UFSM

Trabalho de Graduação

Utilização de Agentes Autônomos Na Gerência de Redes: ACD, Um Agente Coletor de Dados

Rafael Kempfer

Curso de Ciência da Computação

Santa Maria, RS, Brasil

2006

Utilização de Agentes Autônomos Na Gerência de Redes: ACD, Um Agente Coletor de Dados

por

Rafael Kempfer

Trabalho de Graduação apresentado ao Curso de Ciência da Computação – Bacharelado, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação.

Curso de Ciência da Computação

Trabalho de Graduação nº 192

Santa Maria, RS, Brasil

2006

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada, aprova o Trabalho de Graduação

Utilização de Agentes Autônomos Na Gerência de Redes: ACD, Um Agente Coletor de Dados

elaborado por

Rafael Kempfer

como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Comissão Examinadora:

Dra. Roseclea Duarte Medina
(Orientador)

Dr. Raul Ceretta Nunes

Dra. Andrea Schwertner Charão

Santa Maria, 19 de Janeiro de 2006.

"A alegria está na luta, na tentativa, no sofrimento envolvido. Não na vitória propriamente dita."

(Mahatma Gandhi)

A minha linda filha, Rafaela

Agradecimentos

Primeiramente, agradeço a minha linda filha, que, mesmo sem falar ainda, com sua alegria e presença constante no meu coração, sempre me deu forças para levantar e seguir em frente.

Agradeço ao meu amor, que me aguentou nos meus momentos de crise, muitas vezes enxugou minhas lágrimas e nunca me deixou desistir ou desanimar.

A minha família, meus pais e meu irmão, por saberem a hora certa de me cobrar e a hora de me levantar o ânimo.

Aos meus queridos amigos de Santa Maria, especialmente ao Marcos por sempre me ajudar na faculdade e ao Claúdio "Covero" por ser um ombro amigo e por saber me distrair quando precisei.

Agradeço também aos meus amigos que ficaram em Santa Rosa, porque sei que sempre torceram por mim e nunca esqueceram que somos como irmãos.

Aos professores do Curso, pelo conhecimento adquirido, especialmente a Prof. Rose que me auxiliou muito bem nesse trabalho.

E por último, mas não menos importante, a essa força maior que rege as nossas vidas. Por me dar a filhota maravilhosa que tenho e por me dar a oportunidade de ocupar um lugar que, infelizmente, é privilégio de poucos no nosso país, a faculdade.

Sumário

Lista de Tabelas

Lista de Figuras

RESUMO

Trabalho de Graduação
Ciência da Computação
Universidade Federal de Santa Maria

Utilização de Agentes Autônomos Na Gerência de Redes: ACD, Um Agente Coletor de Dados

Autor: Rafael Kempfer
Orientador: Dra. Roseclea Duarte Medina
Data e Local da Defesa: Santa Maria, 19 de Janeiro de 2006.

A cada dia, novas redes de computadores são criadas. Além disso, as redes existentes aumentam, com a adição de novos computadores, roteadores, *bridges*, etc; cada um pertencente a fabricantes ou fornecedores diferentes.

Com esse crescimento nas redes de computadores e a importância que elas representam para todo mundo, o gerenciamento eficaz se tornou extremamente relevante. A área de gerência de redes se tornou um campo amplo de pesquisa.

Novas tecnologias estão sendo estudadas para substituir o modelo ultrapassado e ineficaz do gerenciamento centralizado. Uma das soluções é a busca por um gerenciamento através de vários agentes autônomos efetuando tarefas individuais e cooperando entre si para alcançar um objetivo comum, um sistema multi-agente.

Esse trabalho visa o estudo e a construção de um agente móvel, com a tarefa específica de coletar dados, localmente, em cada *host* da rede. O agente proposto é base de um sistema maior, disponibilizando informações para ferramentas de gerenciamento ou para que outros agentes possam executar suas tarefas.

Capítulo 1

INTRODUÇÃO

Os agentes de software ou, simplesmente, agentes, se tornaram alvo de inúmeros estudos e pesquisas. Tudo graças, à possibilidade de utilização de agentes em diversas áreas da Ciência da Computação como: sistemas distribuídos, inteligência artificial, redes de computadores, entre outras.

Um Agente é um sistema computacional, situado num dado ambiente, que tem a percepção desse ambiente através de sensores, tem capacidade de decisão, age de forma autônoma nesse ambiente através de atuadores, e possui capacidades de comunicação de alto-nível com outros agentes e/ou humanos, de forma a desempenhar uma dada função para a qual foi projetado [22]. Apesar de uma definição bem completa, não há uma definição consensual entre os pesquisadores de agentes. Mas podemos notar uma característica importante, que impulsiona várias pesquisas na área de Agentes, a **autonomia**.

O desenvolvimento de agentes inteligentes e agentes para auxílio na educação utilizam a autonomia como característica primordial de um sistema voltado a responder a mudanças no seu ambiente. Como por exemplo, podemos citar o desenvolvimento do AGENTCHÊ [10], um agente de conversação que se comunica através de linguagem natural, com linguajar gauchesco, auxiliando no aprendizado de redes de computadores.

Outras tipos de agentes também despertam interesse de pesquisadores e desenvolvedores, como é o caso de agentes móveis. São agentes livres para "viajar"por *hosts* de uma rede. Criados em um ambiente de execução, ele pode transportar seu estado e código consigo para outro ambiente de execução na rede, onde continua sua execução [5].

Esse tipo de agente, os agentes móveis, proporcionam o estudo e desenvolvi-

mento de sistemas voltados para redes de computadores, como por exemplo:

- Agentes que pesquisam informações distribuídas em uma rede.
- Agentes que levam informações por uma rede, para atualizar os *hosts*.
- Sistemas de agentes para monitoração e notificação.
- Sistemas de agentes que auxiliam no gerenciamento de redes de computadores.

Várias outras aplicações podem ser desenvolvidas com agentes móveis, mas os agentes para gerenciamento de redes de computadores têm grande importância, visto que o atual modelo de gerenciamento centralizado está ultrapassado e não atende com eficiência redes cada vez maiores, mais complexas e de extrema importância.

As redes têm crescido rapidamente e têm envolvido um número cada vez maior de elementos complexos, além do montante de informações operacionais que devem ser monitoradas e processadas crescer drasticamente [2]. Frente a isso, vemos mudar o modo de gerenciamento centralizado para um gerenciamento distribuído seria mais eficiente.

O uso de agentes é uma boa técnica para construção de um sistema de gerenciamento distribuído. A combinação de vários agentes, cada um com uma tarefa específica, trabalhando em conjuntos para alcançar objetivos comuns, pode ser visto como um Sistema Multi-Agente.

Sistemas Multi-Agentes são sistemas constituídos de múltiplos agentes que interagem ou trabalham em conjunto de forma a realizar um determinado conjunto de tarefas ou objetivos. Esses objetivos podem ser comuns a todos os agentes ou não. Os agentes dentro de um sistema multi-agente podem ser heterogêneos ou homogêneos, colaborativos ou competitivos, etc. Ou seja, a definição dos tipos de agentes depende da finalidade da aplicação em que o sistema multi-agente está inserido [12].

Cada agente cumpre a tarefa a qual foi designado. Assim teríamos, por exemplo, agentes responsáveis por:

- coletar dados localmente nos *hosts*;
- interpretar os dados coletados;
- executar tarefas para correção de erros;

- monitorar o tráfego interno e externo como medida de segurança;

Em um sistema dessa espécie, o agente responsável pela coleta dos dados é de extrema importância, visto que, esse agente é a base para o funcionamento do sistema, porque muitos dos outros agentes dependem das informações coletadas para poderem executar as suas tarefas. Além disso, os dados coletados podem ser utilizados por outras plataformas de gerenciamento e também pelas pessoas responsáveis pelo gerenciamento da rede. Um agente coletor é dotado de uma base de conhecimento com regras que o permitem coletar dados usando o protocolo requerido. Estes agentes podem ter um ou mais objetivos que consistem em extrair valores de objetos gerenciados de um ou mais equipamentos da rede [1].

Este trabalho tem por finalidade o desenvolvimento de um agente coletor de dados, o ACD, que se movimenta entre *hosts* de uma rede, executando localmente em cada *hosts* para coletar os dados que foi designado e retornar a origem com os as informações solicitadas. O objetivo de usar um agente móvel para fazer a coleta dos dados é o fato de poder transferir o código e o estado do agente para executar no *host* para onde ele foi mandado. Assim, todas as requisições de dados são feitas e recebidas localmente, causando tráfego na rede apenas no momento em que o agente vai para outro ponto na rede ou retorna a origem. Esse agente é baseado na plataforma de desenvolvimento de agentes, a *Aglets Software Development Kit* (ASDK) e programação Java [7][27].

O objetivo do presente trabalho é construir um agente móvel para coleta de dados, que se comunique localmente com os *hosts* e retorne à origem com os resultados obtidos. A utilização do agente busca:

1. ser simples para se movimentar rapidamente pela rede e eficaz na coleta dos dados;

2. se movimentar pela rede de maneira autônoma, seguindo um itinerário pré-definido;

3. executar em qualquer *host*, independente do Sistema Operacional ou *hardware* utilizado na máquina;

4. manter sua peregrinação, mesmo que um *host* não esteja disponível, passando para o próximo do seu itinerário;

5. coletar os dados localmente em cada *host* em que passa de modo transparente ao usuário da máquina, e carregar consigo esses dados;

6. reduzir o tráfego na rede, causado por requisições feitas de modo centralizado aos agentes SNMP de cada host;

7. obter informações que o modelo centralizado não consegue requisitar, apenas uma execução local pode conseguir;

8. deixar os dados coletados disponíveis para que outros agentes possam utilizá-los, como por exemplo:

   - um agente estacionário que analisa os dados e toma as medidas cabíveis;
   - um agente de conversação, como o AGENTCHÊ [10], que pode ler os dados e usá-los como material nos seus ensinos sobre redes de computadores;
   - uma plataforma de gerenciamento, como o Trauma Zero [26], que pode analisar os dados coletados ou buscar neles informações que a plataforma não conseguiu de maneira centralizada;
   - um Gerente de Rede, que pode verificar as informações coletadas e executar determinadas tarefas a partir da análise dos dados.

9. ser a base de um sistema maior, um sistema de múltiplos agentes buscando executar suas tarefas e utilizando-se dos dados coletados pelo ACD para alcançar os seus objetivos.

Buscando uma base teórica sólida e uma boa explanação sobre o trabalho desenvolvidos, este texto foi organizado da seguinte forma:

- O capítulo 2 trata do gerenciamento de redes como é conhecido atualmente, dando uma ênfase ao uso do protocolo SNMP;
- O capitulo 3 introduz a tecnologia de agentes, sua definição, características, classificação e uso de agentes;
- No capítulo 4 descreve mais sucintamente os agentes com características de mobilidade;

- O capítulo 5 apresenta a plataforma de desenvolvimento de agentes, a Plataforma *Aglets*;
- O capítulo 6 mostra o ACD propriamente dito, sua definição, características e implementação;
- O capítulo 7 apresenta os resultados obtidos;
- E por fim, no capítulo 8, temos as conclusões e em seguida as referências bibliográficas.

Capítulo 2

GERÊNCIA DE REDES

Com um pouco de sorte, uma rede pequena sem um uso significativo pode necessitar uma quantidade limitada de esforços por parte do gerente ou do administrador da rede para deixa-lá de acordo com as exigências da empresa. Como as redes crescem em complexidade, as necessidades para gerenciar crescem a um ponto onde as ferramentas e técnicas de gerenciamento se tornam indispensável para se obter uma rede eficiente e eficaz [21].

A monitoração e o controle dos inúmeros dispositivos que compõem uma rede de computadores, faz com que se torne necessária uma forma de gerenciamento eficaz. Mas a atual forma de gerenciamento centralizada, muitas vezes não atende os requisitos da empresa.

Este modelo centralizado está baseado em interações entre os dispositivos de uma rede e uma estação de gerenciamento. Em cada dispositivo de rede existe um processo que é responsável pela comunicação com o dispositivo de rede, recuperando e alterando as informações. Ele também é responsável por atender as solicitações da estação de gerenciamento, fornecendo informações sobre o dispositivo. A estação, por sua vez, irá processar essas informações e fornecer ao gerente da rede os dados sobre o estado atual da rede. O gerente é a pessoa responsável por analisar os dados da estação e tomar as medidas cabíveis.

Outra opção é o modelo hierárquico. Neste modelo, cada sub-rede possui uma subestação que é responsável pelo recolhimento e processamento das informações da sub-rede. Mas ainda precisam interagir com a estação principal, criando uma hierarquia que pode possuir vários níveis de subestações, dependendo do tamanho da rede [6].

O grande problema com o modelo centralizado é o grande fluxo de informações

no barramento perto da estação de gerenciamento. Em redes cada vez maiores e mais velozes, o modelo hierárquico também apresenta esse problema, mesmo que em proporções às vezes menores.

Segundo Nassif [18], a gerência centralizada também impede que os administradores das redes locais tenham acesso às informações de seu ambiente. Somente os administradores de rede localizados no ponto onde está a plataforma central de gerenciamento é que têm acesso aos dados coletados e a situação da rede.

Outro problema diz respeito aos modelos de gerência não permitirem uma reação rápida das estações de gerenciamento às freqüentes mudanças que ocorrem nas redes de alta velocidade. Muitas vezes, mudanças ocorridas na rede ou nos dispositivos acabam não chegando até o conhecimento do gerente.

As aplicações para gerência de redes disponíveis no mercado utilizam uma arquitetura cliente-servidor típica baseada em protocolos padronizados como o SNMP (*Simple Network Management Protocol*) e RMON (*Remote MONitoring*). Estas ferramentas possuem características proprietárias que dificultam sua integração com aplicações de terceiros, apesar do uso de protocolos padrão. Além disso, possuem alto custo de aquisição e, principalmente, de implantação e operação [19].

2.1 *Simple Network Management Protocol*

Com o crescimento das redes de computadores, especialmente da Internet, tornou-se evidente a necessidade de ferramentas de gerenciamento melhores. O SNMP foi definido na RFC 1157, publicada em 1990. Junto com a RFC 1155, que trata de informações de gerenciamento. Esse protocolo se tornou bastante usado e com o tempo as deficiências foram aparecendo. Em 1993 foram publicadas as RFCs 1441 e 1452, que definiram a segunda versão do SNMP, chamado de SNMPv2. E em 2002, a RFC 3410 definiu o SNMPv3 [24].

O protocolo SNMP pertence à camada de aplicação do modelo ISO/OSI, como mostra a figura 2.1, e seu objetivo é facilitar a troca de informações de gerenciamento entre os dispositivos de rede. Com esse protocolo, os gerentes podem, por exemplo, obter a performance da rede e encontrar problemas na mesma.

