Universidade Federal Fluminense
Instituto de Computação
Coordenação do Curso de Pós-Graduação em Computação

Curso: Mestrado em Computação
Disciplina: Introdução aos Sistemas Multi-Agentes

Sistema para Gerenciamento de Redes Baseado em Agentes Móveis

Wanderson Carvalho Bragança

Niterói – Rio de Janeiro – Brasil

Maio de 2009

Universidade Federal Fluminense
Instituto de Computação
Coordenação do Curso de Pós-Graduação em Computação

Sistema para Gerenciamento de Redes Baseado em Agentes Móveis

Wanderson Carvalho Bragança

Trabalho apresentado a Professora Viviane Torres da Silva, D.Sc., da disciplina de Introdução aos Sistemas Multi-Agentes do curso de Mestrado em Computação da Universidade Federal Fluminense.

Área de Concentração JADE, Agentes Móveis, Sistemas Distribuídos.

Niterói – Rio de Janeiro – Brasil
Maio de 2009

RESUMO

Este trabalho tem como objetivo descrever, abordar e implementar a tecnologia de agentes para o gerenciamento de redes locais, tais como gerenciamento da arquitetura técnica das redes, tráfego de pacotes, gerenciamento de dados e arquivos, usando um *framework* para o desenvolvimento de aplicações orientadas a agentes, chamado JADE (*Java Agent Developtment Framework*).

Palavras chave:JADE, Agentes Móveis, Sistemas Distribuídos.

LISTA DE ABREVIATURAS

ACC	Agent Communication Channel
ACL	Agent Communication Language
MAS	Agent Management System
API	Application Program Interface
ARP	Address Resolution Protocol
DF	Directory Facilitator
FIPA	Foundation For Intelligente Physical Agents
FTP	File Transfer Protocol
GUI	Graphical User Interface
ICMP	Internet Control Message Protocol
JADE	Java Agent Development Framework
JDK	Java Development Kit
JPCAP	Java Package For Packet Capture
JRE	Java Run-Time Environment
JVM	Java Virtual Machine
KQML	Knowledge Query And Manipulationlanguage
LGPL	Lesser Gnu Public License
LIBPCAP	Packet Capture And Network Monitoring Library For Linux
MTP	Media Transfer Protocol
RARP	Reverse Address Resolution Protocol
RMA	Remote Management Agent
RMI	Remote Method Invocation
SMA	Sistema Multiagente
SGRBA	Sistema de Gerenciamento de Redes Baseado em Agente móveis
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
WINPCAP	Packet Capture And Network Monitoring Library For Windows
WWW	World Wide Web

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

A tecnologia de agentes tem adquirido nos últimos anos uma importância cada vez maior em muitos aspectos da computação, o gerenciamento de redes é uma delas.

O uso de agentes móveis pode trazer diversas vantagens às aplicações e aos seus usuários, tais benefícios são:

1. Redução do tráfego da rede com interações que podem ser realizadas localmente, independentemente da latência da rede;
2. Execução assíncrona e descentralizada, permitindo que o usuário desconecte da rede quando os agentes executarem uma tarefa;
3. Habilidade de detectar mudanças no ambiente da execução e de reagir de acordo com elas, simplificando o desenvolvimento dos sistemas distribuídos que são mais robustos e tolerantes a falhas.

Uma desvantagem dos agentes móveis é a necessidade de se instalar uma plataforma do agente em cada máquina que os agentes forem visitar. A outra, é que quanto mais seguro, mais baixo é o desempenho do sistema. Mesmo assim é importante mencionar a segurança. Pois este é um problema difícil e enfrentado por todas as tecnologias que desenvolvem sistemas distribuídos.

Este trabalho descreve e implementa uma ferramenta de gerenciamento de redes baseadas em agentes móveis usando o *framework* JADE.

JADE[1] é um *framework*[2] implementado em JAVA que facilita a implementação de sistemas multi-agentes de acordo com as especificações da FIPA, através de um conjunto de ferramentas que suporta as fases de depuração e implementação. A plataforma de agente pode estar distribuída entre diferentes máquinas (o qual não necessita estar executando o mesmo sistema operacional) e a configuração pode ser controlada via uma interface gráfica do usuário, GUI (*Graphical User Interface*), de forma remota.

1 JADE é um *framework* feito em JAVA que serve para auxiliar o desenvolvimento de sistemas Multi-Agentes.

2 *Framework* usa-se este termo para designar uma aplicação ou conjunto de aplicações que servem de suporte ao desenvolvimento de software num determinado contexto.

A arquitetura de comunicação oferece flexibilidade e eficiência. Toda sua comunicação é feita via troca de mensagens, onde JADE cria e agencia uma fila privada, de entradas de mensagens ACL[3], para cada agente. Além disso, trabalha com todos os aspectos que não fazem parte do agente em si, e que são independentes das aplicações tais como: codificação e interpretação de mensagens e ciclo de vida dos agentes.

3 ACL é uma linguagem de comunicação, estabelecida pela FIPA, que define a codificação e sintaxe da comunicação entre agentes.

2. JUSTIFICATIVA

O uso de agentes móveis podem simplificar a implementação de sistemas distribuídos. A implementação é centrada no agente, que só precisa ser definido uma única vez.

A tecnologia de agentes móveis pode ser aplicada em ambiente descentralizado. Um agente pode migrar de um host para outro em busca de informações de forma eficiente, executam suas tarefas e retornam assim que a conexão permite, sem a necessidade de ficar enviando requisições e respostas como no modelo centralizado (cliente/servidor).

Os Agentes Móveis requerem apenas a instalação da plataforma do agente em cada *host* a ser visitado.

O framework JADE é um ambiente amigável com outras tecnologias, e propõe uma infra-estrutura de suporte e desenvolvimento de sistemas multi-agentes, tornando a implementação mais simples e robusta.

Por estar em constante atualização e apresentar portabilidade o JADE se torna uma ótima opção para o desenvolvimento de sistemas multi-agentes.

