Universidade Estadual de Campinas – UNICAMP
Faculdade de Engenharia Elétrica e de Computação – FEEC
Departamento de Engenharia de Computação e Automação Industrial – DCA

Sistemas Baseados em Agentes Móveis: uma Abordagem Alternativa para o Desenvolvimento de Sistemas Distribuídos

Paulo César de Oliveira

Orientador: Prof. Dr. Eleri Cardozo (FEEC – UNICAMP)

Dissertação de Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como requisito parcial para a obtenção do título de Mestre.

Dezembro de 1997

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

OL4s

Oliveira, Paulo César de
Sistemas baseados em agentes móveis: uma abordagem alternativa para o desenvolvimento de sistemas distribuídos. / Paulo César de Oliveira.-- Campinas, SP: [s.n.], 1997.

Orientador: Eleri Cardozo
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Software - Desenvolvimento. 2. Processamento eletrônico de dados - Processamento distribuído. I. Cardozo, Eleri. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Comissão Julgadora

Dr. Eleri Cardozo – DCA/FEEC/UNICAMP

Dr. Michel Daoud Yacoub – DECOM/FEEC/UNICAMP

Dr. Kleber Xavier Sampaio de Souza – CNPTIA/EMBRAPA

Dr. Fernado Antonio Campos Gomide – DCA/FEEC/UNICAMP (Suplente)

Publicações

L.A.G. Oliveira, P.C. Oliveira, E. Cardozo, "An Agent-Based Approach for Quality of Service Negotiation and Management in Distributed Multimedia Systems", *Proc. of the 1st International Workshop on Mobile Agents*, Berlin, Alemanha, 1997.

P. C. Oliveira, E. Cardozo, "Mobile Agent-Based Systems: an Alternative Paradigm for Distributed Systems Development", *Anais do 15.º Simpósio Brasileiro de Redes de Computadores*, São Carlos, Brasil, 508-524, 1997.

L.A.G. Oliveira, P.C. Oliveira, L.F. Faina, E. Cardozo, "QoS Agency: An Agent-based Architecture for Supporting Quality of Service in Distributed Multimedia Systems", *Proc. of the IEEE Conference on Protocols for Multimedia Systems – Multimedia Networking*, Santiago, Chile, 1997

Agradecimentos

A Deus por ter me concedido capacidade, força e perseverança para concluir este trabalho. Sem a sua graça e suas bençãos, dadas por meio de Jesus, nada seria possível em minha vida. A Ele seja o reconhecimento e o agradecimento mais profundo do meu coração.

Aos meus pais, pelo incentivo e pelos esforços dispensados de forma constante para prover todos os meios necessários à minha educação. Também os agradeço pelo exemplo de vida que sempre deram a nós, seus filhos, fundamentado em valores como respeito, honestidade e humildade.

Ao prof. Dr. Eleri Cardozo pela valiosa orientação e pelo apoio na condução de todo o trabalho de mestrado.

À Empresa Brasileira de Pesquisa Agropecuária, em especial ao CNPTIA, cujo suporte foi de grande importância para a realização deste trabalho.

Aos amigos Affonso, Faina e Rodrigo pelas experiências de trabalho em equipe que produziram frutos expressivos em nossa vida pessoal e profissional.

À Stella, pelo brilho especial e pela alegria que sua presença trouxe à minha vida.

A toda a minha família, em especial à minha avó Laura, pelo carinho e amor dedicados a mim durante toda a minha vida.

Ao Kleber pelo acompanhamento do trabalho de mestrado realizado junto à Embrapa.

Aos amigos da Embrapa, em especial a Fernando, Stanley, Evandro, Carla e Luciana pelo companheirismo e apoio.

"Filho meu, não te esqueças dos meus ensinos, e o teu coração guarde os meus mandamentos;

porque eles aumentarão os teus dias, e te acrescentarão anos de vida e paz.

Não te desamparem a benignidade e a fidelidade; ata-as ao teu pescoço; escreve-as na tábua do teu coração.

E acharás graça e boa compreensão diante de Deus e dos homens.

Confia no SENHOR de todo o teu coração, e não te estribes no teu próprio entendimento.

Reconhece-o em todos os teus caminhos, e ele endireitará as tuas veredas.

Não sejas sábio aos teus próprios olhos: teme ao SENHOR e aparta-te do mal;

Será isto saúde para o teu corpo, e refrigério para os teus ossos.

Filho meu, não rejeites a disciplina do SENHOR, nem te enfades da sua repreensão.

Porque o SENHOR repreende a quem ama, assim como o pai ao filho a quem quer bem."

Provérbios de Salomão (Bíblia Sagrada), capítulo 3, versos de 1 a 12

Sumário

Os termos *agente* e *sistema baseado em agentes* têm sido amplamente utilizados em diversas áreas da computação, havendo uma grande diversidade no que diz respeito à sua definição. Este trabalho se concentra em agentes de software (chamados apenas de agentes), e os define como programas autônomos que recebem autoridade de seus donos para agir em favor deles na execução de tarefas. Visando completar as tarefas que lhes foram atribuídas, agentes podem se comunicar com outros programas, com o ambiente que os abriga ou com seres humanos. Sistemas baseados em agentes são sistemas de software compostos por agentes.

Agente móvel é um tipo particular de agente que têm a capacidade de se mover numa rede de computadores heterogênea a fim de realizar as tarefas delegadas por seu dono. A abordagem de sistemas baseados em agentes móveis (SBAMs) constitui-se como uma alternativa potencialmente singular para o desenvolvimento de sistemas em rede – ou distribuídos. Entretanto, trata-se de uma abordagem emergente e de recente concentração de esforços de pesquisa e desenvolvimento. Desta forma, há um número muito reduzido de experiências de utilização de SBAMs e infra-estruturas para suporte a SBAMs estão em fase inicial de construção.

Esta dissertação visa explorar sistemas baseados em agentes móveis como abordagem de desenvolvimento de sistemas distribuídos. O domínio de qualidade de serviços em sistemas multimídia distribuídos foi escolhido como estudo de caso para avaliar se as potencialidades de tal abordagem se mostram verdadeiras na prática. Assim sendo, um conjunto de infra-estruturas para suporte a SBAMs foi examinado e um SBAM para monitoramento de qualidade de serviço foi desenvolvido. Considerações a respeito da experiência de desenvolvimento e da aplicação do SBAM desenvolvido em casos de teste concluem esta dissertação.

Abstract

The terms *agent* and *agent-based system* are widely used in many areas of the computer science field, with no consensus definition for them. On the contrary, there is a diversity of understanding about the meaning of these terms. This work focuses on software agents (therein called as agents), and defines them as autonomous programs that act on behalf of their owners. In order to accomplish tasks assigned by their owners, agents may communicate with other programs, with their hosting environment, and with human beings. Agent-based systems are software systems composed by agents.

Mobile agent is a particular type of agent which has the ability to move across a heterogeneos computer network aiming to accomplish tasks delegated by its owner. The mobile agent-based systems (MABS) approach constitutes a potentially unique alternative for networked (or distributed) systems development. However, it is an emerging approach, and research and development efforts have been recently focused on it. Thus, there is a reduced number of experiences regarding the deployment of this approach, and frameworks for supporting MABS are in early phases of development.

This dissertation aims to explore mobile agent-based systems as an approach for distributed systems development. The domain of quality of service in distributed multimedia systems was chosen as a case study, in order to evaluate if the potential of MABS is accomplished in real contexts. Thus, a set of of frameworks for supporting MABS was examined and a MABS for quality of service monitoring was developed. Considerations regarding this development experience and the employment of the developed MABS in test cases conclude this dissertation.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1. Motivação

O desenvolvimento contínuo na área de redes de computadores tem promovido a utilização crescente de sistemas em rede – ou distribuídos. No início da década de 90, a automação de processos nas empresas se baseou em redes corporativas, sistemas de informação distribuídos e computação para suporte a grupos de trabalho[1]. No mesmo período, porém fora do domínio empresarial, um fenômeno de enorme proporção emergiu: a rede Internet. Este fenômeno se constitui correntemente como a grande alavanca para a popularização de redes de computadores e sistemas em rede, visto que a Internet é utilizada tanto por usuários em suas casas quanto por empresas, como instrumento para acesso a informações dos mais variados tipos, tornadas disponíveis em amplitude mundial. Além disso, a recente abertura da Internet para fins comerciais a tornou um meio para mercado eletrônico.

O crescente e amplo uso de sistemas em rede tem levado à consolidação de uma demanda por paradigmas de desenvolvimento que atendam requisitos como:

- rápido desenvolvimento e rápida manutenção: constantes mudanças na informática e nos requisitos de usuários requerem que sistemas sejam desenvolvidos e evoluídos em períodos de tempo reduzidos;
- independência de plataforma: sistemas devem executar em diversas plataformas de hardware e sistemas operacionais. Devem ainda ser desenvolvidos uma única vez e utilizados transparentemente nas plataformas desejadas;
- facilidade de integração: padrões abertos devem ser adotados a fim de que sistemas possam interagir facilmente com outros sistemas;
- facilidade de uso: sistemas devem prover aos seus usuários facilidade de operação e também personalização de funções (adaptação às preferências de usuários);

[1] *"workgroup computing"*

- flexibilidade: sistemas devem se basear em arquiteturas flexíveis nas quais seus componentes tenham interfaces e comportamento bem definidos. Desta forma, o esforço e os custos de evolução se tornam reduzidos. Além disso, componentes não devem assumir apenas papéis fixos (tais como clientes e servidores, produtores e consumidores);

- eficiência: recursos computacionais e de rede devem ser utilizados eficientemente;

- suporte à computação móvel: computadores móveis e dispositivos leves normalmente se conectam a uma rede de forma intermitente e em intervalos de tempo reduzidos para ter acesso a serviços remotos. Além disso, seus recursos computacionais são geralmente limitados. Sistemas em execução nestes tipos de equipamento devem explorar o máximo possível a capacidade de processamento de nós da rede onde residem os provedores de serviços. Também o número de conexões com aqueles nós, bem como o volume de dados transferidos (respostas à requisições de serviços) durante estas conexões, devem ser otimizados.

A partir do final da década de 80, o então emergente modelo Cliente/Servidor foi amplamente adotado para o desenvolvimento de sistemas distribuídos. Neste modelo, sistemas são particionados em componentes (clientes e servidores) que assumem papéis fixos. Na maioria dos casos, componentes do tipo cliente são específicos para uma determinada plataforma, e implementados separadamente para cada ambiente de execução. Desta forma, o esforço e os custos de desenvolvimento e evolução de sistemas se tornam elevados.

O mecanismo de Chamada de Procedimento Remoto[2] (RPC) é o mais utilizado para a comunicação entre componentes de sistemas Cliente/Servidor (figura 1.1). Da mesma forma que a chamada de procedimento local, RPC se baseia em conexões síncronas entre um cliente e um servidor, isto é, o processo cliente tem sua execução suspensa na chamada do procedimento remoto até que o retorno seja recebido. Assim sendo, recursos permanecem alocados mesmo quando não são utilizados (ineficiência de utilização) e custos de comunicação se tornam maiores do que o necessário.

[2] *"Remote Procedure Call"*

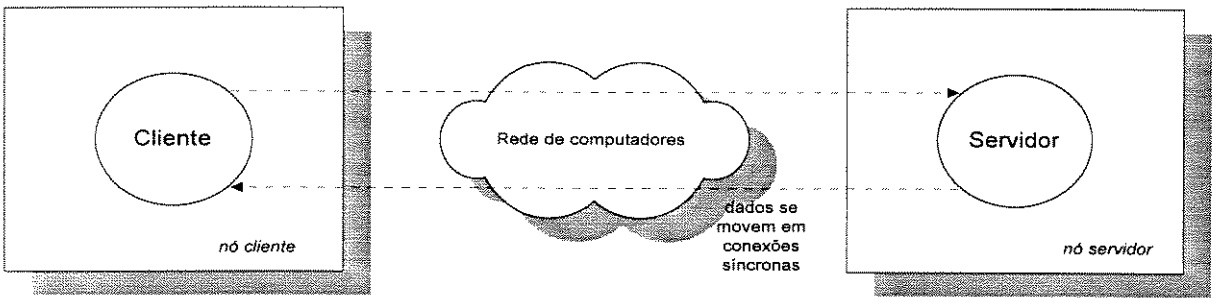

Figura 1.1: RPCs para comunicação entre componentes de sistemas no modelo Cliente/Servidor

A abordagem baseada em agentes de software[3] constitui-se numa alternativa para o desenvolvimento de sistemas distribuídos. Embora os termos *agente* e *sistema baseado em agentes* (SBA) sejam amplamente utilizados em diversas áreas da computação, não há uma definição de consenso para o seu significado.

Agentes são considerados nesta dissertação como programas autônomos que recebem autoridade de seus donos (usuários ou outros programas) para agir em favor destes na execução de tarefas. Visando completar as tarefas que lhes foram atribuídas, agentes podem se comunicar com outros programas, com o ambiente que os abriga ou com seres humanos. A definição adotada se concentra nas seguintes características fundamentais de agentes:

- autonomia: um agente pode tomar decisões próprias, baseando-se nas metas, preferências e políticas definidas por seus donos ou a eles associadas;
- delegação: usuários ou outros programas podem delegar tarefas a agentes e revesti-los com autoridade para agir em seu favor;
- comunicação: habilidade que agentes têm de interagir com outros programas (agentes ou não), com ambientes que os abrigam e com seres humanos;
- flexibilidade: agentes têm interfaces e comportamento bem definidos, e não assumem papéis fixos; eles podem agir como clientes, servidores, observadores, etc., dependendo de suas necessidades correntes;
- eqüidade: agentes comportam-se como pares; não há relações de hierarquia entre eles.

[3] no escopo deste trabalho serão chamados apenas de agentes

As características apontadas como fundamentais nesta dissertação refletem a visão de que um sistema de software baseado em agentes é composto por entidades (agentes) autônomas que executam tarefas a elas atribuídas (delegadas) por seus donos. Cada entidade é vista como par pelas outras entidades que compõem o sistema e exerce papéis flexíveis com interfaces e comportamento bem definidos. Modularidade e encapsulamento são enfatizados através de agentes. Além disso, paralelismo pode ser explorado, visto que agentes são entidades autônomas.

Embora não consideradas como fundamentais, as características abaixo estão extremamente relacionadas a agentes:

- cooperação: agentes podem agir de forma colaborativa, visando atingir metas comuns;
- inteligência: habilidade apresentada por agentes de raciocínio e aprendizado a partir de interações com seres humanos, outros programas e ambientes que os abrigam;
- mobilidade: agentes podem se mover numa rede de computadores heterogênea, visando progressivamente completar tarefas que lhes foram atribuídas.

A característica de mobilidade, aliada às características fundamentais, torna *sistemas baseados em agentes móveis* (SBAMs) uma abordagem singular para o desenvolvimento de sistemas distribuídos. Diferentemente do modelo Cliente/Servidor baseado em RPC, em SBAMs agentes (compostos por dados e código) se movem na rede, o que é ilustrado na figura 1.2.

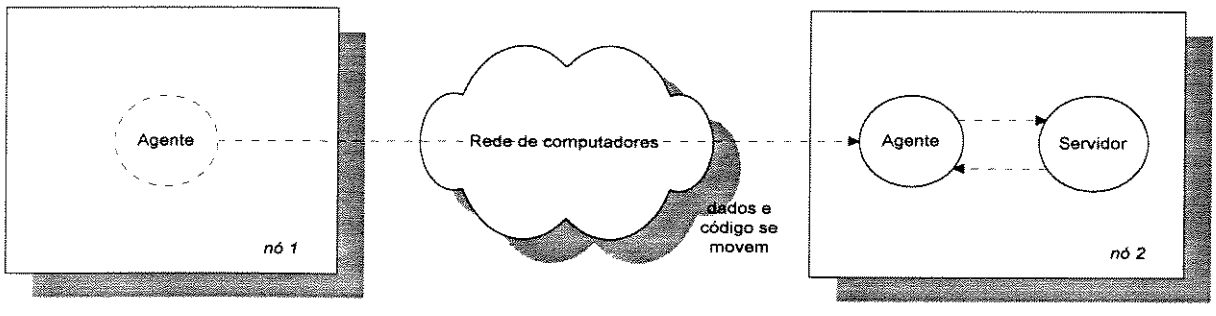

Figura 1.2: Migração de um agente móvel para a utilização de recursos remotos

Visto que agentes se movem em redes potencialmente heterogêneas, seu código deve ser executado de maneira idêntica em cada nó para o qual eles podem ser transportados [Ling95]. Desta forma, o seu código deve ser independente de plataforma.

Sistemas baseados em agentes móveis apresentam as seguintes vantagens:

- estruturação intuitiva de sistemas distribuídos: componentes de sistemas são entidades autônomas, com vida própria [ANSA95];
- arquitetura flexível para computação distribuída: agentes têm interfaces e comportamento bem definidos, e não assumem papéis fixos [Nwan96];
- computação assíncrona: agentes podem ser despachados enquanto outras tarefas são executadas por seus donos [Nwan96];
- redução de custos de comunicação: ao contrário de paradigmas clássicos, agentes se movem até os locais onde recursos (por exemplo, uma base de dados) estão disponíveis para realizar tarefas; portanto não é necessário manter conexões síncronas (estáticas) entre requisitantes e provedores de serviços [ANSA95] [Nwan96]. Esta vantagem se torna concreta nos casos em que o tempo de migração (para o local onde o recurso reside, bem como o retorno com a resposta da tarefa) de um agente móvel é menor do que o tempo de resposta a uma requisição em conexões síncronas;
- suporte à operação desconectada[4]: computadores móveis e dispositivos leves (tais como Assistentes Digitais Pessoais[5]) normalmente se conectam de forma intermitente a uma rede. Aplicações clientes em execução naqueles equipamentos podem iniciar um agente contendo uma requisição de um serviço e despachar o agente durante uma rápida sessão de conexão. A resposta pode vir numa conexão realizada num momento posterior [ANSA95] [Harr95] [Nwan96];
- suporte a aplicações clientes baseadas em máquinas com recursos computacionais restritos: agentes executam tarefas em nós para onde migram, diminuindo a carga de processamento no nó em que foram despachados [ANSA95] [Harr95] [Nwan96].

[4] *"fire and forget"*
[5] *"Personal Digital Assistants"* (PDAs)

As três primeiras vantagens listadas anteriormente estão relacionadas a sistemas baseados em agentes em geral, isto é, não são exclusivas para SBAMs.

A abordagem baseada em agentes móveis contempla os requisitos de paradigmas de desenvolvimento de sistemas distribuídos expostos anteriormente de forma mais abrangente que o modelo Cliente/Servidor. Independência de plataforma é um atributo básico de agentes móveis, já que eles podem ser executados em qualquer nó de uma rede heterogênea. O requisito de flexibilidade, ao contrário do modelo Cliente/Servidor, é atendido já que agentes possuem interfaces e comportamento bem definidos e não assumem papéis fixos. Além disso, pelo fato de agentes enfatizarem encapsulamento e modularidade, evoluções de sistemas podem ser realizadas com impactos reduzidos e em intervalos de tempo menores.

O caráter assíncrono da abordagem baseada em agentes móveis permite maior eficiência na utilização de recursos e redução de custos de comunicação. A exploração de paralelismo através de agentes também pode trazer benefícios neste sentido.

O suporte à computação móvel pode ser realizado de forma mais abrangente e eficiente como conseqüência do caráter assíncrono de agentes, do suporte à operação desconectada, e do suporte a aplicações clientes baseadas em máquinas com recursos computacionais restritos. Agentes podem ser transportados durante conexões rápidas para realizar tarefas nos nós que oferecem serviços, utilizando a capacidade de processamento destes nós. No caso de grande transferência de volumes de dados, agentes podem primeiramente filtrar informações nos nós que oferecem serviços antes de transferi-las. Assim sendo, a utilização de recursos computacionais e de rede é otimizada.

Os requisitos de rápido desenvolvimento e rápida evolução, e de facilidade de integração, podem ser atendidos se ambientes de desenvolvimento de sistemas baseados em agentes móveis se fundamentarem em estratégias como Desenvolvimento Rápido de Aplicações[6] e em padrões abertos, respectivamente. Agentes podem ser empregados para auxiliar seres humanos na execução de tarefas e para a personalização de programas de acordo com as preferências de seus usuários [Maes94]. Neste caso, são chamados de assistentes pessoais de software e introduzem facilidade no uso de sistemas computacionais.

[6] *"Rapid Application Development"* (RAD)

O artigo: *"Mobile Agents: Are they a good idea?"* [Harr95] reforça o argumento de que SBAMs se constituem numa abordagem singular para o desenvolvimento de sistemas distribuídos. Os autores afirmam que correntemente não há outra alternativa que apresente o mesmo conjunto de funcionalidades e vantagens que a abordagem baseada em agentes móveis.

Apesar de ser uma área de recente concentração de esforços de pesquisa e desenvolvimento, sistemas baseados em agentes móveis têm atraído o interesse e o investimento de um conjunto expressivo e crescente de universidades, centros de pesquisa e empresas. Entretanto há um número muito reduzido de experiências de utilização desta abordagem.

Vários pesquisadores, entre eles Guilfoyle [Guil95] e Janca [Janc96], afirmam que nos próximos anos a abordagem baseada em agentes promoverá grande impacto sobre sistemas computacionais, através da sua adoção completa ou parcial como paradigma de desenvolvimento.

1.2. Objetivos

Este trabalho visa explorar sistemas baseados em agentes móveis como abordagem para o desenvolvimento de sistemas distribuídos. O domínio de qualidade de serviço em sistemas multimídia distribuídos foi escolhido como estudo de caso para avaliar se as potencialidades de tal abordagem se mostram verdadeiras na prática.

1.3. Contribuições

Este trabalho contém uma revisão bibliográfica abrangente a respeito de sistemas baseados em agentes, com ênfase em agentes móveis. Além disso, um conjunto de infra-estruturas correntemente existentes para suporte a sistemas baseados em agentes móveis é examinado de forma criteriosa. Outra contribuição diz respeito à integração de uma infra-estrutura de computação distribuída aberta (baseada no padrão CORBA[7] [OMG97]), com uma infra-estrutura para suporte a SBAMs (Aglets Workbench [IBM96a]), visando o desenvolvimento de um sistema baseado em agentes móveis para monitoramento de qualidade de serviço em sistemas multimídia distribuídos.

[7] *"Common Object Request Broker Architecture"*

1.4. Organização da dissertação

O conteúdo desta dissertação está organizado conforme descrito na seqüência. O capítulo 2 apresenta uma revisão bibliográfica abrangente sobre sistemas baseados em agentes, com ênfase para SBAMs. No capítulo 3 um conjunto de infra-estruturas existentes para suporte a sistemas baseados em agentes móveis é examinado. O aspecto de qualidade de serviço em sistemas multimídia distribuídos é abordado no capítulo 4, como estudo de caso para a aplicação de SBAMs. O capítulo 5 apresenta a experiência de desenvolvimento de um SBAM para monitoramento de qualidade de serviço em sistemas multimídia distribuídos. Finalmente, o capítulo 6 trata de conclusões e trabalhos que podem ser realizados como extensão desta dissertação.

Capítulo 2

Revisão Bibliográfica

Este capítulo apresenta um panorama geral sobre sistemas baseados em agentes (SBAs). Inicialmente destacam-se a origem e as linhas de pesquisa desta abordagem. Definições e características de agentes presentes na literatura bem como propostas de classificação de agentes são abordadas em seções específicas. Na seqüência são apresentados domínios de aplicação e exemplos de SBAs. Uma seção dedicada a agentes móveis trata de aspectos relevantes associados a este tipo de agente. Finalmente, os esforços correntes de padronização na área de sistemas baseados em agentes são descritos.

2.1. Origem e linhas de pesquisa

Segundo Nwana [Nwan96], a origem do conceito de agentes vem do modelo de atores proposto por Hewitt [Hewi77] em 1977, no início das atividades de pesquisa em Inteligência Artificial Distribuída (IAD). Atores são definidos como objetos autocontidos, interativos e de execução concorrente. O modelo de atores, por sua vez, fundamentou uma das grandes áreas no campo de IAD chamada sistemas multiagentes [O'Har96].

Sistemas multiagentes (SMAs) são compostos por entidades autônomas (agentes) que trabalham de forma coordenada para a resolução de problemas que estão além de suas capacidades individuais. Aspectos de gerência e coordenação do comportamento de agentes são de extrema relevância. A abordagem de sistemas baseados em agentes emergiu de SMAs, que atualmente podem ser vistos como um subconjunto de SBAs, voltado para IAD.