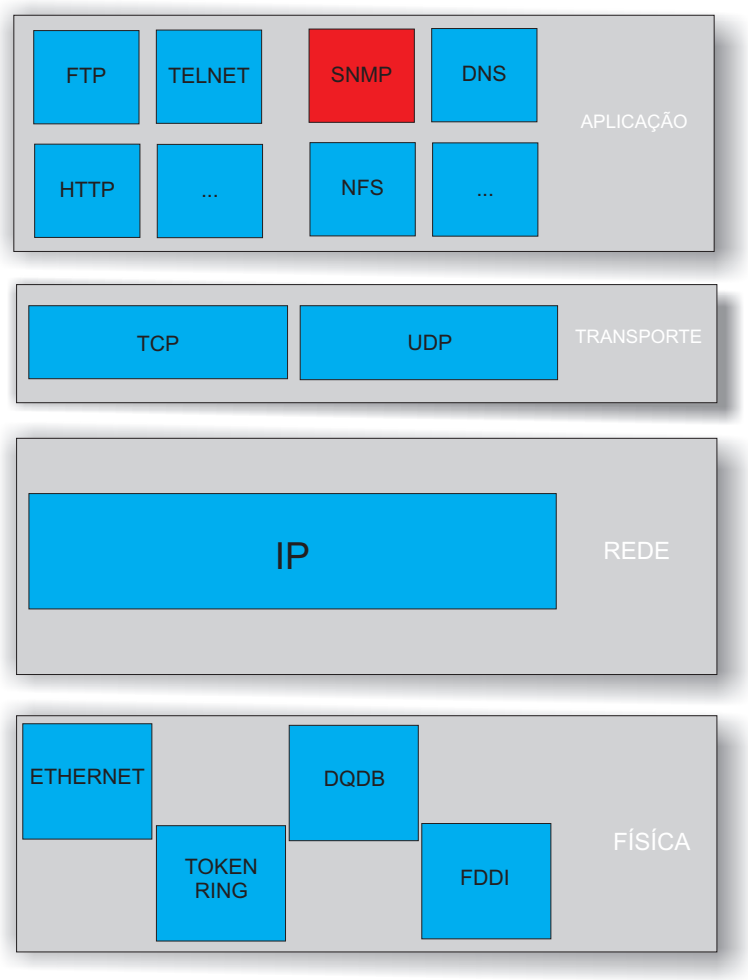

Figura 2.1: SNMP na camada de aplicação

2.1.1 O Modelo SNMP

Segundo [9], podemos definir o modelo SNMP de uma rede gerenciada em quatro componentes:

Nós Gerenciados

Recursos como *hosts*, *bridges*, *hubs*, roteadores, impressoras e outros dispositivo que são capazes de comunicar informações de status com o mundo externo. Para poder ser gerenciado, um nó deve ser capaz de executar um Agente SNMP, que é responsável por coletar e alterar informações do dispositivo.

Essas informações coletadas pelo agente são armazenadas em um banco de dados local na forma de variáveis que têm a função de descrever o estado e o histórico do equipamento. Os agentes também podem alterar essas variáveis, afetando as

operações dos nós gerenciados.

Estações de Gerenciamento

Estações de gerenciamento são computadores que executam um software de gerenciamento de redes, responsável por se comunicar com os agentes, fazendo o gerenciamento efetivo da rede. Através destas estações os gerentes podem obter dados relevantes sobre os dispositivos e alterar as suas variáveis quando preciso.

As estações de gerenciamento são compostas por um conjunto de aplicações de gerenciamento, uma interface de controle e uma base de dados extraídos dos componentes da rede.

Informações de Gerenciamento

São uma coleção de dados sobre cada dispositivos que são armazenadas em uma base de informações. Essa base de informações é conhecida como MIB (*Management Information Base*), que define todos os objetos gerenciados pelo agente.

Para o modelo de gerenciamento SNMP, a base de informações é padronizada, onde cada dispositivo tem um conjunto de informações que devem ser observadas, e ainda, com que formato devem ser mantidas [2].

Um Protocolo de Gerenciamento

O protocolo de gerenciamento, SNMP, é responsável pela comunicação entre os agentes nos dispositivos gerenciados e a estação de gerenciamento.

Como definido anteriormente, o SNMP opera na camada de aplicação. Ele utiliza o protocolo UDP (*User Datagram Protocol*) para efetuar a comunicação entre os agentes e as estações de gerenciamento.

Como o protocolo UDP não é orientado a conexão, todas as transferências devem aguardar confirmação. Assim sendo, quando a estação de gerenciamento solicita alteração de alguma variável, ela deve aguardar a confirmação do agente. Caso isso não ocorra, a solicitação é reenviada.

2.1.2 Representação dos Dados

Vários equipamentos, de diferentes fabricantes são usados em uma rede, o que pode dificultar a comunicação entre eles. Por isso, é necessário a padronização das estruturas de dados, dos formatos das mensagens e a definição das regras de codificação.

Segundo [9], para tornar a comunicação entre equipamentos produzidos por diferentes fornecedores possível, é essencial que todos esses objetos sejam definidos de

uma forma padronizada e neutra (em termos de fornecedor). Além disso, uma forma padronizada é necessária para que os objetos sejam codificados para transferência através da rede.

Para tentar solucionar isso, o SNMP baseia-se na sintaxe abstrata ASN.1 *Abstract Syntax Notation*, que foi tirada do modelo OSI, sendo que, na verdade, o SNMP usa apenas um conjunto simplificado desta notação.

2.1.3 Operações de Gerenciamento

O SNMP define sete mensagens básicas de gerenciamento que podem ser divididas em 3 grupos de operações: operações de leitura, de alteração e de informação. A tabela 2.1, adaptada de [9], mostra seis mensagens, sendo a sétima a mensagem de resposta.

Tabela 2.1: Tipos de operações SNMP

Mensagem	Descrição
Get-request	Solicita o valor de uma ou mais variáveis.
Get-next-request	Solicita a variável seguinte a esta.
Get-bulk-request	Extrai uma tabela longa ou um grande bloco de dados.
Set-request	Atualiza uma ou mais variáveis.
Inform-request.	Mensagem enviada entre gerentes, cujo objetivo é descrever uma MIB local.
SnmpV2-trap.	Relatório sobre *traps* que é enviado de um agente para um gerente.

As operações de leitura são executadas pela estação de gerenciamento com o intuito de obter os valores de determinadas variáveis dos nós gerenciados. São definidas pelas mensagens *GET*.

Assim como as operações de leitura, as de alteração também são efetuadas pela estação de gerenciamento, mas com o objetivo de alterar ou atualizar o valor de uma variável. A mensagem que define essas operações é o *SET*.

Já as operações de informação possuem dois tipos de mensagens, uma é executada por um gerente para avisar a outro sobre quais variáveis está gerenciando e a outra pelo agente, para informar ao gerente sobre a ocorrência de *traps*. *Traps*

são situações significativas que ocorreram no agente, sobre as quais o gerente deve tomar conhecimento, analisar e tomar as medidas cabíveis.

2.1.4 Formato das Mensagens

As mensagens SNMP são divididas em duas partes: o cabeçalho e a unidade de dado do protocolo, o PDU (*Protocol Data Unit*). O cabeçalho é composto pela versão (*version*) e um nome comunitário (*community*). O nome comunitário tem a função de definir um meio de acesso para um grupo de estações de gerenciamento e como forma de autenticação.

A segunda parte da mensagem, o PDU, varia de acordo com a operação desejada. A figura 2.2, adaptada de [2], mostra a estrutura da mensagens para as operações: *get-request, Get-next-request, Set-request, Get-response e Trap*.

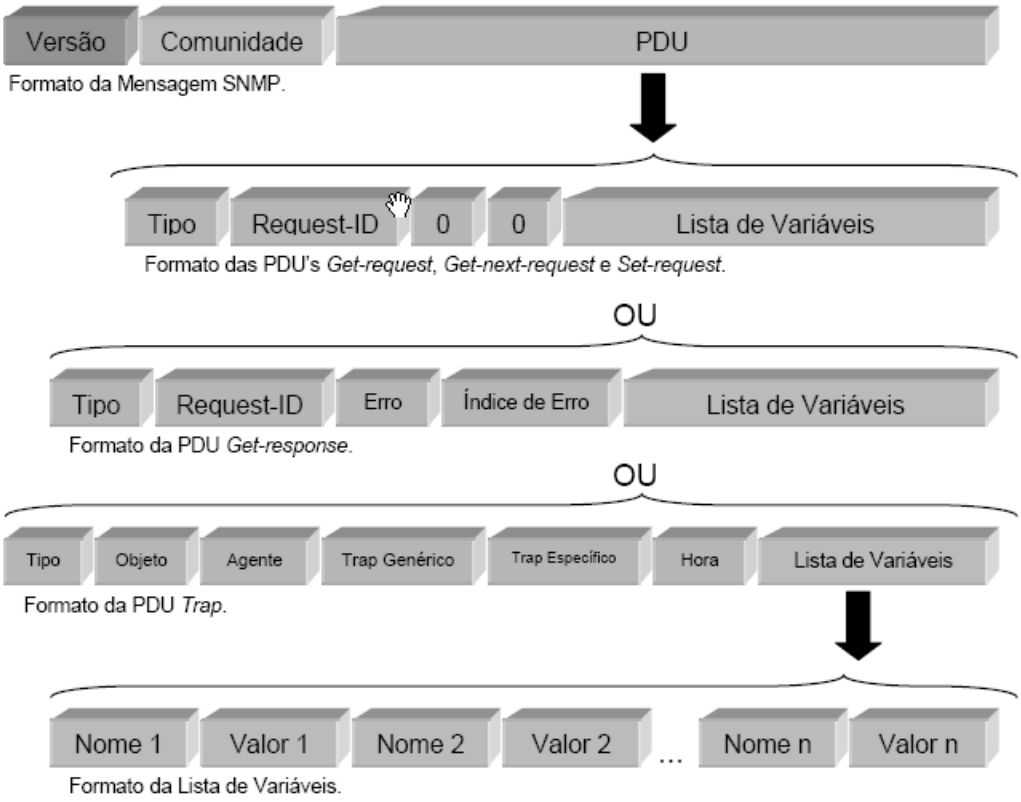

Figura 2.2: Formato das Mensagens SNMP, [2]

Os campos das mensagens definidas na figura 2.2 são:

- Tipo, define o tipo PDU (*Get-request, Set-request, Trap*, etc.);

- *Request-ID*, identificação única, especificada para cada PDU enviada;
- *Erro*, indica se alguma exceção ocorreu durante o processamento da mensagem;
- Índice de Erro, quando diferente de zero, indica qual variável da lista causou a exceção;
- Lista de variáveis, lista de nomes (OID) de objetos da MIB e valores correspondentes;
- Objeto, tipo de objeto que gerou a PDU *trap*;
- Agente, endereço ou identificação do agente que gerou a *trap*;
- *Trap* genérico, tipo de evento;
- *Trap* específico, código específico da *trap*;
- Tempo, informa a hora em que o evento ocorreu.

No SNMPv2 foi inserido a operação *Get-request-bulk*, e os campos da PDU passam a ser:

- Tipo, *Request-ID* e variáveis: análogos as anteriores;
- *Nonrepeaters*: Especifica o número de variáveis na lista de variáveis de ligação para onde um sucessor lexicográfico retorna.
- *Max-Repetitions*: Especifica o número de sucessos lexicográficos que são retornados das variáveis remanescentes na lista de variáveis de ligação.

Capítulo 3

AGENTES

3.1 DEFINIÇÃO

O termo agente é usado sobre várias definições, sendo que não existe uma geral e bem aceita entre os autores, o que se vê são conceitos dados de acordo com o uso de agente. Para Franklin [17], os trabalhadores envolvidos na pesquisa de agentes oferecem uma variedade de definições, sendo que, cada uma espera explicar o seu uso da palavra "agente".

No escopo desse trabalho, a definição que será usada é a de Lobato [11], onde agente é uma entidade física ou virtual que reúne as seguintes características:

- Possui uma representação parcial de seu ambiente, sendo capaz de perceber e atuar nele dirigido por um conjunto de objetivos;
- Possui mecanismos que lhe permitem comunicar-se com outros agentes;
- Possui habilidades especiais e oferece serviços, utilizando seus próprios recursos;
- É capaz de reproduzir a si mesmo;
- Possui comportamento que tende à satisfação de seus objetivos, levando em consideração as restrições e aspectos anteriores.

A figura 3.1, retirado de [11], mostra um esquema típico de um agente, fazendo uma analogia ao corpo humano, com os olhos representando os seus sensores e braços e pernas como atuadores do agente em um suposto ambiente.

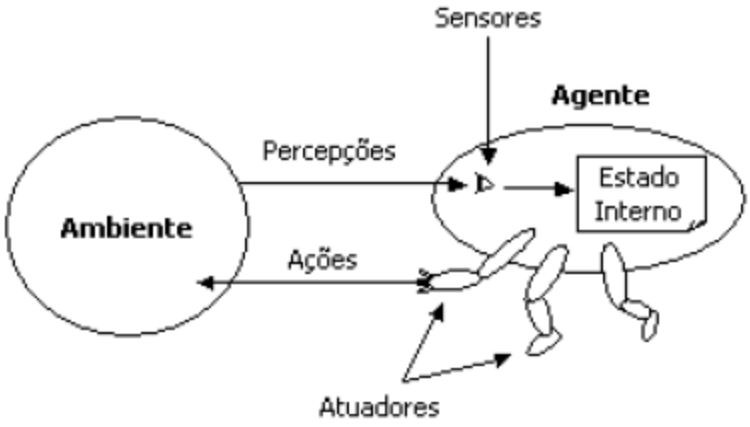

Figura 3.1: Esquema típico de um agente, [11]

3.2 CLASSIFICAÇÃO

A classificação dos agentes, dada por Nwana [8], apresenta várias dimensões de classificação dos agentes existentes, sendo elas:

1ª Dimensão: Mobilidade. Os agentes podem ser ativamente móveis, vagando por uma rede de acordo com o seu itinerário; ou permanecerem executando no local onde foram criados, chamados estacionários;

2ª Dimensão: Reatividade. Os agentes reativos executam suas tarefas no momento em que forem estimulados, sem ter memória do que já foi realizado e sem previsão de ação futura. Em contraposto, os agentes cognitivos possuem a capacidade de raciocinar sobre ações tomadas no passado e planejar ações futuras;

3ª Dimensão: Propriedades Ideais. Classificação de acordo com três propriedades consideradas fundamentais por alguns cientistas: autonomia, agem autonomamente para atingir suas metas; aprendizagem, adaptam-se ao usuário e ao ambiente e aprendem com suas experiências; cooperação, usam linguagens e protocolos padrões para atingir metas comum. Conforme mostra a figura 3.2;

4ª Dimensão: Papel Assumido na Aplicação. Classificação de acordo com as tarefas executadas por um agente e a forma como os seus objetivos serão alcançados dentro da aplicação;

5ª Dimensão: Agentes Híbridos. Combinam duas ou mais das outras dimensões em um único agente.

Figura 3.2: Características Ideais, [8]

Os agentes podem aparecer em várias dimensões descritas anteriormente. Mas Bernardes [13] apresenta uma redução nas combinações dessas várias dimensões, que não seriam possíveis de representação gráfica simples, a uma lista arbitrária, mas que cobre a maioria dos tipos de agentes em investigação atualmente. Desta forma, identifica-se sete tipos de agentes:

1. Agentes Colaboradores - possuem autonomia e, principalmente, poder de cooperação com outros agentes para executar tarefas para seus donos;

2. Agentes de Interface - possuem autonomia e capacidade de aprender com as ações executadas em um programa. É uma espécie de assistente pessoal que monitora estas ações, com o propósito de desenvolver um modelo de habilidades do usuário e automaticamente ajudá-lo quando surgirem problemas;

3. Agente móveis - possuem a capacidade de se movimentar por diversos nodos de uma rede, independente do seu tamanho, interagindo com máquinas, coletando informações e retornando após executar as suas tarefas designadas por um usuário, um programa ou, até mesmo, outro agente;

4. Agente de Informação (para Internet) - tem o papel de gerenciar, manipular e ordenar informações oriundas de várias fontes distribuídas. Estes são utilizados

em máquinas de busca, comércio eletrônico, etc., auxiliando seus usuários no gerenciamento de informações disponíveis na Internet;

5. Agentes Reativos - estes agentes não possuem nenhuma forma de representação simbólica interna dos seus ambientes, atuando com um tipo de comportamento baseado em estímulo-resposta;

6. Agentes Híbridos - são agentes que possuem uma integração das várias filosofias existentes, como, por exemplo, um agente de interface reativo estático;

7. Agentes Inteligentes - seriam agentes capazes de apresentar todas as características citadas possíveis de autonomia, aprendizagem e cooperação, mas que, na realidade, representam apenas o desejo dos pesquisadores atualmente.

