Infra-estrutura GTA/AG: Um Ambiente de Desenvolvimento de Sistemas Baseados em Agentes *

Marcelo Gonçalves Rubinstein
rubi@gta.ufrj.br

Otto Carlos Muniz Bandeira Duarte
otto@gta.ufrj.br

Grupo de Teleinformática e Automação (GTA)
Universidade Federal do Rio de Janeiro
COPPE/EE - Programa de Engenharia Elétrica/Departamento de Eletrônica
Caixa Postal 68504 - CEP 21945-970 - Rio de Janeiro - RJ - Brasil
FAX: + 55 21 2906626

Resumo

Este trabalho propõe um ambiente de desenvolvimento de sistemas baseados em agentes denominado Infra-estrutura GTA/AG (GTA/AGente). Esta infra-estrutura visa prover serviços genéricos de migração, comunicação, execução, segurança e tolerância a falhas, permitindo a um desenvolvedor de aplicações baseadas em agentes focar-se nos detalhes das aplicações, sem se preocupar com a infra-estrutura. Por ser uma estrutura modular e flexível, é possível escolher e configurar os módulos de acordo com as características do sistema desejado, bem como criar novos módulos. O modelo de arquitetura utilizado e os módulos do servidor de agentes, de transferência, comunicação, segurança e tolerância a falhas, são descritos.

Abstract

This paper proposes a development environment for agent-based systems denominated GTA/AG Infrastruture (GTA/AGent). This infrastructure aims to provide generic services like migration, communication, execution, security, and fault tolerance, permitting an agent-based applications developer to focus in applications details, without worrying about the infrastructure. Since it is a modular and flexible structure, it is possible to choose and configure modules by system characteristics, and create new modules. The architecture model and transport, communication, security, and fault tolerance modules of the agent server are described.

1 Introdução

Com o extraordinário desenvolvimento da rede mundial de computadores, a Internet, três fatos principais têm sido motivo de grande preocupação por parte da comunidade de redes e também de desenvolvedores de aplicativos: a quantidade de informação disponível pelo mundo, o custo de movimentação de dados e a pequena experiência em computação de muitos usuários. A quantidade de informação disponível em diferentes sítios por todo o mundo não tem sido acompanhada pela capacidade de um usuário de procurar, selecionar e recuperar estas informações. Além disso, a movimentação de grandes quantidades de dados através da rede possui um custo significativo. Por último, o número de usuários com pouco

*Este trabalho foi realizado com recursos da UFRJ, FUJB, PROTEM-CC, CNPq, CAPES e COFECUB.

conhecimento em computação tem crescido e deverá continuar a crescer rapidamente. Uma possível solução para estes problemas consiste na utilização de agentes.

Os **agentes** são programas que ajudam um usuário a realizar tarefas, agindo em interesse deste usuário. Caso os agentes possam executar suas tarefas em diferentes computadores da rede, os agentes são denominados **agentes móveis**. Os agentes podem mover-se para o lugar onde os dados estão realmente armazenados e, utilizando inteligência, selecionar as informações que o usuário necessita. Com isso, a quantidade total de dados das mais diversas fontes não é transferida, economizando-se banda passante, tempo e dinheiro. Os agentes também facilitam a vida dos usuários com pequeno conhecimento de computação, funcionando como interfaces com o usuário "espertas" que podem detectar e resolver problemas e aprender as preferências e os hábitos do usuário, automatizando ações do cotidiano.

O modelo cliente-servidor tem sido utilizado há anos em sistemas distribuídos. Na maioria dos casos, o cliente e o servidor são executados em máquinas diferentes (figura 1), por conseqüência aumentando os esforços e os custos de desenvolvimento e de manutenção de clientes em plataformas diversas. A chamada remota de procedimentos *Remote Procedure Call* - RPC) é o mecanismo mais utilizado para a comunicação entre clientes e servidores. O modelo de programação remota é normalmente utilizado por agentes no lugar da RPC, a fim de melhorar o desempenho através da eliminação do tráfego intermediário na rede. Neste modelo, um agente representando um cliente pode ser transportado para um hospedeiro a fim de interagir com o(s) servidor(es) deste hospedeiro, realizando suas tarefas localmente (figura 2). Existem duas vantagens principais em se utilizar o modelo de programação remota no lugar da RPC. A primeira vantagem consiste no fato que somente resultados selecionados de cálculos ou pedidos complexos serão enviados, diminuindo o tráfego na rede pois não haveria necessidade de transmitir comandos, todos os dados e resultados intermediários. Outra vantagem é que como um agente sabe como acessar os dados que necessita utilizar no hospedeiro, para realizar sua tarefa, o hospedeiro pode ser simplificado pois não necessita necessariamente reconhecer uma linguagem complexa de pedidos ou saber como realizar cálculos complexos.