3. PROBLEMÁTICA

O problema a ser resolvido se limita à tarefa de descrever e implementar a tecnologia de agentes móveis, no desenvolvimento de uma aplicação para o gerenciamento de rede, usando o *framework* JADE. E dentro desta perspectiva, surge a seguinte indagação: “Uma ferramenta de gerenciamento de redes baseadas em agentes móveis poderá realmente capturar pacotes de um ou mais *host* para auxiliar com eficácia no monitoramento e nos estudos de fluxos de informações da rede?”.

4. HIPÓTESE

A hipótese para responder à questão supra levantada é a confirmação de que a ferramenta de gerenciamento de redes baseadas em agentes móveis realmente poderá auxiliar com eficácia no monitoramento e nos estudos de fluxos de informações, devido às características que apresenta.

5. OBJETIVO

5.1. OBJETIVOS GERAIS

Este trabalho tem como objetivos gerais:

Analisar a teoria referente ao tema de gerenciamento de redes baseado em agentes móveis usando o *framework JADE;*

Desenvolver uma ferramenta de gerenciamento de redes baseadas em agentes móveis visando reduzir o tempo gasto em gerenciamento e o número de mensagens trocadas na realização desta tarefa.

Proporcionar maior flexibilidade no gerenciamento de redes.

Facilitar o trabalho dos administradores de redes no que tange ao monitoramento de acesso à internet, a partir do relatório gerado pelo sistema.

5.2. OBJETIVOS ESPECÍFICOS

Descrever e implementar em JAVA, um sistema de gerenciamento de redes usando agentes móveis.

Executar o sistema implementado, para validação do mesmo.

Monitorar o acesso à *internet,* de forma descentralizada, ou seja, diretamente de um *host* escolhido pelo administrador;

Apresentar à comunidade acadêmica os resultados obtidos através deste estudo, para servir de referencial em estudos futuros, tanto na continuidade deste como comparativo em outras aplicações.

6. GERENCIAMENTO DE REDES E SISTEMAS DISTRIBUÍDOS HETEROGÊNEOS

O rápido crescimento da *Internet* e das redes de computadores, juntamente com a necessidade de gerenciamento, em relação ao funcionamento dos sistemas descentralizados autônomos, têm levado os desenvolvedores de sistemas a trabalharem na construção de ferramentas para auxiliar os administradores de redes, em suas tarefas de gerenciamento e controle.

A utilização de agentes que são instruídos através de *softwares* programados a realizarem tarefas, às vezes repetitivas em domínios locais ou distribuídos, tem ajudado muito na administração do controle e monitoramento dos sistemas.

As redes e os sistemas distribuídos heterogêneos têm exigido mudanças e atualizações freqüentes. A garantia nestes sistemas distribuídos heterogêneos e nas redes é definida pela incorporação de tecnologias que suportam a integração entre várias tecnologias. Essa incorporação garante propriedades funcionais, tais como: a confiabilidade, a segurança, a tolerância a falhas, a autonomia e a mobilidade.

6.1. TAREFAS DE GERENCIAMENTO

O uso dos recursos da rede precisa ter suas atividades controladas e monitoradas, para possibilitar o diagnóstico dos problemas. A tarefa de monitoramento e controle resulta na gerência da rede, que tem a função de obter informações, tratá-las e apontar soluções às falhas do sistema.

Eis algumas tarefas de gerenciamento que devem ser realizadas para a manutenção e funcionamento adequado da rede e do sistema:

- Rede
    - Monitorar estado e tráfego em conexões;
    - Manter os nós da conexão ativos;

- Sistema
    - Acesso: automatizar a distribuição de arquivos;
    - Identificar: manter inventário de *hardware*;
    - Avaliar e planejar: gerenciar recursos compartilhados e *softwares* instalados;
    - Implantar: instalar e manter atualizadas as versões de Sistemas Operacionais e *Softwares*;

6.2. GERENCIAMENTO DE SOFTWARES INSTALADOS

Um gerenciamento de software instalado se dá através de políticas e procedimentos, conhecimento dos recursos dos *softwares* e estabelecimento de rotinas de verificação.

Em gerência de *software* é interessante criar e manter um registro de todas as licenças, termos e condições, inventários de sistemas e utilização de *software*, para facilitar a identificação de necessidades, problemas e dificuldades, bem como melhorar a eficiência do uso dos *softwares* e reduzir o risco de softwares sem licença que esteja em uso.

Mas, o gerenciamento de software não se restringe ao controle de inventário. É necessário criar um sistema que gerencie todo seu ciclo de vida, desde a aquisição e o uso até a baixa. Isso certamente trará resultados positivos. Um gerenciamento de *software* apropriado economiza tempo e dinheiro, mantém o mesmo e as informações compatíveis em toda a entidade e torna fácil para adaptar-se a mudanças.

Por questões de segurança ou políticas de uso, administradores de uma rede local têm a necessidade de saber quais os programas que os usuários estão utilizando, por exemplo: uma empresa não permite que os funcionários usem programas de bate-papo em determinados horários. Então há uma necessidade de monitorar os acessos aos programas para se cumprir as determinações das políticas de uso da rede.

A tecnologia de agentes permite que os desenvolvedores implementem agentes inteligentes de modo que monitorem os programas instalados nas máquinas dos usuários de uma rede local ou remota enviando relatório para o administrador da rede toda vez que um novo programa for instalado.

7. AGENTES MÓVEIS

Para Aridor et al (1998) os agentes móveis apresentam: redução do tráfego da rede em sistemas distribuídos, com grande volume de troca de informações, permitindo que as tarefas sejam executadas com interações locais ou ainda movidas para execução no local de armazenamento dos dados. Assim, ao invés de mover os dados para serem processados, move-se o processamento até os dados.

7.1. CONCEITO

Um agente móvel é um programa que pode migrar de uma máquina para outra em uma rede de sistemas computadorizados, e cumprir uma tarefa especificada pelo seu proprietário. Trabalha de forma autônoma e comunica-se com outros agentes e sistemas da máquina visitada. Durante a migração, o agente carrega consigo, seu código e o tipo do estado da execução. O estado de execução depende da linguagem de programação utilizada, em JAVA, por exemplo, o agente é colocado em série (que é um objeto de uma classe específica) e não contem a informação sobre o estado da máquina virtual de JAVA, mas ele o compreende. Em cada *host* que os agentes móveis visitam, há uma necessidade de um *software* especial, que é responsável para receber e executar os agentes, fornecendo um ambiente seguro de execução, e diversos serviços para os agentes (estáticos) que residem neste *host*.