Duas linhas de pesquisa na área de SBAs são propostas por Nwana [Nwan96]: a primeira que engloba o período de 1977 até o presente momento, e a segunda que se iniciou em 1990 e também se estende até o presente momento.

A primeira linha tem uma forte influência de IAD. Seu trabalho iniciou concentrando-se em questões de escopo macro relativas a SBAs tais como: interação e comunicação entre agentes, decomposição e distribuição de tarefas, coordenação, cooperação e

resolução de conflitos via negociação. Uma grande meta desta linha é especificar, analisar, projetar e integrar sistemas compostos por vários agentes colaborativos. Estas questões de escopo macro enfatizam sociedade de agentes ao invés de agentes individuais. Além disso, trabalhos nesta linha também se caracterizam por pesquisa e desenvolvimento em questões teóricas, arquiteturais e de linguagem [Wool95].

A segunda linha reflete a ampla utilização do termo agente em diversas áreas da computação além de IAD, tais como sistemas distribuídos e engenharia de software. Ela retrata uma maior diversidade nos tipos de agentes sendo investigados.

A próxima seção deste capítulo apresenta definições e característica de agentes presentes nas duas linhas de pesquisa propostas por Nwana [Nwan96].

2.2. Definições e características de agentes

Embora os termos agente e sistema baseado em agentes sejam utilizados amplamente por pesquisadores em várias áreas da computação, não há uma definição de consenso para estes termos. Ao contrário, há uma grande diversidade no entendimento do que é um agente e quais são suas características fundamentais. Para ilustrar tal diversidade, na seqüência estão transcritas algumas definições retiradas da literatura:

> "um agente é um programa que auxilia um usuário na execução de uma tarefa (ou um conjunto de tarefas), possivelmente através da manutenção de um estado persistente e da comunicação com seu dono, outros agentes ou seu ambiente" [Ling95]

> "agentes são componentes de software que se comunicam com seus pares através de troca de mensagens numa linguagem expressiva de comunicação de agentes" [Gene94]

> "um agente autônomo é um sistema situado num ambiente e parte deste ambiente, que percebe e age sobre o ambiente, no tempo, a fim de completar sua agenda de trabalho, de maneira a afetar o que ele percebe no futuro" [Fran96]

> "agentes inteligentes são entidades de software que executam um conjunto de operações em favor de um usuário ou de um outro programa, com algum grau de independência ou autonomia, e desta maneira, empregam algum conhecimento ou representação das metas ou desejos do usuário" [IBM96b]

"agente é um sistema de computador que é concebido ou implementado usando conceitos que são normalmente aplicados a seres humanos. Além disso, deve conter as propriedades listadas abaixo:

autonomia: agentes realizam atividades sem intervenção direta de seres humanos, e têm algum controle sobre suas ações e estado interno;

habilidade social: agentes interagem com outros agentes (e possivelmente com seres humanos) através de algum tipo de linguagem de comunicação de agentes;

reatividade: agentes percebem seu ambiente e respondem no tempo a mudanças nele ocorridas;

pró-atividade: agentes não somente agem em resposta ao seu ambiente, eles são capazes de exibir comportamento orientado por metas, tomando a iniciativa." [Wool95]

A tabela 2.1 mostra características fundamentais de agentes mais freqüentemente adotadas na literatura, seus sinônimos e sua definição.

Característica	Sinônimos	Definição	Referências
delegação		o usuário confia ao agente parte de ou toda uma tarefa	[Ches95] [Gilb96] [Ling95] [IBM96b] [Nwan96]
autonomia		agentes têm controle sobre suas próprias ações e estado interno	[Fran96] [Wool95] [Belg95] [IBM96b] [Nwan96]
habilidade social	comunicabilidade, habilidade de comunicação, sociabilidade	capacidade de interação com outros programas, seres humanos e ambiente externo	[Wool95] [Belg95] [Gilb96] [Gene94]
reatividade	perceber e agir	resposta, no tempo, a mudanças no ambiente externo	[Fran96] [Wool95] [Belg95]
orientação por metas	pró-atividade	ação não apenas em resposta ao ambiente externo; capacidade de exibir comportamento dirigido por metas, tomando a iniciativa	[Fran96] [Wool95] [Nwan96]

Tabela 2.1: Características fundamentais de agentes freqüentemente adotadas na literatura

O termo agente aparece muitas vezes acompanhado de qualificadores, tais como: agentes autônomos, agentes pessoais e agentes inteligentes. Além disso, vários sinônimos para agentes também podem ser encontrados, incluindo: robôs de software, assistentes pessoais de software, robôs baseados em tarefas[1] e robôs para suporte a usuários[2]. Tais qualificadores e sinônimos enfatizam uma ou mais características consideradas como fundamentais por quem os define.

As características de delegação e autonomia estão fortemente presentes nas duas linhas de trabalho propostas por Nwana [Nwan96] e constituem o núcleo do conceito de agentes. A característica de habilidade social também está relacionada a agentes de forma marcante. Reatividade e orientação por metas estão associadas à primeira linha, mais voltada para IAD.

Outras características de agentes presentes na literatura englobam:

- cooperação: capacidade de interação com outros agentes visando atingir metas comuns; para tanto, agentes devem possuir habilidade social [Nwan96];
- raciocínio e aprendizado: à medida que agentes reagem ao e/ou interagem com seu ambiente externo, promovem uma mudança no seu comportamento (adaptabilidade) [Nwan96];
- persistência: capacidade de manter um estado interno consistente no tempo [Belg95];
- personalização: o usuário determina como o agente interage [Gilb96];
- mobilidade: habilidade de se mover numa rede [Gilb96].

Nesta dissertação, agentes são considerados como *programas autônomos que recebem autoridade de seus donos (usuários ou outros programas) para agir em favor destes na execução de tarefas. Visando completar as tarefas que lhes foram atribuídas, agentes podem se comunicar com outros programas, com o ambiente que os abriga ou com seres humanos.*

[1] *"taskbots"*
[2] *"userbots"*

Na seção 1.1 estão descritas as características fundamentais de agentes adotadas nesta dissertação. Tais características refletem a visão de que um sistema de software baseado em agentes é composto por entidades (agentes) autônomas que executam tarefas a elas atribuídas (delegadas) por seus donos. Cada entidade é vista como par pelas outras entidades que compõem o sistema e exerce papéis flexíveis com interfaces e comportamento bem definidos. Modularidade e encapsulamento são enfatizados através de agentes. Além disso, paralelismo pode ser explorado, visto que agentes são entidades autônomas.

A abordagem baseada em agentes é um paradigma genérico de desenvolvimento de sistemas. Entretanto, por ser uma proposta recente, é necessário que um conjunto de métodos, técnicas e ferramentas de suporte seja introduzido para que esta abordagem se consolide.

2.3. Classificação de agentes

Uma classificação de agentes visa apresentar vários tipos de agentes, organizados através de critérios. Tais critérios podem refletir, por exemplo, as características associadas a agentes, os papéis que eles desempenham, etc. A maneira mais precisa de se representar uma classificação de agentes seria uma matriz onde cada critério de classificação estaria associado a uma dimensão. Como há um número muito grande de possíveis critérios, uma representação matricial não seria de fácil leitura e compreensão.

Entretanto, vários esquemas de classificação têm sido propostos na literatura baseados em conjuntos reduzidos de critérios. Nesta seção, serão apresentados alguns deles, cujos autores são: Nwana [Nwan96], Franklin e Graesser [Fran96], Stone e Veloso [Ston96] e FIPA[3] [FIPA97].

Nwana [Nwan96] aponta uma série de possíveis critérios que podem ser utilizados para classificação:

- mobilidade: capacidade de se mover numa rede. Agentes podem ser estáticos ou móveis;
- arquitetura [Wool95]: agentes podem ser deliberativos ou reativos. Agentes deliberativos possuem um modelo interno simbólico de raciocínio e fazem uso de

[3] *"Foundation for Intelligent Physical Agents"*

planejamento e negociação para realizar coordenação com outros agentes. Agentes reativos não contém modelos simbólicos do seu ambiente, e agem usando um comportamento do tipo estímulo/resposta para responder ao estado presente do ambiente no qual estão inseridos;

- características básicas: autonomia, aprendizado e cooperação. Tais características permitem a obtenção de quatro tipos de agentes: agentes inteligentes, agentes colaborativos, agentes de interface e agentes colaborativos que aprendem. Agentes inteligentes possuem as três características básicas. Agentes colaborativos se baseiam nas características de cooperação e autonomia. Agentes de interface se caracterizam por autonomia e aprendizado. Agentes colaborativos que aprendem se baseiam nas características de cooperação e aprendizado;

- papéis desempenhados: agentes da Internet ou agentes de informação auxiliam na gerência de informação em redes de longa distância como a Internet.

O conjunto de possíveis critérios é então combinado para gerar uma lista de tipos (tipologia) de agentes, ilustrada pela figura 2.1, que engloba: agentes colaborativos, agentes de interface, agentes móveis, agentes de informação, agentes reativos, agentes inteligentes, agentes híbridos (que combinam dois ou mais critérios) e sistemas de agentes heterogêneos (conjunto integrado de dois ou mais agentes que pertencem a dois ou mais tipos diferentes de agentes).

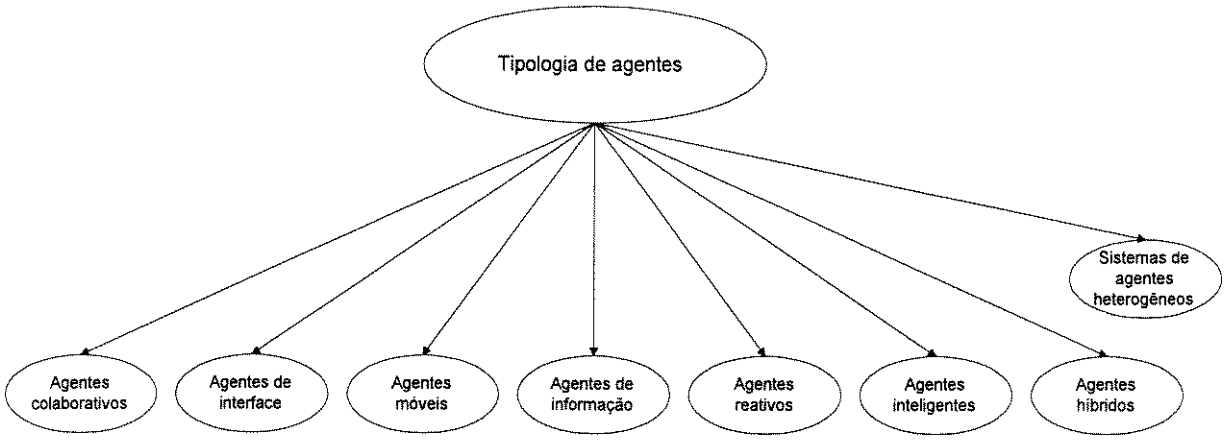

Figura 2.1: Classificação de agentes segundo Nwana

Franklin e Graesser [Fran96] também listam critérios que podem ser utilizados para a classificação de agentes de software, baseando-se em propriedades (características básicas ou não) de agentes, tarefas que eles executam, arquitetura, etc. Os autores afirmam que tais critérios podem ser combinados de forma a gerar uma classificação multidimensional. Uma taxonomia para agentes autônomos é então proposta (reproduzida na figura 2.2). Agentes de software são considerados como uma categoria específica de agentes autônomos e divididos em três tipos: agentes para tarefas específicas, agentes para entretenimento e vírus de computador.

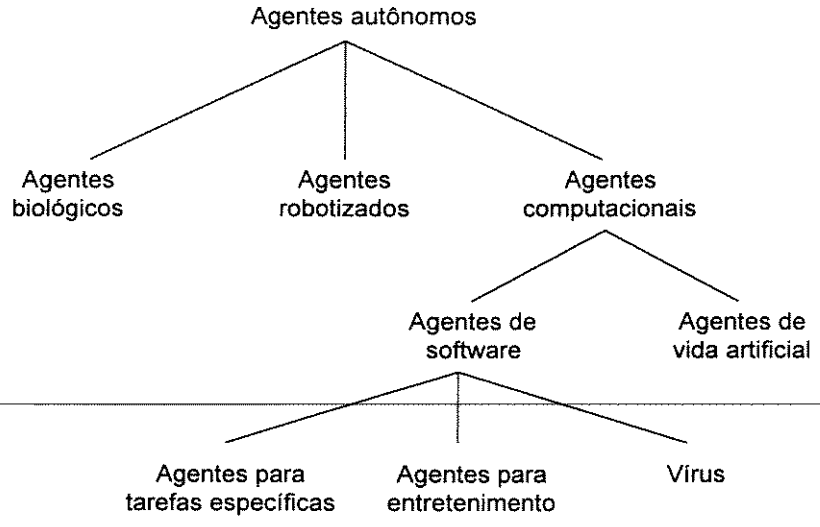

Figura 2.2: Taxonomia para agentes autônomos proposta por Franklin e Graesser

Stone e Veloso [Ston96] apresentam uma taxonomia para sistemas multiagentes baseada em dois aspectos, que eles consideram os mais importantes: grau de heterogeneidade e grau de comunicação. O grau de heterogeneidade diz respeito ao tipo de conhecimento associado ao agente: redundante ou especializado. Agentes homogêneos normalmente contém todo o conhecimento (tipo redundante) para executar qualquer parte de uma tarefa. Agentes heterogêneos tendem a ser projetados para executar subtarefas específicas, fazendo uso de conhecimento especializado. Comunicação considera o grau em que os agentes se comunicam, caso eles se comuniquem, não os protocolos disponíveis a eles.

Através desses dois aspectos são propostas três categorias de classificação: agentes homogêneos não comunicativos, agentes heterogêneos não comunicativos e agentes heterogêneos comunicativos (figura 2.3).

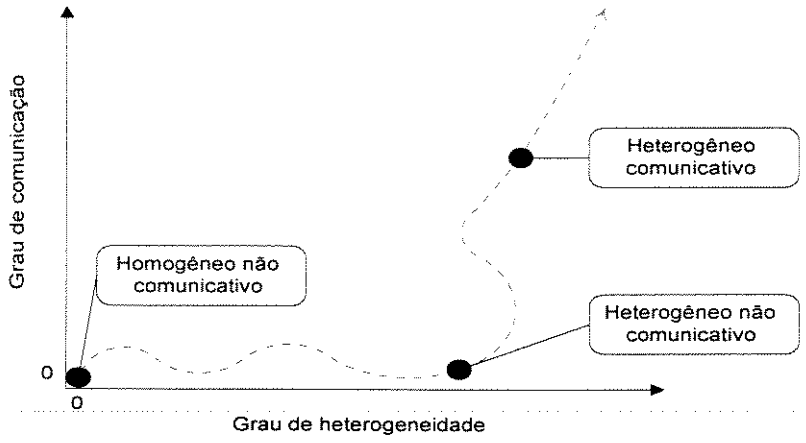

Figura 2.3: Categorias de classificação de sistemas multiagentes proposta por Stone e Veloso

A FIPA [FIPA97] é uma associação internacional que visa produzir especificações de consenso sobre tecnologias genéricas de agentes. Esta associação define uma lista de tipos de agentes baseada em seus atributos composta por: agentes autônomos, agentes de entretenimento, agentes de informação, agentes inteligentes e agentes de interface. A tabela 2.2 apresenta tal lista.

Tipo	Definição
agentes autônomos	Agentes que habitam em ambientes complexos e dinâmicos, percebem e agem autonomamente nestes ambientes e, ao fazê-lo, executam tarefas ou atingem metas
agentes de entretenimento	Mundos simulados, interativos, que provêem entretenimento para um usuário
agentes de informação	agentes que têm acesso potencial a muitas fontes de informação e são capazes de agregar e manipular informação obtida nessas fontes para responder requisições de busca de usuários e/ou agentes
agentes inteligentes	agentes que executam um conjunto de operações em favor de um usuário ou outro programa com algum grau de independência
agentes de interface	agentes que provêem assistência a usuários que estão lidando com outros agentes

Tabela 2.2: Classificação de tipos de agentes segundo a FIPA

A maioria dos trabalhos de classificação existentes até o presente momento não se preocupa em gerar uma lista extensiva de tipos de agentes, mas sim em apresentar alguns tipos vistos como mais importantes sob a ótica dos seus autores. Na seqüência deste capítulo será abordado o tipo de agentes que é o foco desta dissertação: agentes móveis.

2.4. Domínios de aplicação e exemplos de sistemas baseados em agentes

Visto que sistemas baseados em agentes é um paradigma genérico de desenvolvimento de sistemas, sua gama de aplicação é bastante vasta. Nwana [Nwan96] aponta alguns domínios onde SBAs estão sendo aplicados ou cuja aplicação está sendo investigada, dentre os quais estão: gerência de fluxo de trabalho, gerência de rede, controle de tráfego aéreo, reengenharia de processos de negócios, "garimpo" de dados[4], recuperação/gerência de informações, comércio eletrônico, educação, assistentes digitais pessoais, correio eletrônico, bibliotecas digitais e bases de dados inteligentes.

Nesta seção são apresentados alguns exemplos de aplicações de sistemas baseados em agentes. O primeiro é o trabalho de Maes [Maes94] que retrata a utilização de agentes como assistentes pessoais que auxiliam usuários na execução de tarefas. O segundo exemplo é o projeto Pleiades [Plei94], que utiliza agentes no domínio de tomada de decisão organizacional, para a execução de tarefas de recuperação, filtragem e fusão de informação em fontes distribuídas.

Maes [Maes94] tem trabalhado com agentes que provêem assistência personalizada a usuários para a execução de várias tarefas, incluindo: estabelecimento de horários de reuniões, filtragem de notícias eletrônicas, e seleção de livros, músicas e outras formas de entretenimento. Maes utiliza uma abordagem de aprendizado artificial[5], onde agentes aprendem os hábitos de seu usuário através de interações ao longo do tempo. O processo de aprendizado envolve quatro aspectos:

- observação e imitação do usuário;
- recebimento de retorno positivo ou negativo do usuário;
- recebimento de instruções explícitas do usuário;
- consulta a outros agentes para aconselhamento.

Com o passar do tempo, os agentes se tornam mais úteis à medida que eles acumulam conhecimento de como o usuário lida com certas situações. Gradualmente, tarefas que eram inicialmente executadas por usuários podem ser executadas por agentes. Caricaturas são utilizadas pelos agentes para apresentar seu estado interno ao usuário, conforme ilustrado na figura 2.4.

[4] *"data mining"*
[5] *"machine learning"*

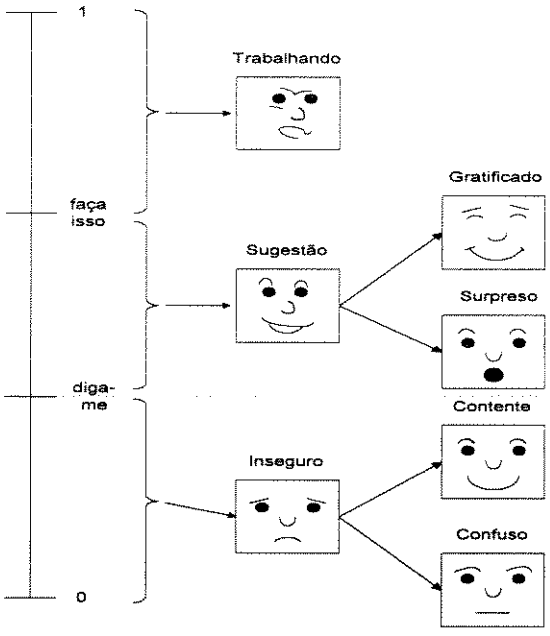

Figura 2.4: Caricaturas que apresentam o estado "emocional" interno de agentes para usuários

O projeto Pleiades [Plei94] tem como foco a colaboração entre agentes que se comunicam no domínio de tomada de decisão organizacional. Tal projeto está baseado numa arquitetura distribuída, ilustrada pela figura 2.5, que contém duas camadas de abstração: a primeira contendo agentes colaborativos para tarefas específicas; e a segunda contendo agentes colaborativos de informação específica.

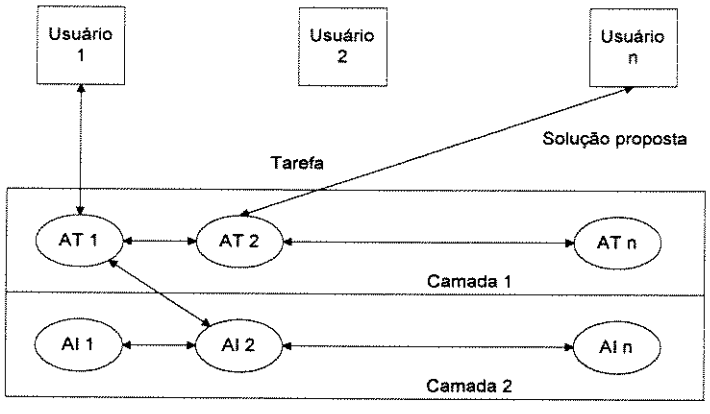

Figura 2.5: Arquitetura adaptada do sistema distribuído Pleiades

Agentes para tarefas específicas (ATs) executam uma tarefa específica para seu usuário, por exemplo marcar reuniões, interagindo com outros ATs para resolver conflitos

ou integrar informações. De forma a obter a informação necessária, eles fazem requisições de informações para agentes de informação específica (AIs). AIs podem colaborar entre si para prover a informação requisitada pelos agentes da camada 1. No final, ATs propõem soluções, às vezes intermediárias, para seus usuários.

Esta arquitetura foi utilizada para desenvolver aplicações como o sistema de atendimento de visitantes da Universidade Carnegie Mellon. Nesse sistema, primeiramente agentes de atendimento tem acesso a outras fontes de informação *"on line"* para determinar as áreas de interesse do visitante, seu nome e sua organização. Em seguida, utilizando a informação obtida, a base de dados de pessoal é pesquisada para determinar professores que possam atender o visitante, retornando dados como seus endereços eletrônicos e números de telefone.

Em terceiro lugar, o agente de atendimento a visitantes seleciona uma lista inicial de professores a serem contatados, e gera mensagens eletrônicas que são despachadas para os agentes de agenda dos professores selecionados. Os agentes de agenda são consultados visando saber se os visitantes podem ser recebidos pelos professores e em que horário. As respostas enviadas são então agregadas. Por fim, o agente de atendimento cria a agenda para o visitante, que contém as salas e os horários reservados para as várias reuniões com professores.

2.5. Agentes móveis

No que diz respeito ao local onde são executados, agentes podem ser estáticos (também chamados de fixos ou estacionários) ou móveis. Agentes estáticos residem numa mesma máquina durante toda a sua existência. Agentes móveis possuem a habilidade de se mover numa rede de computadores heterogênea, visando completar progressivamente tarefas que lhes foram atribuídas por seus donos. São também chamados de agentes itinerantes [Ches95] ou transportáveis [Gray95] [Kota94].

Esta seção apresenta um breve histórico sobre despacho de programas para execução remota. Além disso, mostra algumas definições existentes na literatura para o termo agentes móveis. Desafios, desvantagens e domínios de aplicação de sistemas baseados em agentes móveis são também abordados nesta seção. Por fim, aspectos de infra-estruturas para suporte a SBAMs são discutidos.

2.5.1. Histórico

A idéia de se despachar um programa para execução em um computador remoto é bastante antiga [Ches95]. A motivação normalmente era que o computador local não atendia os requisitos necessários (unidade central de processamento, memória) para executar o programa ou porque o computador remoto tinha acesso direto a um recurso com que o programa interagia.

Esquemas de despacho de programas podem ser exemplificados através de *"batch jobs"* que eram enviados para mainframes na década de 60, e execução remota de programas do tipo *"script"* na década de 70, para permitir processamento distribuído em tempo real. Um exemplo mais recente é o conceito de correio ativo[6], no qual serviços de correio eletrônico têm sido usados para entregar *"scripts"* executáveis.

A linha de produtos da General Magic [Gene96], baseada na linguagem Telescript, foi a primeira implementação de uma infra-estrutura para suporte a sistemas baseados em agentes móveis, e tornou-se disponível para uso comercial em 1994. Esta infra-estrutura é examinada no capítulo 3, seção 3.2.