O Agente Coletor de Dados, objetivo deste trabalho, é definido como um agente híbrido. Possui as características de: **Mobilidade**, sua principal característica; **reatividade**, porque não guarda memória sobre o que foi realizado e **autonomia**, pois o agente "caminha"pela rede e executa as suas tarefas autonomamente.

3.3 AGENTES *versus* OBJETOS

Desde o surgimento dos agentes, a uma grande dificuldade em distinguir e separar o que é AGENTE e o que é, simplesmente, um OBJETO. Vários autores tentam diferenciar os dois paradigmas, a maioria se baseando na característica de autonomia dos agentes.

O paradigma de agentes, surge como uma evolução natural da orientação a objetos. Como os objetos, a abstração agente possui uma memória e um comportamento. Porém, ela não é uma entidade passiva, como os objetos tradicionais. A priori, um agente pode ser definido como um objeto ativo, autônomo, social e com capacidade de aprendizado [16].

A tabela 3.1, apresentada por [11], faz um comparativo entre agentes e objetos.

Como descrito, as principais diferenças estão na capacidade dos agentes executarem autonomamente trocando mensagens entre si e não invocando métodos uns dos outros.

Tabela 3.1: Agentes X Objetos - Comparativo

Objetos	Agentes
Definidos pela terminologia da orientação a objetos: atributos, métodos, mensagens, etc.	Coleção de entidades que executam ações autonomamente.
Os objetos são usados por quaisquer outros objetos para executar ações.	Estado interno complexo, com propósito bem definidos.
Objetos têm papéis definidos, não sendo flexíveis em seus objetivos.	Agentes mudam seus papéis dinamicamente, de acordo com a aplicação.
No modelo de objetos distribuídos, cada objeto contribui de maneira independente para alcançar o objetivo da aplicação.	Em um sistema Multi-Agentes, várias entidades cooperam entre si para alcançar um único objetivo
Objetos invocam métodos.	Agentes conversam entre si.
Objetos esperam por eventos específicos para começar a executar.	Agentes decidem ativamente quais eventos devem ser acionados.

3.4 APLICAÇÕES DE AGENTES

A utilização de agentes não está restrita a apenas uma área da Ciência da Computação. O seu uso pode ser visto na Gerência de Redes, Sistemas Distribuídos, Inteligência Artificial, etc.

Algumas aplicações de agentes são descritas por [10]:

Prestação de Serviços: Modelo centrado em redes, com todas as informações armazenadas e o cliente só consulta quando necessário. Os agentes estabelecem a interface e gerenciam a informação, tornando a estrutura de hardware e software mais simples, diminuindo custos. Grandes empresas como Oracle, IBM e Sun já apostaram nesse tipo de modelo;

Assessoria e Suporte Empresarial via Internet: Implantação de um laboratório virtual de suporte técnico e de serviços, proporcionando um canal direto de comunicação entre a universidade e a iniciativa privada. Seriam disponibilizados os serviços de assessoria em planejamento e controle de produção, redução de custos de administração de capital de giro, consultor de compras, sistemas de apoio a decisão e análise de custos;

Consultas sobre Decisões de Compras: Oferece os serviços básicos de negociação inteligente, assessoria inteligente e sistemas de consulta de informações;

Simulação Interativa de Ambientes: Imitação de seres humanos atuando em papéis específicos em ambientes simulados, como o desenvolvimento de pilotos inteligentes autônomos, em uma batalha simulada;

Simulação de Campos de Batalha: Simulação de agentes inteligentes com o objetivo de diminuir o potencial humano necessário para controlar as entidades inimigas. É usado uma hierarquia de agentes no exército, com restrições para os agentes de nível mais baixo da hierarquia;

Logística e Simulação: Cada agente suporta uma tarefa específica e possui suas próprias estratégias de solução, sendo capaz de solicitar serviços de outros agentes;

Gerenciamento de Serviços em Redes de Comunicação: Comunidade de agentes que cooperam entre si para resolver uma variedade de problemas complexos em uma rede de telecomunicações;

Comunicação entre Agentes: Uso da linguagem de comunicação KQML (*Knowledge Query and Manipulation Language*) entre agentes e a compatibilidade com www;

Realidade Virtual: Possui aplicações em diversas áreas, com destaque em treinamento e educação, sendo utilizados agentes inteligentes atuando como participants humanos para monitorar as atividades do usuário;

Atores Virtuais e Ambientes Virtuais: Ambiente virtual é um mundo simulado por computador, constituído por agentes, objetos e processos. Um ator virtual representa um modelo computacional de uma figura humana, podendo mover-se e agir em um ambiente virtual.

Capítulo 4

AGENTES MÓVEIS

4.1 DEFINIÇÃO

Agentes móveis, ou simplesmente AM, são processos de software cuja característica principal é a capacidade de se transportar entre diferentes nós da rede, ou entre diferentes redes, como no caso da Internet [3].

Estes agentes são entidades de software com duas características principais: a capacidade de se movimentar pela rede seguindo algum padrão ou regra e a capacidade de se comunicar com outros agentes e com os dispositivos de rede. Essas características permitem ao agente móvel atuar em diferentes localizações de uma rede.

Granville [6] define que todos os AM possuem um conjunto de modelos que definem seu comportamento: um modelo para o ciclo de vida do agente, um modelo computacional, um modelo de segurança e um modelo de comunicação. O ciclo de vida de um agente define como o mesmo se comportará durante sua existência, em que circunstâncias haverá clonagem do agente, quando uma instância deve ser excluída, etc. O modelo de comunicação define a forma como os agentes se comunicam entre si e com o ambiente. Além dos modelos citados, um modelo importante para agente móvel é o de locomoção que define como um agente move-se na rede. De acordo com este modelo, um agente é capaz de decidir, por exemplo, qual o próximo nodo a ser visitado em uma rede. Se isso já ocorreu o agente precisa decidir que outra ação de locomoção deve ser tomada. Quando um agente migra, a sua execução é suspensa no local onde ele está e o agente é transportado, junto com o seu código, dados e estado de execução, para um outro "lugar"na rede onde a execução é retomada.

Com essas definições, chegamos a estrutura de um AM:

- Estado: necessário para continuar a execução após o deslocamento;
- Implementação: código do agente;
- Interface: para comunicação com outros agentes ou com usuários;
- Identificador: único e imutável;
- Autoridades: autor e dono.

4.2 GERENCIAMENTO COM AM

O avanço tecnológico vem deixando as redes de computadores mais complexas, velozes e heterogêneas, o que torna as arquiteturas atuais de gerenciamento inadequadas. Algumas situações freqüentemente ocorrem nestas redes: problemas relacionados ao tráfego de dados, fluxo de gerenciamento complexo, dificuldade de manter a consistência dos dados, protocolo e interações não são suficientemente rápidos para acompanhar modificações quase que instantâneas nas redes de alta velocidade [6].

Certas tarefas exigem rapidez na coleta e na disponibilização de informações e eficiência na análise destas informações por parte do administrador da rede ou do software de gerenciamento. Mas muitas vezes, esses requisitos não são alcançados devido ao grande volume de informações que é gerado. O modelo centralizado de gerenciamento pode causar, em algumas situações, um grande tráfego de informações perto da estação gerenciadora. E mesmo em redes com gerenciamento descentralizado, os segmentos de rede podem causar tráfego indesejado na troca de mensagens entre as estações de gerenciamento e os elementos gerenciados.

Os AM podem ser responsáveis pela recuperação de informações nos elementos gerenciados, tarefas de alterações de variáveis da MIB, segurança da rede e outras áreas do gerenciamento. O uso de diversos agente, cada um com sua atividade específica, é definido como um sistema multi-agente.

Um sistema multi-agente no gerenciamento é responsável pela manipulação e processamento dos dados de uma rede, transformando-os em informações de gerência. Para a composição do sistema multi-agente, são identificadas algumas atividades essenciais: coleta de dados dos dispositivos da rede, a classificação destes

dados, análise dos dados a fim de transformá-los em informações de gerência e apresentação das informações.

Os AM coletam informações da MIB ou do agente SNMP estacionário e posteriormente retornam à aplicação de gerenciamento ou gerente os resultados da busca. Os AM ficam encarregados de alterar listas de roteamento, realizar configurações para garantir o balanceamento de carga e a conectividade de um segmento da rede.

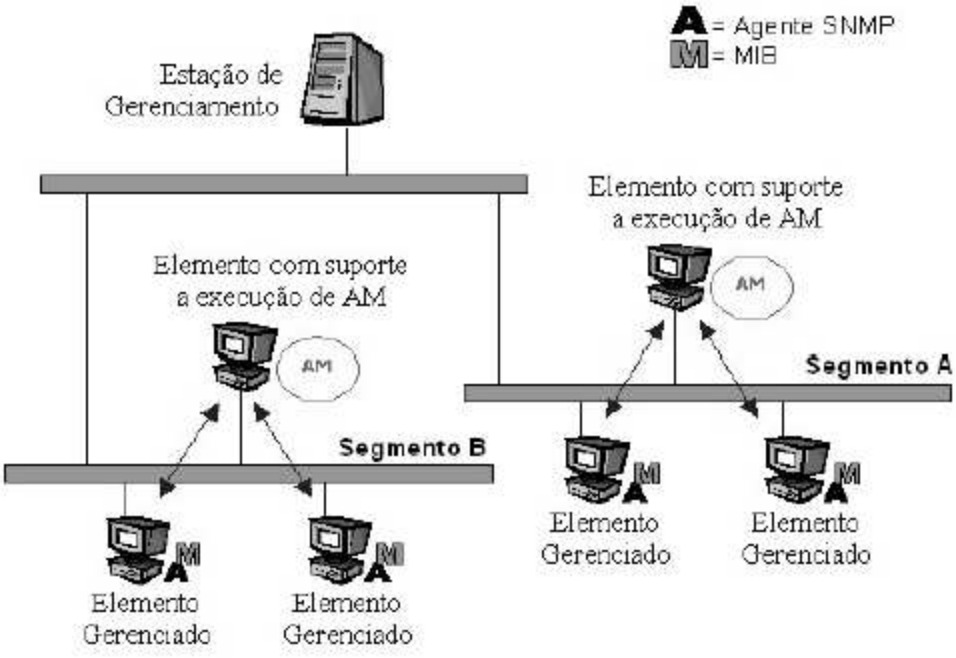

Figura 4.1: Funcionamento de uma aplicação de gerência SNMP que utiliza Agentes Móveis, Reis [2]

Reis [2] define um modelo, no qual são utilizados vários AM, cada um direcionado a realizar operações em um segmento de rede específico. Depois que a estação de gerenciamento inicia os AM, estes são enviados a elementos específicos da rede e de lá, trocam mensagens SNMP com os agentes SNMP localizados em cada elemento gerenciado. A troca de mensagens sendo feita localmente é uma das grandes vantagens deste modelo. Depois disso os AM transferem-se para estação de gerenciamento, juntamente com as informações coletadas.

4.3 AM *versus* SNMP

Como descrito na seção 2.1, o gerenciamento através do modelo SNMP, provê uma forma de gerenciamento centralizado, onde os dados são requisitados as estações gerenciadas através de requisições feitas pela rede. Com o uso de Agentes Móveis, esses dados são coletados localmente em cada estação gerenciada, com o uso de agentes que se transferem pela rede e coletam estes dados.

Este modelo de coleta de dados, com o uso de agentes móveis, possui muitas vantagens sobre o modelo tradicional. Lange [5] define 7 razões para o uso de agentes móveis. A seguir apresentaremos as razões que melhor se aplicam à coleta de dados, comparando o modelo de gerenciamento SNMP com o uso de AM:

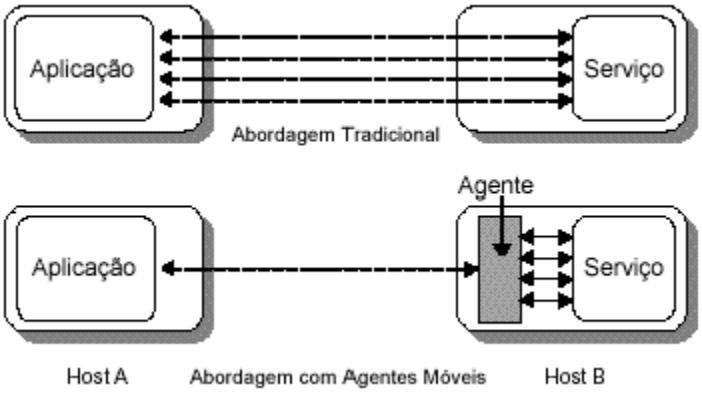

Figura 4.2: Execução local, Lobato [11]

- Reduzem o tráfego na rede: na abordagem tradicional, as requisições aos dados da estação gerenciada são feitas seqüencialmente através da rede, enviando o pedido e recebendo o resultado. Com AM, o agente é enviado até a estação gerenciada e, então, faz todas as requisições aos dados localmente, como mostra a figura 4.2.

- Reduzem a latência da rede: com a solicitação de dados feita pelo modelo centralizado, o uso da rede por estas transferências pode prejudicar o tempo de resposta da estação gerenciadora a outros serviços. Com os AM, a rede fica livre para a estação gerenciadora usá-la, enquanto o AM está fazendo a sua coleta localmente no *host*.

- Executam assíncronamente e autonomamente: se a conexão a um *host* for perdida no meio de uma transferência, ela deverá ser refeita novamente no modelo tradicional. Ou ainda, se um *host* estiver indisponível, o gerente deverá acionar a plataforma novamente em outro momento. No caso dos AM, se um *host* estiver inacessível, o agente parte para outro endereço e posteriormente pode voltar automaticamente para o *host* perdido. Mas se o agente estiver em uma estação gerenciada e esta perde conexão, o AM não é destruído, apenas suspenso até a conexão retornar, mantendo os dados já coletados. A figura 4.3 mostra como isso ocorre.

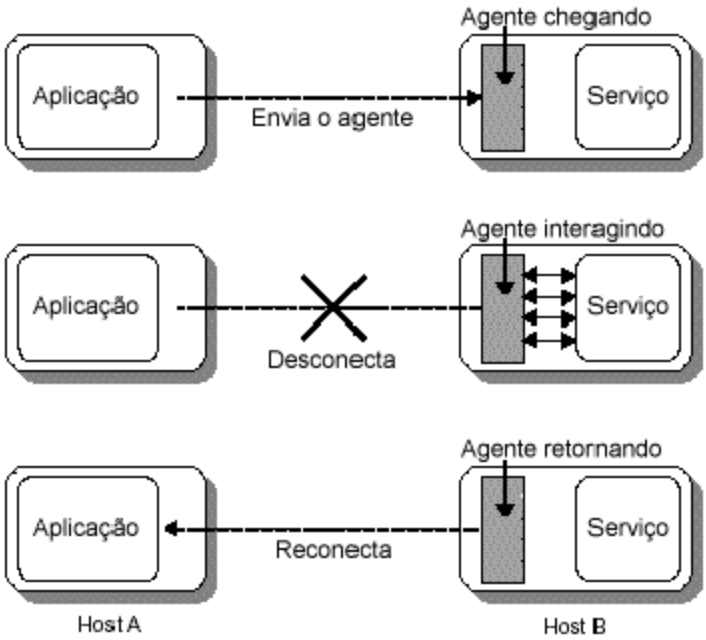

Figura 4.3: Execução autônoma, Lobato [11]

- Encapsulam protocolos: em uma rede heterogênea, as máquinas podem executar com protocolos diferentes ou em versões diferentes. E a manutenção da padronização destes protocolos é muito dispendiosa. Os AM encapsulam estes protocolos, podendo executar de acordo com cada *host*.