A migração dos agentes é baseada em uma estrutura que pode fornecer os serviços necessários na rede. Os serviços incluem o acesso aos recursos e aplicações locais, troca de informações (mensagens) entre agentes, serviços de segurança básico e criação de novos agentes, dentre outros.

Os agentes móveis são atualmente um tópico quente no domínio de sistemas distribuídos. A tecnologia de agentes móveis pode ajudar a projetar soluções inovadoras nos atuais sistemas cliente/servidor.

7.2. MODELO DE AGENTES

O termo agente está sendo usado cada de vez mais na literatura. Várias definições já foram propostas, de uma maneira principal os agentes podem ser classificados em cinco níveis.

- **Agentes de execução:** responsáveis por contribuir com o envolvimento de habilidades de nível de planejamento íntegro do grupo.
- **Agentes de colaboração:** responsáveis pela execução de uma ação independente, porém a colaboração dada deve ser integrada às demais ações do grupo.
- **Agentes de contribuição**: responsáveis em dar contribuição direta à uma ação que envolve um único indivíduo.
- **Agentes de comunicação:** fica responsável pelo gerenciamento das comunicações entre agentes, exemplo, envia mensagens para um grupo de agentes ou coordena as mensagens recebidas.
- **Agentes de serviços:** são responsáveis para oferecer serviços de baixo nível aos agentes ou aos componentes do sistema. As tarefas são simples e compreendidas implicitamente.

7.3. SEGURANÇA

Os ambientes computacionais ligados em rede e com acesso a *Internet*, são suscetíveis a ataques maliciosos. Os ataques podem ocorrer de diversas maneiras:

- Agente atacando uma plataforma de agente, através de ataque ao sistema operacional ou captura de informações para alterá-las;
- Plataforma atacando agente através da intromissão ou mascaramento;

- Agentes atacando agentes de outras plataformas através da repudiação e;
- Outras entidades atacando os agentes ou a plataforma através de um elemento externo.

Para se prevenir contra esses problemas que são desvantagens para os agentes é necessário que se adote uma política de uso do ambiente com autenticação de usuários, proteção da plataforma, controle e monitoramento ao acesso dos aplicativos locais e, ainda, a plataforma deve fornecer confidencialidade do código e do estado do agente. Os agentes devem ser autenticados.

7.4. SISTEMAS MULTI-AGENTES

Os Sistemas Multi-Agentes(SMA) emergiram como uma das áreas mais importantes de pesquisa e de desenvolvimento na tecnologia de informação nos anos 1990. Um (SMA) é composto por vários agentes, que são tipicamente capazes de cooperar para resolver os problemas que estão além das habilidades de apenas um agente. Os sistemas Multi-agentes são importantes porque foram desenvolvidos para ter a aplicabilidade muito diversa, por exemplo, tanto na área de controle de processo industrial, quanto na área de comércio eletrônico.

SMA’s são sistemas desenvolvidos e implementados usando vários agentes que interagem entre si, dentro de um mesmo ambiente, com a finalidade de ajuda mútua em soluções de problemas.

Para o SMA resolver problemas, os agentes devem:

- Comunicar-se entre si;
- Coordenar suas atividades e;
- Negociar em caso de conflitos.

Portanto, SMA é um conjunto de agentes autônomos, possivelmente pré-existentes que trabalham na solução de problemas que está além da capacidade de um único agente.

Os SMA’s formam uma subárea da Inteligência Artificial Distribuída e concentram-se no estudo de agentes autônomos em um universo multiagente. Para os Sistemas Multi-agentes, um é agente autônomo quando tem independência nas ações em relação a outros agentes. Cada agente carrega dentro de si, intrinsecamente, o conjunto de comportamentos que definem sua capacidade de agir como tal. Se dotado de inteligência, a definição de seu comportamento lhe permitirá tomada de decisões em suas ações.

O objetivo dos SMA’s é criar mecanismos genéricos para a coordenação de tais agentes possibilitando que o sistema como um todo (em geral chamado de uma sociedade de agentes) funcione de forma adequada e eficiente.

8. JPCAP

Para realização deste trabalho, foi utilizado o framework JADE para criação dos agentes. Um dos principais problemas enfrentados refere-se à interação do agente em JADE com o dispositivo de rede, ou seja, um agente de captura implementado totalmente em JAVA. Com isso foi utilizado a *JPCAP(Java package for packet capture)* biblioteca de captura de pacotes para linguagem JAVA.

JPCAP é um pacote organizado, composto de bibliotecas de classes que permitem implementações que capturem ou enviem pacotes através de uma rede de computadores.

Jpcap é baseado em *libpcap/winpcap*. Portanto, aplicações utilizando a JPCAP podem ser desenvolvidas para qualquer sistema operacional que suportem a *libpcap/winpcap*. Atualmente, Jpcap tem sido testado em sistemas operacionais como: FreeBSD 3.x, em *Linux RedHat*, *Fedora Core*, em Solaris, e em *Microsoft Windows* 2000/XP/Vista.

Jpcap suporta os seguintes tipos de protocolos: *Ethernet*, IPv4, IPv6, ARP/RARP, TCP, UDP, e ICMPv4. Outros tipos de pacotes são capturados como os pacotes primitivos que contém todos os dados dos pacotes. Isto permite que as aplicações do JAVA analisem pacotes com tipos não suportáveis.

9. JADE

9.1. O QUE É JADE

Para ilustrar os conceitos de desenvolvimento de sistemas multi-agentes usando o *framework* JADE é ideal que se tenha conhecimento prévio da tecnologia JAVA. Tecnologia porque além da linguagem oferece outros componentes como a *Interface de Programação de Aplicativos* (API) e a *máquina virtual JAVA*(JVM).

JADE é um *framework* de código fonte aberto que simplifica a implementação e execução de SMA's de um *middleware*[4] de acordo com as especificações da FIPA.