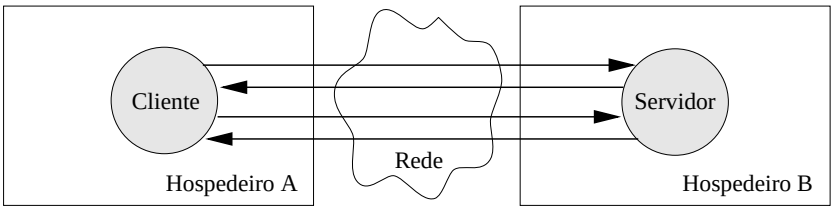

Figura 1: Modelo Cliente-Servidor.

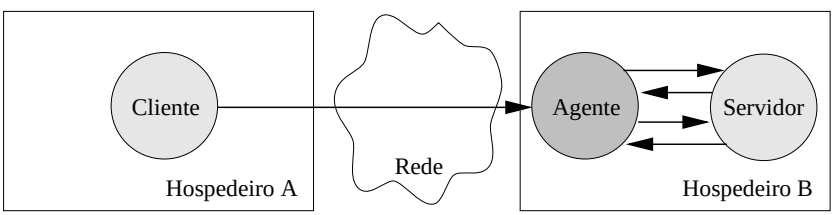

Figura 2: Modelo de Programação Remota.

Vários sistemas baseados em agentes estão disponíveis, como o TACOMA (*Tromsφ And COrnell Moving Agents*) [1], o Agent Tcl [2] e o Telescript [3]. No projeto TACOMA, foi desenvolvido um protótipo de plataforma de execução com um mecanismo flexível para a movimentação e a distribuição de agentes em redes de computadores. O Agent Tcl é um sistema de agentes transportáveis que dá suporte a várias linguagens e mecanismos de transporte, provê migração e comunicação transparentes e provê segurança efetiva e tolerância a falhas. O Telescript é uma linguagem orientada a objetos que contém facilidades para a construção de aplicações baseadas em agentes. Este sistema está sendo substituído pelo Odyssey [4] totalmente desenvolvido em Java. Outro sistema de agentes que utiliza Java como linguagem de implementação é o Aglets da IBM [5]. Esforços de padronização também estão sendo realizados pela FIPA (*Foundation for Intelligent Physical Agents*) [6], pela Agent Society [7] e pela OMG (*Object Management Group*) [8]. A FIPA é uma associação internacional de companhias e organizações criada para produzir especificações de tecnologias genéricas de agentes. A Agent Society é uma organização internacional industrial e profissional estabelecida para ajudar no desenvolvimento de tecnologias de agentes. A OMG está trabalhando de modo a estabelecer padrões industriais para a tecnologia de agentes móveis e interoperabilidade entre sistemas de agentes como o Odyssey e o Aglets.

Este trabalho propõe um ambiente de desenvolvimento de sistemas baseados em agentes denominado Infra-estrutura GTA/AG (GTA/AGente). As principais metas de projeto deste ambiente consistem em utilizar uma linguagem de programação de agentes portável, interpretada e facilmente integrável ao WWW e possibilitar flexibilidade em relação à serviços genéricos de migração, comunicação, execução, segurança e tolerância a falhas, agregando as principais vantagens dos sistemas de agentes já desenvolvidos. Por ser uma estrutura modular e flexível, é possível escolher e configurar os módulos de acordo com as características do sistema desejado, bem como criar novos módulos.

Este trabalho está organizado da seguinte maneira: na seção 2 são apresentados os agentes, na seção 3 é apresentada a estrutura dos agentes móveis, a seção 4 descreve a infra-estrutura GTA/AG e na seção 5 são apresentadas as considerações finais.

2 Agentes

O estudo de agentes engloba várias áreas tais como sistemas distribuídos, recuperação de informações, comércio eletrônico e inteligência artificial e com isso, existem muitas definições para agentes. Basicamente, a definição de agentes a ser utilizada neste trabalho consiste em programas que ajudam um usuário a realizar tarefas, agindo em interesse desse usuário.

2.1 Características de Agentes

Os agentes são encarregados de realizar tarefas e podem operar sobre requisitos fornecidos. Para executar estas tarefas, os agentes podem utilizar quatro capacidades: agência, inteligência, interface com o usuário e mobilidade [9].