2.5.2. Definições

Da mesma forma que o termo agente, também não há uma definição de consenso sobre o que é um agente móvel. Embora haja concordância que um agente móvel pode se mover numa rede carregando consigo seu código e dados internos, o mesmo não é verdade no que se refere ao seu estado de execução. A seguir são transcritas algumas definições encontradas na literatura para este termo:

> "agentes móveis são programas, tipicamente escritos em uma linguagem de *script*, que podem ser despachados de um computador cliente e transportados para um computador servidor para execução" [Harr95]

> "agentes móveis são processos computacionais de software capazes de vagar por redes de longa distância, interagindo com nós desconhecidos, colhendo informações em favor de seus donos e voltando após o término de obrigações definidas por seus usuários" [Nwan96]

[6] *"active mail"*

"agentes itinerantes são programas despachados de um computador, vagando por um conjunto de servidores na rede até que suas tarefas tenham sido completadas; são processos em movimento que progressivamente completam tarefas movendo-se de um lugar para outro" [Ches95]

"agentes transportáveis são capazes de suspender sua execução num determinado sistema computacional (nó), transportar-se para um outro nó da rede e retomar sua execução a partir do ponto em que ela foi suspensa" [Kota94]

"um agente transportável é um programa com nome que pode migrar de uma máquina para outra numa rede heterogênea. O programa escolhe quando e para onde migrar. Ele pode suspender sua execução num ponto arbitrário, se transportar para outra máquina e retomar sua execução na nova máquina" [Gray95]

Kotay e Kotz [Kota94] bem como Gray [Gray95] enfatizam que o transporte de um agente móvel envolve também a migração de seu estado de execução.

2.5.3. Desafios e desvantagens

Embora sistemas baseados em agentes móveis se constituam numa alternativa promissora para o desenvolvimento de sistemas distribuídos, existem alguns desafios para a consolidação desta abordagem. Muitos dos desafios estão relacionados à verificação de que as vantagens da utilização de SBAMs são na prática uma realidade. Dentre eles pode-se citar:

- eficiência: ambientes de execução de SBAMs requerem recursos computacionais significativos? Comparando-se com outros métodos, a transmissão de uma transação ou requisição de consulta via um agente móvel resulta em mais ou menos tráfego de rede? [Harr95]

- desempenho: qual será o efeito de se ter centenas, milhares ou milhões de agentes numa rede? [Nwan96]

- flexibilidade e robustez: SBAMs provêem realmente um método mais flexível e robusto de comunicação? [Harr95]

- serviços de interoperabilidade/comunicação/intermediação[7]: como prover serviços de intermediação e lista de diretórios para a localização de máquinas de execução[8] e/ou serviços específicos? Como executar um agente escrito em uma linguagem, numa máquina de execução baseada em outra linguagem? [Nwan96]

- tolerância a falhas: como assegurar que um agente não foi terminado ou que sua execução não foi interrompida de forma anormal?

- a escala geográfica de redes públicas de longa distância e a ausência de acordos de administração formal entre os muitos domínios que compõem as redes levantam problemas para a navegação e a autenticação de agentes móveis [Ches95]

- segurança: Como assegurar que o agente não é ou não está infectado por um vírus? Como assegurar a inviolabilidade do agente por entidades externas?

Com respeito aos desafios apresentados é unânime que segurança é ponto vital para a consolidação da abordagem de sistemas baseados em agentes móveis. Isto porque o controle e monitoramento de agentes móveis é muito difícil, e se faz necessária a proteção de ambientes contra ações nocivas executadas por agentes neles recebidos.

Algumas desvantagens de SBAMs transcritas abaixo são apontadas no trabalho de Harrison et al. [Harr95]. Elas estão diretamente relacionadas ao desafio na área de segurança em SBAMs e, segundo os autores, requerem um estudo aprofundado e grande inovação técnica:

- necessidade de ambientes de execução com grande segurança;

- limitações funcionais e de desempenho como conseqüência do nível de segurança necessário;

- necessidade de mecanismos de controle e rastreamento de vírus de computador.

[7] *"brokerage"*
[8] *"engines"*

2.5.4. Domínios de aplicação de sistemas baseados em agentes móveis

Sistemas baseados em agentes móveis têm potencial de aplicação para qualquer tipo de sistema distribuído. Como exemplo, os seguintes domínios de aplicação podem ser citados: recuperação de informação, gerência de rede, comércio eletrônico e computação móvel [Ling95].

No domínio de recuperação de informação, agentes móveis podem conter uma requisição de busca de seu dono e varrer a rede buscando e migrando para fontes de informação que mais se adeqüem à requisição. Tal estratégia pouparia trabalho a usuários, tendo-se em vista o grande e crescente número de fontes disponíveis. Gerência de redes, principalmente redes de grande porte, poderia ter tarefas importantes que consomem tempo, tais como monitoramento e detecção de falhas, realizadas por agentes móveis.

À medida que o comércio eletrônico vem se expandindo através da Internet, agentes móveis podem ser empregados na localização de ofertas de menor valor, para realizar negócios ou fornecer informações para a sua realização. Como último exemplo de domínio de aplicação de agentes móveis, computação móvel pode ser suportada de forma mais abrangente e eficiente por SBAMs. Normalmente dispositivos e computadores móveis estão ligados de forma intermitente à rede e têm recursos computacionais mais restritos. O caráter assíncrono e o suporte à operação desconectada providos por SBAMs são de grande valor nesse contexto.

O cenário descrito a partir do parágrafo seguinte, que trata de planejamento e reserva de viagens aéreas, ilustra a utilização de agentes móveis no domínio de comércio eletrônico. Tal cenário também explora o emprego de SBAMs em computação móvel e foi adaptado de Chess et al. [Ches95] e General Magic [Gene96].

João é um consultor na área de agricultura irrigada que deseja fazer uma viagem aérea de negócios num período de dois dias, partindo de São Paulo para Fortaleza numa quinta-feira. Dependendo do valor de uma possível alteração no preço da passagem, João pensa em ficar até domingo e então voltar para São Paulo. Ele possui um computador portátil, que pode ter acesso a uma rede pública, onde há um SBAM para planejamento e reserva de viagens. João então executa o sistema e apresenta sua identificação. Seus dados pessoais são mostrados e as seguintes informações, que compõem o plano de viagem, são requisitadas pelo sistema: datas desejadas de viagem, forma de pagamento e

suas preferências (por exemplo: faixa de preços, companhias aéreas, localização no avião, setor fumante ou não fumante).

O sistema de planejamento e reserva de viagens faz uso de informações registradas anteriormente sobre as preferências de João para fazer sugestões ao seu dono. Após a entrada dos dados, tal sistema então cria um agente móvel com a autoridade de João cuja tarefa é levantar as melhores alternativas de viagem no âmbito do plano estabelecido. Os endereços de um conjunto de agências de viagem podem ser fornecidos por serviços do tipo "páginas amarelas" ou pelo próprio usuário. Os passos seguintes são então tomados:

1. o agente se transporta para a próxima agência de viagem não visitada. Na primeira vez, a migração é feita a partir do computador móvel, durante o período em que João está conectado à rede pública. O sistema de planejamento e reserva de viagens pode ser executado mesmo quando o computador não está conectado;

2. o agente interage com o provedor de serviços de planejamento e reserva de viagens da agência que o recebeu, apresentando as informações referentes ao usuário que ele representa e recebendo informações de número do vôo, preço e assento, tanto para uma viagem na sexta-feira quanto no domingo;

3. uma classificação das ofertas é feita pelo agente baseando-se nas preferências do usuário. Se as ofertas da agência corrente atendem melhor as preferências do usuário, o agente faz então a reserva e descarta possíveis reservas estabelecidas anteriormente. A reserva fica pendente por um certo tempo negociado com o provedor.

Após a visita a um número mínimo de agências ou obtido um número mínimo de ofertas de viagem que atendam aos requisitos do seu dono, o agente móvel retorna ao computador portátil de João e mostra os resultados obtidos para confirmação ou seleção. Após a decisão por parte de João, um agente é enviado para a agência escolhida, efetua a compra do bilhete e retorna ao computador móvel de seu dono com a confirmação ou não da compra. A figura 2.6 ilustra este cenário.

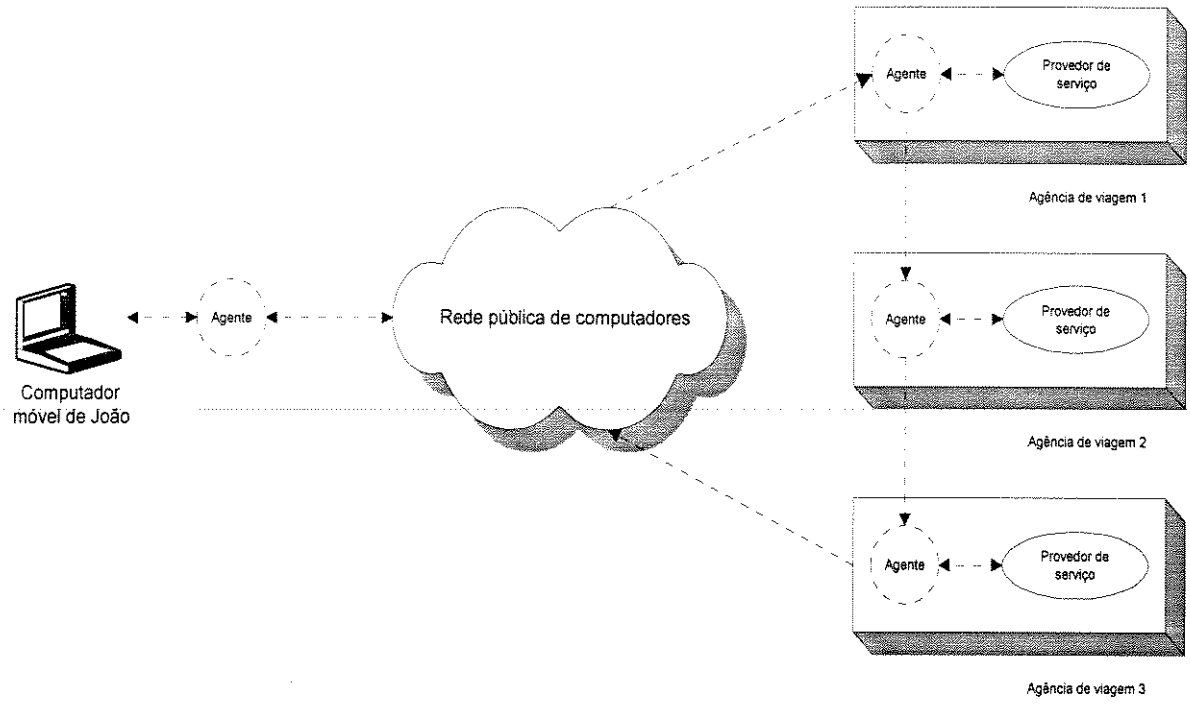

Figura 2.6: SBAM para planejamento e reserva de viagens aéreas

2.5.5. Infra-estruturas para suporte a sistemas baseados em agentes móveis

Uma infra-estrutura para suporte a SBAMs é um sistema de software distribuído numa rede de computadores possivelmente heterogênea, que permite a criação, a execução, a migração e o término de agentes, bem como a comunicação entre agentes, agentes e outros programas e agentes e seres humanos.

Segundo Green et al. [Gree97] uma infra-estrutura para suporte a SBAMs deve conter grande parte dos seguintes modelos: modelo de agentes, modelo de ciclo de vida, modelo computacional, modelo de segurança, modelo de comunicação e modelo de navegação. Na seqüência, serão apresentados e discutidos tais modelos e aspectos relevantes a eles relacionados. Finalmente, um esquema de decomposição de infra-estruturas em blocos funcionais é mostrado tendo como base os aspectos abordados nesta seção.

Modelo de agentes

O modelo de agentes define a estrutura interna de um agente, provendo suporte às características previstas pela infra-estrutura, como por exemplo: autonomia e delegação. Algumas considerações importantes sobre delegação são tratadas na seqüência.

Agentes foram concebidos inicialmente como entidades que executam tarefas em favor de seus donos. Embora agentes possam ser utilizados na execução de qualquer tarefa, um campo de extrema aplicação de agentes é a sua utilização no auxílio a usuários para a execução de tarefas rotineiras que consomem grandes intervalos de tempo.

Maes [Maes94] levantou uma questão importante no que diz respeito à delegação: "Como assegurar que usuários se sintam confortáveis na delegação de tarefas a agentes?". Ela crê que a técnica de aprendizado artificial pode responder a esta pergunta. Maes desenvolveu trabalhos fundamentando-se nesta técnica, onde um usuário é capaz de construir incrementalmente um modelo das competências e limitações de agentes, baseado nas capacidades de raciocínio e aprendizado por eles utilizadas para resolver tarefas anteriores.

Agentes devem conter uma identificação de quem (pessoa, organização, programas) eles representam. Tal identificação deve ser globalmente entendida e passível de autenticação em qualquer nó da rede onde a infra-estrutura reside. Na infra-estrutura para agentes itinerantes proposta por Chess et al. [Ches95], um agente contém um passaporte com a autenticação do dono.

Modelo de ciclo de vida

Este modelo define os diferentes estados de execução de um agente móvel e os eventos que levam à transição de estados. Assim sendo, ele está fortemente ligado ao modelo computacional, que descreve como a execução ocorre. Os dois modelos de ciclo de vida mais difundidos correntemente são: modelo de processo persistente e o modelo baseado em tarefas.

No modelo de processo persistente, há um estado de inicialização chamado de "início" que passa a "em execução" no momento em que o processo (agente) é executado. Quando o agente móvel migra para um nó, o processo no estado "em execução" é verificado e suspenso, e o seu estado é alterado para "congelado". Ao chegar no nó, o contexto de execução do processo é restaurado e ele volta ao estado "em execução" a partir do ponto em que foi suspenso. Quando o processo alcança seu final ele passa ao estado de "terminado".

No modelo baseado em tarefas, também há um estado de inicialização chamado de "início". Entretanto, dependendo de um conjunto de condições, o processo continua através de uma rede de tarefas, onde cada tarefa tem seu próprio estado. Quando um agente se

move para um outro nó, o contexto da tarefa correntemente em execução é perdido. Antes da migração o agente deve indicar qual tarefa deve ser executada quando ele for retomado no nó destino. Quando o processo alcança seu final ele passa ao estado de "terminado".

Modelo computacional

O modelo computacional define como um agente móvel é executado quando ele está no estado "em execução". A execução ocorre no contexto de um ambiente de execução baseado em um ou vários processadores (onde cada processador pode ser uma unidade central de processamento de um computador ou um processador abstrato de uma máquina virtual). Parte deste modelo é composto por um conjunto de instruções primitivas que definem uma linguagem de programação baseada em agentes móveis. Tal linguagem estabelece as habilidades computacionais de um agente como, por exemplo, *"multithreading"*. Desenvolvedores de sistemas baseados em agentes móveis têm acesso aos outros modelos através do modelo computacional.

Infra-estruturas para suporte a SBAMs não devem se basear em uma única linguagem. Ao contrário, elas devem suportar várias linguagens e prover mecanismos de interação entre agentes escritos em diferentes linguagens. Para tanto, uma linguagem de comunicação de agentes padronizada, tal como proposta por Genesareth e Ketchpel [Gene94], pode ser usada.

Com respeito à sua estrutura, as linguagens de programação podem ser imperativas, funcionais ou declarativas. Os dois últimos tipos se aplicam principalmente a SBAs com ênfase em representação de conhecimento, visando prover inteligência a agentes. Linguagens imperativas, principalmente procedurais e orientadas a objetos, são normalmente aplicadas em SBAMs.

Quanto à forma de execução, linguagens podem ser compiladas, interpretadas ou compiladas para uma linguagem portável interpretada. A primeira forma tem maior aplicação para sistemas baseados em agentes estáticos, já que código compilado é específico para uma dada plataforma. As outras duas são mais apropriadas para SBAMs, cujo código é único e empregado em diversas plataformas. Entretanto, tais formas de execução introduzem um decréscimo no desempenho de SBAMs, devido à interpretação do código.

Há um desafio chave no que concerne à definição de um modelo computacional que possa atender requisitos de mobilidade e de inteligência. Agentes móveis têm seu código

independente de plataforma, e normalmente são compactos para que possam facilmente migrar numa rede e serem recuperados no caso de falhas. Com a introdução de inteligência, agentes móveis poderiam carregar consigo grandes bases de conhecimento acumulado ao longo de seu itinerário, além de ter que contar com ambientes de execução independentes de plataforma para manipulação de conhecimento.

Modelo de segurança

Segurança é um aspecto fundamental relativo a infra-estruturas para suporte a SBAMs. Isto porque em tese um agente móvel pode migrar livremente para qualquer nó de uma rede, e mecanismos de segurança devem existir para prevenir ações nocivas por parte de agentes que chegam num determinado nó. Por outro lado, agentes também não devem ser adulterados nos nós que os recebem.

Embora segurança seja tão relevante é sabido que a tecnologia de computação corrente não permite que tal aspecto seja tratado por completo em SBAMs. Por exemplo, a verificação com certeza total de que um agente arbitrário não é um vírus de computador é atualmente impossível. Entretanto, outras funcionalidades importantes de segurança podem ser providas, tais como: autenticação do dono de um agente, verificação de autorizações do dono de um agente para execução num nó, controle de acesso a recursos, registro de ações e verificação de integridade.

A autenticação visa o estabelecimento não ambíguo da identidade do dono de um agente. Isto pode ser realizado através da inclusão de um certificado de chave pública do dono como parte de um agente, fazendo uso de técnicas de criptografia [Ches95].

O controle de acesso a recursos permite que não haja acesso indevido a recursos por parte de agentes. Para a realização de um controle efetivo, agentes podem conter informações que representem suas necessidades e sua capacidade corrente de utilização de recursos – por exemplo, quando for necessário fazer o pagamento pela sua utilização. A infra-estrutura deve verificar a disponibilidade de recursos antes de permitir a execução de um agente quando este é recebido num nó. Da mesma forma, após o despacho de agentes, recursos de um nó devem ser liberados para uso posterior.

Registro de ações visa o armazenamento de informações sobre as ações que um determinado agente realizou, com propósito de descobrir autores de possíveis ações nocivas e realizar estatísticas de uso. Mecanismos de verificação de integridade visam

proteger agentes móveis de alterações indevidas por parte do ambiente que os abriga num nó da rede. Técnicas de criptografia podem ser usadas com este objetivo [Ches95].

Modelo de comunicação

Visando a realização de tarefas, agentes podem se comunicar com outros programas (sejam eles agentes ou não), com seres humanos e com ambientes que os abrigam. A comunicação pode ter objetivos como: acesso a serviços, cooperação e negociação entre partes conflitantes. Protocolo é a implementação de um modelo de comunicação. Devido ao fato de existirem na prática vários protocolos implementados, é provável que uma infra-estrutura suporte mais do que um modelo de comunicação.

A comunicação entre agentes pode ser local ou remota, síncrona ou assíncrona. Ela engloba a interação entre agentes existentes em ambientes de execução diferentes (possivelmente escritos em linguagens de programação diferentes) e entre agentes em execução em diferentes infra-estruturas.

O trabalho de Genesareth e Ketchpel [Gene94] introduz o conceito de engenharia de software baseada em agentes, no qual aplicações são escritas como agentes de software. Agentes são componentes de software que se comunicam com seus pares através da troca de mensagens em uma linguagem de comunicação de agentes (LCA) expressiva. Nessa abordagem, agentes usam uma linguagem comum para comunicação, com uma semântica independente dos próprios agentes.

Pesquisadores do Esforço de Compartilhamento de Conhecimento[9] da Agência de Projetos de Pesquisa Avançados[10] definiram três componentes para uma LCA: vocabulários; uma linguagem interna chamada Formato de Troca de Conhecimento[11] (KIF) [Gene92] contendo dados simples, expressões, regras e restrições; e uma linguagem externa chamada Linguagem de Manipulação e Busca de Conhecimento[12] [Fini93] que é uma camada lingüística sobre KIF que provê informação contextualizada para comunicação eficiente.

[9] *"Knowledge Sharing Effort"*
[10] *"Advanced Research Projects Agency"* (ARPA)
[11] *"Knowledge Interchange Format"*
[12] *"Knowledge Query and Manipulation Language"* (KQML)

Agentes podem também se comunicar com outros programas tais como sistemas de gerência de base de dados, visando ter acesso a serviços e recursos disponíveis num determinado nó.

A comunicação entre agentes e seus donos pode ser realizada com propósitos de notificação e confirmação de ações. Para tanto, são apropriados mecanismos como correio eletrônico e interfaces gráficas – quando, por exemplo, agentes fazem uso de uma janela para requisitar a confirmação de seu dono a respeito de ações a serem executadas.

Agentes também devem ser capazes de interagir com os ambientes que os abrigam visando fazer uso das capacidades oferecidas por tais ambientes para a execução de suas tarefas. Por exemplo, o acesso a informações estruturais de um outro agente com propósito de comunicação.

Modelo de navegação

Infra-estruturas para suporte a SBAMs devem prover migração transparente de agentes entre nós de uma rede. A migração envolve mecanismos de seleção de destino, despacho, transporte e recebimento de agentes.

Visto que agentes se movem numa rede para completar suas tarefas, a seleção de destinos promissores para migração é uma questão primária. Agentes podem ter itinerário pré-definido, mas normalmente sua rota de viagem é estabelecida dinamicamente. Mecanismos para oferta e busca de serviços devem ser providos para auxiliar a composição de um itinerário. Exemplos de tais mecanismos são: busca por palavra chave e roteamento semântico – que tenta selecionar destinos baseando-se nos objetivos que um agente tem que cumprir. Serviços de diretórios ou "páginas amarelas" podem ser utilizados como fundamento para mecanismos de seleção de serviços.

O despacho de agentes engloba os passos de: liberação dos recursos em uso, verificação da integridade do agente, armazenamento da estrutura do agente para tolerância a falhas, codificação da estrutura do agente para transporte e o envio do agente através de um mecanismo de transporte.

O transporte trata do envio de agentes de um nó para outro na rede através de mecanismos diversos, tais como protocolos padrão (TCP/IP, HTTP, X.25) e correio eletrônico. O recebimento de agentes engloba: descodificação de um agente para a recuperação de sua estrutura, autenticação e autorização, verificação de disponibilidade de

recursos necessários, entrega da estrutura de um agente para o seu ambiente de execução e armazenamento da estrutura com propósito de tolerância a falhas.

Esquema de decomposição funcional

A figura 2.7 ilustra um esquema representando alguns dos blocos funcionais mais relevantes associados a infra-estruturas para suporte a SBAMs, tomando como base os modelos discutidos anteriormente.

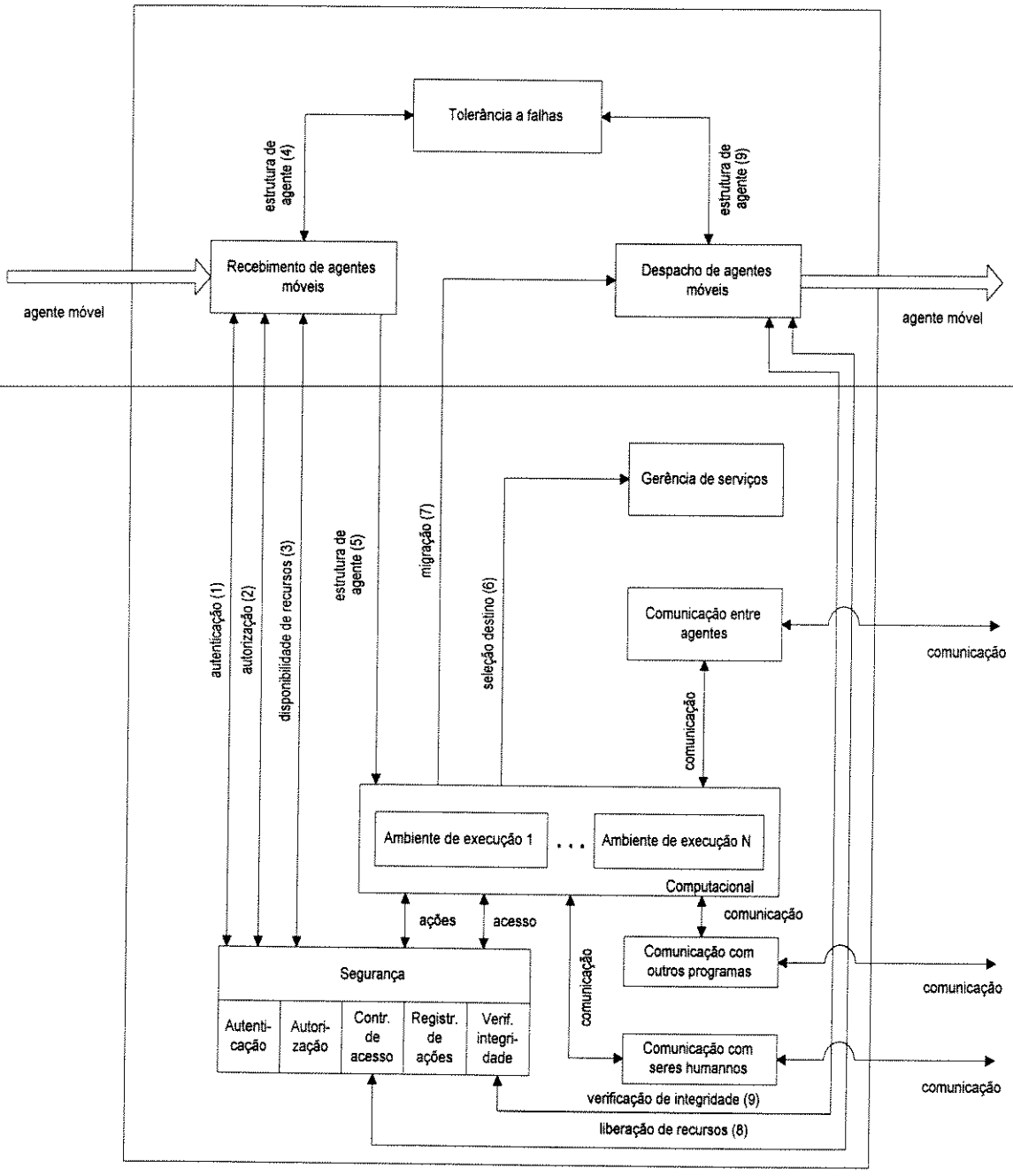

Figura 2.7: Decomposição funcional de infra-estruturas para suporte a SBAMs

Os seguintes blocos funcionais podem ser reconhecidos na figura 2.7: segurança, recebimento de agentes móveis, tolerância a falhas, computacional, comunicação, gerência de serviços e despacho de agentes móveis.