A plataforma do agente pode ser distribuída através das máquinas e a configuração pode ser controlada através de uma interface gráfica - remotamente. A configuração pode ser alterada no *run-time* movendo agentes de uma máquina para outra, quando requerido.

O JADE é totalmente implementado e composto de vários pacotes JAVA, dando ao programador várias interfaces abstratas para adaptá-las a sua preferência.

9.2. OS AGENTES EM JADE

9.2.1. Classe Agente

A classe agente representa uma classe base para criação de agentes, consequentemente, do ponto de vista do programador, um agente em JADE é simplesmente uma instancia da classe *Agent* nas quais os programadores deverão escrever seus próprios agentes como subclasses da classe `Agent`. Isto

4 *Middleware* é um *software* que serve de ponte entre dois programas diferentes, convertendo o formato das informações de forma que ambos se entendam.

implica que os agentes criados herdem as características básicas para realizar interações com a plataforma do agente (registro, configuração, gerência remota...) e um conjunto de métodos que podem ser chamados para executar o comportamento (por exemplo: o envio e recebimento de mensagens que usam protocolos padrão da interação e de registro com diversos domínios).

O modelo computacional de um agente é o multitarefa, onde as tarefas (ou os comportamentos) são executadas simultaneamente. Cada funcionalidade/serviço fornecido por um agente deve ser executado como um ou mais comportamento.

Um programa interno a classe *Agent* invisível ao programador, controla automaticamente a programação dos comportamentos.

9.3. DEFINIÇÃO DE COMPORTAMENTOS *(BEHAVIOURS)*

As tarefas designadas a um agente no jade no JADE são definidas como “comportamentos”. Um comportamento representa uma tarefa que um agente pode realizar, e é executado como um objeto de uma classe que herda da classe *jade.core.behaviours*.

Um agente pode realizar diversas tarefas simultâneas em resposta aos diferentes eventos externos. A fim de fazer uma gerência eficiente, cada agente do JADE é composto de uma única *thread* de execução e todas suas tarefas são modeladas e podem ser executadas como objetos da classe *jade.core.behaviours*. Os agentes *Multi-threaded* também podem ser executados, mas nenhum suporte específico (exceto sincronizar a fila de mensagens ACL) é fornecido pelo JADE.

Para um agente desenvolver uma tarefa especifica deve-se primeiro definir uma ou mais subclasses da classe *behaviours,* instanciar os objetos e adicioná-los a lista de tarefas do agente. A classe do agente, que deve ser herdada pelos programadores de agentes, disponibiliza dois métodos:

addBehaviour(Behaviour) e *removeBehaviour(Behaviour),* que controlam a fila de tarefas de um agente específico. Nota-se que os comportamentos e os sub-comportamentos podem ser adicionados sempre que houver necessidade e não somente dentro do método *Agent.setup().*

No JADE um comportamento pode ser instanciando de uma das subclasses da classe *jade.core.behaviours,* que são:

- ***jade.core.behaviours.behaviours***
- ***jade.core.behaviours.CompositeBehaviour***
- ***jade.core.behaviours.SimpleBehaviour***

Os principais métodos da classe *behaviours* são:

- ***action():*** principal método da classe *behaviours* este método contém as instruções que vai determinar as ações do comportamento. É nele que são implementadas as tarefas dos comportamentos.
- ***Block():*** permite bloquear um comportamento.
- ***Done():*** retorna um valor *booleano*, especificando se o comportamento terminou ou não, caso o retorno seja `true`, ou seja, finalizou, o comportamento tem que ser removido da fila de comportamentos do agente, se *false*, executa o método *action()* novamente.

9.4. COMUNICAÇÃO ENTRE AGENTES

A habilidade de comunicar-se de uma maneira complexa, como trocar informações, expressar idéias, instruções e perguntas, emoções e interagir em um nível social, são um dos aspectos mais importantes dos seres humanos. Assim sendo, um dos principais objetivos do desenvolvimento de agentes é fazer com que agentes artificiais progridam ao nível em que possa haver uma comunicação entre humanos e agentes artificiais.

9.4.1. Linguagens de Comunicação

Visando resolver o problema de comunicação entre agentes, um grupo resolveu então criar um padrão para comunicação entre agentes, Assim, os agentes não precisam necessariamente ter sido desenvolvido utilizando as mesmas ferramentas e modelos. Contudo, deve concordar no uso de uma linguagem comum para comunicação. Algumas especificações de Linguagem de Comunicação de Agentes (LCA) foram propostas como objetivo de viabilizar a comunicação entre agentes desenvolvidos em projetos diferentes e possibilitar a descrição das regras que regem o uso de comunicação entre eles. Dentre as propostas de LCA existentes, duas se destacam: KQML e FIPA-ACL.

As linguagens de comunicação entre agentes, como FIPA-ACL e o KQML, se tornaram o foco das atenções nesses últimos anos. Pois se mostraram eficazes para estabelecer comunicação entre agentes, ou ainda, para interação simples do humano-agente, mas estão longe de refletir a complexidade de uma comunicação humana. Para que haja comunicação entre os agentes, os *framework* para o desenvolvimento de agentes devem conter: representações formais das ações e de ontologias das ações, a integração da informação no contexto da situação - que servem como uma base para executar o comportamento - e finalmente uma comunicação entre agentes inteligentes.

9.4.2. FIPA-ACL

FIPA – (Fundação para desenvolvimento de agentes físicos inteligentes) é uma organização internacional que dedica a promover a indústria de agentes inteligentes abertamente, desenvolvendo as especificações que suportem a interoperabilidade entre agentes e a aplicação baseada em agentes. (Isto ocorre com a colaboração aberta entre as organizações membro, que são empresas, universidades, etc.).

A FIPA-ACL é uma linguagem que permite a comunicação entre agentes.

Uma característica fundamental dos sistemas multi-agentes é que os agentes individuais se comunicam e se interagem. Isto é realizado com a troca das mensagens e, para se compreenderem é crucial que os agentes concordem com o formato e a semântica destas mensagens. O JADE segue padrões da FIPA de modo que os agentes do jade pudessem interagir satisfatoriamente com os agentes escritos em outras linguagens e plataformas.