- São naturalmente heterogêneos: em uma rede de computadores existe máquinas de todos os tipos, que variam tanto no hardware, quanto no software. Os AM se movimentam pelos *hosts* e coletam os dados sem importar qual configuração cada máquina possui.

Capítulo 5

PLATAFORMA AGLETS

5.1 FILOSOFIA

Desenvolvida originalmente por *Danny Lange* e *Mitsuro Oshima* sob o nome de *Aglets Workbench*, da IBM Japão, em 1996, a *Aglets* é uma plataforma para desenvolvimento de Agentes Móveis, totalmente desenvolvida em linguagem Java. Posteriormente, a IBM decidiu liberar a plataforma para a comunidade de código-aberto, é então quando a *SourceForge* aparece [7].

No começo, a *SourceForge* apenas desenvolvia a correção de erros na versão 1.x da IBM, mas com algumas mudanças a comunidade ficou responsável pela plataforma e lançou a versão 2.x. Essa versão trouxe uma grande melhoria em relação as séries da IBM, que foi o fato de ser compatível com versões mais atuais da *Java Development Kit*(JDK) [27].

A plataforma pode ser adquirida da forma gratuita, tanto no sua versão 1.x, desenvolvido pela IBM, quanto na versão 2.x da *SourceForge*, que responde pelo nome de *Aglet Software Development Kit*(ASDK2.0). O código fonte da ASDK2.0 também pode ser obtido através da Internet e está sob os termos de licenciamento da *IBM Public License* [23].

5.2 DEFINIÇÃO

Aglets é definida pelo *SourceForge*[7], como uma biblioteca Java para desenvolvimento de agentes móveis. Aglets é simples, pois segue o paradigma de desenvolvimento de *applets* onde a única diferença está na definição de alguns métodos para implementar os agentes. É seguro e flexível.

5.3 CARACTERÍSTICAS

Os autores descrevem a plataforma Aglets a partir do modelo conceitual de um sistemas de agentes Java, como mostra a figura 5.1 tirado de [11]. Neste modelo, a plataforma de agentes fornece uma API com as funcionalidades básicas para gerenciamento, migração e comunicação de agentes, na forma de pacotes Java. O TCP/IP é usado como o principal mecanismo de transporte. O servidor de agentes é *multithread* e executa sobre a JVM (*Java Virtual Machine*). Seguindo este modelo conceitual, está a descrição dos três níveis superiores: *Agents, Agent API e Agent Runtime Environment*, que podem ser vistos dentro da plataforma Aglets como: *Aglets, Aglets API e Ambiente de Execução Aglets*, respectivamente.

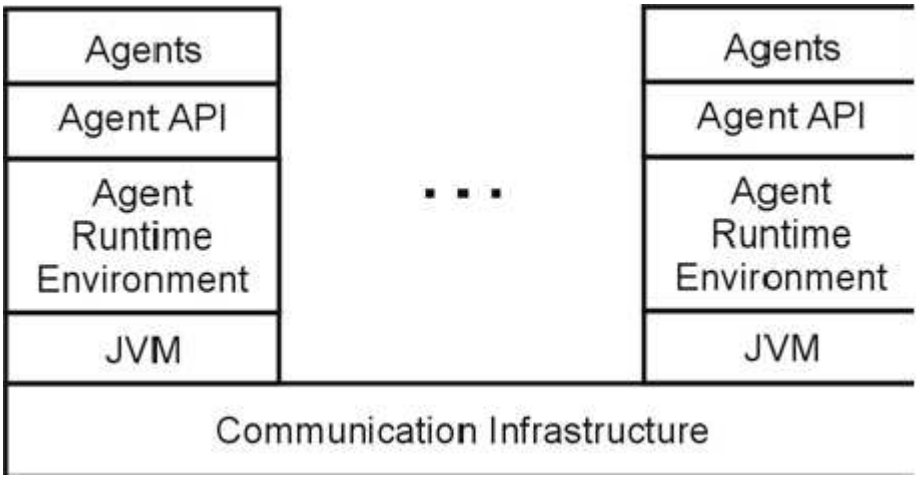

Figura 5.1: Modelo Conceitual de Agentes Móveis baseado em Java, Lobato [11]

Na camada *Aglets* estão os objetos Java, que são uma combinação de agentes de software com o modelo *applets* Java. Com o modelo de código móvel *applet*, os *aglets* são capazes de se mover através dos *hosts* de uma rede. Também se caracterizam por apresentar a própria *threads* de controle, serem dirigidos a eventos e se comunicarem através de passagem de mensagens.

A ASDK é a plataforma para programação de agentes *aglets*, que possui os métodos necessários à definição de modelos de eventos e de comunicação. Essa API proporciona uma programação *aglets* que não necessita de modificações na Máquina Virtual, permitindo total portabilidade dos códigos, sendo possível executá-los em qualquer máquina, independente de *hardware*, sistemas operacionais ou características particulares da API, bastando apenas suportar a sua interface de programação.

A interface de programação se caracteriza, também, por ter sido projetada para tirar proveito das características Java que suportam os requisitos de agents móveis, o que da simplicidade e flexibilidade a API.

O Ambiente de execução *aglet* é um servidor de agentes, responsável pela execução dos *aglets*. Este servidor, o *Aglet Server*, conta com um aplicativo gráfico que auxilia na configuração de privilégios de acesso e segurança no servidor e na manipulação dos agentes, denominada *Tahiti*.

Capítulo 6

ACD - AGENTE COLETOR DE DADOS

Este capítulo visa mostrar o agente desenvolvido para coleta de dados em uma rede de computadores. Nas seções que seguem será dada a definição e características do agente, padrões de projeto utilizados, estrutura e implementação do agente.

6.1 DEFINIÇÃO

Na abordagem dada anteriormente para um sistema multi-agente no gerenciamento de redes de computadores, vimos que a melhor configuração é de agentes trabalhando e cooperando entre si para alcançar um objetivo comum. Para a construção de um sistema nesses moldes, a implementação de um Agente Coletor de Dados é imprescindível, visto que esse agente é a base para o funcionamento do sistema, porque muitos dos outros agentes dependem das informações coletadas para poderem executar as suas tarefas.

O ACD (Agente Coletor de Dados) tem, por função, a movimentação pela rede, seguindo um itinerário (*hosts* pré-estabelecidos), sendo que, em cada *host*, ele deverá obter os dados relevantes sobre a máquina em questão, independente do Sistema Operacional e do *hardware* deste *host*.

De posse dos dados, o ACD (ou os ACDs), deverá retornar à máquina gerenciadora, a qual iniciou o mesmo, e apresentar os dados coletados dos *hosts* que foram efetivamente alcançados. Já na máquina gerenciadora, de posse dos dados coletados, o sistema grava os dados em arquivo para serem utilizados por outros sistemas como:

- Sistema Multi-Agente: outros agentes responsáveis por analisar os dados coletados e tomar as medidas cabíveis.

- Plataformas de Gerenciamento: os dados são disponibilizados para que uma plataforma de gerenciamento comum faça uso deles como informação sobre as máquinas, não necessitando consultar as máquinas através do modelo centralizado. Como exemplo, o ACD disponibiliza os dados obtidos na MIB de cada máquina do seu itinerário, para serem utilizados no Trauma Zero [26].

- Outros agentes: agentes com tarefas e características próprias podem fazer uso destes dados em suas execução. Por exemplo, o AGENTCHÊ, desenvolvido por Schopf [10], pode utilizar esses dados como base de conhecimento para os seus ensinamentos.

Os dados coletados pelo ACD são retirados da MIB de cada *host*. Ele pode ser configurado para percorrer qualquer sub-árvore da MIB, dependendo de o que se deseja obter.

A figura 6.1, mostra os dados coletados pelo ACD, através do agente móvel, sendo disponibilizado para serem utilizados pelo AGENTCHÊ [10] ou por uma plataforma de gerenciamento.

6.2 CARACTERÍSTICAS

Nessa seção será fornecida uma visão geral das características do ACD. Maiores detalhes sobre a implementação e funcionamento do Agente serão dadas nas seções posteriores.

O ACD apresenta algumas características básicas de agentes apresentadas anteriormente, com a mobilidade como ponto primordial do ACD.

Abordaremos o ACD de acordo com o seu objetivo e utilizando as definições de [3]:

Autonomia: funciona sem a intervenção direta de usuários, exercendo controle sobres suas próprias ações. Definir o ACD como um objeto com sua própria *thread* é uma forma prática de implementar autonomia em Java. A autonomia do ACD não é total, pois em algum dado momento ele deverá ser iniciado. A responsabilidade pela inicialização é de outro agente, chamado de Agente

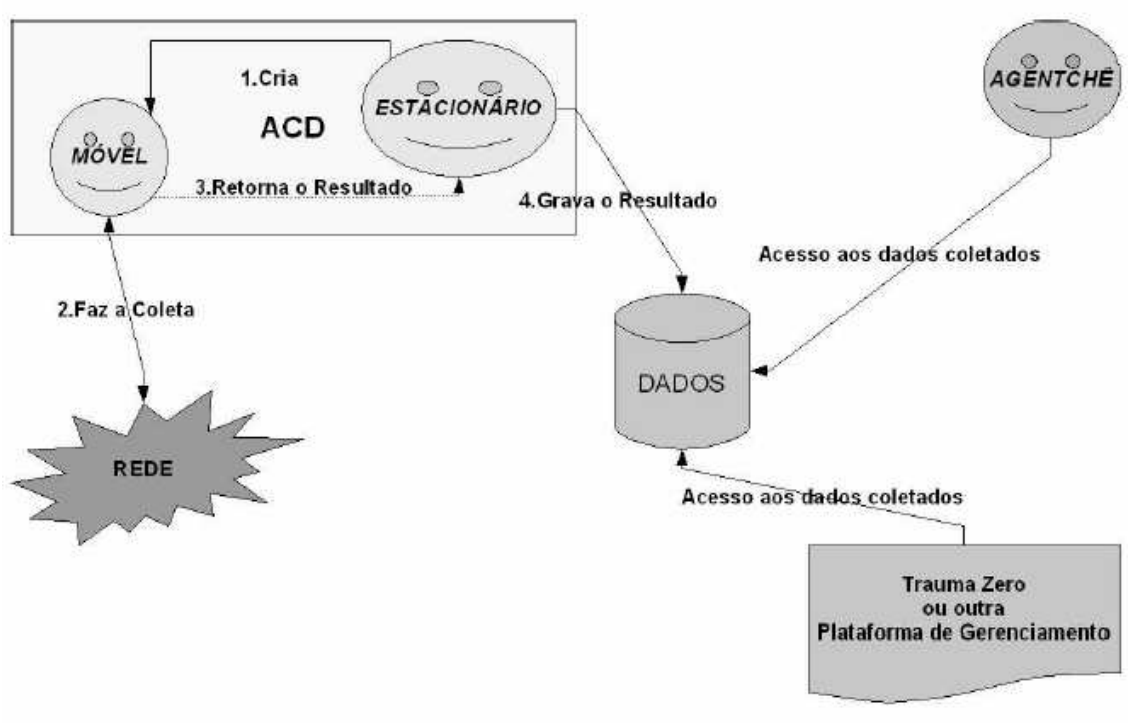

Figura 6.1: Disponibilidade dos Dados

Senior (que será discutido mais tarde). É neste momento de inicialização que o agente recebe o seu itinerário.

Habilidade Social: interação com usuários e outros agentes através das diretrizes de comunicação. Na comunicação entre os *aglets* as mensagens não são enviadas diretamente aos agentes, e sim, através de objetos *proxy*, que são vistos como uma interface de troca de mensagens entre os *aglets*. O ACD se comunica através da passagem de objetos e não da invocação de métodos.

Persistência: os agentes mantém consistentes seu estado durante sua existência. Quando um agente móvel chega a um novo *host* ele continua a sua execução a partir do ponto em que foi suspensa no endereço anterior. No caso do Agente Coletor, ele continua a sua execução no novo endereço, fazendo nova coleta e armazenando juntamente com os dados coletados anteriormente;

Mobilidade: habilidade do agente em mover-se através de uma rede, ocupando diferentes nodos e recursos ao longo do tempo. Quando o ACD se movimenta, ele chega ao seu destino com o mesmo estado em que estava na origem. Isso é

possível porque, antes de iniciar a sua transferência, o ACD serializa os dados atuais de sua execução.

O modelo de programação do ACD é baseado em eventos. A ASDK utiliza um modelo de delegação de eventos semelhantes ao implementado em Java, no qual objetos geram eventos quando certas ações são realizadas em um sistema.

O ACD, assim como um agente *aglet*, possui um estado, comportamento, identidade e localização. A localização, nada mais é, do que o contexto de execução de um ACD. Segundo [4], o contexto é um objeto estacionário que fornece meios para gerenciar *aglets* em seu ambiente de execução. Uma das principais funções do contexto é dar segurança ao *host* em que o ACD está sendo executado contra ataques mal intencionados que tentam afetar o funcionamento da máquina. Isto é possível porque o contexto de um agente atua no sentido de limitar o escopo de atuação dos *aglets* nos *hosts* da rede.

6.3 MODELAGEM E IMPLEMENTAÇÃO

Esta seção apresentará a modelagem, desenvolvimento e implementação do ACD. Para modelagem foi utilizada a linguagem UML - *Unified Modeling Language*, que é uma linguagem para especificação, documentação, construção e visualização de artefatos de sistemas de *softwares*[20].

6.3.1 Análise de Requisitos

6.3.1.1 Objetivo Geral

O objetivo geral do ACD é apresentar um sistema para coleta de informações de gerenciamento de redes de computadores utilizando a tecnologia de agentes móveis.

6.3.1.2 Objetivos Específicos

- Construir um agente capaz de se movimentar por uma rede de computadores, independente da configuração de *hardware* ou *software* das máquinas.
- Permitir que o agente coletor esteja apto a fornecer as informações para outras ferramentas ou agentes.
- A utilização de agentes móveis pra acessar dados relevantes de gerenciamento de uma rede.

6.3.1.3 Clientes Potenciais

- Outros desenvolvedores que desejam utilizar o Agente Coletor para compor um sistema Multi-Agente para Gerenciamento de Redes.
- Ferramentas de Gerenciamento tradicionais que utilizariam o ACD para fazer a obtenção dos dados.

6.3.1.4 Funcionalidade do Sistema

O Agente Coletor deve apresentar algumas funções, sendo algumas essenciais e outras menos importantes. Os requisitos funcionais do ACD são:

- Permitir a atualização de endereços na lista de *hosts*, como inserção, exclusão e alteração.
- Definir os parâmetros de iteração com o Agente SNMP.
- Compor um arquivo com os resultados obtidos.

6.3.2 Casos de Uso

Um caso de uso é uma descrição narrativa de um processo no domínio do sistema, ou mais especificamente, é um documento narrativo que descreve a seqüência de eventos utilizada por um ator do sistema a fim de completar uma determinada tarefa [14].