A **agência** é o grau de autonomia e autoridade investida no agente, medida através de sua interatividade. Existem três níveis de agência: interatividade com o usuário, interatividade com a aplicação e interatividade com o agente. A interatividade com o usuário indica um agente auto-suficiente que interage somente com o usuário. A interatividade com a aplicação indica interação com o usuário e com as aplicações. A interatividade com o agente indica interação com o usuário, as aplicações e outros agentes.

A **inteligência** é o grau de raciocínio e comportamento aprendido: a habilidade do agente em aceitar as metas do usuário e realizar a tarefa a ele conferida. Existem

quatro níveis de inteligência: preferência, raciocínio, planejamento e aprendizado. No mínimo, deve existir alguma declaração de preferências, talvez na forma de regras, com uma máquina de inferência ou algum outro mecanismo de raciocínio para agir sobre estas preferências. Níveis maiores de inteligência incluem um modelo do usuário ou alguma outra forma de entender e raciocinar a respeito do que o usuário quer que seja feito. Além disso, o agente pode planejar os meios de alcançar este objetivo. Níveis mais altos de inteligência podem ser obtidos através de sistemas que aprendem e se adaptam ao seu ambiente, ambos em termos dos objetivos do usuário e dos recursos disponíveis ao agente.

A **interface com o usuário** é utilizada quando existe uma interação direta do usuário com o agente.

A **mobilidade** corresponde ao grau de movimento dos agentes através da rede. Existem dois níveis de mobilidade: fixa e móvel. Um agente pode ser fixo, residindo na máquina cliente (por exemplo, para gerenciar uma interface com o usuário) ou instanciado no servidor. Um agente pode ser móvel: criado em uma máquina e enviado para outra para execução em um ambiente seguro. Um agente móvel pode ser transportado de uma máquina para outra no meio de sua execução e carregar com ele dados acumulados sobre o seu estado.

2.2 Agentes Móveis

Os agentes móveis, também chamados itinerantes, são agentes criados para executar suas tarefas, para seus usuários, em diferentes computadores da rede. Estes agentes podem viajar para agências na quais irão apresentar suas credenciais e obter acesso a serviços e dados gerenciados pelas agências. As agências podem servir como "corretores" ou "casamenteiros", agrupando agentes com interesses similares e metas compatíveis e provendo um ponto de encontro no qual os agentes possam interagir de modo seguro [10]. A mobilidade de agentes conduz a uma extensão do modelo cliente-servidor: clientes enviam partes deles mesmos para o servidor (ou servidores) para execução.

Várias vantagens podem justificar a utilização de agentes móveis em certas aplicações, tais como [11]:

- redução do custo: caso uma aplicação exija uma grande quantidade de informações a ser veiculada pela rede, pode ser mais econômico enviar um agente que execute a tarefa onde estão localizadas estas informações. Desta forma, o agente pode filtrar e selecionar somente as informações relevantes e transferí-las para a sua origem;
- processamento assíncrono: um agente pode ser enviado através da rede, continuando suas tarefas em outros nós. Enquanto o agente se encontra fora de seu nó de origem, não é necessário que o nó permaneça em operação. Este processamento assíncrono pode ser utilizado na computação móvel;
- divisão de processamento: computadores com pequena capacidade de processamento podem ser utilizados de maneira eficiente para armazenar e processar dados que são filtrados e enviados por agentes móveis a partir de grandes servidores de informações com maior capacidade de processamento;
- flexibilidade: agentes móveis se contituem como uma alternativa para o modelo cliente-servidor, podendo também complementá-lo.

3 Estrutura dos Agentes Móveis

Os agentes móveis consistem de código, estado e atributos. O **código** de um agente móvel é um programa que define o comportamento do agente. Obviamente um código de um agente pode ser escrito em qualquer linguagem, porém um agente deve poder ser executado de uma mesma maneira em qualquer hospedeiro para o qual possa se mover

[10]. Para isto, as linguagens que são interpretadas diretamente ou compiladas para uma linguagem intermediária baseada em interpretador que possa ser facilmente transportada e executada sem recompilação são as mais adequadas. Como exemplos, tem-se: Tcl, Perl e Java.

O ambiente proposto provê suporte à linguagem Java. Esta linguagem apresenta um melhor desempenho em relação às linguagens interpretadas, é portável e está tendo grande utilização na comunidade da Internet [12]. O atual interesse em sistemas de agentes móveis é devido quase que inteiramente à grande adoção de Java [13]. Outras vantagens de Java em relação a outras linguagens de implementação de agentes são a integração com a infra-estrutura WWW e as bibliotecas de suporte sofisticadas e padronizadas [14].