Formato de uma mensagem FIPA-ACL

```
(inform
:sender agent1
:receiver agent5
:content (valor ZZZ 150)
:language PT
:ontology xxxxx)
```

9.5. IMPLEMENTACÃO DE SISTEMAS MULTI-AGENTES EM JADE

O JADE possibilita a conformidade sintática e, onde possível, a conformidade semântica com especificações da FIPA para implementações de sistemas multi-agentes.

9.5.1. A plataforma do Agente de acordo com as especificações da FIPA

O modelo padrão de uma plataforma de agentes, definido pela FIPA, é representado na seguinte figura.

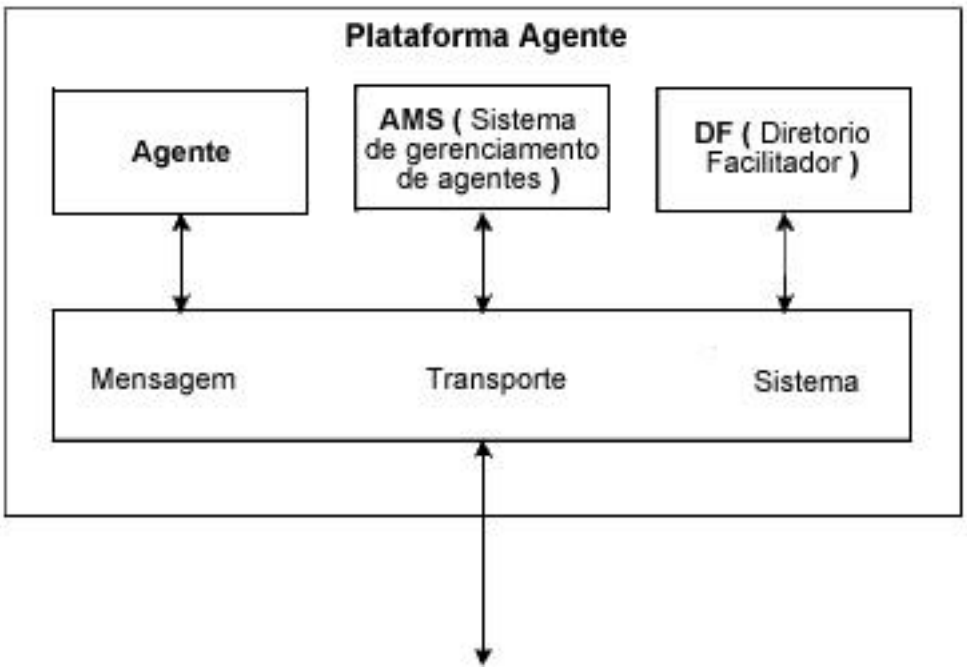

Figura 1 Referência da arquitetura de uma plataforma de agentes conforme a FIPA

Como ilustrado na figura 2, segue abaixo as definições das estruturas que compõem uma plataforma de agentes:

O sistema de gerenciamento de agentes (AMS) é o agente que exerce o controle sobre o acesso e o uso da plataforma do agente. Em cada plataforma só pode existir um AMS. Ele fornece o serviço de páginas brancas e o ciclo de vida dos agentes, mantendo um diretório de identificadores do agente (AID) e de estado do agente. Cada agente deve registrar-se em um AMS a fim iniciar um (AID) válido.

O diretório facilitador (DF) é o agente que fornece o serviço de páginas amarelas por padrão na plataforma, onde após serem registrados os agentes passam a categoria de associados e podem ter suas características consultadas, alteradas ou ainda ser excluído do banco de dados.

O sistema de transporte de mensagens, também chamado de canal de comunicação de Agente (ACC), é o componente de *software* que controla toda a troca de mensagens dentro da plataforma.

O JADE cumpre inteiramente com essa arquitetura; quando a plataforma do JADE é lançada, o AMS e o DF são criados automaticamente e o canal de comunicação (ACC) é ajustado para permitir a comunicação de mensagens. A

plataforma do agente pode ser dividida em diversos *host*. Somente uma aplicação de Java, e conseqüentemente somente uma máquina virtual (JVM), são executadas em cada *host*. Cada JVM é um recipiente básico de agentes, que fornece um ambiente completo do tempo de funcionamento para a execução do agente e, permite que diversos agentes executem simultaneamente no mesmo *host*. O *Main-Container*, ou o *front-end*, são os *containers* de agentes onde são criados o AMS e o DF, e o registro do RMI que é usado internamente por JADE. Os outros *containers* do agente conectam ao *container Main-Container* e fornece um ambiente *run-time* completo para a execução de agentes do JADE.

10. ESTUDO DE CASO

10.1. ARQUITETURA DO SISTEMA SGRBA

A arquitetura proposta é baseada em uma especificação modular implementada em um ambiente composto por agentes móveis e agentes estáticos. Os agentes móveis desempenham o papel de agentes de captura.

O sistema é composto de duas partes principais.

1. Agente *SGRBA* fica responsável por todo ciclo de vida dos agentes, como criação, destruição atribuição de comportamentos, envio de mensagens e geração da interface gráfica para interação com usuário.

2. *AgenteEscravo*, este agente é o responsável pela captura de pacotes, monitoração das atividades dos usuários e envio de relatórios para agente *SGRBA*.

De uma maneira geral o agente *SGRBA* envia um agente móvel (*AgenteEscravo*) para executar a tarefa de gerenciamento na rede. Ao chegar a um *host* especifico o agente proposto nessa aplicação, inicia o processo de captura de pacotes e armazenando as informações obtidas na execução da tarefa. Ao final da tarefa, o *AgenteEscravo* envia as informações para o agente *SGRBA*.

O agente *SGRBA* pode enviar mais de um agente para fazer à mesma tarefa a *hots* diferentes, tendo assim um menor tempo de resposta. Para que o agente possa coletar informações de cada *host*, as máquinas da rede que são visitadas pelos agentes são previamente preparadas para hospedar temporariamente o agente. Isso implica na instalação do ambiente JADE.

Para que o agente não se mova para uma máquina que não esteja ativa: antes de migrar, o agente *SGRBA* verifica, com AMS (*Agent Management System*), e lista na GUI todas as plataformas disponíveis. Esta verificação é feita pelo

comportamento *PegaLocaisDisponiveis*. A figura a seguir mostra como o processo é feito.