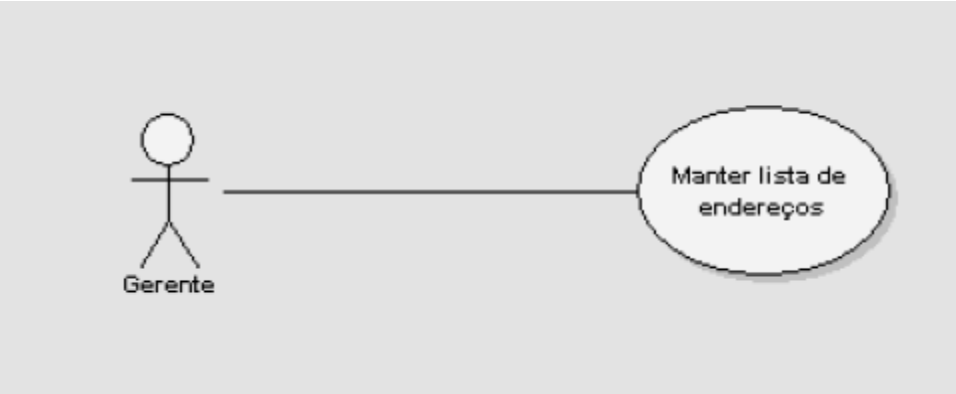

Figura 6.2: Diagrama de Caso de Uso: Manter lista de endereços

Na seção a seguir são apresentados os casos de uso do Agente Coletor de Dados.

6.3.2.1 Manter lista de endereços

A lista de endereços compõe o itinerário do agente. São todos os *hosts* que o ACD visita durante a sua execução. Neste caso de uso, as funções referentes a manutenção da lista, como: inserção de novos endereços, exclusão e atualização são concebidas. A figura 6.2 mostra o diagrama para este caso de uso.

Caso de Uso:	Manter Lista de Endereços
Atores:	Gerente
Finalidade:	Atualizar a lista de endereços usada como itinerário do agente
Descrição:	1. Este caso de uso começa quando o gerente solicita a lista de endereços. 2. O sistema apresenta a lista com os endereços iniciais. 3. O usuário realiza as atualizações desejadas, que podem ser: alteração de um endereço, inserção de novos endereços ou remoção. 4. O sistema armazena as atualizações realizadas.

Figura 6.3: Diagrama de Caso de Uso: Fazer coleta dos dados

6.3.2.2 Fazer a coleta dos dados

No caso de uso: "fazer a coleta dos dados", o agente percorre o seu itinerário, coletando os dados em cada *host*. São requeridas as funções de: definir os parâmetros de iteração com o agente SNMP e compor um arquivo com os resultados obtidos.

Caso de Uso:	Fazer Coleta dos Dados
Atores:	Gerente e *hosts*
Finalidade:	Mandar o Agente móvel para rede com o intuito de fazer a coleta doa dados.
Descrição:	1. Este caso de uso começa quando o gerente deseja realizar uma coleta nos *hosts* do itinerário. 2. O gerente informa os parâmetros para comunicação com o agente SNMP e pesquisa nas subárvores da MIB. 3. O gerente inicia a coleta. 4. O sistema envia o AM para a rede. 5. O agente móvel se comunica com o agente SNMP do *host*. 6. O agente recebe os dados requisitados. 7. O agente armazena os resultados obtidos junto com os outros obtidos nos *hosts* anteriores. 8. Se ainda existirem *hosts* a serem visitados, o agente repete os passos anteriores de comunicação e coleta. 9. O agente retorna para a máquina inicial com a coleta obtida. 10. O agente informa que terminou a coleta. 11. O sistema grava os resultados obtidos em um arquivo e informa ao usuário.
Seqüência Alternativa:	No caso de algum *host* inacessível, o agente se transfere para o próximo, se houver.

Se algum endereço do itinerário estiver inacessível, o agente precisará tomar uma decisão alternativa, passando para o próximo endereço da lista. Para saber se um *host* está acessível, ele testa o mesmo antes de iniciar a transferência para ele. A figura 6.3 mostra o diagrama para este caso de uso.

6.3.3 Diagrama de Casos de Uso

A finalidade do diagrama é apresentar um diagrama de contexto, através do qual é possível detectar rapidamente quais são os atores de um sistema e as principais

maneiras, segundo as quais eles utilizam [14]. A figura 6.4 apresenta o diagrama de casos de uso do Agente Coletor de Dados.

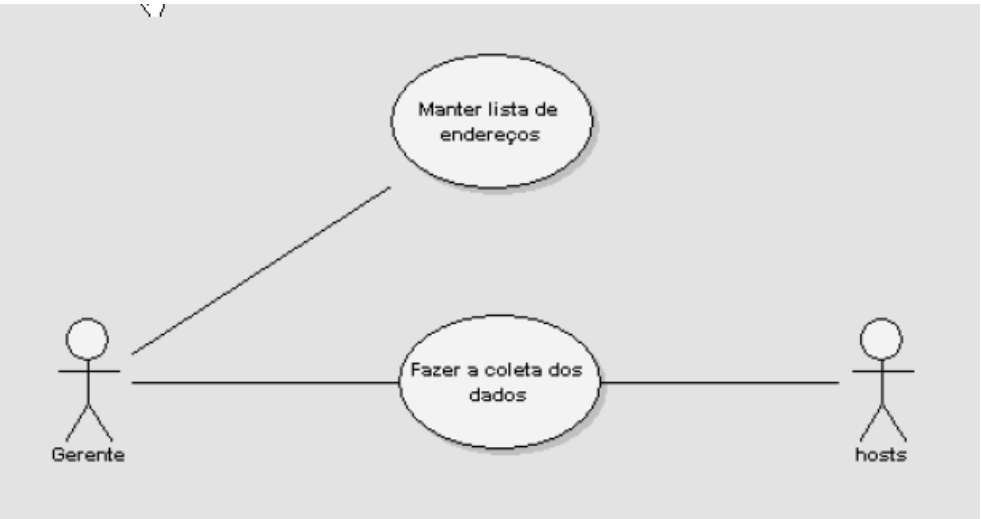

Figura 6.4: Diagrama de casos de uso do ACD

6.3.4 Padrões de Projeto Utilizadas

A plataforma *Aglets* reconhece um bom número de padrões de projetos. Estes padrões, implementados em Java, foram adicionados logo no inicio do desenvolvimento, na primeira versão da plataforma *Aglets Workbench*.

Nesta seção será apresentado com mais ênfase os padrões *Master-Slave* e *Itinerary*, que foram utilizados na confecção do ACD.

6.3.4.1 Padrão *Master-Slave*

O padrão *Master-Slave* é fundamental no projeto de agentes móveis. Este padrão define que um agente *master*, geralmente estacionário, cria e define tarefas à um agente *slave*(móvel). Logo em seguida, o agente *slave* é enviado ao seu destino, onde ele executará o que lhe foi determinado. Após essa execução, ele retorna com os resultados ao *host* original.

A idéia principal é utilizar as classes abstratas *Master* e *Slave* para especificar os papéis dos agentes no cumprimento de uma tarefa: retornar um *slave*, ou enviá-lo a um outro *host*, iniciar a execução de uma tarefa, tratar exceções que ocorrem durante a execução, etc [11].

Na implementação do ACD, dois agentes foram criados constituindo o padrão *Master-Slave*: o agente *Senior*, como agente *master*; e o agente *Coletor*, como agente *slave*. Cada agente tem tarefas distintas: o agente *Senior* é um agente estacionário, responsável pela criação do agente *Coletor* e apresentação dos dados coletados pelo mesmo; já o agente *Coletor* tem a capacidade de se mover até os *hosts* e executar as suas tarefas, retornando com os dados coletados para o agente *Senior*.

6.3.4.2 Padrão *Itinerary*

Este padrão é responsável pelo tratamento das questões voltadas ao itinerário de um agente. As suas principais funções são: manter uma lista dos destinos a serem visitados; definir um esquema de roteamento através dos *hosts* da rede; tratar exceções como o que fazer se um determinado destino não é encontrado ou fazer o retorno do agente a destinos anteriormente visitados, e indicar o próximo *host* da rede para onde o agente deve ser transferido [15].

A construção do Agente com base no padrão *Itinerary*, coloca a responsabilidade de navegação do agente para um objeto à parte, instânciado a partir de uma classe projetada para realizar essa tarefa. O objeto Agente cria o objeto itinerário e o inicia com dois argumentos: uma lista de destinos (*hosts*) e uma referência que mantém a ligação entre os dois objetos. O itinerário é mantido sempre junto do agente, independente da sua movimentação. O método *go* do objeto itinerário faz com que o agente se mova para o próximo destino ou retorne ao *host* original.

6.3.4.3 Junção dos Padrões

O ACD é implementado a partir de uma combinação dos dois padrões de projetos citados anteriormente. Com a combinação dos padrões *Master-Slave* e *Itinerary*, é possível implementar o agente estacionário *Senior*, responsável pelo gerenciamento do ACD, e o agente *Coletor* móvel, que segue um itinerário para percorrer os *hosts* da rede e executar as tarefas desejadas.

Contudo, para haver a combinação entre os padrões, o agente *Coletor* do padrão *Master-Slave*, teve que ser alterado para que pudesse suportar múltiplos *hosts*, afim de que pudesse executar a sua coleta a cada novo destino do itinerário. As modificações feitas foram: associação do *Coletor* a um objeto que controla o seu itinerário e modificação do mesmo para que ele possa seguir ao próximo destino, após o seu objetivo completado no *host* corrente.

As classes utilizadas para implementação do ACD, utilizando os padrões *Master-Slave* e *Itinerary*, estão descritas no pacote *com.ibm.agletx.patterns* [7].

Com a utilização dessas classes é possível associar ao agente móvel uma lista de destinos, *hosts* que devem ser percorridos para execução das tarefas pré-destinadas, indicar o próximo *host* e fazer o retorno do agente para a origem.

6.3.5 Diagrama de Classes

Um diagram de classes ilustra as especificações para as classes e interfaces de uma aplicação. As informações típicas incluem: (1) classes, associações e atributos; (2) interfaces, com suas operações e constantes; (3) métodos; (4) informações do tipo de atributos; (5) navegabilidade; e (6) dependências [14].

A figura 6.5 mostra o diagrama de classes do ACD.

6.3.5.1 Estrutura

O coletor é um agente reflexivo, não tem passado nem futuro, com a única função de coletar periodicamente os dados desejados de uma estação da rede, formatá-los e enviá-los à aplicação gerente que analisará os dados recebidos do coletor [1].

A estrutura do ACD é definida da seguinte forma:

- *Aglet*, superclasse do ASDK que deve ser implementada, sobrescrevendo sues métodos, para poder criar um agente Aglet.
- *hostsStore*, estrutura responsável por armazenar os *hosts* destinos do itinerário. Ela armazena o nome e o endereço do *host*.
- *hostsList*, classe responsável por ler um arquivo texto, contendo os endereços de *hosts* na rede, e inicializar um vetor *hostsStore* com esses endereços.
- *Senior*, agente estacionário que cria o agente móvel, dando-lhe o seu itinerário, criado pela (*hostsList*). Os dados coletados pelo agente móvel são entregues ao agente *Senior*, que grava-os em um arquivo texto para futura análise.
- *Coletor*, agente móvel, que percorre a lista de itinerários, coletando os dados localmente em cada *host* seguindo regras específicas para comunicação com o agente SNMP. Essa classe implementa o Padrão do Projeto *SearcherSlave* definido no ASDK. Esse padrão é responsável pela confecção do itinerário e movimentação do Coletor pela rede.

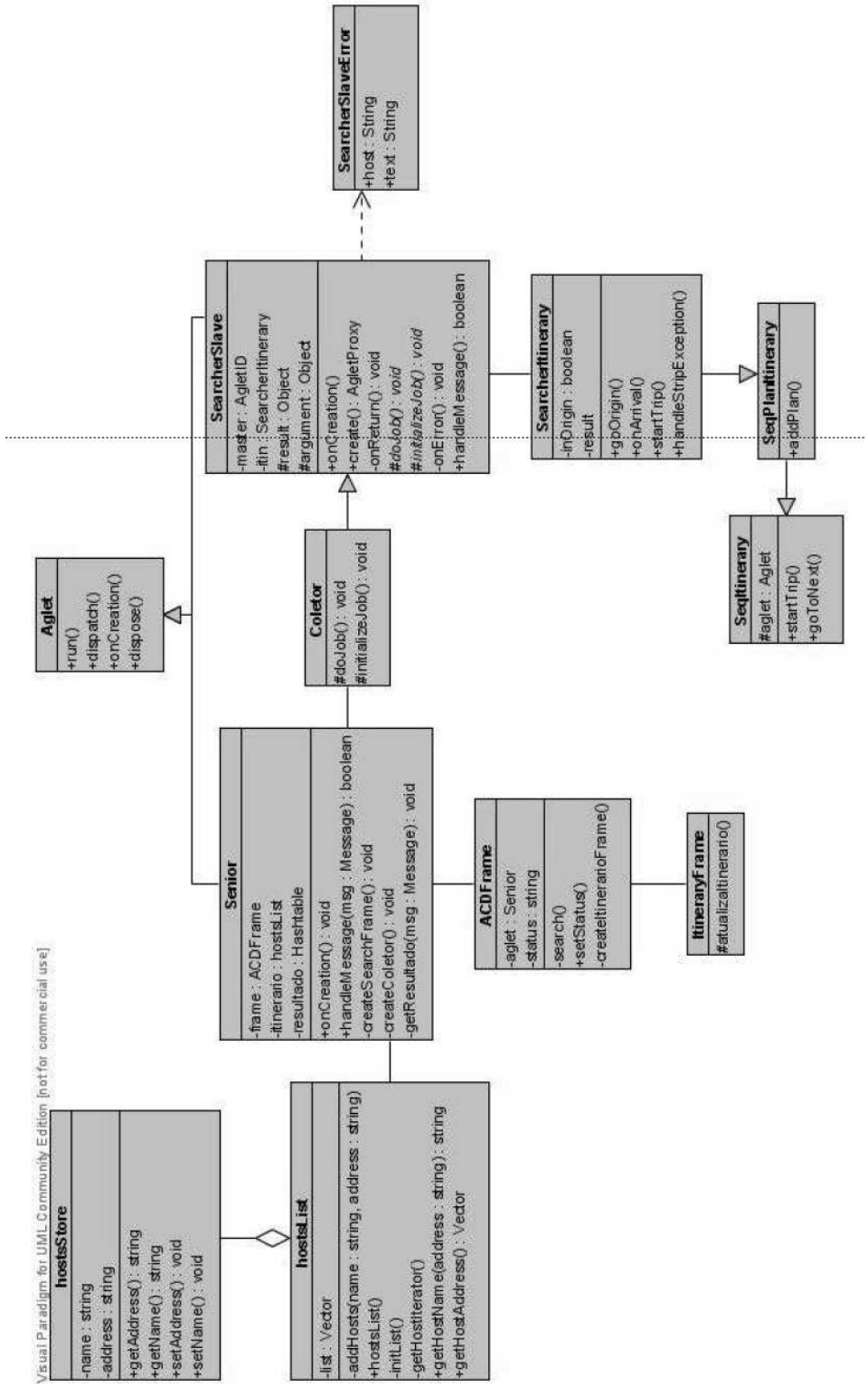

Figura 6.5: Diagrama de Classes do ACD

- classe *ACDframe*, define a interface gráfica do agente *Senior*, com a possibilidade de iniciar a coleta, visualizar o itinerário e visualizar os resultados.
- classe *ItineraryFrame*, define a interface gráfica para mostrar o itinerário, possibilitando ao usuário atualizar os endereços.