O bloco de segurança disponibiliza os serviços de autenticação, autorização, controle de acesso, registro de ações e verificação de integridade. Há uma forte interação deste bloco com o sistema operacional em execução num nó.

O bloco de recebimento de agentes móveis recebe de um mecanismo de transporte a estrutura codificada de um agente. Em seguida a estrutura é descodificada, verifica-se se o dono do agente é reconhecido e se ele tem permissão de execução num dado nó (interagindo com o bloco de segurança). Os recursos requeridos por um agente são também verificados. Caso os passos anteriores tenham alcançado sucesso, a estrutura do agente é repassada ao bloco de tolerância a falhas e é também entregue ao ambiente de execução apropriado.

O bloco de tolerância a falhas visa prover meios para que agentes e as informações que eles carregam possam ser recuperados em caso de falha na rede ou no nó que os abriga, de forma que o sistema possa voltar à sua execução normal o mais rápido possível. Persistência de agentes é uma estratégia para suportar tolerância a falhas, onde agentes são armazenados em memória não volátil para possível recuperação.

O bloco computacional é composto por vários ambientes de execução que podem suportar diversos modelos computacionais. Agentes são instanciados e passam a ter vida própria quando entregues ao ambiente de execução apropriado. A partir de instruções executadas pelos agentes, o bloco computacional interage com os blocos de segurança (para registro de ações e controle de acesso), de comunicação (para prover comunicação entre agentes, agentes e seres humanos e agentes e outros programas), de gerência de serviços (para seleção de destinos de viagem), e de despacho de agentes móveis (para possibilitar a migração de agentes).

O bloco de comunicação está dividido em três partes: comunicação entre agentes, comunicação com outros programas (por exemplo, sistemas de gerência de bases de dados), e comunicação com seres humanos (por exemplo, via correio eletrônico). A comunicação entre agentes deve suportar as modalidades síncrona e assíncrona, local e remota e também permitir a interação entre agentes em diferentes ambientes de execução e em diferentes infra-estruturas.

O bloco de gerência de serviços é um sistema distribuído de oferta e busca global de serviços, para prover a seleção de destinos de migração.

O bloco de despacho de agentes móveis interage com o bloco de segurança para a liberação dos recursos em uso pelo agente que deseja migrar e para a verificação da integridade da estrutura do agente. É responsável pela entrega desta estrutura ao bloco de tolerância a falhas, pela codificação da estrutura para transporte e pelo envio da estrutura codificada para o destino desejado via mecanismo de transporte.

Na figura 2.7, os passos numerados de 1 a 5 são tomados para o recebimento e execução de agentes. Os passos de 6 a 9 tratam do despacho de agentes de um nó.

2.6. Padronização

Interoperabilidade é a habilidade que um sistema tem de interagir com outros sistemas de forma transparente. Para que um SBA contenha esta habilidade, sua infra-estrutura de suporte deve se basear em padrões abertos, não proprietários. Como sistemas baseados em agentes é uma área emergente, esforços de padronização estão começando a surgir. Dentre eles pode-se destacar a Fundação para Agentes Físicos Inteligentes[13] (FIPA) [FIPA97] e Sociedade de Agentes[14] [Agen97].

A FIPA é uma associação internacional sem fins lucrativos de companhias e organizações que visa promover o desenvolvimento e a elaboração de especificações de tecnologias de agentes, utilizáveis por um grande número de aplicações, provendo a elas um alto nível de interoperabilidade. Trabalhos de elaboração de especificações têm sido conduzidos pela FIPA nas seguintes áreas: gerência de agentes, linguagem de comunicação de agentes, integração de agentes com outro software, assistentes pessoais e gerência de redes.

A Sociedade de Agentes é uma organização internacional, sem propósitos lucrativos, de indústrias e profissionais estabelecida para colaborar no desenvolvimento e surgimento de mercados e tecnologias de agentes inteligentes. Dentre suas atividades destacam-se:

- desenvolver e implementar interfaces e protocolos abertos para intercomunicação e interoperabilidade entre agentes diversos. Nesta linha, estão sendo desenvolvidos um modelo de referência para sistemas abertos baseados

[13] *"Foundation for Intelligent Physical Agents"*
[14] *"Agent Society"*

em agentes, uma proposta de uma plataforma comum para SBAs e um protocolo simples de transferência de agentes;

- facilitar a colaboração e a transferência de tecnologia através da troca de informações sobre desenvolvimentos na área de SBAs.

No campo de SBAMs, um outro esforço de padronização é de grande relevância: a especificação de uma Facilidade para Agentes Móveis[15] (MAF) pelo Grupo de Gerência de Objetos[16] (OMG) [OMG97]. O OMG é uma organização composta por mais de 750 membros contendo vendedores, desenvolvedores e usuários de software. Estabelecido em 1989, sua missão é promover a teoria e a prática de tecnologia orientada a objetos para o desenvolvimento de sistemas computacionais distribuídos. Para tanto, o OMG tem como meta estabelecer uma arquitetura comum para aplicações orientadas a objetos, baseando-se em especificações amplamente disponíveis.

A Arquitetura de Gerência de Objetos[17] do OMG contém quatro componentes principais:

- intermediário de requisição de objetos[18] (ORB): um barramento de software que permite que objetos possam interoperar num ambiente distribuído. Por interoperar entende-se que objetos podem fazer transparentemente requisições para outros objetos locais ou remotos e deles receber respostas;

- serviços comuns de objetos: serviços no nível de sistema, que complementam a funcionalidade de um ORB. Alguns desses serviços, atualmente padronizados pelo OMG, incluem: ciclo de vida (operações para criação, cópia, migração e remoção de objetos), nomes (localização de objetos por nome) e persistência (armazenamento de objetos em memória não volátil);

- facilidades comuns: serviços de uso direto por objetos de aplicação. Contêm especificações de serviços de alto nível divididos em duas categorias: horizontais (serviços necessários na maioria dos domínios de aplicação), e verticais (serviços para áreas especializadas de mercado).

- objetos de aplicação: objetos específicos que compõem aplicações de usuários.

---

[15] *"Mobile Agent Facility"*
[16] *"Object Management Group"*
[17] *"Object Management Architecture"* (OMA)
[18] *"Object Request Broker"*

O ORB é o núcleo da arquitetura de gerência de objetos. Sua especificação é chamada de CORBA[19]. Objetos CORBA desempenham o papel de clientes quando requisitam uma operação, ou de servidores quando recebem, atendem e respondem às requisições. A figura 2.8 ilustra a interação entre cliente e servidor via ORB.

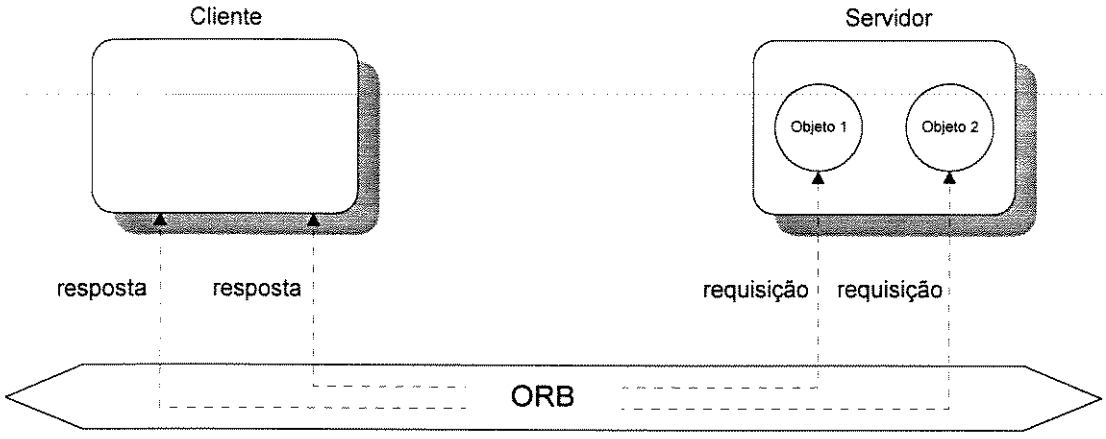

Figura 2.8: Interação entre cliente e servidor via ORB

Objetos CORBA interoperam através da especificação de interfaces entre eles escritas numa linguagem neutra de definição de interfaces[20] (IDL). Em IDL são descritos os limites de um objeto, bem como as suas interfaces com potenciais clientes. IDL é independente de implementação, entretanto os métodos nela especificados podem ser traduzidos para linguagens de programação que contenham mapeamentos CORBA[21] (correntemente apenas Java, C, C++ e Smalltalk).

MAF visa estender as capacidades de um ORB para dar suporte à implementação de agentes móveis. Recentemente foi feita pelo OMG uma requisição de propostas (ORBOS RFP 11) para a especificação de uma MAF. Em junho de 1997, uma proposta conjunta foi submetida ao OMG por instituições importantes que têm atuado no campo de SBAMs desde o seu surgimento. Tais instituições englobam a Crystaliz, Inc. [Crys97], General Magic, Inc. [Gene96], GMD FOKUS [GMD97] e IBM Corporation [IBM96a]. Esta proposta conjunta é examinada no próximo capítulo (seção 3.6).

[19] *"Common Object Request Broker Architecture"*
[20] *"Interface Definition Language"*
[21] CORBA *"bindings"*

Capítulo 3

Infra-estruturas para Suporte a Sistemas Baseados em Agentes Móveis

Este capítulo inicialmente apresenta e examina as implementações existentes de infra-estruturas para suporte a sistemas baseados em agentes móveis (SBAMs). Os modelos e aspectos tratados no capítulo anterior são tomados como base na discussão das características particulares de cada infra-estrutura. Finalmente, a proposta conjunta de especificação para uma Facilidade de Agentes Móveis[1] submetida ao Grupo de Gerência de Objetos[2] em 02/06/1997 é discutida. Tal proposta retrata uma visão consensual de instituições importantes correntemente envolvidas no campo de SBAMs, e contempla questões que terão impacto sobre as infra-estruturas existentes.

3.1. Introdução

Esforços de desenvolvimento de infra-estruturas para suporte a SBAMs emergiram a partir de 1994. A tecnologia Telescript [Gene96], baseada na linguagem de mesmo nome, foi concebida e desenvolvida do princípio, enquanto outras infra-estruturas foram construídas baseando-se em linguagens de programação existentes e seus respectivos ambientes de execução. Neste contexto, são exemplos relevantes as linguagens: Obliq [Card95], Scheme [MIT97], Phyton [Phyt97], Perl [Perl97] e Tcl/Tk [SunS97].

Obliq é baseada em Modula 3 e foi desenvolvida na empresa Digital Equipment Corporation. Ela é uma linguagem interpretada, orientada a objetos e contém capacidades de *"multithreading"*. Objetos Obliq são locais, entretanto instruções podem ser despachadas para execução remota, mantendo conexões de rede. Não se trata de uma solução suficientemente abrangente para suporte a SBAMs.

Scheme é uma linguagem baseada em LISP, desenvolvida no Instituto de Tecnologia de Massachussets. Ela contém um conjunto diferenciado de implementações, sendo que Scheme48 é uma versão beta teste reduzida, portável, com base em arquitetura de máquina virtual. Entretanto é utilizada apenas em contextos acadêmicos.

---

[1] *"Mobile Agent Facility"*
[2] *"Object Management Group"*

Phyton é uma linguagem interpretada, orientada a objetos com ambiente de execução disponível em várias plataformas. Porém sua utilização se restringe a um conjunto reduzido de usuários.

Perl é uma linguagem de *"scripts"* eficiente e compacta, voltada para o processamento de textos. Seu escopo é limitado como solução genérica para suporte a SBAMs.

Tcl/Tk também é uma linguagem de *"scripts"*, interpretada, com capacidades de programação gráfica e de fácil extensão. Entretanto, não apresenta uma série de características correntemente desejáveis para suporte a SBAMs, tais como modularização de código, orientação a objetos e *"multithreading"*.

A partir de meados de 1996, a evolução na tecnologia Java [Java97] causou um grande impacto sobre o campo de sistemas distribuídos e consequentemente em SBAMs. A linguagem Java é orientada a objetos, interpretada, portável e contém capacidades de *"multithreading"*. Seu ambiente de execução é baseado numa arquitetura de máquina virtual.

A tecnologia Java engloba facilidades amplas para suporte a segurança e distribuição. Com respeito a este último aspecto, a versão 1.1 introduziu capacidades de extrema relevância: a invocação de método remoto[3] (RMI) e a seriação de objetos[4]. A primeira consiste em um mecanismo de interação entre objetos, onde um objeto pode invocar métodos de um outro objeto remoto de forma síncrona. A segunda diz respeito à codificação e descodificação de classes para transporte. Estas capacidades podem ser empregadas na construção de infra-estruturas para suporte a SBAMs, facilitando a implementação de serviços como migração de agentes e comunicação remota.

As linguagens apresentadas anteriormente e seus respectivos ambientes de execução não reúnem um conjunto expressivo de características e vantagens tal qual a tecnologia Java. Por isso deixaram de ser foco de concentração de esforços de desenvolvimento de infra-estruturas para suporte a SBAMs.

A tabela 3.1 apresenta um conjunto de implementações existentes de infra-estruturas para suporte a sistemas baseados em agentes móveis, sua origem e linguagens suportadas.

---

[3] *"Remote Method Invocation"*
[4] *"Object Serialization"* (OS)

Infra-estrutura	Origem	Linguagem
Tabriz	indústria	Telescript
Odyssey	indústria	Java
Agent Tcl	academia	Tcl/Tk
Aglets Workbench	indústria	Java
Voyager	indústria	Java
TACOMA	academia	C e Tcl/Tk
Mole	academia	Java
Ara	academia	Tcl/Tk e C/C++

Tabela 3.1: Implementações existentes de infra-estruturas para suporte a SBAMs

Dentre as infra-estruturas listadas, TACOMA [Joha95], Mole [Mole97] e Ara [Ara97] são esforços de pesquisa em contexto acadêmico que necessitam de maior amadurecimento e exposição externa.

Tendo em vista o quadro apresentado até aqui, as implementações correntemente existentes de infra-estruturas para suporte a SBAMs de maior relevância são: Tabriz [Gene96] e Odyssey, Agent Tcl [Gray95], Aglets Workbench [IBM96a] e Voyager [Obje97]. Tais infra-estruturas serão examinadas com detalhe na seqüência, tendo como base os modelos (agentes, ciclo de vida, computacional, segurança, comunicação e navegação) e aspectos tratados no capítulo anterior.

Embora estando ainda em fase inicial de desenvolvimento, cabe fazer uma nota aos projetos Agentes e Objetos Móveis [MOA97] e MAGNA [MAGN97], que podem trazer contribuições importantes no que diz respeito ao assunto tratado neste capítulo.

3.2. Tabriz e Odyssey

A infra-estrutura Tabriz para suporte a sistemas baseados em agentes móveis foi apresentada pela General Magic no ano de 1994 e se tornou a primeira do gênero disponível comercialmente. Sua arquitetura é composta por uma linguagem de programação chamada Telescript, um interpretador para Telescript e protocolos de comunicação para transporte de agentes. A figura 3.1 ilustra tal arquitetura.

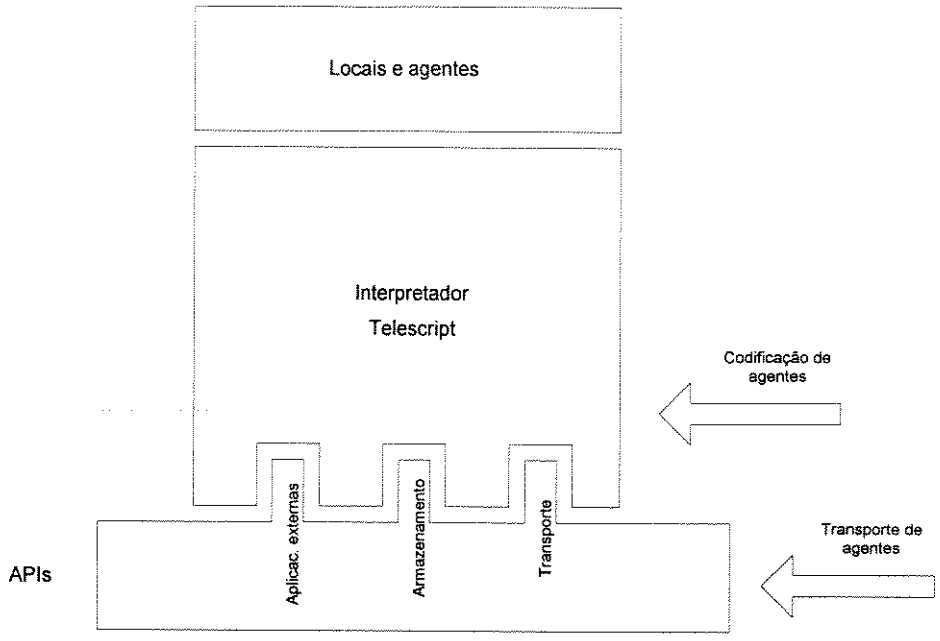

Figura 3.1: Arquitetura da infra-estrutura Tabriz

A linguagem Telescript é orientada a objetos e compilada para uma linguagem portável interpretada (de mesmo nome). Ela contém três conceitos principais: agentes, locais[5] e viagem. Locais compõem uma rede de computadores e oferecem serviços a agentes que neles são recebidos. Agentes se movem viajando de um local para outro. Viagens se dão através da instrução *go*.

O interpretador implementa a linguagem Telescript através da manutenção e execução de locais e agentes que ocupam tais locais. Os protocolos permitem que dois interpretadores se comuniquem de forma que agentes possam ser transportados na ocorrência da execução da instrução *go*.

No modelo de agentes da infra-estrutura Tabriz, um agente é implementado como um objeto Telescript. O aspecto de delegação é tratado através da atribuição de autoridades a agentes. Autoridades identificam pessoas ou organizações do mundo real que os agentes representam. Agentes são capazes de discernir autoridades, e não é permitido anonimato nem falsificação de autoridades. Tabriz implementa o modelo de processo persistente para ciclo de vida, onde o estado de execução do agente móvel é mantido quando o agente viaja de um local para outro.

[5] *"places"*

As características da linguagem Telescript apresentadas anteriormente descrevem o modelo computacional desta infra-estrutura. O modelo de segurança é implementado através de autoridades, regiões, identidades e permissões. Uma região é um conjunto de locais que são operados pela mesma autoridade. Identidades distinguem agentes e locais com mesma autoridade. Controle de acessos é realizado via permissões. Um programa Telescript, executado num interpretador, não tem acesso direto ao processador, à memória, aos dispositivos periféricos e ao sistema de arquivos presentes em um nó.

Comunicação entre agentes é implementada através dos conceitos de encontro[6] e conexão. Encontros permitem que agentes possam chamar procedimentos de outros agentes presentes num mesmo nó. Conexões permitem comunicação remota entre agentes. Correio eletrônico e *"paging"* são suportados para comunicação entre agentes e seres humanos. Código C pode ser adicionado a agentes para permitir a comunicação com outros sistemas.

O modelo de navegação se baseia nos conceitos de local e viagem, bem como na instrução *go*. Não há mecanismo de seleção de destinos. Os protocolos de comunicação entre interpretadores são responsáveis pelo transporte de agentes e pela sua codificação e descodificação. Tabriz suporta os protocolos TCP/IP, X.25 e SMTP (correio eletrônico) para comunicação entre interpretadores.

Tolerância a falhas é abordada através de persistência. Agentes Telescript e a informação que eles carregam consigo são armazenados em memória não volátil onde quer que eles estejam.

A infra-estrutura Tabriz é robusta e abrangente. Entretanto é uma tecnologia proprietária, cara e que requer recursos computacionais significativos. Assim sendo, a General Magic recentemente descontinuou o produto, partindo para o desenvolvimento de uma alternativa baseada na linguagem Java, que foi chamada de Odyssey.

Odyssey implementa parcialmente a funcionalidade e os conceitos da infra-estrutura Tabriz. Sua versão corrente é 1.0 beta 2. Odyssey tem como característica relevante o suporte aos seguintes mecanismos de transporte: RMI da linguagem Java, Modelo de Objetos Comum Distribuído[7] da Microsoft e Protocolo Internet para comunicação entre

[6] *"meeting"*
[7] *"Distributed Common Object Model"* (DCOM)

ORBs[8] do Grupo de Gerência de Objetos[9]. Trata-se de um esforço em fase inicial, cujas capacidades não podem ser completamente exploradas pela falta de documentação.

3.3. Agent Tcl

Agent Tcl é uma infra-estrutura em desenvolvimento na Faculdade Dartmouth, construída como extensão sobre o sistema de *"scripts"* Tcl/Tk. O núcleo de tal sistema foi alterado para que o estado interno de um *"script"* em execução pudesse ser capturado. Devido ao fato de ser desenvolvida sobre Tcl/Tk, Agent Tcl é independente de plataforma. Sua implementação correntemente disponível, versão 1.1, ainda é funcionalmente limitada, isto é, apenas pequena parte da arquitetura proposta está operacional. Em breve a versão 1.2 deverá ser lançada contendo avanços nesse sentido.

A arquitetura da infra-estrutura Agent Tcl, ilustrada pela figura 3.2, tem quatro níveis. O nível mais baixo é uma interface de programação[10] para mecanismos de transporte disponíveis. O segundo nível é um servidor que deve ser executado em cada nó de uma rede, responsável por tarefas tais como:

- rastrear agentes em execução no nó onde reside;
- receber agentes que chegam num nó, autenticando a identidade de seus donos e passando o agente autenticado para o interpretador adequado;
- prover um espaço de nomes para agentes, permitindo que agentes possam trocar mensagens neste contexto;
- prover acesso à memória não volátil para que agentes possam armazenar seu estado interno. Tal estado pode ser recuperado no caso de falhas.

O terceiro nível da arquitetura consiste em um interpretador para cada linguagem disponível. Um interpretador tem 4 componentes: o próprio interpretador, um módulo de segurança, um módulo de estado (que captura e restabelece o estado interno de um agente em execução), e uma interface de programação (que interage com o servidor para tratar de migração, comunicação e verificação). O último nível da arquitetura é composto por agentes.