Além do código, um agente carrega um **estado** que permite que suas ações subseqüentes sejam baseadas em suas ações passadas. Com isto, um agente pode retomar o trabalho de onde parou, após mudar para outro hospedeiro. O estado de um agente contém o ponto de execução no novo hospedeiro e as variáveis necessárias para a sua execução. O *Java Development Kit* (JDK 1.1) [15] contém um mecanismo chamado *Object Serialization* que suporta a codificação de objetos em um fluxo de objetos e a remontagem do objeto a partir deste fluxo. De modo a armazenar e recuperar objetos, o estado dos objetos é representado de uma forma serializada, suficiente para reconstruir os objetos, e todos os objetos referenciados por um objeto são armazenados e recuperados ao mesmo tempo, a fim de se manter as relações existentes entre os objetos. Neste ambiente, o estado de um agente é implicitamente capturado quando da migração de um agente para outro hospedeiro, através do mecanismo *Object Serialization*, como no serviço implementado por Teixeira e Junqueira [12]. De maneira diferente, no TACOMA um programador precisa capturar explicitamente o estado de um agente.

A terceira parte de um agente móvel consiste de **atributos** que descrevem o agente, seus requisitos e sua história para a infra-estrutura. Isto inclui dados como um identificador único de agente, o dono do agente como um endereço destinatário para resultados intermediários, mensagens de erro ou reclamações quanto a um mau comportamento de um agente, o tempo e o lugar de origem e a história do movimento. Os atributos também impoẽm limitações na mobilidade do agente, tais como datas de expiração, "fique dentro do domínio `ufrj.br`" e outras (seção 4.3.3). Os agentes também expressam seus requisitos de recursos (seção 4.3.3), tais como a quantidade de memória necessária, limitações (por exemplo "R$ 1,00 deixado para pagar por serviços de recuperação de informações") e requisitos para o ambiente no qual necessitam executar suas tarefas (como "acesse um banco de dados bibliográficos de acordo com os padrões do `GTA`"). Um agente carrega informação de autenticação que deixa a infra-estrutura verificar as credenciais de qualquer usuário que queira parar ou chamar novamente o agente ou verificar seu progresso, assim como checar quaisquer senhas ou chaves que o agente necessite para obter acesso a informações restritas, se necessário. Parte destes atributos é acessível ao agente, porém o agente não é capaz de modificar seus atributos.

4 Infra-estrutura GTA/AG

Os agentes devem interagir com seus hospedeiros para utilizar seus serviços e negociar serviços com outros agentes. Os agentes móveis devem ainda podem se mover em redes de computadores. O ambiente que permite estas ações é denominado infra-estrutura de agentes e a entidade que provê suporte para os agentes em um hospedeiro particular é chamada servidor de agentes. Além destes serviços básicos, uma infra-estrutura de agentes deve também prover segurança a fim de proteger o hospedeiro (e os agentes) de acessos não autorizados e espionagem (seção 4.3.3).

Neste trabalho é proposto um ambiente de desenvolvimento de sistemas baseados em agentes denominado GTA/AG (GTA/AGente). Esta infra-estrutura visa prover serviços genéricos como migração, comunicação, execução, segurança e tolerância a falhas, permitindo a um desenvolvedor de aplicações baseadas em agentes focar-se nos detalhes das aplicações, sem se preocupar com a infra-estrutura. Por ser uma estrutura modular e flexível, é possível escolher e configurar os módulos de acordo com as características do sistema desejado, bem como criar novos módulos. De acordo com Lingnau et al. [10], uma infra-estrutura geral para agentes móveis deve prover mecanismos, não policiamento, para ser aceita de um modo geral.

A maioria das infra-estruturas existentes, enquanto diferindo consideravelmente na realização, usam a mesma solução básica para portabilidade e segurança: elas não executam os agentes na máquina real de processador, memória e sistema operacional, mas em uma máquina virtual, usualmente um interpretador e um sistema em tempo de execução, os quais escondem os detalhes da arquitetura do sistema hospedeiro assim como confinam as ações dos agentes ao ambiente restrito [16].

A base do modelo de arquitetura utilizado para a infra-estrutura de agentes (figura 3) é o **servidor de agentes** (seção 4.3). Este servidor está encarregado de prover diversas funções como transporte de agentes, segurança, comunicações de agentes com o hospedeiro, com outros agentes e com seus donos, e armazenamento persistente. Um ambiente apropriado para a execução de um agente, denominado **ambiente em tempo de execução** (seção 4.2), é a interface entre um agente e o seu hospedeiro. Suas principais funções são proteger o hospedeiro de agentes, capturar o estado de um agente e interagir com o servidor. O usuário (dono) de um agente interage com a infra-estrutura através de um **cliente**.