6.3.5.2 Definição do Itinerário

A função desta parte da implementação é definir um itinerário para o agente móvel a partir de informações sobre *hosts* que compõe a rede. O *Coletor* se movimenta entre os *hosts* especificados no itinerário montado a partir da leitura de um arquivo que mantém os endereços da rede e os nomes dos *hosts*.

A *hostsList* é a classe responsável pela leitura do arquivo e inicialização do vetor de destinos. O vetor é uma estrutura de dados chamada *hostsStore*, quem armazena o endereço e o nome do *host*.

A classe *hostsList* possui três métodos públicos: o método *getHostName*, retorna o nome de um *host* a partir de um endereço dado *address*, o método *getHostNames* retorna um vetor com os nomes de todos os *hosts* que compõe o itinerário. Já o método *getHostAddress* retorna um vetor com todos os endereços do itinerário.

6.3.5.3 Agente Estacionário

O agente estacionário, chamado de *Senior*, é o responsável pela criação do agente móvel, passando-lhe o seu itinerário, e pelo recebimento dos dados coletados.

O itinerário é definido a partir da criação de um instância da classe *hostsList*. Já a criação do agente móvel é feita chamando um método da classe *SearcherSlave*. Essa classe é originária da junção dos padrões de projeto *Master-Slave* e *Itinerary*. Os dados recebidos, após a volta do agente móvel, são gravados em outro arquivo .txt para serem utilizados posteriormente.

O *Senior* é definido como uma subclasse da classe abstrata *Aglet*. Essa declaração é importante quando se está definindo um agente. A partir dela, o agente herda atributos e métodos desta classe abstrata. Os métodos herdados pelo agente *Senior* foram: *onCreation* e *handleMessage*.

O atributo *itinerario* irá armazenar a lista de endereços a serem percorridos; e ao atributo *resultado* que receberá os resultados obtidos pelo *Coletor*.

O método *onCreation* foi sobrescrito para fazer a chamada do método *createSearchFrame*, responsável por criar a interface gráfica do agente *Senior*. Esse

método, *createSearchFrame*, cria uma instância da classe *ACDFrame*, que implementa a interface gráfica.

O método *handleMessage*, herdado da superclasse, faz o tratamento das mensagens recebidas pelo *Senior*. Nesta implementação, três tipos de mensagens estão sendo tratadas:

- *dialog*, que manda criar a interface gráfica;
- *createColetor*, que chama o método *createColetor* para criar o agente *Coletor* e iniciar a coleta;
- *result* que avisa ao *Senior* que o resultado chegou e então o método *getResultado* é iniciado para tratar a mensagem.

O método *createColetor* é o responsável por iniciar o agente de coleta e passar-lhe o itinerário. O agente Coletor é criado, chamando o método *SearcherSlave.create* definido nos padrões de projetos. O método *create* recebe seis argumentos, definidos assim na classe *SearcherSlave*:

1. *URL url*: a url da classe aglet, no caso o agente *Coletor*;
2. *String name*: o nome do agente que será criado;
3. *AgletContext context*: o *aglet* contexto em que o agente *Coletor* está sendo criado;
4. *Aglet master*: o agente *master*, ou criador do agente móvel, no caso o agente *Senior*;
5. *Vector itinerary*: o vetor de endereços de destinos;
6. *Object Argument*: Um argumento adicional, que foi lista dos nomes de todos os *hosts*.

O método *getResultado*, que é chamado quando o agente *Coletor* retorna a origem com os dados, retira o resultado, que foi passado como argumento da mensagem recebida pelo *Senior*, sinaliza a interface gráfica que o agente terminou a coleta e grava o resultado em um arquivo, através do método *gravaResultado*.

Os detalhes de movimentação do *Coletor* são transparentes para o desenvolvedor, já que estão encapsulados nas classes que implementam o padrão *Itinerary*.

6.3.5.4 Agente Móvel

O agente móvel, chamado de *Coletor*, que foi criado pelo agente *Senior*, tem a função de se comunicar com o agente SNMP em cada *host* do seu itinerário, coletando e armazenando as informações.

O agente estende a classe *SearcherSlave*, que faz parte das classes que implementam os padrões de projeto *Master-Slave* e *Itinerary*. *RESULT* é um atributo da superclasse *SearcherSlave*.

O método *doJob*foi sobrescrito para efetuar a coleta dos dados em cada host que o agente passa. Para definir a comunicação com o agente SNMP, foi utilizada uma API Java chamada *AdventNet SNMP API* [25].

É criado uma conexão snmp, através do método *SnmpTarget* e em seguida são configurados alguns parâmetros, como: Versão do SNMP a ser usado, *host* que será pesquisado, nome do *community* e o *Objetc ID* onde será feito a coleta.

É percorrido a sub-árvore dada pelo *OID*, sempre testando o OID que está sendo pesquisado está dentro da sub-árvore. Para efetuar essa pesquisa são feitos sucessivos *getnext* e os resultados vão sendo armazenados.

Depois, é verificado se foram obtidos algum resultados, em caso negativo é sinal que estourou o *timeout*. Por fim, a conexão com o SNMP é finalizada, os resultados são armazenados em RESULT e o agente parte para o seu próximo destino.

6.3.6 Diagrama de Seqüência

Um diagrama de seqüência da fase de projeto de uma aplicação ilustra a interação através de mensagens entre instâncias presentes no diagrama de classes, isto é, os diagrams de seqüência da fase de projeto mostram os detalhes do comportamento do sistema associado às operações invocadas pelos usuários [14].

As figuras 6.6 e 6.7, mostram os diagramas de seqüência do ACD, desde o momento em que o sistema é iniciado, passando pela movimentação do agente pela rede retornando com os resultados.

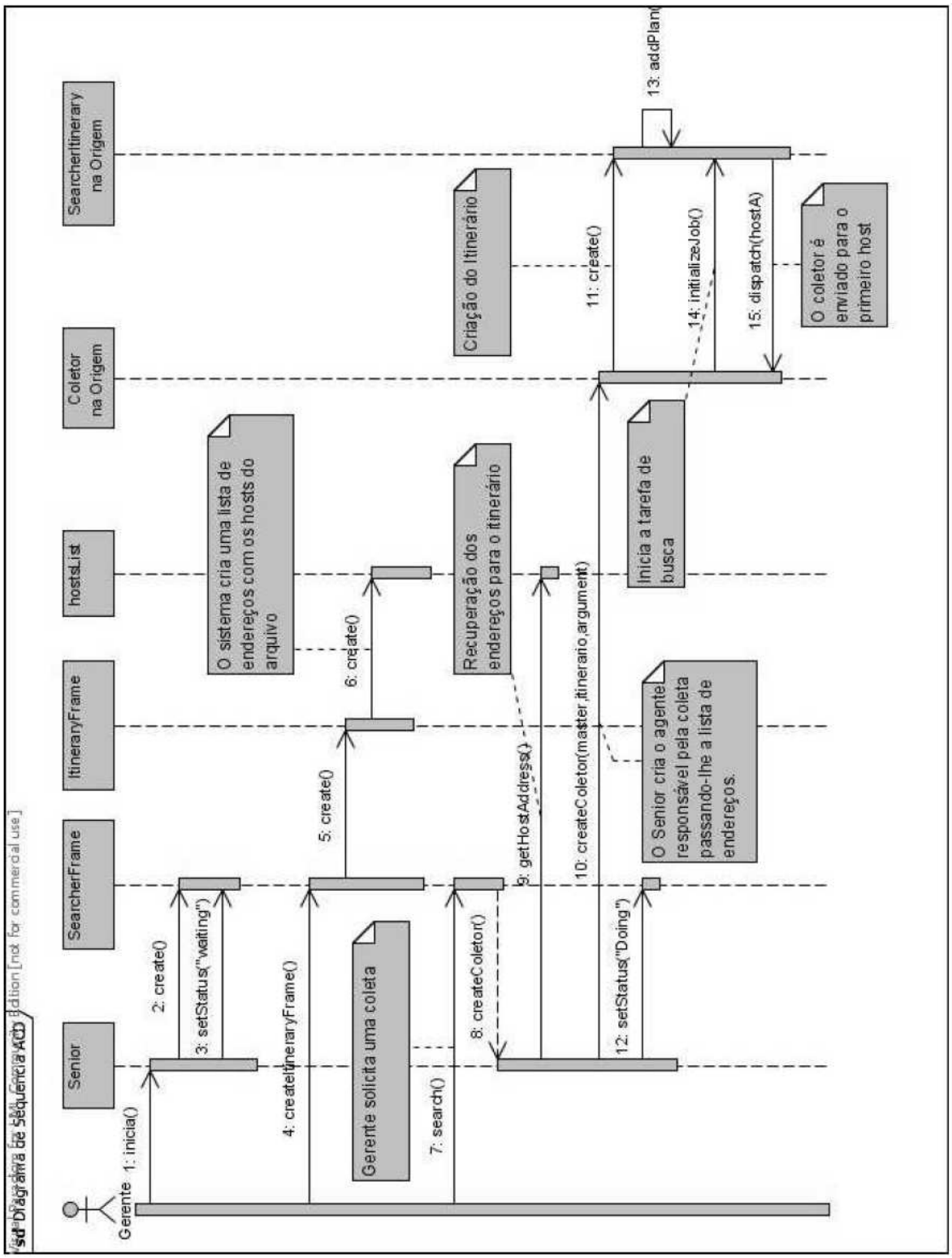

Figura 6.6: Diagrama de Seqüência do ACD - Inicio da Coleta

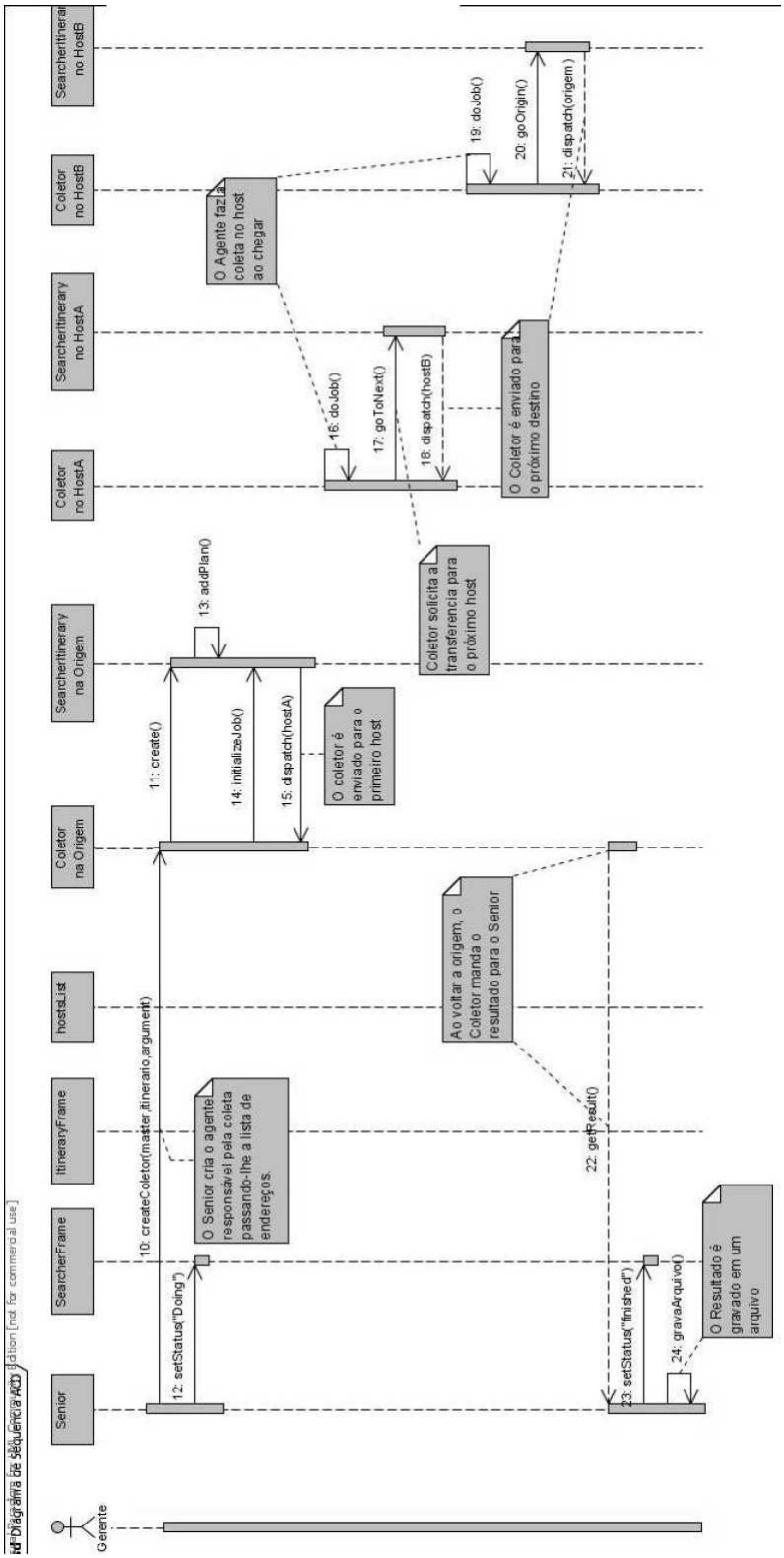

Figura 6.7: Diagrama de Seqüência do ACD - Movimentação pela rede e retorno do resultado

Capítulo 7

VALIDAÇÃO DO ACD

Para validação do Agente Coletor, precisava-se que ele se transportasse efetivamente entre *hosts* de uma rede. Durante a implementação do ACD e os primeiros testes foram feitos em uma máquina apenas, onde através da plataforma *Aglets*, com a utilização da interface gráfica *tahiti*, simulou-se três computadores, utilizando-se portas diferentes de comunicação. A figura 7.1 ilustra essa simulação.

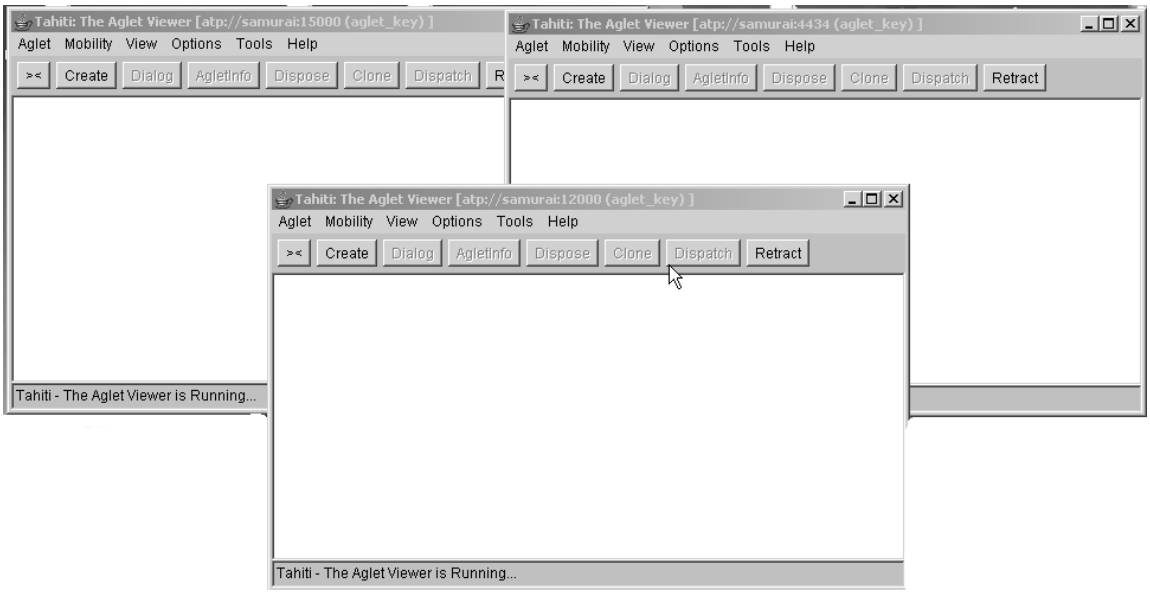

Figura 7.1: Simulação de três computadores, usando *Tahiti*

7.1 INSTALAÇÃO DO ASDK 2.02

Como a versão inicial da IBM, a *Aglets Workbench*, é apenas compatível com a versão 1.2 do JDK, optamos pela versão da *SourceForge*, o ASDK 2.02.