[8] *"Internet Inter-ORB Protocol"* (IIOP)
[9] *"Object Management Group"*
[10] *"Application Programming Interface"*

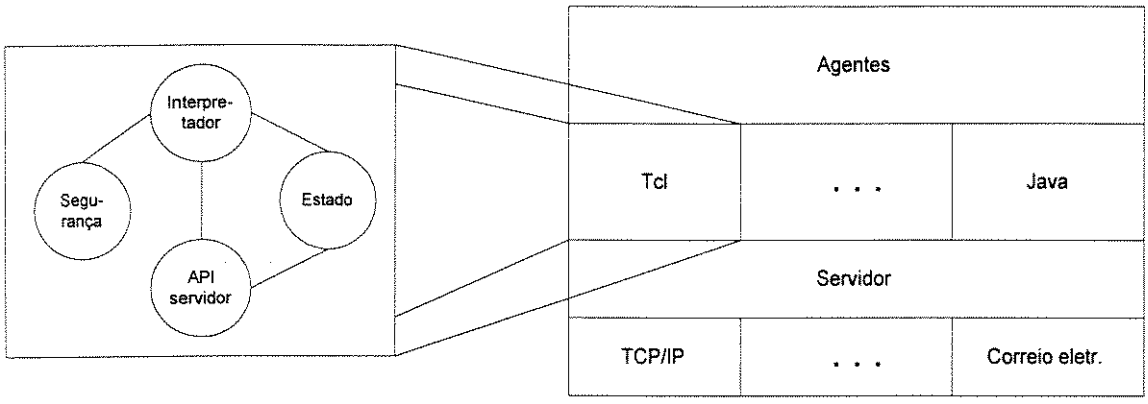

Figura 3.2: Arquitetura da infra-estrutura Agent Tcl

As modificações realizadas sobre a linguagem Tcl englobam instruções que são utilizadas por agentes para migração, comunicação e criação de agentes filhos.

O modelo de agentes da infra-estrutura Agent Tcl é o *"script"* Tcl estendido, e seu modelo computacional está baseado nas características do sistema Tcl/Tk apresentadas anteriormente. Esta infra-estrutura implementa o modelo de processo persistente para ciclo de vida, onde o estado de execução do agente móvel é capturado e transportado quando o agente viaja.

O modelo de segurança é atualmente provido de forma rudimentar. Um conjunto de nós autorizados é estabelecido por um administrador e qualquer agente, mensagem ou conexão que não tiver origem naqueles nós é ignorado.

A comunicação entre agentes é implementada de duas formas: mensagens (para comunicação assíncrona), e encontros[11] (para transferência síncrona de dados). As facilidades para construção de interfaces gráficas providas pela linguagem Tk podem ser utilizadas para a comunicação com seres humanos.

O modelo de navegação está baseado na execução da instrução *agent_jump*. Quando tal instrução é executada no código de um agente, o seu estado interno é capturado e transportado para outro nó da rede, através do módulo de estado do interpretador. O servidor no nó destino é responsável por receber o agente e repassá-lo ao interpretador adequado. Correntemente, apenas o protocolo TCP/IP é suportado para transporte. Não há mecanismos de seleção de destinos.

[11] *"meetings"*

Tolerância a falhas não é correntemente suportada. A implementação completa da funcionalidade de servidores proverá capacidades básicas para suporte a persistência.

3.4. Aglets Workbench

A infra-estrutura Aglets Workbench (AWB) permite a manipulação de agentes móveis escritos em Java. Ela foi desenvolvida pela IBM no Laboratório de Pesquisa de Tóquio e disponibilizada para uso externo a partir de meados de 1996. Atualmente, encontra-se na versão alfa teste 5.

A arquitetura desta infra-estrutura consiste dos seguintes componentes:

- interface de programação[12] (API) Aglet;
- camada de execução Aglets;
- interface de transporte e comunicação de agentes[13] (ATCI);
- camada de transporte.

A API Aglet contém uma série de classes e interfaces Java que implementam abstrações do modelo de objetos Aglet. As abstrações mais relevantes presentes neste modelo são:

- aglet: um objeto capaz de se mover autonomamente de um nó para outro numa rede Internet;
- contexto: provê meios para manter e gerenciar aglets em execução num ambiente uniforme independente da arquitetura da máquina em que se encontram;
- proxy: um invólucro, ou escudo, que provê um meio comum para acesso a um aglet, protegendo-o de ações indevidas;
- mensagem: um objeto trocado entre dois aglets, utilizado para comunicação.

A camada de execução Aglets é a implementação da API Aglet. A camada de transporte é responsável pelo transporte de um agente para o destino desejado na forma de

[12] *"Application Programming Interface"*
[13] *"Agent Transport and Communication Interface"*

uma seqüência de bytes que contém definições de classes, bem como o estado seriado do agente (utilizando-se da técnica de seriação de objetos, presente na tecnologia Java). Esta camada também é definida como uma API, chamada de interface de transporte e comunicação de agentes, e permite que a camada de execução Aglets faça uso da camada de transporte de uma maneira independente de protocolo.

A implementação da ATCI é responsável pelo envio e recebimento de agentes, bem como pelo estabelecimento de comunicação entre eles. A versão corrente da AWB usa o Protocolo de Transferência de Agentes[14] (ATP), que é um protocolo no nível de aplicação, para a transmissão de agentes móveis. O ATP é modelado sobre o protocolo HTTP, e pode ser utilizado para transferir o conteúdo de um agente de uma forma independente de infra-estrutura para suporte a SBAMs. Para possibilitar a comunicação entre agentes, o ATP também suporta troca de mensagens.

Do ponto de vista operacional, a infra-estrutura Aglets Workbench contém facilidades para sua utilização. Dentre elas, destacam-se o gerente gráfico de agentes chamado Tahiti e o integrador da AWB com a tecnologia *"World Wide Web"*, de nome Fiji. Tahiti permite que algumas funcionalidades de agentes móveis aglets possam ser executadas via interface gráfica, tais como: criação, término e migração. Fiji permite que applets Java também sejam capazes de manipular agentes aglets.

O modelo de agentes da AWB é o objeto aglet. Esta infra-estrutura implementa o modelo de ciclo de vida baseado em tarefas. O estado de execução não é migrado juntamente com o código e os dados do agente. Os eventos presentes no modelo de ciclo de vida da Aglets Workbench são:

- criação: ocorre dentro de um contexto. Um identificador único é atribuído ao aglet criado, que é inserido no contexto e iniciado;
- clonagem: uma cópia quase idêntica de um aglet é criada no mesmo contexto. As diferenças estão nos identificadores e no fato da execução ser reiniciada;
- despacho: um aglet é transportado de um contexto para outro, onde sua execução é reiniciada;
- recolhimento: um aglet é trazido de um contexto para outro onde a requisição de recolhimento foi feita. É inverso ao evento de despacho;

[14] *"Agent Transfer Protocol"*

- desativação: armazenamento em memória não volátil de um aglet;
- ativação: recuperação de um aglet armazenado em memória não volátil para um contexto;
- término: a execução de um aglet é terminada e o agente é removido de seu contexto.

A figura 3.3 ilustra os eventos que compõem o modelo de ciclo de vida da infra-estrutura Aglets Workbench.

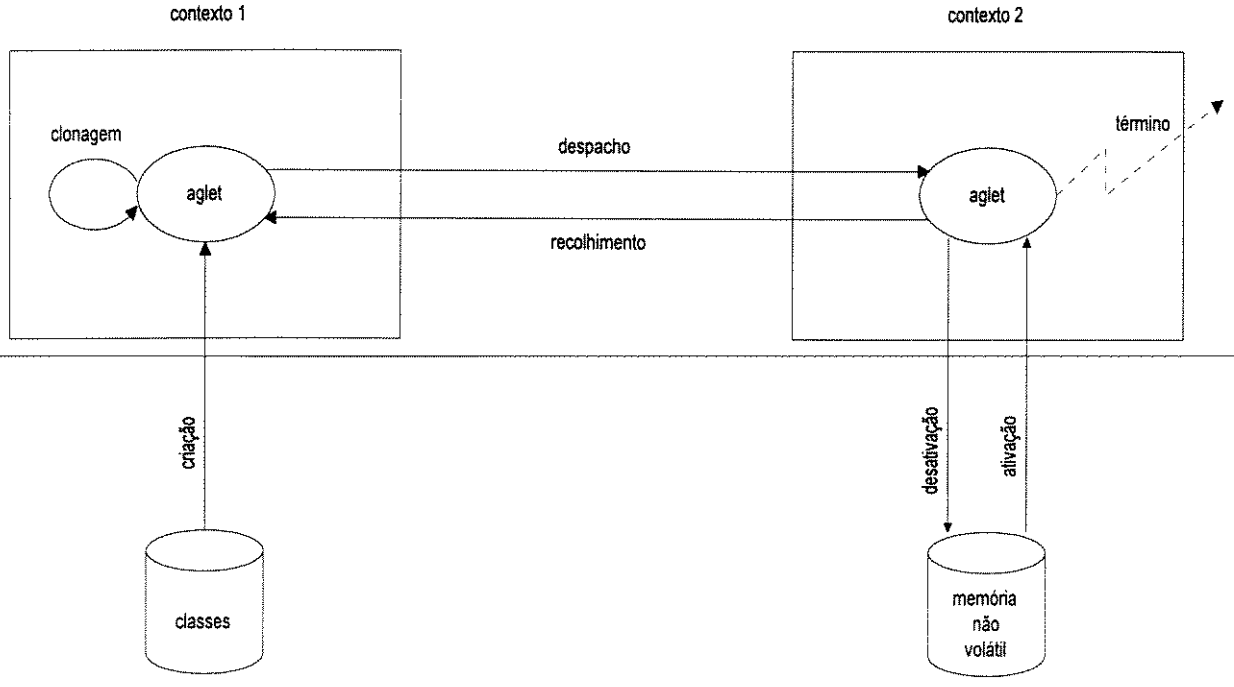

Figura 3.3: Eventos que compõem o modelo de ciclo de vida da AWB

O modelo computacional da AWB está baseado nas características do ambiente de execução da tecnologia Java descritos anteriormente. Entretanto, o carregador de classes Java é substituído por um carregador particular da infra-estrutura Aglets Workbench. Este é responsável por carregar dinamicamente, quando necessário, código interpretável[15] Java referente às definições de classes de agentes aglet e de outros objetos manipulados por aglets. O carregador de classes Aglet é acionado em eventos como criação, execução e migração de agentes.

[15] *"bytecode"*

Outra característica particular do modelo computacional é a abstração contexto, que provê um ambiente uniforme onde aglets são executados independentemente da arquitetura do sistema no qual eles se encontram.

O modelo de segurança da infra-estrutura AWB faz uso das características suportadas pela linguagem Java, como por exemplo a checagem de consistência realizada pelo verificador de *"bytecode"*. Um esquema de nomes únicos global para agentes permite sua identificação de forma não ambígua. Adicionalmente, esta infra-estrutura permite que aglets possam ser considerados como confiáveis ou não. A definição deste atributo está estritamente relacionada ao controle de acesso de recursos num dado nó. Finalmente um gerente de segurança é provido, contendo uma série de métodos específicos para ações relativas à segurança.

A comunicação entre agentes é realizada na infra-estrutura Aglets Workbench através do mecanismo de troca de mensagens, englobando os modos síncrono e assíncrono. Agentes podem se comunicar localmente ou remotamente. O protocolo ATP é utilizado para suportar o mecanismo de troca de mensagens.

As capacidades de construção de interfaces gráficas oferecidas pelo Pacote de Janelas Abstratas[16] (AWT) da tecnologia Java permitem a interação entre seres humanos e agentes.

O modelo de navegação está baseado na abstração chamada itinerário que pode conter um conjunto de destinos de migração dinamicamente assinalado. Aglets também podem migrar através da instrução *dispatch*. O protocolo ATP é um componente chave do modelo de migração, possibilitando que agentes possam ser enviados e recebidos por servidores. Aglets são transformados numa seqüência de bytes quando são migrados.

A técnica de seriação de objetos é utilizada para transportar os dados que compõem o estado de um aglet, bem como objetos que são manipulados pelo agente. As definições de classes de agentes e dos objetos por eles manipulados são carregadas dinamicamente durante a migração através do Carregador de Classes Aglet[17]. Não há mecanismos de seleção de destinos.

A figura 3.4 ilustra a migração de um agente na infra-estrutura AWB.

[16] *"Abstract Windowing Toolkit"*
[17] *"Aglet Class Loader"*

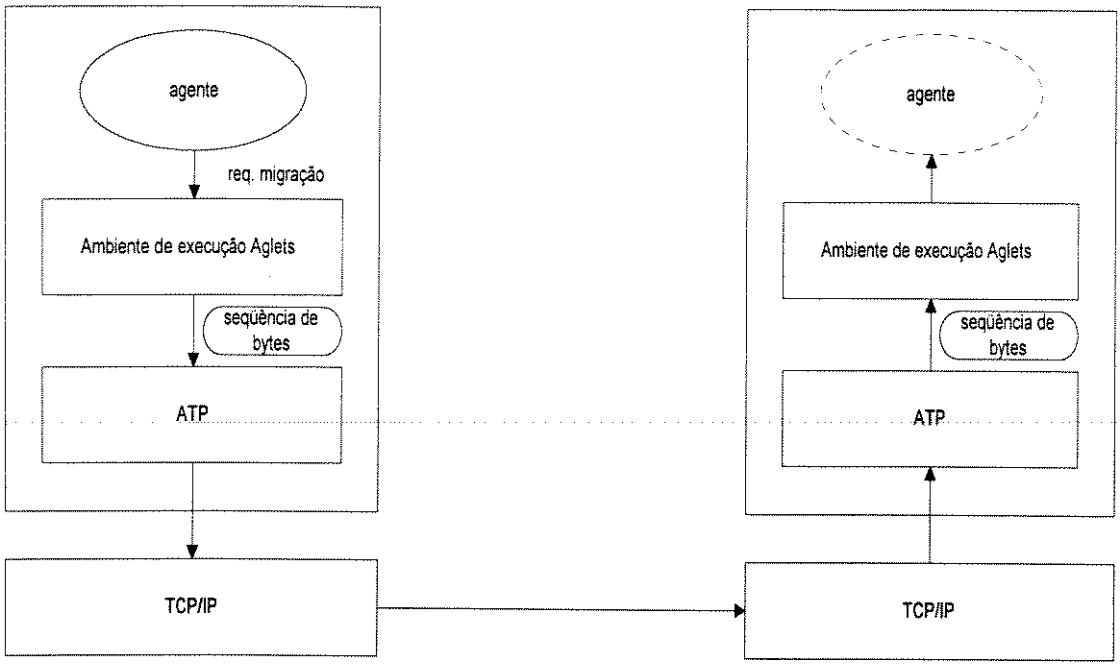

Figura 3.4: Migração de agentes na Aglets Workbench

Quanto ao aspecto de tolerância a falhas, a AWB não o trata em nenhum nível. Entretanto, as capacidades de desativação e ativação podem ser utilizadas pelo desenvolvedor para prover persistência e recuperação.

Aglets Workbench é uma das infra-estruturas para suporte a SBAMs mais utilizadas correntemente. Tal fato é conseqüência de uma série de fatores, tais como: um modelo abrangente e bem projetado, facilidade de operação, boa documentação, grande exposição a contextos externos ao seu ambiente de desenvolvimento e utilização da tecnologia Java. Entretanto, a IBM não definiu seus planos futuros de cessão e comercialização da AWB.

3.5. Voyager

Voyager é uma plataforma baseada na tecnologia Java cujo principal componente é um intermediário de requisições de objetos[18] (ORB) que provê suporte a objetos e agentes móveis. Além do ORB, serviços de persistência, comunicação em grupo e diretório de nomes fazem parte desta plataforma. A versão corrente é a 1.0, lançada como produto em setembro de 1997 pela empresa ObjectSpace.

[18] *"Object Request Broker"*

Um conceito fundamental introduzido por esta infra-estrutura é o de tornar uma classe *habilitada para uso remoto*[19]. Isto ocorre quando instâncias de uma classe podem ser criadas fora do espaço de endereçamento local de um programa, e caso tais instâncias sejam capazes de receber mensagens como se fossem locais.

A implementação do conceito de habilitação para uso remoto se dá através de classes virtuais que se constituem como estruturas centrais da plataforma Voyager. O utilitário vcc[20] é responsável pela geração de classes virtuais a partir de qualquer código fonte ou interpretável Java.

Instâncias de classes virtuais (objetos virtuais) podem ser criadas remotamente. Quando uma instância é criada, ela recebe um identificador global único. Sua identificação é registrada no serviço de diretórios da plataforma. Se o código da classe original não estiver disponível remotamente, ele é carregado automaticamente pelo carregador de classes em rede. O acesso a objetos virtuais se dá através de referências virtuais.

O serviço de diretórios de nomes é outro componente da plataforma que permite que objetos remotos possam ser encontrados, sem conhecimento prévio de sua localização. Ele se baseia num *"alias"* dado a objetos. Diretórios em rede também podem ser criados e conectados para formar um serviço de nomes federativo em grande escala.

O suporte à comunicação em grupo é realizado através do conceito de espaço. Neste conceito, grupos locais de objetos são agregados em subespaços, que por sua vez constituem um grupo lógico de maior escala chamado de espaço. Quando uma mensagem é recebida num subespaço, cópias idênticas são feitas e enviadas para os subespaços mais próximos antes de que a mensagem seja entregue a cada objeto no subespaço local.

A comunicação entre objetos é proporcionada através de troca de mensagens. Mensagens são enviadas para referências virtuais de objetos que, por sua vez, as encaminham para os objetos remotos associados. Mensagens são divididas em duas categorias: síncronas (o envio da mensagem bloqueia o remetente até que o valor de retorno seja recebido) e assíncronas. Na seqüência estão listados os tipos de mensagens assíncronas:

- sem retorno: o envio da mensagem não bloqueia o remetente e a mensagem não tem valor de retorno;

[19] *"remote-enabled"*
[20] *"virtual code compiler"*

- futura: o envio da mensagem não bloqueia o remetente e o valor de retorno é obtido posteriormente;
- multidestinos sem retorno: todos os objetos de um espaço recebem a mensagem, que não tem valor de retorno;
- multidestinos seletiva: um subconjunto de objetos de um espaço recebe a mensagem.

Qualquer objeto virtual seriado pode migrar de um programa para outro na plataforma Voyager. O mecanismo de seriação é uma característica da linguagem Java. A migração é ativada pelo recebimento da mensagem *moveTo*. Antes de um objeto migrar, ele deixa um objeto especial local[21] para repassar mensagens e lidar com requisições futuras de conexões.

Um agente é um tipo especial de objeto na infra-estrutura Voyager, que contém todas as características de objetos descritas anteriormente. Entretanto, um agente pode se mover autonomamente. Isto se dá quando ele envia a mensagem *moveTo* para si mesmo, contendo o destino da migração e o método a ser executado quando chegar no seu destino.

O modelo de agentes da infra-estrutura Voyager é o objeto agente tendo como base o conceito de objeto virtual. O estado de execução não é migrado juntamente com um agente. A ação a ser executada no destino é especificada antes da migração. Portanto, o modelo de ciclo de vida é baseado em tarefas.

Quanto ao modelo computacional, Voyager se fundamenta nas características do ambiente de execução da tecnologia Java. O carregador de classes em rede é responsável por carregar automaticamente códigos de classes não disponíveis localmente.

O modelo de segurança é implementado através de um gerente de segurança, seguindo o conceito da tecnologia Java. A sua instalação é opcional e sua funcionalidade básica engloba os aspectos de autenticação, autorização e controle de acesso. Quando um objeto é criado, ele recebe um identificador global único. Sua identificação é registrada no serviço de diretórios da infra-estrutura.

A comunicação entre agentes é realizada através do mecanismo de troca de mensagens. Os tipos de mensagens suportados pela Voyager foram descritos

[21] *"forwarder"*

anteriormente. Comunicação em grupo é efetuada de forma eficiente fazendo-se uso do conceito de espaço.

Se um agente deseja interagir eficientemente com um objeto remoto, ele pode "migrar para aquele objeto" e então enviar mensagens Java locais. Isto se dá quando o agente envia a mensagem *moveTo* para si mesmo, contendo a referência virtual do objeto remoto e o método a ser executado quando chegar no seu destino. As capacidades de construção de interfaces gráficas oferecidas pelo AWT da tecnologia Java permitem a interação entre seres humanos e agentes.

O modelo de navegação está baseado no tratamento da mensagem *moveTo*. Qualquer agente (objeto virtual seriado autônomo) pode migrar de um programa para outro na plataforma Voyager. Apenas código e dados internos são migrados. Não há mecanismo de seleção de destinos, e o protocolo TCP/IP é utilizado para transporte.

Tolerância a falhas é suportada pela infra-estrutura Voyager através de persistência. Um objeto persistente contém uma cópia de segurança em uma base de dados. Voyager inclui um sistema de armazenamento de objetos de alto desempenho chamado VoyagerDb. Além disso, esta infra-estrutura provê uma camada de interface para permitir que desenvolvedores possam fazer uso de qualquer outra base de dados, seja ela relacional ou orientada a objetos.

Das infra-estruturas examinadas, Voyager contém a melhor documentação. A sua utilização é sem custos para uso comercial, com algumas restrições. Trata-se de uma infra-estrutura robusta, bem projetada e com funcionalidades amplas. Versões beta teste foram colocadas a disposição para uso externo desde abril de 1997.

Entretanto, o ORB Voyager não implementa a especificação aberta CORBA[22] (seção 2.6). Ele também não é um ORB no sentido em que tal termo é definido naquela especificação, isto é, um barramento de software que permite que objetos possam interoperar num ambiente distribuído.

A integração bidirecional entre Voyager e ORBs que implementam a especificação CORBA está prevista para a próxima versão do produto. A adição desta propriedade será de grande valor para a consolidação da infra-estrutura Voyager.

[22] *"Common Object Request Broker Architecture"*

3.6. Facilidade de agentes móveis

Esta seção apresenta e examina a proposta conjunta de especificação de uma Facilidade de Agentes Móveis[23] (MAF) [MAFS97], submetida ao Grupo de Gerência de Objetos[24] (OMG) em junho de 1997 por um conjunto de instituições importantes que tem atuado no campo de SBAMs. Tais instituições englobam a Crystaliz, Inc. [Crys97], General Magic, Inc. [Gene96], GMD FOKUS [GMD97] e IBM Corporation [IBM96a].

MAF visa estender as capacidades de um intermediário de requisições de objetos (ORB) [25] para dar suporte à implementação de agentes móveis, e será mais um componente das Facilidades Comuns[26] presentes na Arquitetura de Gerência de Objetos[27] do OMG. Facilidades comuns incluem especificações de serviços de alto nível divididos em duas categorias: horizontais (serviços necessários na maioria dos domínios de aplicação), e verticais (serviços para áreas especializadas de mercado).

O primeiro capítulo da proposta conjunta aborda um Modelo Conceitual Comum que reflete a visão consensual dos proponentes a respeito de aspectos fundamentais de sistemas baseados em agentes móveis.

Inicialmente são introduzidos e definidos alguns conceitos básicos. Dentre eles destacam-se:

- agente: um programa de computador que age autonomamente em favor de uma pessoa ou organização. Cada agente tem sua própria *"thread"* de execução de forma que possa executar suas tarefas sob sua própria iniciativa;
- agente móvel: um agente que não está limitado ao sistema onde começou sua execução. Tem a habilidade única de se transportar de um sistema para outro numa rede;
- estado de agente: quando um agente migra, transporta seu código e estado consigo. O estado pode ser seu estado de execução, ou valores de atributos de um agente que podem auxiliá-lo a determinar o que fazer quando a execução é retomada no destino;

---

[23] *"Mobile Agent Facility"*
[24] *"Object Management Group"*
[25] *"Object Request Broker"*
[26] *"Common Facilities"*
[27] *"Object Management Architecture"* (OMA)

- sistema de agentes: é uma plataforma que pode criar, interpretar, executar, transferir e terminar agentes;
- local: quando um agente se transfere, ele migra entre ambientes de execução chamados de locais. Um local é um contexto dentro de um sistema de agentes onde um agente pode ser executado;
- regiões: um conjunto de sistemas de agentes que tem a mesma autoridade, mas que não são necessariamente do mesmo tipo;
- infra-estrutura de comunicações: provê serviços de chamada remota de procedimento, de nomes e de segurança para sistemas de agentes.

A figura 3.5 ilustra alguns dos conceitos descritos anteriormente.