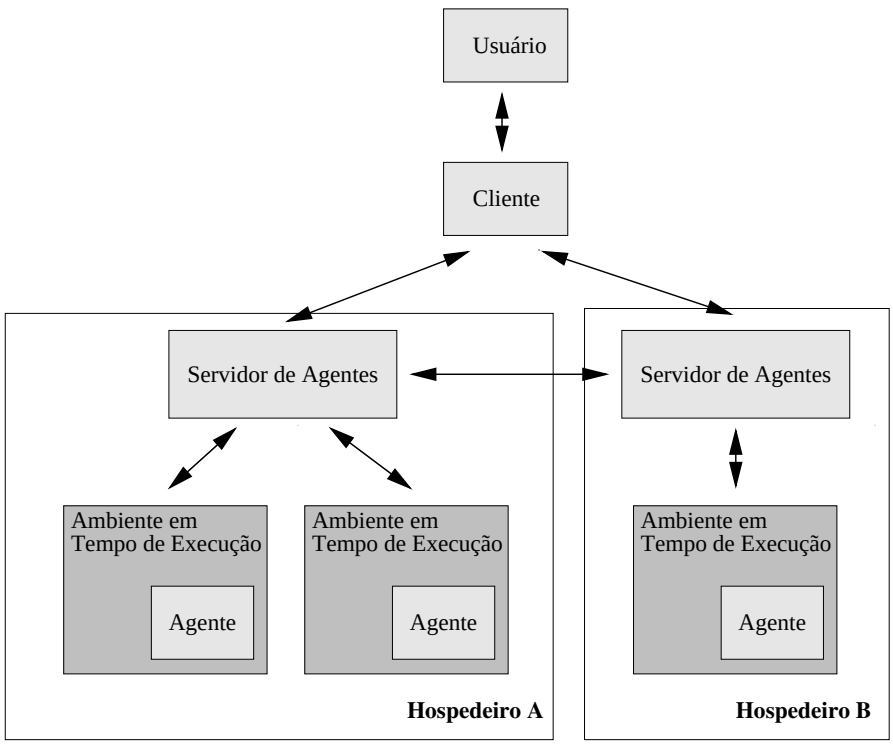

Figura 3: Modelo de Arquitetura.

4.1 Cliente

O cliente deixará o usuário submeter um agente para execução, descobrir seu estado, parar ou chamar novamente um agente e realizar outras operações, se necessário. O cliente não necessita de uma conexão permanente com o resto da infra-estrutura de agentes, pois enquanto o agente se encontra fora de seu hospedeiro origem, não é necessário que o hospedeiro permaneça em operação. Este processamento assíncrono pode ser utilizado em computação móvel [17].

A comunicação entre os agentes e seus usuários na infra-estrutura GTA/AG é realizada através de correios eletrônicos ou interfaces gráficas. No caso da utilização de correios eletrônicos, este modo de comunicação assíncrono se encaixa bem ao fato que o usuário de um agente pode estar desconectado ou utilizando um terminal móvel. As interfaces gráficas podem ser escritas em Tcl/Tk.

4.2 Ambiente em Tempo de Execução

A infra-estrutura de agentes deve prover suporte em tempo de execução para os agentes por várias razões [10]:

- necessidade de serviços de mobilidade e comunicação, com isso, operações apropriadas devem estar disponíveis na linguagem de programação do agente. Devem ser providas operações que deixem o agente se mover para outro servidor e comunicar resultados intermediários ao seu dono;
- os agentes não são supostos fazer má utilização do acesso ao sistema hospedeiro, por exemplo, formatando um disco rígido do sistema. A melhor maneira de forçar este tipo de segurança é desabilitar este tipo de operação em uma primeira instância ou controlá-la bem de perto, de modo que os argumentos de operações não confiáveis sejam cuidadosamente examinados e as operações rejeitadas, caso possam vir a danificar o sistema;
- necessidade de funcionalidades orientadas ao problema do agente. Por exemplo, um agente cujo objetivo consiste em realizar uma pesquisa bibliográfica poderia requisitar acesso a uma ferramenta de procura utilizando a mídia texto, ferramenta que seria provida pelos fornecedores de banco de dados para que o agente não precisasse carregá-la consigo.

Para cada agente sendo executado em um servidor, existe um ambiente em tempo de execução dedicado criado pelo servidor. Com isto, um agente não tem acesso direto a outros agentes que estão sendo executados no mesmo hospedeiro. O ambiente é a interface entre o agente e seu hospedeiro e disponibiliza recursos ao agente de modo controlado.