O *download* do JDK 1.5.0 foi feito do site da *Sun MicroSystems*[27]. Já o download da versão 2.02 do ASDK foi efetuado no site do *Aglet*, [7]. Ambos os downloads foram feitos de forma gratuita.

1º Passo

Primeiramente foi instalado o JDK, e em seguida configurado as variáveis de ambiente.

2º Passo

Descompactado o ASDK, configurado as variáveis de ambiente e executado duas vezes o arquivo ant.bat dentro da pasta bin.

Para visualização da movimentação do agente, foi executado a interface gráfica *Tahiti*, através do comando *agletsd -port 12000*, como mostra a figura 7.2.

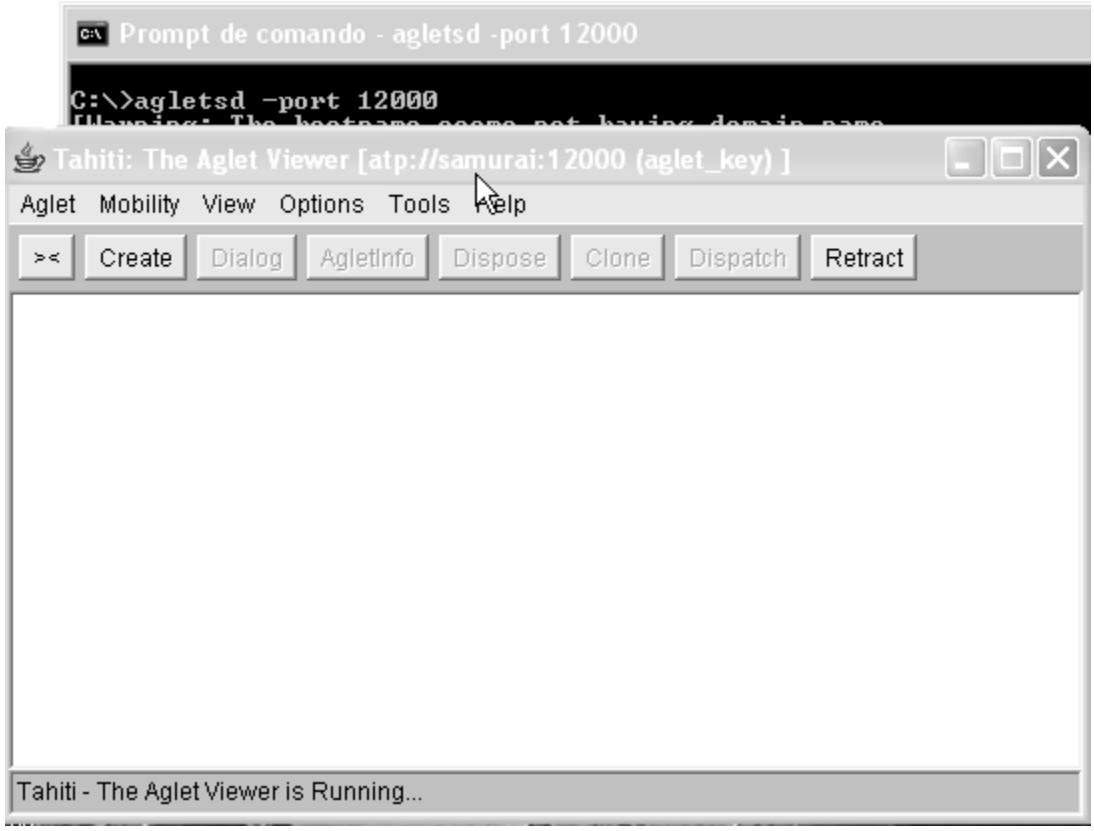

Figura 7.2: Comando e a interface *Tahiti*

7.2 EXECUÇÃO DO ACD

Para efetuar testes mais reais, o agente foi executado em três computadores pertencentes ao Centro de Processamento de Dados da Universidade Federal de Santa Maria. Os três computadores estavam rodando versões do Linux, a tabela 7.1 mostra a configuração de cada máquina.

Tabela 7.1: Configuração dos computadores da rede

Nome do Computador	S.O.	JDK	ASDK
MUSEU	Debian	JDK 1.5.0 update 5	ASDK 2.02
MARCOS	Gentoo	JDK 1.5.0 update 6	ASDK 2.02
DEWES	Gentoo	JDK 1.5.0 update 5	ASDK 2.02

O computador intitulado MUSEU, foi utilizado como estação de gerenciamento, de onde os agentes eram iniciados e para onde retornavam com os dados da coleta. Os outros computadores se comportaram como nós gerenciados. A tabela 7.2, mostra o comportamento de cada computador e os seus endereços.

Tabela 7.2: Papel de cada computador e seu endereço na rede do CPD

Nome do Computador	Endereço	Papel Desempenhado
MUSEU	atp://museu:4434/	Estação Gerenciadora
MARCOS	atp://marcos:12000/	Nó Gerenciado
DEWES	atp://dewes:13000/	Nó Gerenciado

Diferente das execuções realizadas durante a implementação, onde foi simulado três *hosts* em uma máquina. Aqui foram usadas máquinas diferentes para execução do agente. Foram utilizados três computadores diferentes que estavam rodando a plataforma ASDK para execução do agente *Coletor*. Foi usada a interface gráfica *tahiti* para registrar a passagem do agente móvel pelas máquinas.

O agente *Senior* foi iniciado normalmente e o mesmo criou, instantâneamente, o seu menu gráfico, como mostra a figura 7.3. O agente *Coletor*, assim que disparado, percorreu todos os endereços de destinos, executando suas tarefas, e retornou com êxito a sua origem.

Figura 7.3: Criação do agente *Senior* na máquina MUSEU, e a sua janela gráfica

O agente *Coletor*, no momento que chega no primeiro endereço, mostra sua mensagem de chegada na interface gráfica, como está na figura 7.4.

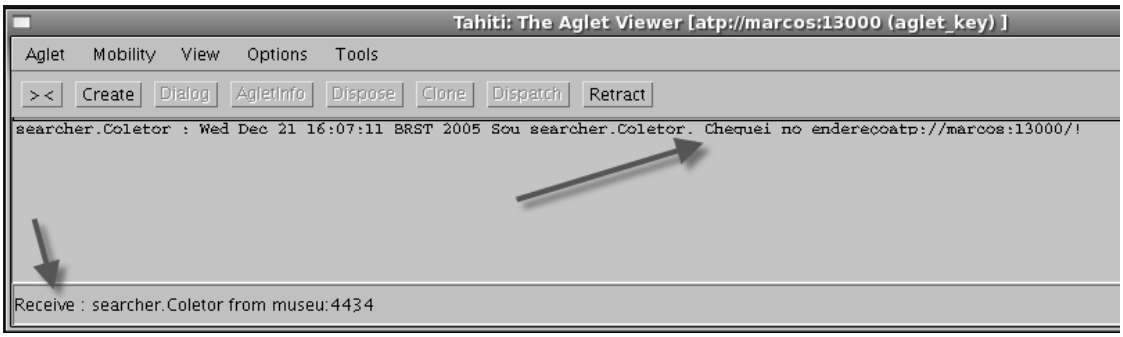

Figura 7.4: O agente chega no primeiro endereço: atp://marcos:13000/

Após sua execução, o agente imprime outra mensagem e se despede anunciando sua saída, a figura 7.5 registra esse momento. O agente então, se dirige ao próximo endereço do seu itinerário e executa novamente os mesmos procedimentos, como visto na figura 7.6.

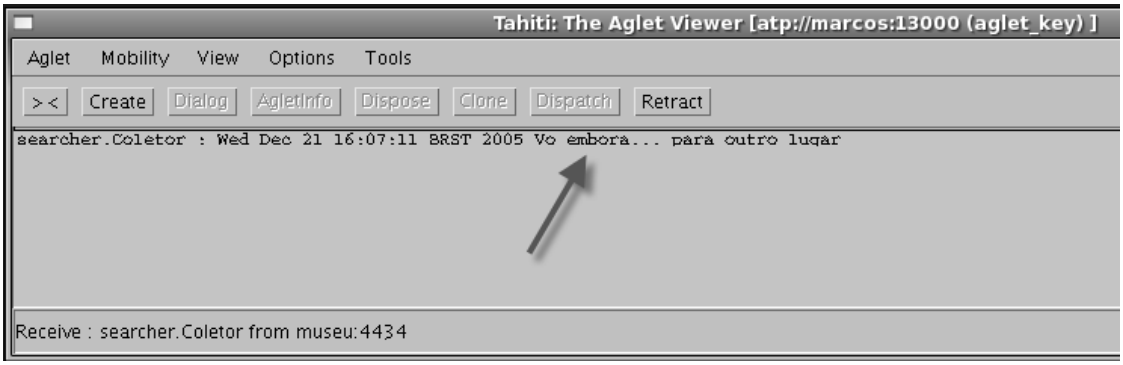

Figura 7.5: Agente se despede após executar suas tarefas.

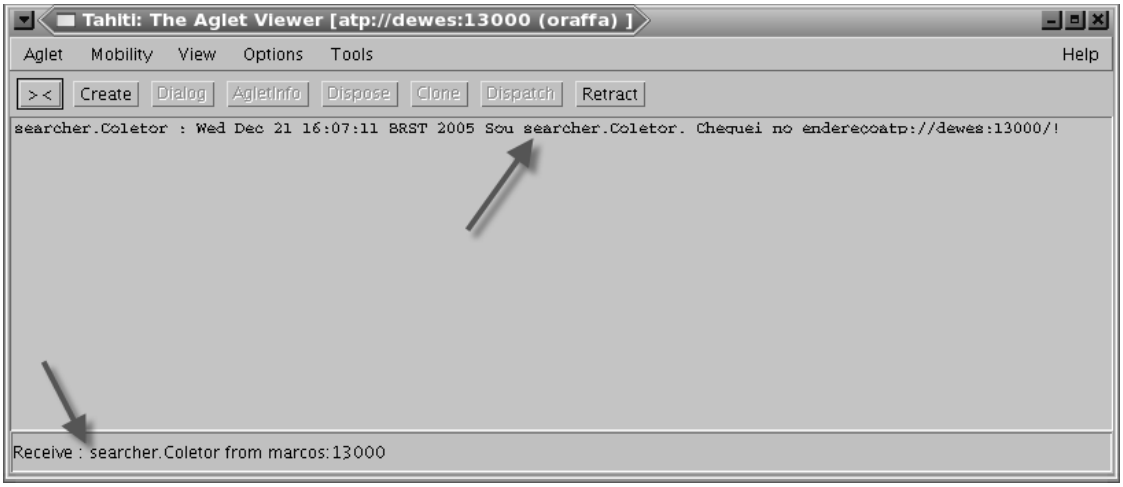

Figura 7.6: O agente *Coletor* chega no segundo endereço: atp://dewes:13000/

Executado suas tarefas, o agente se despede novamente, figura 7.7 e retorna a origem. A figura 7.8 mostra o registro, no rodapé da janela gráfica, do endereço para onde o agente se moveu, a sua origem.

Como na execução simulada, o agente *Senior* recebe os dados do agente *Coletor*, o mesmo é destruído e o agente *Senior* registra o recebimento dos resultados com uma mensagem na interface, figura 7.9. A destruição do agente *Coletor* é feita chamando-se o método *onDispose*, pertencente a definição de *Aglets* da plataforma ASDK. Como essa destruição se procede é um detalhe implementado pela plataforma.

Para proceder essa execução, nenhuma alteração foi necessária no código fonte do agente. Apenas a alteração do arquivo contendo os endereços de destino foi

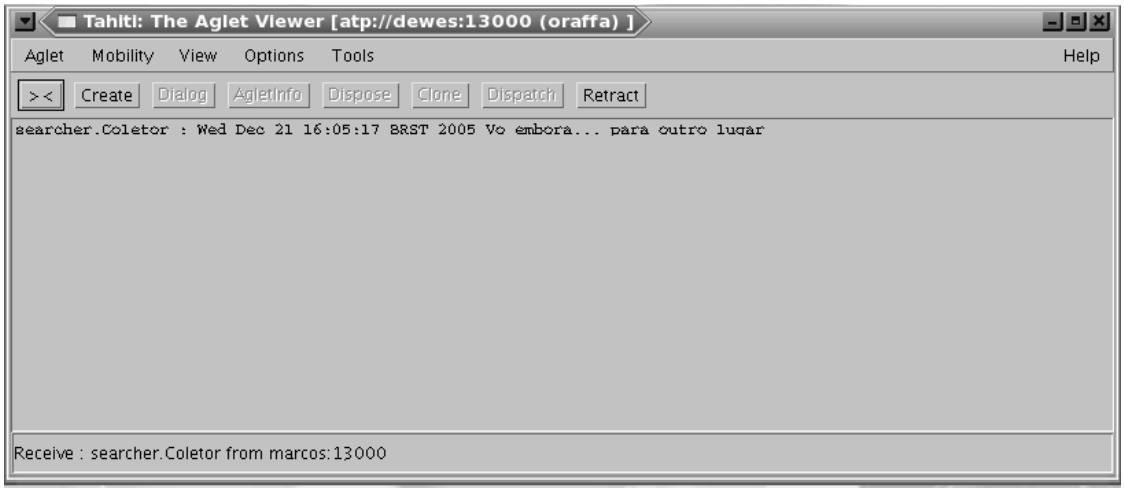

Figura 7.7: Agente se despede após executar suas tarefas.

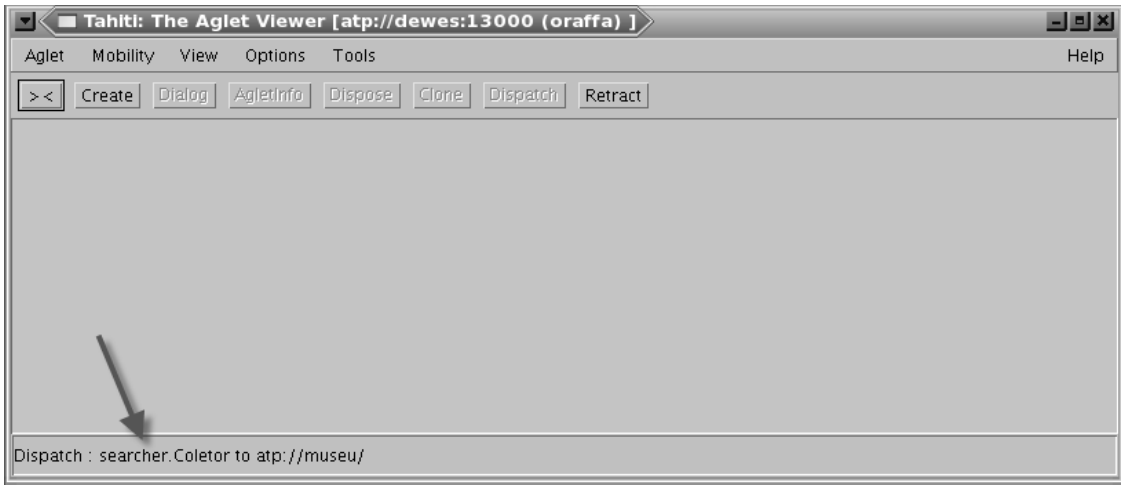

Figura 7.8: Registro do endereço para qual o agente se moveu.

alterado.