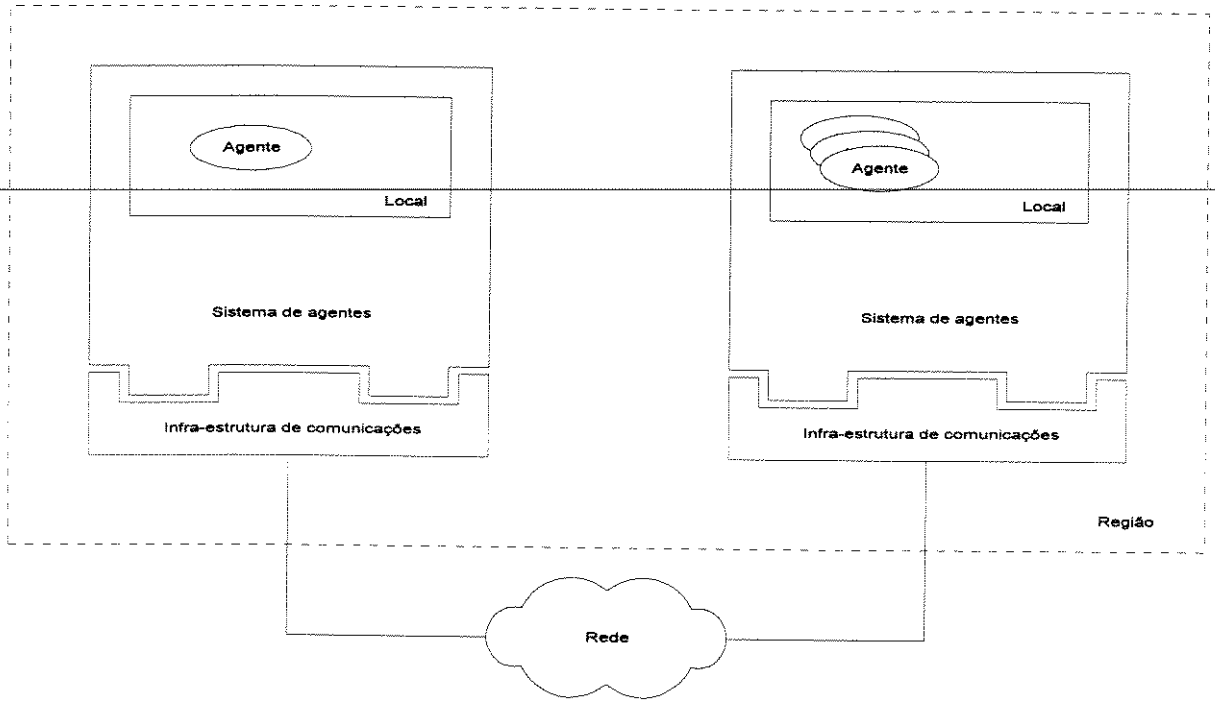

Figura 3.5: Interação entre sistemas de agentes numa mesma região

A definição de estado de agente presente na proposta conjunta não restringe o modelo de ciclo de vida somente ao modelo de estado persistente. Este é um ponto de grande controvérsia na definição do termo agente móvel por pesquisadores da área. Os conceitos de local e regiões foram herdados e estendidos da infra-estrutura Tabriz [Gene96].

Com respeito ainda ao modelo conceitual, a proposta conjunta examina aspectos comuns presentes nas infra-estruturas existentes para suporte a SBAMs. Tais aspectos incluem:

- transferência de um agente: que trata da iniciação da transferência, recebimento de um agente e a transferência de classes (para linguagens que não são orientadas a objetos, corresponde ao código do agente);
- criação de um agente: engloba a criação de uma *"thread"*, a instanciação da classe, geração de um nome global único e início da execução do agente, no contexto de sua *"thread"*;
- fornecimento de localizações e de nomes globais e únicos para agentes: um sistema de agentes deve ser capaz de gerar nomes únicos para si próprio e para agentes e locais que ele cria;
- suporte ao conceito de região: através da interação (cooperação) entre sistemas de agentes de mesma autoridade;
- busca de um agente móvel: para prover comunicação entre agentes e gerência dos mesmos;
- garantia de um ambiente seguro para operações de agentes: um sistema de agentes deve proteger recursos de ações nocivas por parte de agentes. Para tanto, mecanismos como autenticação, controle de acesso e confidencialidade devem ser providos.

Após a apresentação do modelo conceitual, algumas áreas objeto de padronização pela proposta são abordadas, visando prover interoperabilidade entre as diversas infra-estruturas existentes para suporte a SBAMs e o aumento de suas capacidades correntes. Tais áreas são: gerência, rastreamento e transporte de agentes.

Gerência de agentes permite que um agente controle agentes de um outro sistema de agentes. A MAF proposta define interfaces para ações tais como término, suspensão e retomada da execução de agentes. A implementação deste tipo de interoperabilidade é relativamente direta para a maioria das infra-estruturas existentes.

O rastreamento de agentes permite que agentes de diferentes infra-estruturas, registrados num mecanismo proposto, possam ter sua execução rastreada. A

implementação deste tipo de interoperabilidade também é relativamente direta para a maioria das infra-estruturas existentes.

A MAF proposta trata do transporte de agentes através da definição de métodos para o recebimento de agentes e obtenção de suas classes. A implementação deste tipo de interoperabilidade não é simples de ser incorporada pelas infra-estruturas existentes.

Finalmente, os capítulos 2 e 3 da proposta tratam dos detalhes de como a MAF pretende alcançar interoperabilidade. As interfaces da MAF, especificadas em linguagem de definição de interface[28] são apresentadas.

O escopo da proposta conjunta abordada nesta seção ainda é bastante reduzido para o propósito a que ela se destina. Os seus autores atestam que outros aspectos de interoperabilidade devem ser tratados posteriormente, à medida que a indústria alcance maior maturidade, ou que padrões de fato surjam.

As instituições envolvidas na elaboração da proposta já estão preparando mudanças nas infra-estruturas existentes de sua propriedade para atender a especificação contida em [MAFS97]. A IBM, por exemplo, prevê a substituição da ATCI[29] da Aglets Workbench pela interface especificada, no que diz respeito ao transporte de agentes.

[28] *"Interface Definition Language"* (IDL)
[29] *"Agent Transport and Communication Interface"*

em nível de usuário, e vice-versa. Em geral, o jitter tem impacto no nível de sincronização e PER e BER têm impacto sobre a distorção.

A qualidade de serviço em SMDs é função dos recursos alocados para a manipulação de um ou mais meios de representação de informação associado(s) ao serviço. Tais recursos englobam:

- dispositivos de captura e exibição;
- unidade central de processamento;
- memória;
- infra-estrutura de rede;
- software: sistemas operacionais, sistemas para processamento de meios, infra-estrutura para computação distribuída (visto que componentes de um SMD estão geograficamente dispersos) e sistemas de gerência de informação (para manipulação de grandes volumes de dados armazenados em memória não volátil).

Sistemas multimídia distribuídos devem empregar políticas efetivas de gerência de recursos para prover suporte à qualidade de serviço. As políticas dirigem a maneira como recursos são alocados, monitorados, otimizados e liberados. Visto que SMDs são normalmente executados cruzando limites de domínios diferentes, a gerência de recursos deve se basear em mecanismos de negociação. O processo de negociação não é trivial pois pode envolver conflito, isto é, disputa pelos recursos desejados.

O suporte à qualidade de serviço deve ser provido durante todo o ciclo de vida de um sistema multimídia distribuído. Segundo Hafid e Bochmann [Hafi95], tal ciclo pode ser dividido em três fases:

- estabelecimento: onde recursos são alocados para uma sessão;
- gerência: onde recursos são utilizados e gerenciados;
- encerramento: onde recursos são liberados.

A fase de estabelecimento é composta por quatro etapas: especificação de QoS no nível de usuário, mapeamento de parâmetros de QoS no nível de usuário para parâmetros

no nível de sistema, negociação da QoS requerida e reserva de recursos. Na primeira etapa, normalmente são especificados os valores desejados juntamente com margens toleradas.

A fase de gerência contém três etapas:

- monitoramento de QoS: a qualidade de serviço corrente é medida através da obtenção de valores de parâmetros no nível de sistema. A partir dos valores medidos, verifica-se se o SMD está honrando a QoS estabelecida na fase anterior;

- adaptação de QoS ocorre quando a qualidade de serviço alcançada não é a desejada, porém está dentro de limites de tolerância;

- renegociação de QoS ocorre quando há a violação de valores mínimos para parâmetros de qualidade de serviço ou quando o usuário solicita a renegociação. Ela é similar à negociação exceto pelo fato de que o SMD continua sua execução enquanto a renegociação é conduzida.

A partir da análise do ciclo de vida proposto por Hafid e Bochmann [Hafi95], três aspectos importantes relacionados à qualidade de serviço podem ser destacados: o mapeamento de parâmetros no nível de usuário para o nível de sistema, o mapeamento dos parâmetros no nível de sistema para a quantidade de recursos necessários para se atender a QoS desejada, e a negociação/renegociação de recursos conforme necessário.

A realização dos aspectos levantados no parágrafo anterior demanda sistemas que atendam os seguintes requisitos de forma efetiva: distribuição, descentralização, autonomia e flexibilidade. Distribuição é uma conseqüência da dispersão geográfica dos recursos manipulados. Descentralização permite a execução de ações em paralelo, aumentando a eficiência do sistema. Tais ações podem englobar, por exemplo, o mapeamento de parâmetros de QoS nos diversos nós componentes de um SMD.

A negociação e a renegociação podem ser conduzidas de forma eficiente por entidades autônomas, isto é, capazes de tomar suas próprias decisões com respeito a tarefas a elas delegadas no processo de negociação. Flexibilidade é de extrema relevância devido ao caráter dinâmico de sistemas multimídia distribuídos. Neste contexto, parâmetros de QoS e configurações de SMDs podem mudar com freqüência, como resultado de ações de usuários ou dos próprios SMDs.

Propostas iniciais de suporte a QoS em sistemas multimídia distribuídos se caracterizaram por seguirem arquiteturas centralizadas [Camp93]. Mais recentemente, abordagens baseadas no modelo Cliente/Servidor [Kerh94] e em intermediação[4] [Nahr95] foram desenvolvidas. Na primeira, aplicações cliente interagem com servidores (gerentes de recursos) para negociação e monitoramento de recursos. Na abordagem de intermediação, um protocolo de negociação permite a interação entre gerentes de recursos (intermediários) localizados em diferentes nós de uma rede. Um intermediário age como comprador, quando obtendo recursos, ou como vendedor, quando oferecendo recursos disponíveis.

Aurrecoechea et al. [Aurr95] apresentam uma revisão ampla sobre as arquiteturas existentes de suporte a QoS em SMDs mais promissoras. Nenhuma delas é suficientemente abrangente para atender os requisitos descritos anteriormente. A abordagem de agentes é potencialmente capaz de fazê-lo. A próxima seção apresenta um modelo baseado em agentes para suporte a qualidade de serviço em sistemas multimídia distribuídos. Tal modelo é utilizado nesta dissertação como estudo de caso para o emprego de sistemas baseados em agentes móveis.

4.2. Um modelo baseado em agentes para negociação e gerência de qualidade de serviço em sistemas multimídia distribuídos

Oliveira et al. [Oliv97] propõem em seu trabalho um modelo baseado em agentes para a negociação e gerência de qualidade de serviço em sistemas multimídia distribuídos. Tal modelo segue o ciclo de vida de Hafid e Bochmann [Hafi95] e explora as potencialidades de SBAMs como paradigma de desenvolvimento de sistemas distribuídos.

A arquitetura do modelo contém uma agência de qualidade de serviço em cada nó de uma rede onde um dado SMD é executado. Cada agência de QoS é composta por contratos, um servidor de contratos, uma fábrica de agentes, e agentes que podem ser estáticos ou móveis (figura 4.2).

[4] *"brokerage"*

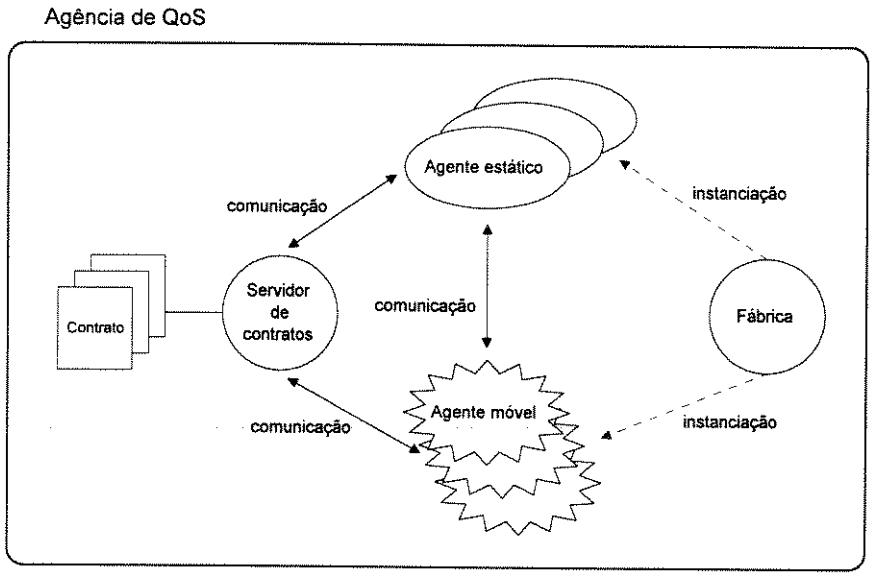

Figura 4.2: Componentes de uma agência de QoS

Um contrato descreve os fluxos que compõem um SMD, bem como os parâmetros de QoS associados aos fluxos. Fluxos são descritos através do tipo do meio associado, o seu nó de origem e os seus nós de destino. Parâmetros são especificados como um conjunto de atributos e valores associados. Um atributo descreve um parâmetro. Os valores contêm a qualidade desejada bem como faixas de tolerância. O servidor de contratos gerencia contratos armazenados em uma base.

Uma fábrica cria e termina agentes estáticos e móveis. O modelo define sete tipos de agentes estáticos e três tipos de agentes móveis. Os agentes pertencentes à primeira categoria são:

- agente de interface: interage com o usuário na fase de estabelecimento do ciclo de vida de um SMD para a especificação da qualidade de serviço desejada, e para informar os valores de QoS obtidos durante a fase de gerência;
- mapeador de QoS: mapea valores de parâmetros de QoS no nível do usuário para o nível de sistema na fase de estabelecimento, e vice-versa na fase de gerência;
- estimador de recursos: executa o mapeamento entre valores de parâmetros de QoS no nível de sistema para os recursos computacionais necessários a atender a qualidade especificada;

- gerente de recursos: é responsável pela gerência de recursos num dado nó. Ele interage com o sistema operacional e com a infra-estrutura de comunicação para reservar, alocar e desalocar recursos;

- negociador local: é responsável pela negociação e renegociação de recursos necessários no nó em que reside. Ele interage com o estimador de recursos para executar sua tarefa;

- monitor de QoS: tem como função medir a qualidade dos fluxos com destino no nó em que reside. Além disso, a partir de valores de parâmetros de QoS relativos a fluxos com origem no nó em que reside, informa outros agentes para que ações cabíveis (por exemplo, adaptação e renegociação) possam ser tomadas;

- adaptador de QoS: tenta corrigir situações onde a qualidade desejada não está sendo honrada, porém está dentro de limites de tolerância. Para tanto, ele interage com o gerente de recursos.

Além de agentes estáticos, três tipos de agentes móveis são definidos por Oliveira et al. [Oliv97]. Estes tipos de agentes têm tarefas com escopo global, isto é, a sua realização depende de ações tomadas em várias agências de QoS. Os agentes móveis são:

- negociador de contrato: negocia um contrato com todos os seus participantes na fase de estabelecimento do ciclo de vida. Ele migra para as agências envolvidas, tentando obter os recursos necessários pelos fluxos que compõem o contrato;

- monitor de contrato: migra, durante a fase de gerência, para as agências envolvidas em um contrato visando obter valores de parâmetros de QoS. É também responsável por relatar para a origem, a qualidade dos fluxos exibidos em seus destinos;

- renegociador de contrato: sua função é similar ao agente negociador de contrato. Entretanto ele realiza sua tarefa na fase de gerência, como resultado da violação de valores mínimos para parâmetros de qualidade de serviço ou quando o usuário solicita a renegociação.

Um cenário de negociação de contrato envolvendo duas agências de QoS, extraído de [Oliv97], é ilustrado na figura 4.3. Nesta figura, agentes estáticos são representados por elipses e agentes móveis por estrelas. Entidades externas são representadas por

retângulos. Setas numeradas indicam interações ordenadas no tempo. A negociação é parte da fase de estabelecimento do ciclo de vida de SMDs proposto por Hafid e Bochmann [Hafi95].

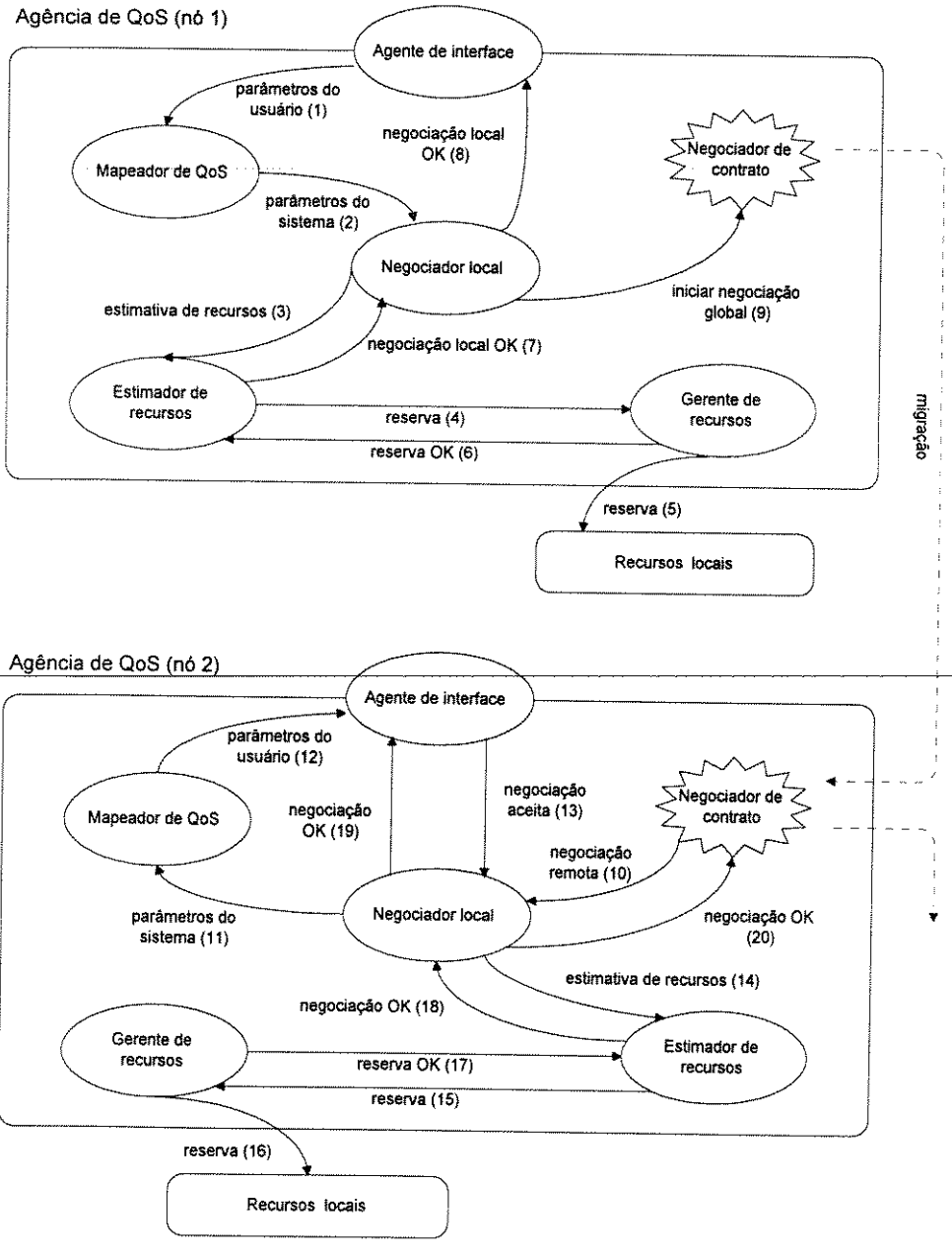

Figura 4.3: Cenário de negociação global de contrato

No cenário de negociação, primeiramente, o usuário de um SMD define um contrato contendo os fluxos e a qualidade de serviço a eles associada. O agente de interface recebe tal informação e passa os valores de parâmetros no nível de usuário, com respeito aos fluxos que envolvem o nó local, para o mapeador de QoS (interação 1). Este, por sua vez,

calcula os valores correspondentes para parâmetros no nível de sistema e os envia para o negociador local (2). O processo de negociação local é então iniciado e engloba as interações de 3 a 7.

O negociador local tenta fazer a reserva dos recursos necessários no nó em que reside, e para isso interage com o estimador de recursos (3). Os recursos estimados são então enviados ao gerente de recursos através de uma requisição de reserva (4). Se os recursos necessários estiverem disponíveis, o gerente de recursos então faz a reserva (5) e notifica o sucesso ao estimador de recursos (6). O processo de negociação local termina quando o agente negociador local é informado a respeito do sucesso da negociação (7).

Em seguida, o resultado da negociação local é apresentado ao usuário pelo agente de interface e um processo de negociação remota (global) é iniciado. Um agente móvel negociador de contrato é responsável por tal processo. Para tanto, ele migra para as agências QoS que se localizam nos outros nós associados aos fluxos do contrato em negociação.

Quando o negociador de contrato chega em uma agência de QoS remota, o processo de negociação é conduzido de uma maneira similar à negociação local (interações de 10 a 20). O agente móvel passa para o negociador local os parâmetros no nível de sistema (10), que podem ser apresentados ao usuário da estação local pelo agente de interface após tratamento pelo mapeador de QoS. O usuário então pode confirmar a negociação (13). Em seguida, o negociador local interage com o estimador de recursos, iniciando uma seqüência de etapas idênticas à negociação local descrita anteriormente (interações de 14 a 18).

Caso a negociação tenha tido sucesso o usuário é notificado pelo agente de interface e o agente negociador de contrato é notificado pelo negociador local (20). O processo de negociação global então continua com a migração do negociador de contrato para a próxima agência de QoS.

Um cenário de gerência de contrato numa agência de QoS, extraído de Oliveira et al. [Oliv97], é ilustrado na figura 4.4. A convenção de representação da figura 4.3 é também adotada para esta figura.

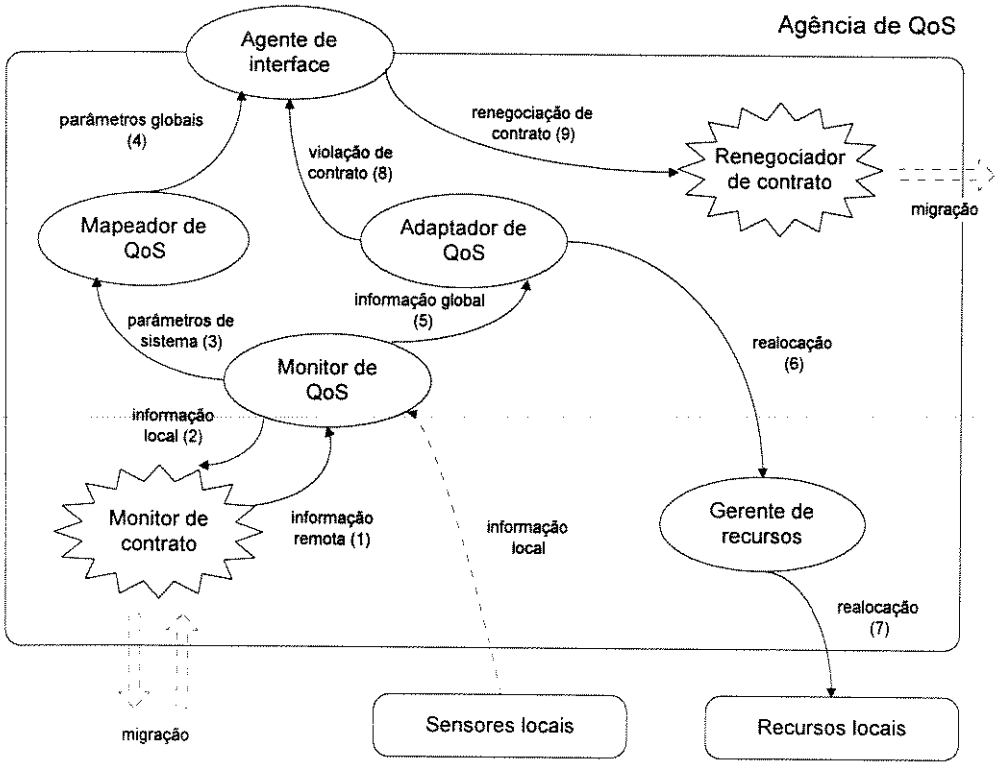

Figura 4.4: Cenário de gerência global de contrato

A gerência é iniciada por um agente monitor de contrato que migra para as agências de QoS localizadas nos nós associados aos fluxos do contrato sob gerência. Ao chegar numa agência, o monitor de contrato interage com o monitor de QoS local. Nesta interação, o agente móvel informa como está a qualidade de serviço nos destinos dos fluxos com origem no nó corrente (1) e recebe informações de QoS dos fluxos com destino no nó corrente (2).

