência de Redes. Dissertação de mestrado do CPGCC-UFSC. Florianópolis, outubro, 1999.

[DUA99] DUARTE, Otto Carlos M. B.; RUBINSTEIN, Marcelo G. Analyzing Mobile Agent Scalability in Network Management. IEEE LANOMS'99 Latin American Network Operations and Management Symposium. Rio de Janeiro. Dezembro, 1999.

[GER00] Gerência de Redes. Fevereiro, 2000. Site http://penta2.ufrgs.br

[GRA00] Plataforma Grasshopper. Março, 2000. Site http://www.grasshopper.de

[JAV00] Java Documents and API. Agosto 2000. Site: http://java.sun.com

[KOC97] KOCK, Luiz Fernando. Agentes Autônomos para Gerenciamento de Redes. Dissertação de mestrado do CPGCC-UFSC. Florianópolis, outubro, 1997.

[LOR98] LORENSET, Vera Lúcia. Gerenciamento Distribuído TMN: uma Experiência em Supervisão de Alarmes com CORBA. Dissertação de mestrado do CPGCC-UFSC. Florianópolis, abril, 1998.

[MAG92] Revista Data Communications Magazine. Maio 1992.

[OSH98] OSHIMA, M.; LANGE, D.. Programming and Deploying Java Mobile Agents with Aglets. Addison Wesley.

[RMO00] RMON Overview. Agosto 2000. Site http://support.baynetworks.com/library/tpubs/html/router/soft110/114070B/N_24.HTM

[RMN00] RMON MIB. Agosto 2000. Site http://www.ietf.org/html.charters/rmonmib-charter.html

[SIL00] SILVA, Jorge. L. C.; SILVA, Arnoldo N.; SOUZA José N. Modelos de Implementação de Gerência de Redes baseados na Arquitetura de Redes Ativas e no Conceito de Delegação de Autoridade. Universidade Federal de Pernambuco e Universidade Federal do Ceará. Artigo submetido ao Simpósio Brasileiro de Redes de Computadores. Março, 2000.

[STA99] STALLINGS, W.. SNMP, SNMPv2, SNMPv3 and RMON1 and 2, Third Edition. Addison Wesley. Massachusetts, EUA. 1999.

[TEN96] TENNENHOUSE, David L.; WETHERALL, David J.. Towards an Active Network Architecture. Laboratório de Ciência da Computação, Instituto de Tecnologia de Massachusetts – MIT. 1996. Site http://www.sce.carleton.ca/netmanage/activeNetworks/mmcn96.html.

[TOW00] Towards secure and distributed management in the Internet. Agosto 2000. Site http://www.ibr.cs.tu-bs.de/~schoenw/resume.shtml

[VOY00] Plataforma Voyager. Abril, 2000. Site http://www.objectspace.com