Com essa execução foi possível constatar que o Agente Senior criou o Agente Coletor. Este, por sua vez, percorreu todo o itinerário estabelecido retornando ao ponto de saída. A mobilidade, sua principal característica e o objetivo mais importante, foram alcançados.

7.2.1 Dados Recebidos

O agente *Coletor* retornou à sua origem com os dados e o agente *Senior* conseguiu gravá-los em um arquivo.

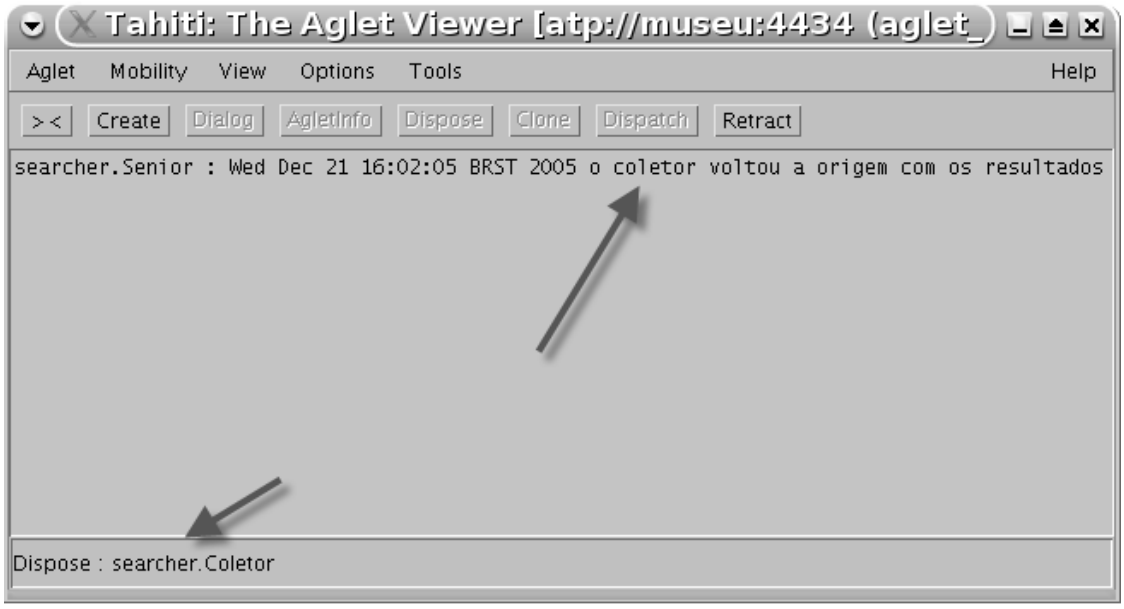

Figura 7.9: Retorno do agente ao ponto de origem.

Na figura 7.10, mostra o exemplo de uma das coletas efetuadas. No caso, foi passado o *Object ID* ".1.3.6.1.2.1.1"e os dados coletados pelo ACD nas máquinas do CPD foram as ilustradas pela figura.

```
OID --> .1.3.6.1.2.1.1

sysDescr.0:-->Hardware: x86 Family 15 Model 4 Stepping 8 AT/AT COMPATIBLE
sysObjectID.0:-->.iso.org.dod.internet.private.enterprises.311.1.1.3.1.1
sysUpTime.0:-->0 hours, 6 minutes, 32 seconds.
sysContact.0:-->dewes@cpd.ufsm.br
sysName.0:-->DEWES
sysLocation.0:-->CPD
sysServices.0:-->76


sysDescr.0:-->Hardware: x86 Family 15 Model 4 Stepping 8 AT/AT COMPATIBLE
sysObjectID.0:-->.iso.org.dod.internet.private.enterprises.311.1.1.3.1.1
sysUpTime.0:-->0 hours, 6 minutes, 32 seconds.
sysContact.0:-->marcos@cpd.ufsm.br
sysName.0:-->MARCOS
sysLocation.0:-->CPD
sysServices.0:-->76
```

Figura 7.10: Dados coletados pelo *Coletor.* no OID .1.3.6.1.2.1.1

A figura 7.11 mostra a coleta no OID .1.3.6.1.2.1.2, feitas nas máquinas do CPD.

```
Sent get request to localhost : 161
ifNumber.0:-->2
ifIndex.1:-->1
ifIndex.65539:-->65539
ifDescr.1:-->MS TCP Loopback interface□
ifDescr.65539:-->NVIDIA nForce Networking Controller - Miniporta do agendador de pacotes□
ifType.1:-->softwareLoopback(24)
ifType.65539:-->ethernet-csmacd(6)
ifMtu.1:-->1520
ifMtu.65539:-->1500
ifSpeed.1:-->10000000
ifSpeed.65539:-->100000000
ifPhysAddress.1:-->
ifPhysAddress.65539:-->00 11 d8 98 04 97
ifAdminStatus.1:-->up(1)
ifAdminStatus.65539:-->up(1)
ifOperStatus.1:-->up(1)
ifOperStatus.65539:-->up(1)
ifLastChange.1:-->0 hours, 0 minutes, 0 seconds.
ifLastChange.65539:-->0 hours, 12 minutes, 52 seconds.
ifInOctets.1:-->823331
ifInOctets.65539:-->16317289
ifInUcastPkts.1:-->20829
ifInUcastPkts.65539:-->19598
ifInNUcastPkts.1:-->0
ifInNUcastPkts.65539:-->2585
ifInDiscards.1:-->0
ifInDiscards.65539:-->0
ifInErrors.1:-->0
ifInErrors.65539:-->0
ifInUnknownProtos.1:-->0
ifInUnknownProtos.65539:-->0
ifOutOctets.1:-->823331
ifOutOctets.65539:-->3122736
ifOutUcastPkts.1:-->20829
ifOutUcastPkts.65539:-->16348
ifOutNUcastPkts.1:-->0
ifOutNUcastPkts.65539:-->2501
ifOutDiscards.1:-->0
ifOutDiscards.65539:-->0
ifOutErrors.1:-->0
ifOutErrors.65539:-->0
ifOutQLen.1:-->0
ifOutQLen.65539:-->0
ifSpecific.1:-->.0.0
ifSpecific.65539:-->.0.0
```

Figura 7.11: Dados coletados pelo *Coletor.* no OID .1.3.6.1.2.1.2

Capítulo 8

CONCLUSÃO

De acordo com os resultados obtidos, o ACD atingiu seus objetivos, pela validação positiva em termos de movimentação pela rede, atingindo todos os *hosts* definidos em seu itinerário e retornando a sua origem; execução em máquinas com Sistemas Operacionais e *hardwares* diferentes, sem que ajustes na implementação tivessem que ser feitos; execução local em cada *host* atingido; continuidade da execução, mesmo que algum ponto não fosse alcançado; obtenção dos dados requisitados, trazendo-os de volta à origem e deixando-os disponíveis para o uso. Os dados retornados foram corretamente disponibilizados para que as outras aplicações pudessem fazer uso. A requisição feita a MIB dos SNMPs dos *hosts* foram atendidas.

Observamos a vantagem de ter um Agente Coletor pelo fato de poder executar em cada ponto de seu itinerário, fazendo localmente as inúmeras requisições ao agentes SNMP, que anteriormente eram feitas através da rede, aumentando o tráfego e a latência da mesma. Um agente que pode se movimentar habilmente pela rede, independente da configuração de *software* ou *hardware* encontrado em cada máquina, executando da mesma forma em cada *host*.

O desenvolvimento desse trabalho também reforçou o aprendizado adquirido no curso. Foram utilizadas no trabalho técnicas de Engenharia de Software, Qualidade de Software, programação Java e linguagem Orientada Objeto, Estruturas de Dados e demais conceitos vistos durante o Curso de Ciência da Computação.

Como sugestões de trabalhos futuros temos o desenvolvimento de um Sistema Multi-Agentes para o gerenciamento de redes de computadores, com a implementação de outros agentes que possam se utilizar dos dados coletados pelo ACD; uma integração maior com o AGENTCHÊ [10], atendo a requisições feitas diretamente pelo agente de conversação ao agente móvel; melhorias na interface do agente *Se-*

nior e na segurança do agente *Coletor*; possibilitar que o agente *Senior* possa dividir itinerários grandes entre mais agentes de coleta.

Referências Bibliográficas

[1] SANTOS, E. E.; KOCH, F. L.; ASSUNÇAO, M. D.; WESTPHALL, C. B. **Agentes Coletores na Gerência de Redes de Computadores** In: Agentes Aplicados a Telecomunicações, *Agent´s Day* - CB Comp, 2003.

[2] REIS, A. L. A.; **Comparação SNMP x Agentes Móveis pra Gerência de Redes.** Dissertação de Mestrado em Ciência da Computação. Universidade Federal de Santa Catarina, 2001. Florianópolis, BR.

[3] SILVA, A.; DELGADO, J.; **Agentes de Softwares: Conceitos e Tecnologias.** 3º Encontro Nacional do Colégio de Engenharia Electrotécnica, Ordem dos Engenheiros, 1997, Lisboa, PT. Disponível em: <http://berlin.inesc.pt/alb/papers/1997/OE-97.pdf>. Acesso em: ago. 2005.

[4] LANGE, D. B. **Móbile Objects and Móbile Agents: The Future of Distributed Computing?.** In: Proceedings of the European Conference on Objects-Oriented Programming'98, 1998.

[5] LANGE, D. B.; OSHIMA, M. **Seven Good Reasons for Móbile Agents** In: Comunications of the ACM, 1999. Disponível em:<http://www.moe-lange.com/danny/docs/7reasons.pdf> Acesso em: ago. 2005.SILVA

[6] GRANVILLE, L. Z. **Agentes Móveis no Gerenciamento de Redes.** Rio Grande do Sul: Universidade Federal do Rio Grande do Sul, 1999. Disponível em: <http://www.inf.ufrgs.br/granville/Docs/AgentesMovéis>. Acesso em: set. 2005.

[7] AGLETS. **The Aglet Software Development Kit.** Disponível em: <http://aglets.sourceforge.net/>. Acesso em: set. 2004.

[8] NWANA, H. S.; **Software Agents: An Overview.** In: Knowledge Engineering Review, vol. 11, nº3, 1996.

[9] TANENBAUM, A. S. **Redes de Computadores.** 3nd edição. Rio de Janeiro, BR: Campus, 1997.

[10] SCHOPF E. C. **AGENTCHÊ:** Agente de Conversação com Linguagem Regionalista Voltado ao Ensino de Tópicos de Redes de Computadores. Rio Grande do Sul: Trabalho de Conclusão de Curso em Ciência da Computação, Universidade Federal de Santa Maria, 2004. Rio Grande do Sul, BR.

[11] LOBATO, C. A.; DAMASCENO, K. N. F.; **Agentes Móveis: Aglets na Busca de Informações** Para: Trabalho de Conclusão de Curso em Ciência da Computação, Universidade Federal do Pará, 2003. Pará, BR.

[12] SILVA, L. A. M. **Estudo e Desenvolvimento de Sistemas Multiagentes usando JADE: Java Agent Development framework** Para: Monografia de Conclusão de Curso em Informática, Universidade de Fortaleza, 2003. Ceará, BR.

[13] BERNARDES, M. C.; **Avaliação do Uso de Agentes Móveis em Segurança Computacional.** Dissertação de Mestrado em Ciência da Computação e Matemática Computacional. Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, 1999. São Paulo, BR.

[14] LARMAN, C. **Utilizando UML e Padrões:** Uma introdução à análise e ao projeto orientado a objetos. Bookman, 2000. São Paulo, SP.

[15] ARIDOR, Y.; LANGE, D.; **Agents Design Patterns: Elemental of Agent Application Design** In: Proceedings of the Second International conference on Autonomous Agents (AGENTS´98), 1998. Disponível em: <http://www.moe-lange.com/danny/patterns.pdf>. Acesso em: ago. 2005.

[16] GIRARDI, R. **Engenharia de Software Baseada em Agentes** In: Procedimento do IV Congresso Brasileiro de Ciência da Computação (CBComp2004), 2004. Santa Catarina, BR.

[17] FRANKLIN, S.; GAESSER, A.; **Is It An Agent, or Just a Program? A Taxonomy for Autonomous Agents** In: Third International Workshop on Agent Theories, Architetures and Languages, 1996. Disponível em:

<http://www.msci.memphis.edu/ franklin/Agentprog.html>. Acesso em: ago. 2005.

[18] NASSIF, L. N.; COSTA, M. F.; RESENDE, L. H. A. **AGA - Sistema de Agentes Móveis no Gerenciamento de Redes Orientado a Aplicação.** In: I Workshop de computação da Região Sul - WORKCOMP-SUL, 2004, Florianópolis, BR. Disponível em: <http://inf.unisul.br/ ines/workcomp/cd/pdfs/2927.pdf>. Acesso em: set. 2005.

[19] SOUZA, A. M.; SIMÕES, M. A. C. **Gerência de Redes de Computadores Utilizando Agentes Móveis Inteligentes**. I Workshop de Trabalhos de Iniciação Científica e Graduação Bahia-Sergipe (WTICG-BASE2003). In: III Escola Regional de Computação Bahia-Sergipe (ERBASE2003), 2003. Salvador, BR. Disponível em: <http://www.professores.fib.br/marcosimoes/Publicacoes/WTICG-BASE2003_GerenciadeRedesdeComputadoresUtilizandoAgentesMoveisInteligentes.pdf>. Acesso em: out. 2005.

[20] VIDEIRA, C.; SILVA, A. **UML, Processos e Ferramentas CASE**. Centro Atlântico, 2001. Disponível em: <http://www.berlin.inespc.pt/alb/livros/uml/index.asp> Acesso em: out. 2005.

[21] HELD, G.; et. al. **Ethernet Networks: Design, Implementation, Operation, Management.** 4th edition. Macon, USA : John Wiley & Sons, 2003.

[22] REIS, L. P. **Agentes Autônomos**, LIACC (NIAD&R), Researche Report, 2002.

[23] IBM Public License. In: Disponível em: <http://oss.software.ibm.com/developerworks/opensource/license10.html>. Acesso em: set. 2004.

[24] The Internet Engineering Task Force. Disponível em: <http://www.ietf.org/>. Acesso em: out. 2005.

[25] AdventNet Inc. Disponível em: <http://www.adventnet.com/index.html>. Acesso em: nov. 2005.

[26] iVirtua Solutions. Disponível em: <http://www.ivirtua.com.br/>. Acesso em: out. 2005.

[27] Sun Microsystems. In: Disponível em: <http://www.sun.com>. Acesso em: set. 2005.

[28] The Apache Software Foundation. Disponível em: <http://www.apache.org>. Acesso em: set. 2005.