A partir da interação com o monitor de contrato, o monitor de QoS passa as informações de qualidade de serviço dos fluxos com origem local para o adaptador de QoS (5) e para o mapeador de QoS (3). Este, por sua vez, calcula os valores de parâmetros no nível de usuário e os envia para o agente de interface (4), para notificação ao usuário. Caso a qualidade de serviço não seja a desejada, mas esteja nos níveis de tolerância, o adaptador de QoS tenta realizar uma realocação de recursos (6 e 7). Por outro lado, se a qualidade de serviços desejada está sendo violada, o agente de interface é notificado pelo adaptador de QoS (8) e um agente de renegociação de contrato pode ser iniciado (9).

O agente de renegociação de contrato tem como tarefa tentar restaurar a qualidade de serviço estabelecida no contrato. Se a renegociação não for possível, o contrato é finalizado e seus fluxos são terminados.

Os cenários de negociação e gerência de contratos apresentados nesta seção permitem um melhor entendimento do modelo proposto por Oliveira et al. [Oliv97]. Tal modelo é bastante abrangente e o seu desenvolvimento completo requer um grande esforço. Para se cumprir os propósitos desta dissertação o seu desenvolvimento parcial é suficiente como estudo de caso para sistemas baseados em agentes móveis.

O modelo proposto por Oliveira et al. [Oliv97] contém três agentes móveis: monitor, negociador e renegociador de contrato. Estes agentes são responsáveis pela realização das etapas de monitoramento, negociação e renegociação do ciclo de vida de um SMD respectivamente.

A realização das etapas de negociação e renegociação está diretamente associada à gerência dos recursos necessários para se atender o nível de qualidade de serviço especificado em um contrato. A gerência de recursos engloba aspectos importantes como a sua reserva e a sua alocação em tempo de execução. Alguns recursos significativos disponíveis para os casos de teste (por exemplo: sistema operacional, protocolos e infra-estrutura de rede) não permitem que tais aspectos sejam tratados. Assim sendo, aquelas etapas não poderiam ser efetivamente realizadas. A etapa de monitoramento de qualidade de serviço foi então adotada como estudo de caso para SBAMs. A experiência de desenvolvimento de um SBAM para monitoramento de QoS em SMDs é tratada no próximo capítulo desta dissertação.

Capítulo 5

Desenvolvimento de um Sistema Baseado em Agentes Móveis para Monitoramento de Qualidade de Serviço em Sistemas Multimídia Distribuídos

Este capítulo relata a experiência de desenvolvimento de um sistema baseado em agentes móveis (SBAM) para monitoramento de qualidade de serviço[1] (QoS) em sistemas multimídia distribuídos (SMDs). Tal sistema se baseia no modelo para negociação e gerência de QoS em SMDs apresentado no capítulo anterior desta dissertação. Casos de teste foram aplicados para examinar o comportamento do SBAM desenvolvido. A análise dos resultados dos casos de teste encerra o conteúdo deste capítulo.

5.1. Escopo e arquitetura de desenvolvimento

A etapa de monitoramento faz parte da fase de gerência do ciclo de vida de sistemas multimídia distribuídos. Nesta etapa, a qualidade de serviço é medida através da obtenção de valores de parâmetros no nível de sistema. A partir destes valores, verifica-se se a qualidade de serviço definida na fase de estabelecimento está sendo honrada. Caso a qualidade de serviço não seja a desejada, porém esteja dentro de limites de tolerância previamente definidos, a etapa de adaptação de QoS é iniciada. Por outro lado, se houver a violação dos valores mínimos para qualidade de serviço, ou se o usuário solicitar, a etapa de renegociação é iniciada.

A arquitetura de desenvolvimento de um SBAM para monitoramento de QoS em SMDs contém dois elementos principais:

- infra-estrutura para suporte a sistemas baseados em agentes móveis: permite criação, execução, migração e término de agentes, bem como comunicação entre agentes, agentes e outros programas e agentes e seres humanos;
- infra-estrutura de computação distribuída aberta: possibilita a interação entre componentes de um sistema distribuído independentemente de hardware,

[1] *"quality of service"*

sistema operacional, protocolo de rede e linguagem de programação. Esta capacidade é chamada de interoperabilidade.

A infra-estrutura para suporte a SBAMs é o núcleo da arquitetura de desenvolvimento. Os agentes componentes do modelo proposto por Oliveira et al. [Oliv97] são manipulados dentro de seu contexto. Para que a etapa de monitoramento de QoS possa ser realizada, agentes têm que se comunicar com programas externos à infra-estrutura para suporte a SBAMs. Estes programas são o servidor de contratos e sensores locais. A comunicação de forma interoperável entre agentes e outros programas é provida pela infra-estrutura de computação distribuída aberta.

Aglets Workbench (AWB) da IBM, examinada na seção 3.4 desta dissertação, foi escolhida como infra-estrutura para suporte a sistemas baseados em agentes móveis. Tal escolha se fundamentou em uma série de fatores que incluem: boa documentação, facilidade de operação, abrangência, robustez, evoluções constantes e grande exposição a usuários externos. Além disso, a AWB está baseada na tecnologia Java, que atualmente é a mais promissora para o desenvolvimento de sistemas baseados em agentes móveis.

A infra-estrutura de computação distribuída aberta selecionada baseou-se na especificação CORBA[2] do Grupo de Gerência de Objetos[3] (OMG). Trata-se de um padrão de interoperabilidade de sistemas amplamente aceito que segue a abordagem de orientação a objetos. O núcleo da especificação CORBA é o intermediário de requisições de objetos[4] (ORB), descrito na seção 2.6. Uma série de produtos disponíveis comercialmente implementam tal padrão, destacando-se a família Orbix da Iona Technologies[Iona97] e a família Visibroker da Visigenic [Visi97]. Ambas contêm implementações da especificação CORBA com suporte a ambientes de execução baseados nas linguagens C++ e Java.

O produto utilizado nesta experiência de desenvolvimento foi o OrbixWeb da Iona Technologies. Trata-se de uma implementação de um ORB com suporte ao ambiente de execução da tecnologia Java. A família Orbix é amplamente utilizada em aplicações práticas e a empresa Iona é atualmente a líder mundial de desenvolvimento e venda de produtos voltados para a Arquitetura de Gerência de Objetos[5] do OMG.

---

[2] *"Common Object Request Broker Architecture"*
[3] *"Object Management Group"*
[4] *"Object Request Broker"*
[5] *"Object Management Architecture"* (OMA)

OrbixWeb e Aglets Workbench são baseados na tecnologia Java. A integração destas infra-estruturas se dá no nível de linguagem de programação, isto é, as capacidades de interoperabilidade providas pelo padrão CORBA são diretamente incorporadas no código de agentes aglets. Para tanto, OrbixWeb gera código Java interoperável seguindo a especificação CORBA de mapeamento de interfaces escritas em IDL[6] para linguagem Java[7].

Da mesma forma, programas externos (servidores) que atendem as requisições de agentes podem se tornar interoperáveis se incorporarem código gerado a partir de mapeamentos IDL especificados pelo padrão CORBA (atualmente disponíveis para linguagens Java, C, C++ e Smalltalk). Agentes desempenham o papel de clientes ORB que fazem requisições para servidores locais.

Com respeito à integração entre OrbixWeb e Aglets Workbench um problema operacional importante foi constatado nesta experiência de desenvolvimento: requisições de agentes para programas externos via OrbixWeb sempre são rejeitadas por motivo de violação de segurança. Tal problema é contornado se a infra-estrutura AWB for executada com o modo de segurança desabilitado. Esta solução entretanto é incabível fora de contextos experimentais.

A agência de QoS é a base do modelo proposto por Oliveira et al. [Oliv97] e consequentemente o componente fundamental do SBAM para monitoramento de qualidade de serviço em sistemas multimídia distribuídos. A próxima seção descreve como a agência de QoS foi implementada a partir da arquitetura de desenvolvimento.

5.2. Implementação da agência de QoS

Um SBAM para monitoramento de QoS conforme modelo proposto por Oliveira et al. [Oliv97] é composto por uma agência de QoS residente em cada nó participante do sistema multimídia distribuído. Esta seção descreve como foram implementados os componentes da agência de QoS que estão envolvidos na realização da etapa de monitoramento. Tais componentes são: contratos, servidor de contratos, fábrica de agentes e agentes.

[6] *"Interface Definition Language"*
[7] Java *"binding"*

Contratos e fluxos foram implementados como objetos Java seriados, isto é, seu código interpretável pode ser migrado quando necessário. Fluxos são identificados pelo tipo do meio manipulado e pelo nó de sua origem. Uma lista de nós destino e um conjunto de parâmetros de QoS em nível de sistema completam a estrutura de um fluxo. Cada parâmetro de QoS é descrito por um nome, valor desejado e limites de tolerância. Contratos são identificados por seu nome e versão. Um contrato pode ter várias versões. A estrutura de um contrato é completada por uma lista de fluxos componentes (figura 5.1).

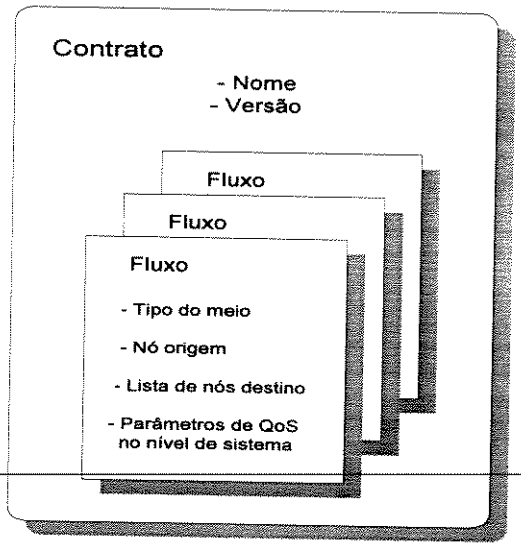

Figura 5.1: Estrutura de um contrato

O servidor de contratos é um sistema responsável por gerenciar contratos armazenados em uma base. Tal sistema foi implementado na linguagem Java e algumas de suas ações foram disponibilizadas de forma interoperável. Para tanto, uma interface IDL foi definida contendo operações como: tornar disponível informações de um contrato e informar qual a versão mais recente de um contrato. A base gerenciada pelo servidor de contratos é um arquivo seqüencial de objetos contrato.

A inserção de contratos na base é realizada no modelo proposto por Oliveira et al. [Oliv97] por agentes de negociação e renegociação. Uma aplicação de cadastro de contratos foi desenvolvida para realizar esta tarefa, já que o desenvolvimento daqueles agentes não faz parte do escopo desta dissertação. Esta aplicação foi escrita em linguagem Java e sua interface permite que usuários executem operações de inserção, alteração e remoção de informações pertinentes a contratos, bem como de navegação pelos objetos

cadastrados na base. As figuras 5.2 e 5.3 mostram interfaces gráficas que compõem a aplicação de cadastro de contratos.

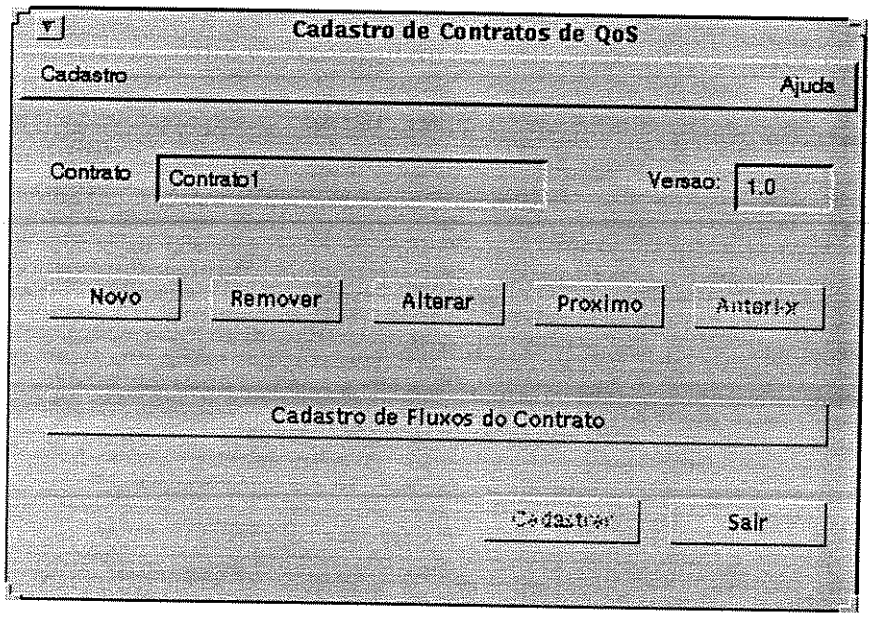

Figura 5.2: Interface principal da aplicação de cadastro de contratos

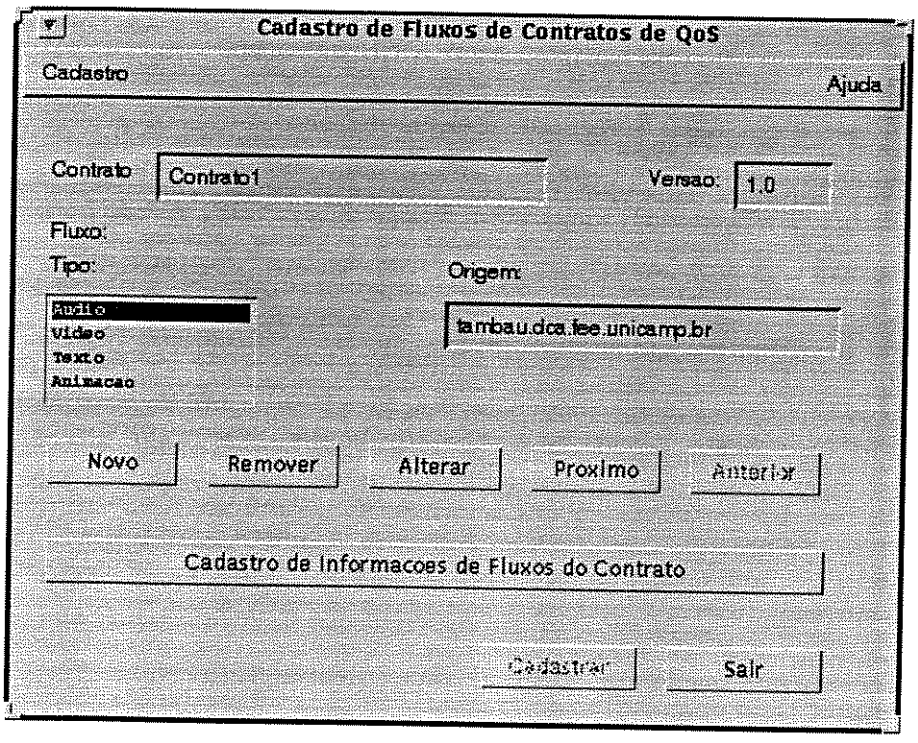

Figura 5.3: Interface de manipulação de fluxos presente na aplicação de cadastro de contratos

A figura 5.2 apresenta a interface principal da aplicação de cadastro de contratos, que permite a manipulação de dados de identificação de contratos (nome e versão). Os fluxos componentes de um contrato podem ser manipulados quando o botão "Cadastro de Fluxos do Contrato" é pressionado, mostrando então a interface apresentada na figura 5.3.

A interface de cadastro de fluxos (figura 5.3) permite a manipulação de dados de identificação de fluxos (tipo do meio e nó origem). As demais informações pertinentes a um fluxo podem ser acessadas por outras interfaces que são apresentadas a partir do momento em que o botão "Cadastro de Informações de Fluxos do Contrato" é pressionado.

A fábrica de agentes tem como função criar e terminar agentes. Nesta experiência de desenvolvimento, este componente da agência de QoS foi implementado em Java como um agente fixo aglet devido a características particulares da infra-estrutura Aglets Workbench.

Para que a fábrica possa instanciar agentes, ela tem que ser previamente criada num contexto de execução. O gerente de agentes Tahiti atualmente é o mecanismo disponível para estabelecer contextos. Assim sendo, a fábrica tem que ser criada a partir do Tahiti, isto é, ela tem que ser um agente. Por outro lado, o término de agentes é facilitado. A infra-estrutura Aglets Workbench provê capacidades abrangentes de comunicação entre agentes que são exploradas pela fábrica (através do envio de mensagens de término para agentes locais e remotos por ela criados).

No modelo proposto por Oliveira et al. [Oliv97] a etapa de monitoramento envolve três tipos de agentes estáticos e um tipo de agente móvel. Os agentes estáticos são: monitor de QoS, mapeador de QoS e agente de interface. O agente móvel é o monitor de contrato, que é responsável pela realização da etapa de monitoramento.

Com respeito aos quatro tipos de agentes que estão envolvidos na realização da etapa de monitoramento, apenas o mapeador de QoS não foi implementado. Isto porque não há ainda na área de sistemas multimídia um estudo abrangente e detalhado estabelecendo parâmetros subjetivos no nível de usuário, bem como seus respectivos mapeamentos para o nível de sistema.

Agentes foram implementados em Java como aglets. O monitor de contrato foi desenvolvido de forma ligeiramente diferente do modelo proposto por Oliveira et al. [Oliv97]. Neste modelo há apenas um agente para monitorar todo o contrato. Como a abordagem de SBAMs enfatiza paralelismo, decidiu-se que a etapa de monitoramento seria realizada por

vários agentes monitores de contrato, sendo que cada um é responsável por monitorar fluxos com mesma origem. Os valores dos parâmetros de QoS obtidos pelo monitor de contrato são enviados para a agência localizada na origem dos fluxos, via mensagens remotas.

A figura 5.4 ilustra um cenário de monitoramento de QoS, contendo um nó origem e um nó destino para um fluxo multimídia. Os componentes da agência de QoS e suas relações com a arquitetura de desenvolvimento podem ser nela observados.

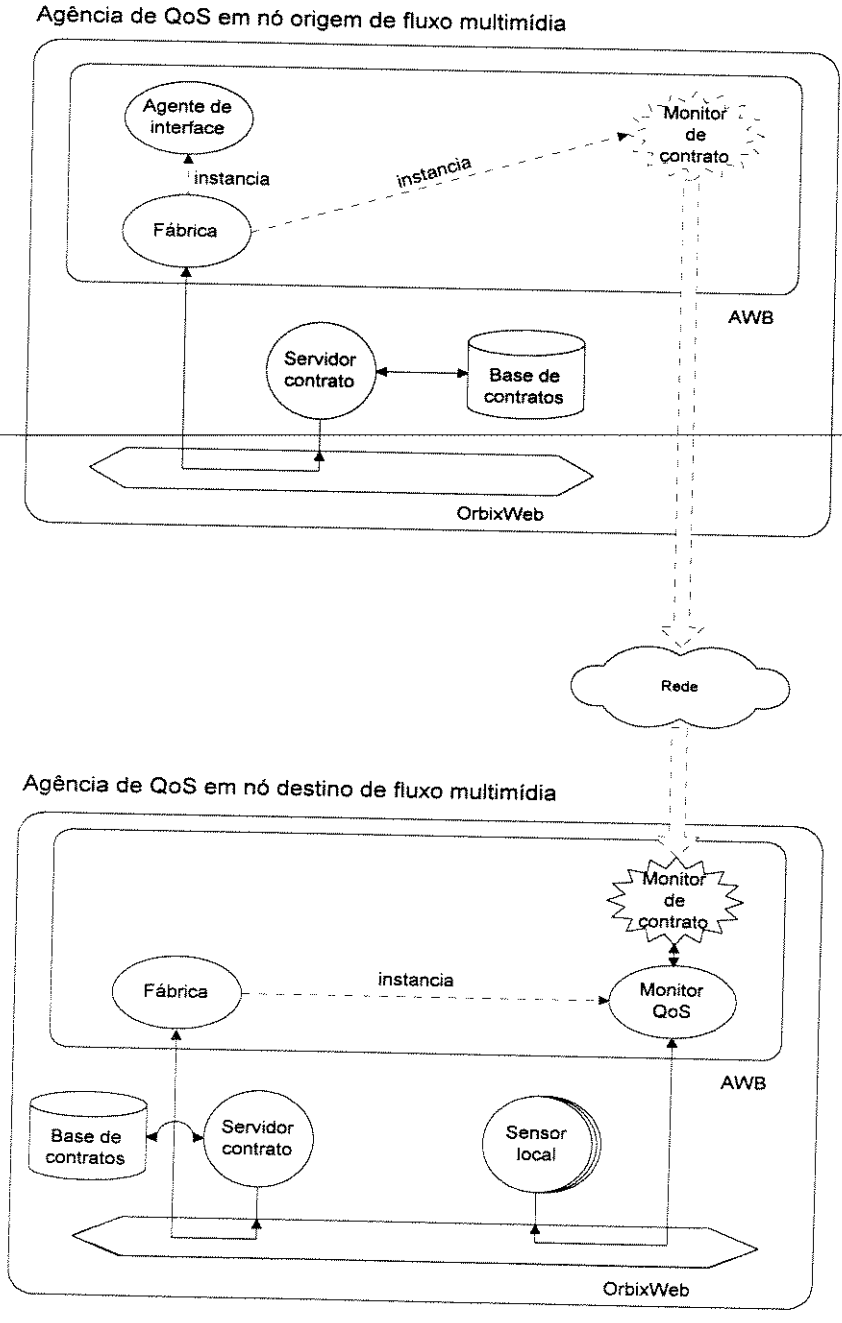

Figura 5.4: Cenário de monitoramento de QoS através do SBAM desenvolvido

A execução normal do SBAM para monitoramento de qualidade de serviço em SMDs requer que duas pré-condições sejam satisfeitas:

- um contrato foi previamente negociado para um sistema multimídia distribuído e está armazenado numa base (através da aplicação de cadastro) acessível pelo servidor de contratos;
- a execução do sistema multimídia distribuído está agendada para iniciar num tempo t_0 e terminar num tempo t_1.

O SBAM desenvolvido tem três momentos distintos de execução. Primeiramente, no tempo (t_0 - Δt), onde Δt é um intervalo de segurança, o sistema é iniciado. Neste momento de preparação, os seguintes passos são tomados em cada nó origem ou destino de fluxos que compõe o contrato sob monitoramento:

- execução do gerente de agentes Tahiti da infra-estrutura Aglets Workbench;
- execução da infra-estrutura de computação distribuída aberta OrbixWeb;
- criação de uma fábrica de agentes (é realizada por intervenção humana através da interface gráfica do Tahiti).

Em seguida, fábricas locais instanciam um agente monitor de QoS em cada nó destino de fluxos do contrato sob monitoramento.

O segundo momento de execução do SBAM desenvolvido ocorre no tempo t_0 quando o monitoramento é efetivamente realizado. Em cada agência que reside em nós origem de fluxos do contrato sob monitoramento, a fábrica local cria um agente de interface e um agente monitor de contrato. O itinerário de um agente monitor de contrato é uma seqüência de nós que são destinos de fluxos com origem no nó onde foi criado.

Após a sua criação, o agente monitor de contrato inicia viagem (migrando para o primeiro nó componente do seu itinerário) visando realizar a etapa de monitoramento. Em seguida, as migrações têm como destino nós subseqüentes do seu itinerário. Quando todo o itinerário é percorrido, o agente completa um ciclo de monitoramento de QoS (todos os destinos de fluxos foram visitados), e retorna ao primeiro nó componente do seu itinerário para iniciar mais um ciclo.

Finalmente, no tempo t_1 as fábricas locais terminam os agentes por elas criados (monitor de QoS, agente de interface e monitor de contrato).

Um programa externo ao SBAM desenvolvido é responsável por notificar o alcance dos três momentos de execução para todas as agências envolvidas em um contrato. As fábricas locais recebem tais notificações e disparam as ações pertinentes. O programa externo foi implementado em Java como um agente fixo aglet (chamado agente externo) para que a sua interação com as fábricas fosse facilitada.