O ambiente em tempo de execução dá suporte à linguagem Java através de um pacote em tempo de execução escrito na mesma linguagem. O interpretador está dividido em três blocos: segurança, estado e API para o servidor (figura 4). O bloco de segurança evita que um agente tome ações maléficas, através do controle de todas as ações dos agentes, o bloco de estado é responsável por capturar o estado de um agente, enquanto que o bloco API para o servidor interage com o servidor para lidar com migração, comunicação e verificação.

4.3 Servidor de Agentes

O servidor de agentes é um programa (como um servidor de correio eletrônico ou um servidor de FTP) que é executado em cada computador que será acessível ao agentes e está encarregado dos agentes que estão sendo executados neste computador. Suas tarefas incluem aceitar agentes, criar o ambiente de execução apropriado, supervisionar as

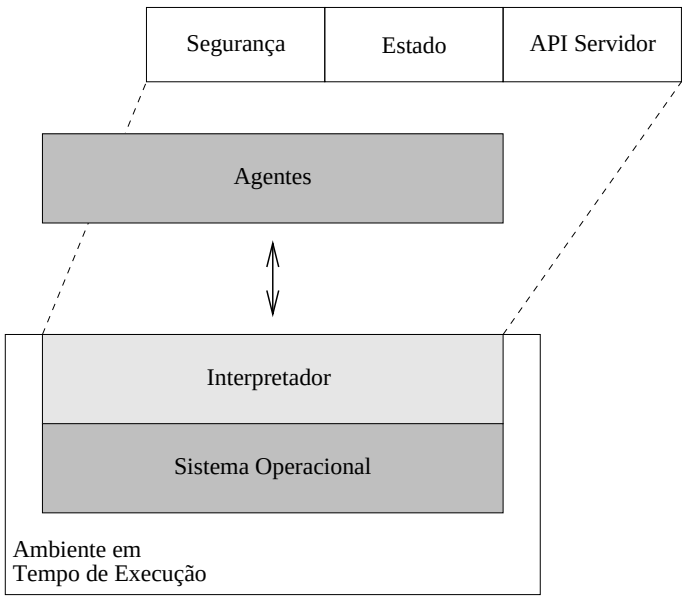

Figura 4: Ambiente em Tempo de Execução.

execuções dos agentes (no meio tempo respondendo pedidos sobre seus estados) e finalizá-las quando necessário. O servidor de agentes também deve organizar a transferência de agentes para outros hospedeiros, gerenciar comunicações entre agentes e entre agentes e seus donos, autenticar e controlar o acesso para todas as operações de agentes e recuperar agentes e as informações carregadas por eles no caso de falhas em computadores e redes.

As funções do servidor de agentes (figura 5) estão divididas em módulos: Módulo de Transferência (T), Módulo de Comunicação (C), Módulo de Segurança (S) e Módulo de Tolerância a Falhas (TF).

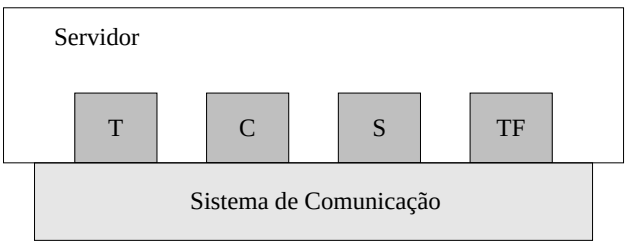

Figura 5: Módulos do Servidor de Agentes.

4.3.1 Módulo de Transferência

Uma das características principais de agentes móveis é seu poder de se mover de um hospedeiro para outro. A diferença entre este tipo de mobilidade e a encontrada em, por exemplo, sistemas de balanceamento de carga que permitem a realocação de processos é que o movimento acontece de acordo com um pedido explícito do agente ao invés de ocorrer de acordo com alguma decisão do sistema operacional (sem o agente notar) [10]. Isto é, a linguagem de implementação do agente inclui uma sentença ou chamada de biblioteca da forma `MOVA-PARA hospedeiro`. Quando esta chamada é executada, o agente é suspenso,

codificado para transmissão, transmitido para o novo hospedeiro, decodificado e instalado para retomar a execução neste novo local, preservando o estado atual e a integridade do agente.

O JDK 1.1 contém um utilitário de arquivamento chamado JAR (*Java ARchive*) que trabalha com um formato de arquivo independente da plataforma utilizada e agrega vários arquivos em um só, suportando compressão. Neste ambiente, a codificação e a decodificação de agentes são realizadas através do JAR. A transmissão de agentes é realizada através do protocolo de grande utilização na Web, o HTTP (*HyperText Transfer Protocol*), por este protocolo estar intimamente ligado à Java.