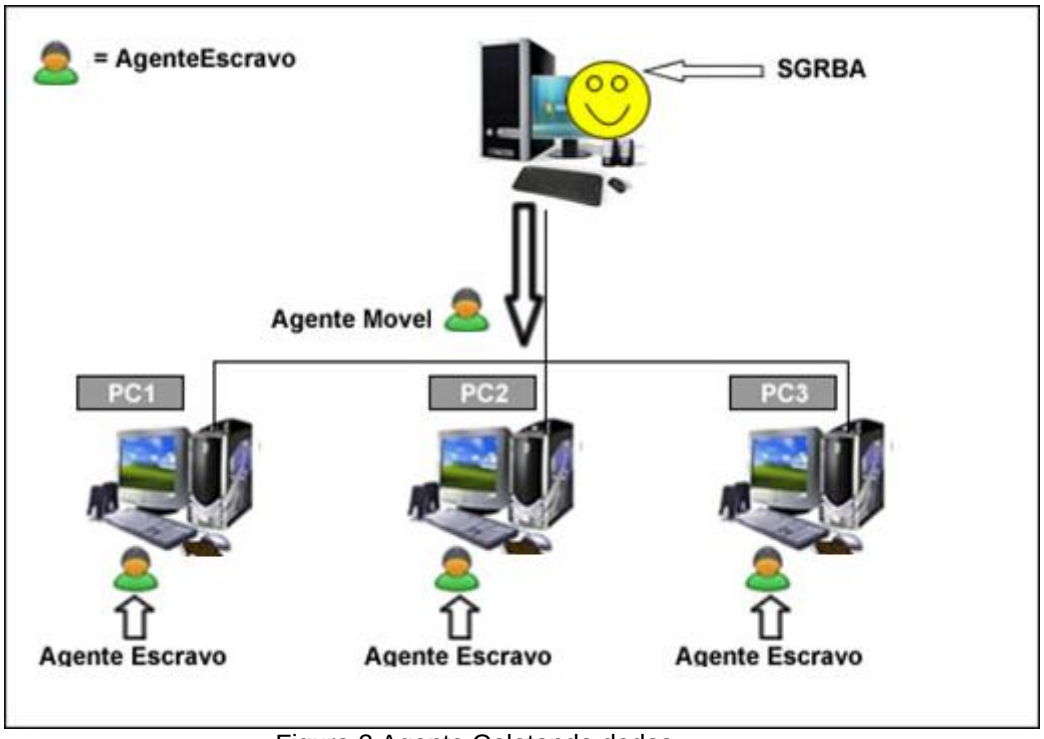

Figura 2 Agente Coletando dados

10.2. CARACTERÍSITICAS DO SISTEMA

10.2.1. Agente SGRBA

O *SGRBA* é o agente principal que herda da classe *GuiAgent*, disponível no JADE, que cria a Interface gráfica para interagir com o administrador da rede. Veja a figura a seguir:

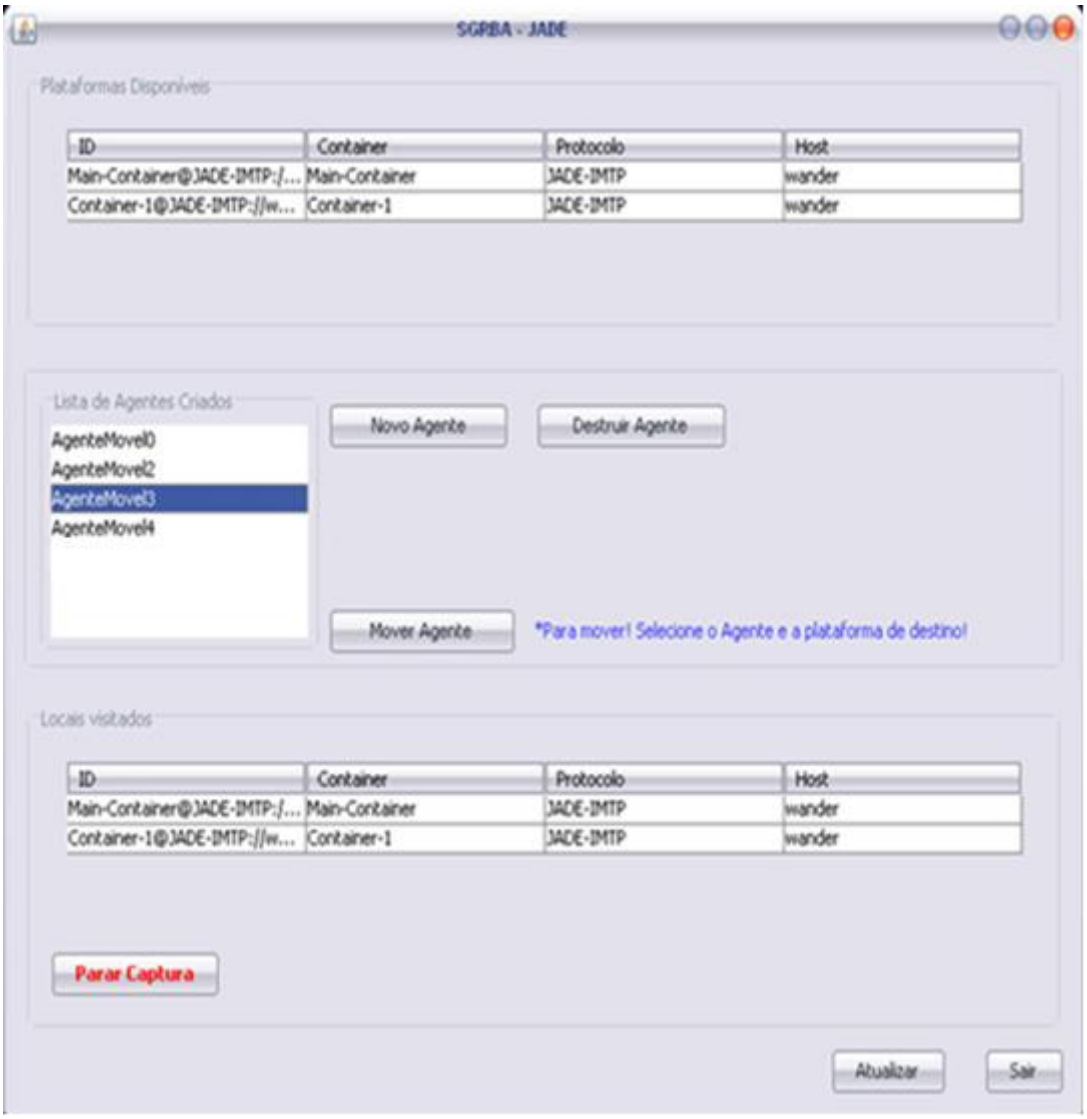

Figura 3 Interface gráfica do Sistema SGRBA

A interface gráfica do SGRBA apresenta-se ao usuário com os seguintes componentes:

Na parte superior encontram-se listadas, as plataformas disponíveis para receber o agente móvel.

Na parte central localiza-se uma lista de agentes criados e opções para criar, destruir, e mover agentes.

Na parte inferior encontram-se listadas os locais visitados.

Ao clicar no botão **“NOVO AGENTE”**, o mesmo emite um evento para o agente `SGRBA` que implementa o método `public void`

onGuiEvent(GuiEvent ev) da interface *jade.gui.GuiAgent* . O agente *SGRBA* recebe o evento, verifica que o evento é para criar um novo agente e executa o código. Veja abaixo como é feita a implentação para criar um novo agente a partir de outro:

```
case NOVO_AGENTE:
    jade.wrapper.AgentController a = null;
    try {
        Object[] args = new Object[2];
        args[0] = getAID();
        String name;
        Cria um nome para o agente
        name = "AgenteMovel"+cont++;
        a = home.createNewAgent(name, MobileAgent.class.getName(), args);
        a.start();
        Adiciona o agente na lista
        agents.add(name);
        Exibe na GUI
        myGui.updateList(agents);
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null,"Problema na criação do Agente ");
    }
    break;
```

Figura 4 Cria um novo agente a partir de outro

DESTRUIR AGENTE – Ao clicar sobre esse botão, o agente que estiver selecionado será *destruído,* o agente *SGRBA* recebe o evento da GUI para destruir o agente e como parâmetro recebe o nome do agente. O agente SGRBA pega o AID do agente envia uma ordem para o mesmo se destruir. O agente recebe a mensagem e executa o método *doDelete()*. Veja na figura a seguir como é feito a ordem para o agente se destruir:

```
case KILL_AGENTE:
    PEGA O NOME DO AGENTE A SER REMOVIDO
    String agentName = (String)ev.getParameter(0);
    LOCALIZA O AGENTE
    AID aid = new AID(agentName, AID.ISLOCALNAME);
    KillAgent ka = new KillAgent();
    ka.setAgent(aid);
    CRIA UM COMPORTAMENTO: ENVIA A ORDEM DE DESTRUIÇÃO PARA O AGENTE
    addBehaviour(new  EnviaMsg(this,new Action(aid, ka)));
    REMOVE O AGENTE DA LISTA
    agents.remove(agentName);
    ATUALIZA A LISTA NA GUI
    myGui.updateList(agents);
    break;
```

Figura 5 Envia ordem de destruição para o agente

MOVER AGENTE – O agente selecionado será deslocado para a plataforma escolhida. Recebe como parâmetros o nome do agente e a plataforma de destino. O agente *SGRBA* pega o AID do agente envia uma ordem para se locomover à plataforma escolhida. O agente recebe a mensagem e executa o método *doMove(localDestino)*. Veja abaixo como é feito a ordem para o agente se mover:

```
case MOVE_AGENTE:
    PEGA O NOME DO AGENTE A SER MOVIDO
    String agentName1 = (String)ev.getParameter(1);
    AID aid1 = new AID(agentName1, AID.ISLOCALNAME);
    PEGA O LOCAL DE DESTINO
    Iterator MoveParameters = ev.getAllParameter();
    nextSite =(Location)MoveParameters.next();

    MobileAgentDescription mad1 = new MobileAgentDescription();
    mad1.setName(aid1);
    mad1.setDestination(nextSite);

    MoveAction ma = new MoveAction();
    ma.setMobileAgentDescription(mad1);
    addBehaviour(new  EnviaMsg(this,new Action(aid1, ma)));

    break;
```

Figura 6 Envia uma ordem para o agente mover-se para outro local

Na parte inferior encontra-se uma tabela com a lista de locais visitados e opções de “iniciar” e parar “captura” de pacotes alem dos botões “atualizar” e “sair”.

ATUALIZAR – essa ação atualiza a lista de plataformas disponíveis executando o método:

```
addBehaviour(new PegaLocaisDiponiveis(this));
```

SAIR – Encerra SGRBA, e destrói todos os agentes criados, finalizando o monitoramento das ações.

10.3. AGENTE DE CAPTURA DE PACOTES

A característica principal do agente de captura *(AgenteEscravo)* proposto nesse projeto, é monitorar o acesso à internet em uma rede, mas de forma descentralizada, ou seja, diretamente de um *host* escolhido pelo usuário.