4.3.2 Módulo de Comunicação

Existem muitas situações onde é mais eficiente comunicar de lugar para outro do que migrar o agente chamador para o lugar onde o agente chamado está [18]. Para realizar trabalhos que tenham alguma utilidade, os agentes devem poder se comunicar com o sistema hospedeiro, com outros agentes e com seus donos. O protocolo HTTP é utilizado para a comunicação entre os agentes.

A comunicação entre os agentes e o sistema hospedeiro também é realizada através do protocolo HTTP. A comunicação entre os agentes e seus donos foi apresentada na seção 4.1.

4.3.3 Módulo de Segurança

Talvez o tópico mais importante para a completa aceitação de sistemas baseados em agentes seja a segurança. Em um sistema de redes de computadores, nem o agente nem as máquinas são "confiáveis" [19]. Muitas pessoas associam agentes movéis às ameaças impostas por programas maliciosos como o vírus e os cavalos de Tróia. Um agente pode atacar uma máquina e ganhar acesso a recursos locais [19]. Deve-se evitar que um servidor adultere os agentes a seu cargo ou mesmo olhe o código de um agente que implementa, por exemplo, uma estratégia de negociação de um competidor.

Criptografia baseada em chave pública RSA é utilizada durante a migração de agentes para protegê-los contra espionagem. A versão 1.2 do JDK, ainda não disponível, incluirá extesões para criptografia através das *Java Cryptography Extensions* [15]. O transmissor de um agente é autenticado através do JAR, utilizando a assinatura digital do dono do agente. Um agente pode ser admitido ou não em um hospedeiro, conforme a verificação de autorização do transmissor de agente e a necessidade de recursos como disco, CPU e memória. O núcleo do interpretador (seção 4.2) controla o acesso aos recursos e a memória do hospedeiro é protegida pela utilização dos interpretadores. A integridade de um agente é verificada também através de chave pública RSA (em um futuro próximo serão utilizadas as *Java Cryptography Extensions*). A memória não volátil, utilizada no Módulo de Tolerância a Falhas (seção 4.3.4), também é protegida. De maneira diferente, o Agent Tcl provê uma segurança rudimentar na qual qualquer agente, mensagem ou conexão que não vier de uma "máquina aprovada" é ignorada. O administador do sistema especifica o conjunto de máquinas aprovadas. O TACOMA provê mecanismos de segurança através de um agente *firewall* que só relata em um disco a interação de um agente com o *firewall* quando da execução da instrução `encontro` e não acessa diretamente processador, memória, sistemas de arquivos ou periféricos.

4.3.4 Módulo de Tolerância a Falhas

Mecanismos de recuperação de agentes e das informações que carregam são utilizados no caso de falhas em computadores e redes, de modo que a execução possa voltar ao

normal o mais rápido possível, assim como descrito em Oliveira et al. [20]. Persistência é utilizada, onde agentes, seus dados e seus estados são armazenados de tempos em tempos em memórias não voláteis. Este armazenamento baseia-se em pontos de checagem, normalmente pontos críticos na execução de um agente. Quando um destes pontos é alcançado, as informações descritas anteriormente são salvas na memória não volátil. No Agent Tcl somente os agentes e seus estados são salvos, enquanto que o TACOMA só salva o estado do agente em um disco logo que o agente chega.

5 Considerações Finais

As principais aplicações de agentes móveis são em recuperação de informações, gerenciamento de redes, comércio eletrônico e computação móvel. Nenhuma destas aplicações requer necessariamente a utilização de agentes móveis, porém caso isto não seja feito, haverá um aumento substancial na carga do sistema (e da rede) e, possivelmente, existirá inconveniência para os usuários. O desenvolvimento de uma infra-estrutura que sirva de base para a comunicação entre os agentes e os hospedeiros e para a mobilidade dos agentes facilita um grande número de serviços e aplicações em redes.

Neste trabalho foi proposto um ambiente de desenvolvimento de sistemas baseados em agentes denominado GTA/AG (GTA/AGente). Esta infra-estrutura provê serviços genéricos de migração, comunicação, execução, segurança e tolerância a falhas, permitindo a um desenvolvedor de aplicações baseadas em agentes focar-se nos detalhes das aplicações, sem se preocupar com a infra-estrutura. Por ser uma estrutura modular e flexível, é possível escolher e configurar os módulos de acordo com as características do sistema desejado, bem como criar novos módulos.