As informações capturadas são gravadas em um arquivo, que é enviado para o agente *SGRBA* quando o *AgenteEscravo* receber o pedido para finalizar o processo de captura. A figura a seguir mostra os pacotes capturados pelo agente *AgenteEscravo.*

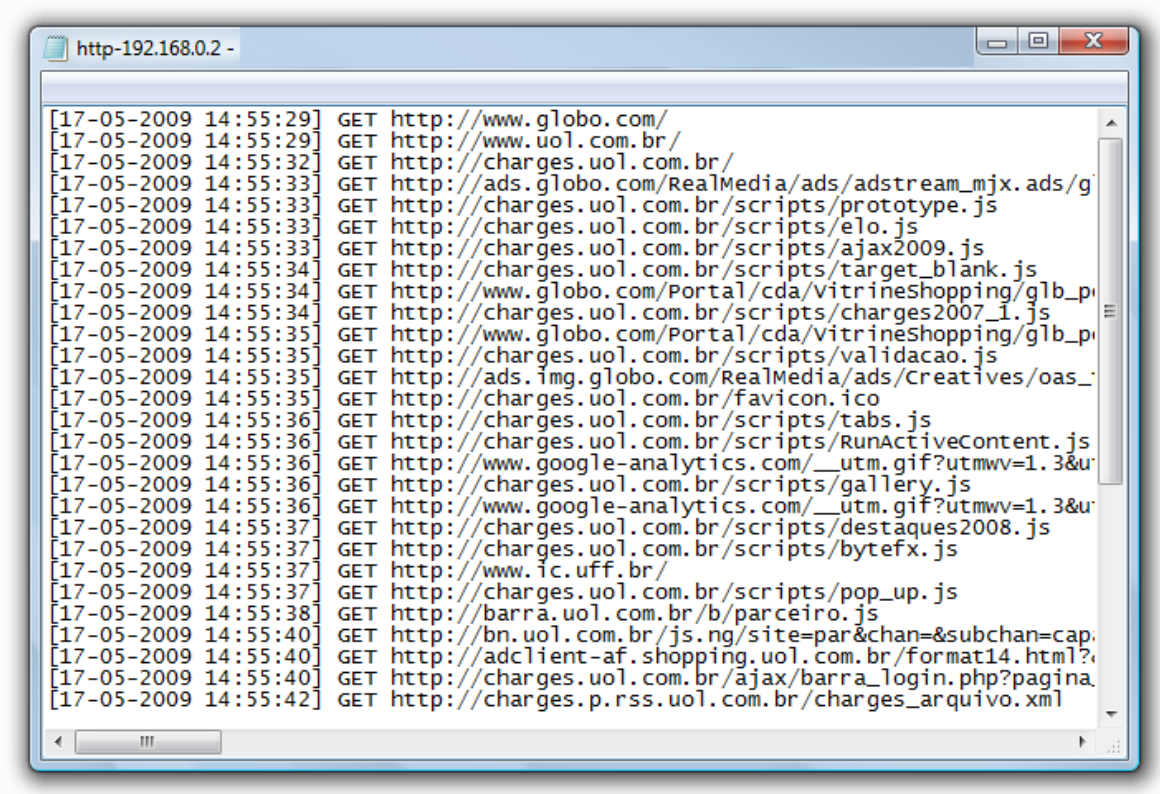

Figura 7 Arquivo com os pacotes capturados pelo *AgenteEscravo*

11. CONCLUSÃO

O objetivo principal foi descrever e implementar uma aplicação de gerenciamento utilizando a tecnologia de agentes móveis que permitisse ao usuário realizar, com facilidade, tarefas de gerenciamento. Daí surgiu o SGRBA (Sistema de Gerenciamento de Redes Baseado em Agente Móveis). O projeto foi desenvolvido em JAVA usando o *framework* JADE para criação de agentes.

Embora a tecnologia de desenvolvimento de agentes, aplicada com sucesso em alguns estudos de caso, sua aplicação em cenários práticos é muito restrita, ainda. Pelo fato de ser relativamente nova, muitas questões ainda precisam ser resolvidas e há muito ainda a ser explorado. Porém, por ser uma ferramenta gratuita e de código aberto e por estar em constante atualização, JADE se torna uma excelente alternativa para o desenvolvimento e suporte de sistemas com agentes móveis.

Um ponto positivo foi conhecer mais detalhadamente o JADE e outro foi fazer a implementação de agentes móveis integrando Jpcap e JADE.

A principal contribuição deste trabalho está na arquitetura do sistema. O SRGBA é totalmente descentralizado. O *host* que executar o SGRBA assume o papel de gerente, o que pode acontecer de qualquer ponto da rede em acesso local ou remoto.

Apesar dos problemas durante o decorrer deste trabalho, principalmente na integração entre Jpcap e JADE, a realização do mesmo foi uma experiência muito enriquecedora e proveitosa.

Conclui-se, então que o objetivo proposto em gerenciar redes através de agentes móveis foi alcançado.

12. REFERÊNCIAS BIBLIOGRÁFICAS

ARIDOR, Yariv & LANGE, Danny B. **Agent Design Patterns: Elements of Agents Application Design**. Proceedings of the Second International Conference on Autonomous Agents. (15 de Julho 1998).

BELLIFEMINE Fabio, CAIRE Giovanni, TRUCCO Tiziana, RIMASSA Giovanni. **JADE Programmer's Guide**. 2003. Documento eletrônico . Disponível em: http://sharon.cselt.it/projects/jade/doc/programmersguide.pdf. Acesso em Abril de 2009

DEITEL, H. M. & P. J. Deitel. **Java**: Como Programar 3a. Edição 2000.

FIPA. Documento eletrônico. Foundation for Intelligent Physical Agents – **FIPA**. Disponível em: < http://www.fipa.org>, Acesso Maio de 2009.

HORSTMANN, Cay. **Core Java 2** - Volume I Fundamentos. Makron Books

JADE PROGRAMMER'S GUIDE. Documento eletrônico. Disponível em: <http://jade.tilab.com/doc/programmersguide.pdf>. Acesso Maio de 2009.

JAVA. ***Java Agent Development framework*** - JADE. Documento eletrônico. Disponível em: <http://jade.tilab.com/ "JADE Home Page">. Acesso em Abril de 2009.

Jpcap **– Java package for packet capture**. Disponível em: http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html, Acesso em Abril de 2009.

LibPcap: Documento eletrônico .Disponível em: http://www.tcpdump.org/, acesso em Abril de 2009.

SILVEIRA, Ricardo A. Modelagem Orientada a Agentes Aplicada a Ambientes Inteligentes Distribuídos de Ensino: **JADE** – Java Agent Framework for learning

WinPcap: *The Windows Packet Capture Library*. Documento eletrônico. Disponível em: http://www.winpcap.org/. Acesso em Abril de 2009.

WinPcap: *The Windows Packet Capture Library* . Documento eletrônico .Disponível em: http://winpcap.mirror.ethereal.com/301a/docs/main.html, Acesso em Abril de 2009.