A base do modelo de arquitetura utilizado para a infra-estrutura de agentes é o servidor de agentes. Este servidor provê diversas funções como transporte de agentes, segurança, comunicações de agentes com o hospedeiro, com outros agentes e com seus donos, e armazenamento persistente. Um ambiente apropriado para a execução de um agente, denominado ambiente em tempo de execução, é a interface entre um agente e o seu hospedeiro. O usuário (dono) de um agente interage com a infra-estrutura através de um cliente.

Como continuação deste trabalho, os módulos de comunicação e de transferência serão estendidos. Em relação à comunicação, a forma de interação agente-servidor e agente-agente será especificada, enquanto que na transferência, o problema do intinerário de lugares a ser seguido por um agente será abordado. Além disso, diversas aplicações serão analisadas de modo a ser feita a escolha de uma aplicação alvo na qual o modelo de programação remota será empregado.

Referências

[1] D. Johansen, R. van Renesse e F. B. Schneider, "An introduction to the TACOMA distributed system", relatório técnico, University of Tromsϕ/Cornell University, Tromsϕ/Ithaca, Noruega/EUA, junho de 1995.

[2] R. S. Gray, "Agent Tcl: a transportable agent system", in *CIKM Workshop on Intelligent Information Agents, Fouth International Conference on Information and Knowledge Management (CIKM 95)*, Baltimore/Maryland, EUA, 1995.

[3] J. White, "Mobile agents white paper", relatório técnico, General Magic, 1996.

[4] "Odyssey". General Magic, disponível em http://www.genmagic.com/agents, 1997.

[5] "Programming mobile agents in Java". IBM Corporation, IBM do Japão, disponível em http://www.trl.ibm.co.jp/aglets, 1996.

[6] L. Chiariglione, "Helping agent technologies get across to the market place". Foundation for Intelligent Physical Agents (FIPA), disponível em http://drogo.cselt.stet.it/fipa/general_agents.htm.

[7] "The Agent Society". Disponível em http://www.agent.org, 1997.

[8] "The mobile agent facility specification". Disponível em http://www.genmagic.com/agents/MAF.

[9] D. Gilbert e P. Janca, "IBM intelligent agents". IBM Corporation, EUA, disponível em http://www.networking.ibm.com/iag/iagwp1.html.

[10] A. Lingnau e O. Drobnik, "An infrastructure for mobile agents: requirements and architecture". Fachbereich Informatik (Telematik), Johann Wolfgang Goethe-Universität, Frankfurt, Alemanha, disponível em ftp://ftp.tm.informatik.uni-frankfurt.de/agents/13dis-paper.ps.gz.

[11] H. S. Nwana, "Software agents: an overview", *Knowledge Engineering Review*, vol. 11, no. 3, pp. 1–40, setembro de 1996.

[12] R. C. Teixeira e F. P. Junqueira, "Implementação de serviço baseado em agentes móveis", relatório técnico, Grupo de Teleinformática e Automação, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, agosto de 1997.

[13] J. Kiniry e D. Zimmerman, "A hands-on look at Java mobile agents", *IEEE Internet Computing*, vol. 1, no. 4, pp. 21–30, julho de 1997.

[14] B. Liberman, F. Griffel, M. Merz e W. Lamersdorf, "Java-based mobile agents - how to migrate, persist, and interact on electronic service markets", in *1st International Workshop on Mobile Agents*, Berlin, Alemanha, pp. 27–38, abril de 1997.

[15] "Java development kit". Sun Microsystems, disponível em http://java.sun.com/products/jdk/1.1, 1998.

[16] H. Peine e T. Stolpmann, "The architecture of the Ara platform for mobile agents", in *1st International Workshop on Mobile Agents*, Berlin, Alemanha, pp. 50–61, abril de 1997.

[17] M. G. Rubinstein e O. C. M. B. Duarte, "Utilizando o paradigma de programação remota em aplicações multimídias distribuídas", relatório técnico, Grupo de Teleinformática e Automação, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, julho de 1997.

[18] J. Baumann, F. Hohl, N. Radouniklis, K. Rothermel e M. Straβer, "Communication concepts for mobile agent systems", in *1st International Workshop on Mobile Agents*, Berlin, Alemanha, pp. 123–135, abril de 1997.

[19] M. Abdalla, W. Cirne, L. Franklin e A. Tabbara, "Security issues in agent based computing", in *XV Simpósio Brasileiro de Redes de Computadores (SBRC 97)*, São Carlos, SP, Brasil, pp. 231–244, maio de 1997.

[20] P. C. de Oliveira e E. Cardozo, "Mobile agent-based systems: an alternative paradigm for distributed systems development", in *XV Simpósio Brasileiro de Redes de Computadores (SBRC 97)*, São Carlos, SP, Brasil, pp. 508–524, maio de 1997.