Da mesma forma que as fábricas locais, o agente externo é criado por intervenção humana através do Tahiti. A sua interface gráfica permite primeiramente que o contrato sob monitoramento seja identificado. Além disso, a interface contém três botões que representam os momentos de execução descritos anteriormente. Quando tais botões são pressionados, notificações (mensagens AWB) são enviadas para fábricas locais presentes em agências de QoS que residem em nós envolvidos no contrato sob monitoramento.

Para que o agente externo se comunique com as fábricas locais é necessário que ele conheça suas *proxys* (invólucros de agentes aglets para os quais mensagens devem ser enviadas). No tempo $(t_0 - \Delta t)$ o agente externo inicialmente interage com o servidor de contratos para obter os nós onde residem as agências de QoS envolvidas no contrato sob monitoramento. Em seguida, envia um agente móvel para cada nó obtido visando descobrir *proxys* de fábricas locais. Os agentes móveis então retornam com a informação desejada que é registrada pelo agente externo.

A próxima seção descreve os casos de teste nos quais foi aplicado o SBAM desenvolvido para monitoramento de qualidade de serviço em sistemas multimídia distribuídos.

5.3. Casos de teste

O sistema baseado em agentes móveis desenvolvido conforme descrito na seção anterior teve seu comportamento examinado através de casos de teste. O objetivo deste procedimento é avaliar se as potencialidades associadas à abordagem de SBAMs se mostram verdadeiras na prática.

Os casos de teste foram executados num ambiente distribuído composto por quatro estações de trabalho Sun sob sistema operacional Solaris, ligadas em rede local Ethernet.

As estações dispõem de dispositivos multimídia incluindo auto-falantes, microfones e câmeras de vídeo.

Um sistema de manipulação de áudio em ambiente distribuído, implementado por Araújo et al. [Arau97], foi utilizado como SMD a ser monitorado. Embora atualmente manipule apenas um meio, tal sistema é suficiente para se cumprir os propósitos desta dissertação. Casos de teste foram estabelecidos a partir de duas diferentes configurações do sistema distribuído (SD) de manipulação de áudio, envolvendo os recursos descritos no parágrafo anterior. Ambas refletem um contrato contendo dois fluxos de áudio com origens diferentes. Portanto, dois agentes móveis, um para cada origem, realizam a etapa de monitoramento. As figuras 5.5 e 5.6 ilustram estas configurações.

A primeira configuração envolve três nós. Um agente monitor de contrato (agente 1) monitora o fluxos com origem no nó 1, isto é, o fluxo 1. Seu itinerário é composto pelo nó 2. O outro agente monitor de contrato (agente 2) monitora fluxos com origem no nó 2, isto é, o fluxo 2. O itinerário deste agente é composto pelos nós 1 e 3.

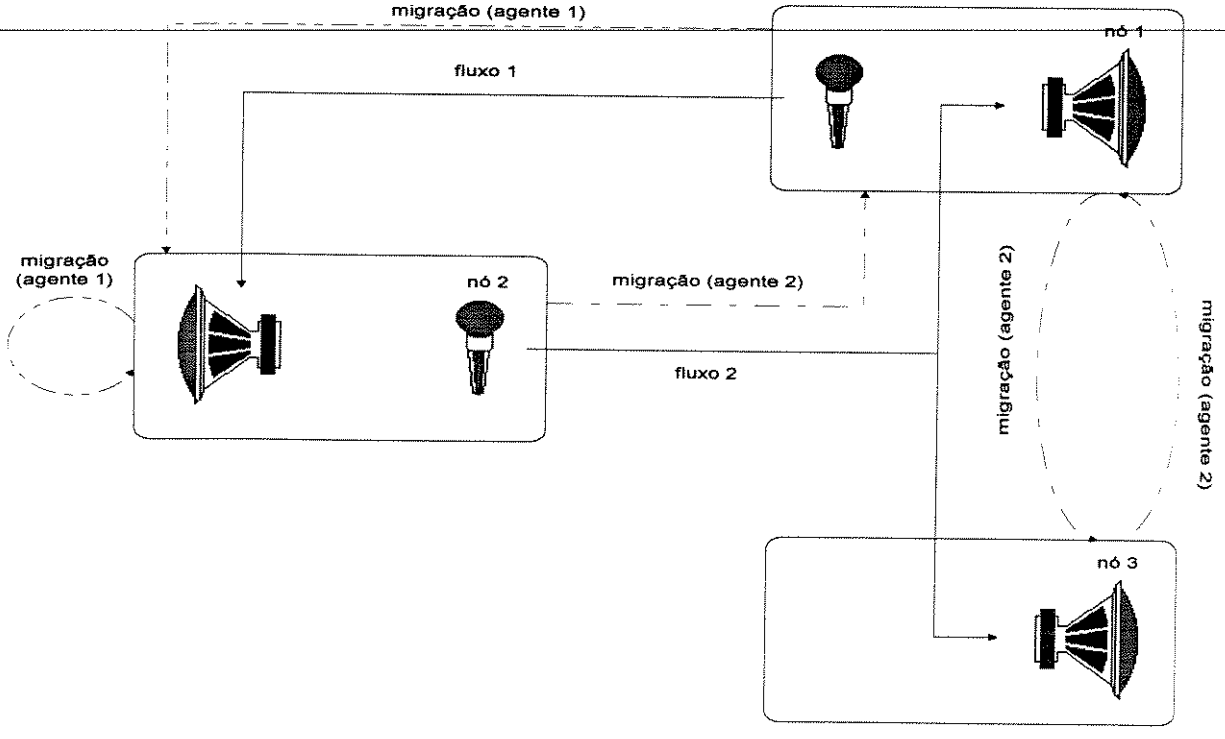

Figura 5.5: Primeira configuração estabelecida para casos de teste

A segunda configuração estabelecida para casos de teste (figura 5.6) envolve quatro nós e é semelhante à primeira. Um agente monitor de contrato (agente 1) monitora fluxos

com origem no nó 1, isto é, o fluxo 1. Seu itinerário é composto pelo nó 2. O outro agente monitor de contrato (agente 2) monitora fluxos com origem no nó 2, isto é, o fluxo 2. O itinerário deste agente é composto pelos nós 1, 3 e 4.

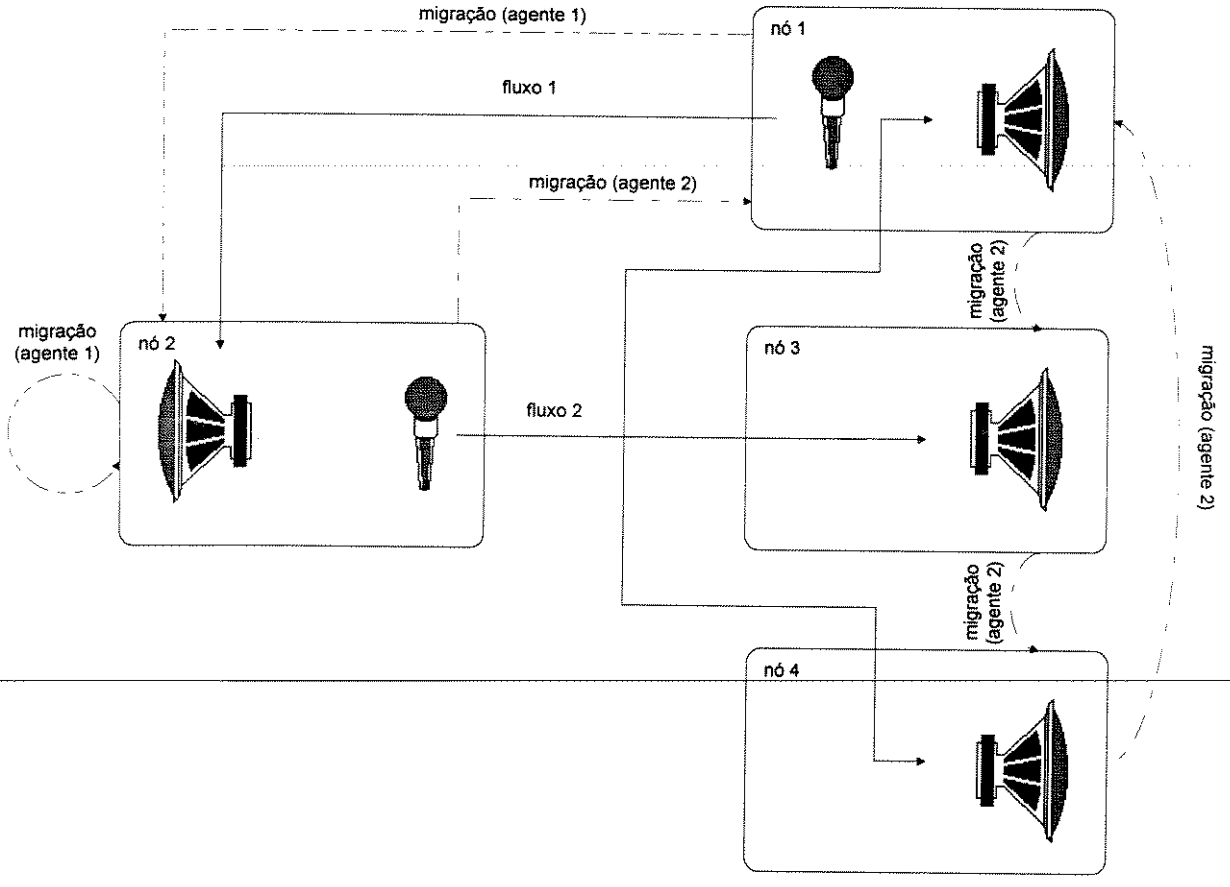

Figura 5.6: Segunda configuração estabelecida para casos de teste

O SD de manipulação de áudio implementa o conceito de canal como mecanismo de comunicação entre componentes de um sistema distribuído. Este conceito está definido na especificação de Processamento Distribuído Aberto[8] proposta pela Organização de Padrões Internacionais[9] [ISO95]. Para a transmissão de fluxos multimídia, o canal implementado utiliza o Protocolo de Tempo Real[10] (RTP) [Schu96]. RTP transmite uma quantidade muito pequena de informação de controle. Por isso é de execução leve e rápida tornando-o de grande aplicabilidade em SMDs. Além disso, RTP calcula e torna disponível valores de parâmetros de QoS em nível de sistema, tais como: taxa de transferência, jitter e taxa de perda de pacotes.

[8] *"Open Distributed Processing"* (ODP)
[9] *"International Standards Organization"* (ISO)
[10] *"Real Time Protocol"*

Os sensores locais presentes em cada nó envolvido no sistema distribuído de manipulação de áudio foram implementados em C++ como objetos componentes de um canal. Através da definição de uma interface IDL (seguindo-se a especificação CORBA), tais objetos se tornaram servidores interoperáveis capazes de fornecer valores de parâmetros de QoS em nível de sistema para qualquer outro programa.

Os sensores locais obtêm os valores de QoS através da interação com o RTP. Tal tarefa é realizada dentro do contexto do canal.

As figuras de 5.7 a 5.9 ilustram interfaces gráficas do agente de interface mostradas durante a execução do SBAM desenvolvido. Tal agente tem como função apresentar a um usuário os valores de parâmetros de qualidade de serviço para fluxos com origem no nó em que reside. Para tanto, inicialmente são mostrados todos os fluxos associados à origem (figura 5.7). A partir da seleção de um fluxo, ao se pressionar o botão "Informações", todos os destinos do fluxo selecionado podem ser apreciados, conforme mostra a figura 5.8.

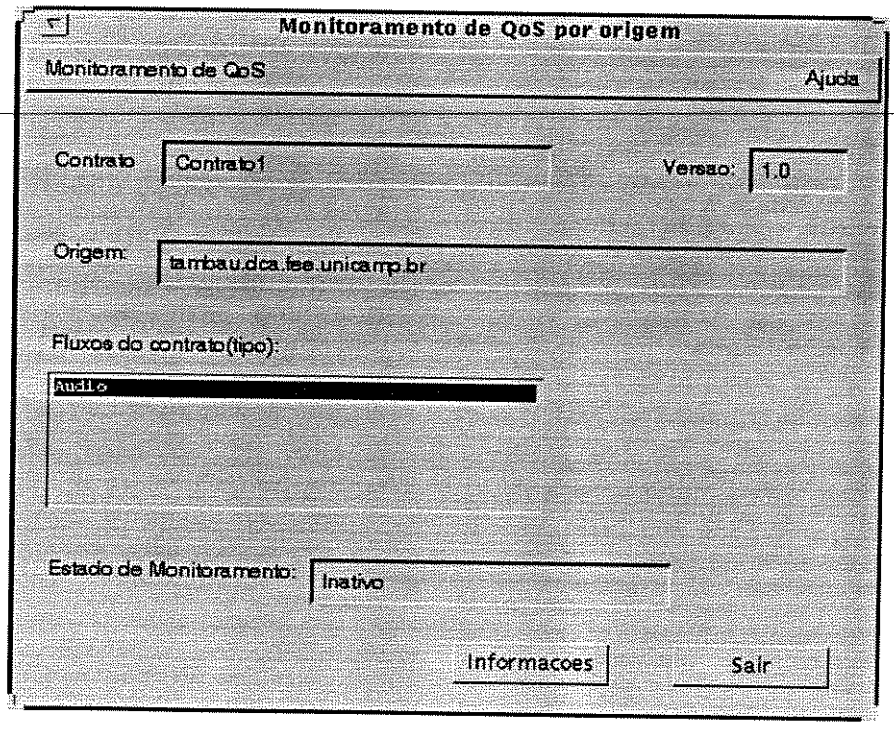

Figura 5.7: Interface gráfica do agente de interface para seleção de fluxo

Após a seleção de um destino o botão "Informações" pode ser pressionado, fazendo com que a interface gráfica ilustrada na figura 5.9 seja apresentada ao usuário.

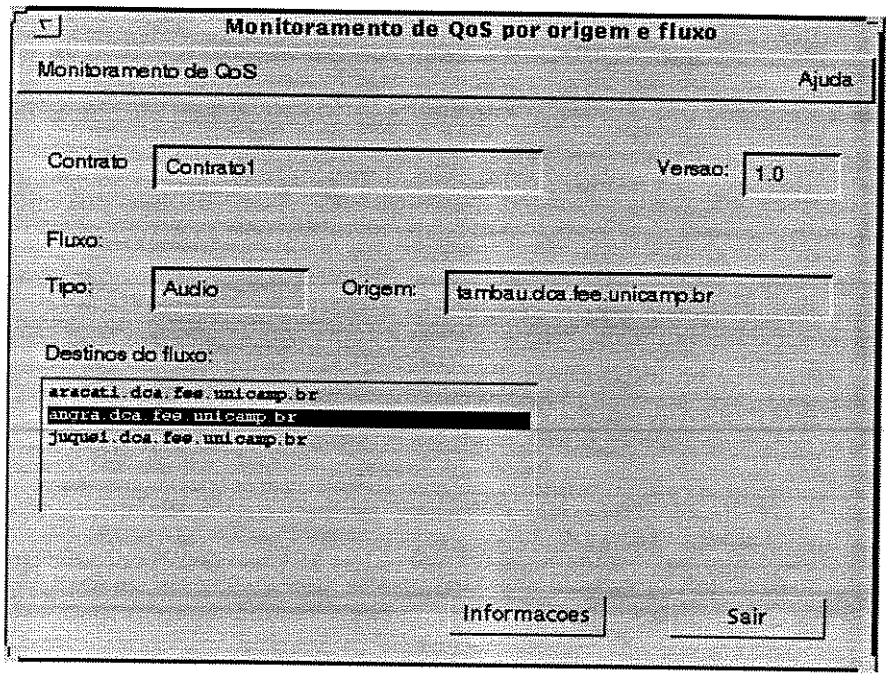

Figura 5.8: Interface gráfica do agente de interface para seleção de destino de um fluxo

A interface gráfica ilustrada pela figura 5.9 mostra os valores dos parâmetros de QoS em nível de sistema para um destino particular de um dado fluxo.

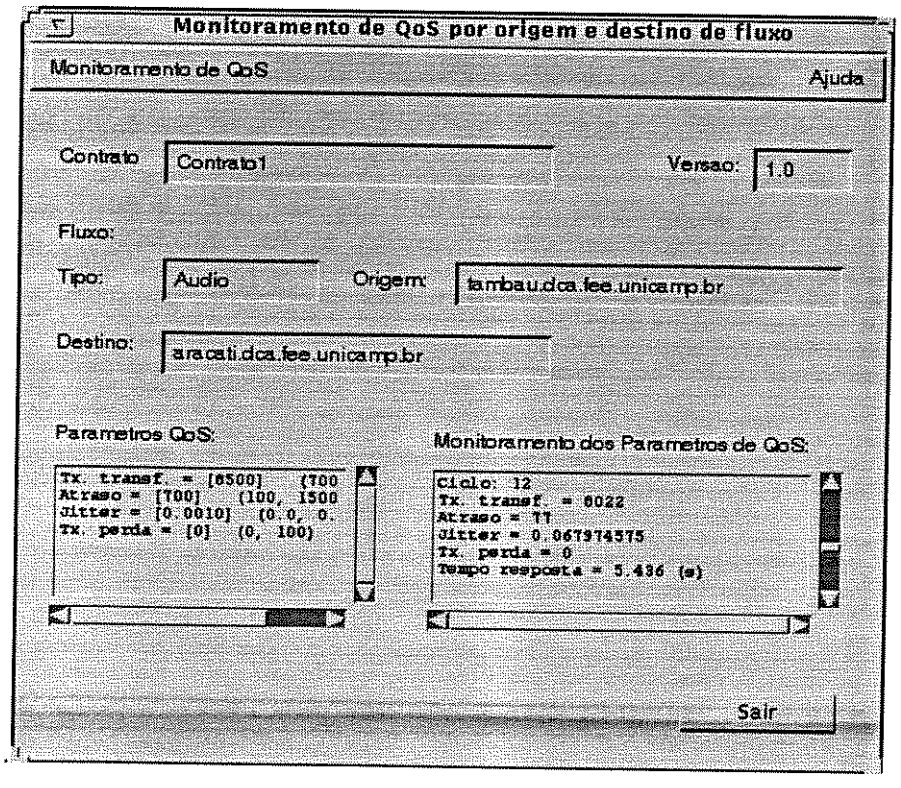

Figura 5.9: Interface gráfica do agente de interface contendo valores de parâmetros de QoS

Capítulo 4

Qualidade de Serviço em Sistemas Multimídia Distribuídos: um Estudo de Caso para Sistemas Baseados em Agentes Móveis

A abordagem baseada em agentes móveis constitui-se numa alternativa promissora para o desenvolvimento de sistemas distribuídos. Entretanto, por ser uma área de recente concentração de esforços de pesquisa e desenvolvimento, existem poucos relatos de sua utilização. Este capítulo descreve o aspecto de qualidade de serviço[1] (QoS) em sistemas multimídia distribuídos (SMDs) e apresenta um modelo baseado em agentes para a negociação e gerência de QoS em SMDs, proposto por Oliveira et al. [Oliv97]. Tal modelo contém agentes estáticos e móveis, e foi adotado como estudo de caso para o emprego de SBAMs.

4.1. Qualidade de serviço em sistemas multimídia distribuídos

Sistemas multimídia são sistemas de computador que manipulam de forma integrada vários tipos de meios de representação de informação. Tais meios podem ser estáticos (por exemplo: texto e gráficos) ou dinâmicos (por exemplo: áudio, vídeo e animação). Freqüentemente, sistemas multimídia são distribuídos, isto é, seus componentes estão localizados em diferentes nós de processamento numa rede local ou de longa distância.

Sistemas multimídia distribuídos que manipulam meios dinâmicos alcançaram grande popularidade recentemente. Isto se deve a dois fatores. Primeiro porque a infra-estrutura necessária (hardware, software e comunicação) está disponível com baixos custos a um grande número de pessoas. Além disso, a interação entre ser humano e computador se torna mais agradável através deste tipo de sistema, já que as informações manipuladas são mais ricas. Exemplos de aplicação de SMDs incluem sistemas de conferência e de ensino à distância.

[1] *"quality of service"*

A seguir são apresentados dados obtidos como conseqüência da realização de três casos de teste que utilizam o SD de manipulação de áudio. Para cada caso de teste foram levantados valores referentes a dois indicadores de tempo visando a análise de desempenho do SBAM desenvolvido. Tais indicadores são: tempo de migração e tempo de resposta. O tempo de migração é o intervalo de tempo em segundos transcorrido no transporte de um agente monitor de contrato entre dois nós consecutivos do seu itinerário.

O tempo de resposta é o intervalo de tempo em segundos transcorrido entre dois recebimentos consecutivos de valores de parâmetros de QoS com respeito a um destino específico de um fluxo, pela agência localizada em sua origem. O agente monitor de contrato é responsável por enviar tais valores obtidos em agências remotas.

O tempo de resposta está diretamente associado ao itinerário do agente. Quanto maior for o número de nós que compõem o itinerário, maior será o tempo de resposta. Isto porque estes nós são visitados de forma seqüencial pelo agente monitor de contrato. Portanto, o itinerário de um agente monitor de contrato deve conter um número reduzido de nós componentes. Caso contrário, recomenda-se que a tarefa de monitoramento por origem seja realizada por vários agentes em paralelo.

5.3.1. Primeiro caso de teste

O primeiro caso de teste se baseia na configuração ilustrada pela figura 5.5. As tabelas A.1 e A.2 (presentes no Apêndice A) apresentam dados obtidos a partir da execução do SBAM desenvolvido.

A tabela A.1 contém, em cada linha, valores para o indicador tempo de migração. Colunas representam nós para os quais agentes monitores de contrato migram. Estes nós são destinos de fluxos específicos.

A tabela A.2 mostra, em cada linha, os valores obtidos para o indicador tempo de resposta. Colunas representam destinos de um fluxo a partir dos quais os valores de parâmetros de QoS são enviados.

A tabela 5.1 sumariza os dados relativos ao primeiro caso de teste, contendo os tempos médios de migração e resposta por nó.

Nó / Tempo	2 (fluxo1)	1 (fluxo2)	3 (fluxo2)
Migração médio (segundos)	0,865	0,673	0,722
Resposta médio (segundos)	1,867	3,085	3,079

Tabela 5.1: Valores médios de indicadores de tempo para o primeiro caso de teste

Com respeito ao indicador tempo de resposta, os valores pertencentes às colunas segunda e terceira são maiores que os da primeira. Conforme esclarecido anteriormente, tempos de resposta estão diretamente relacionados ao número de nós do itinerário associado ao agente monitor de contrato que envia as respostas. O nó 2 é o único componente do itinerário de um agente, enquanto os nós 1 e 3 compõem o itinerário de um outro agente.

O cálculo do tempo de resposta para a primeira coluna engloba uma migração (sempre para o mesmo nó) seguida da obtenção dos valores dos parâmetros de QoS e seu envio para a agência origem do fluxo monitorado. Estas duas últimas atividades são chamadas de tarefa local.

O cálculo do tempo de resposta para as colunas segunda e terceira engloba duas migrações (para o próximo nó do itinerário e retorno ao nó corrente) e a realização de duas tarefas locais, uma em cada nó do itinerário. Portanto, espera-se que tempos de resposta para os nós 1 e 3 sejam aproximadamente duas vezes o tempo de resposta para o nó 2.

A partir da tabela 5.1, pode-se observar que a média dos tempos de migração médios alcançou a marca de 0,753 segundos. Com respeito aos tempos de resposta médios, os valores para as colunas segunda e terceira apresentam o comportamento esperado, isto é, se aproximam de duas vezes o valor da primeira coluna.

5.3.2. Segundo caso de teste

O segundo caso de teste se baseia na configuração ilustrada pela figura 5.6. As tabelas A.3 e A.4 (presentes no Apêndice A) apresentam dados obtidos a partir da execução do SBAM desenvolvido.

A tabela A.3 contém, em cada linha, valores para o indicador tempo de migração. Colunas representam nós para os quais agentes monitores de contrato migram. Estes nós são destinos de fluxos específicos.

A tabela A.4 mostra, em cada linha, os valores obtidos para o indicador tempo de resposta. Colunas representam destinos de um fluxo a partir dos quais os valores de parâmetros de QoS são enviados.

A tabela 5.2 sumariza os dados relativos ao segundo caso de teste, contendo os tempos médios de migração e de resposta por nó.

Nó / Tempo	2 (fluxo1)	1 (fluxo2)	3 (fluxo2)	4 (fluxo2)
Migração médio (segundos)	0,824	0,696	0,913	0,896
Resposta médio (segundos)	1,776	5,331	5,222	5,202

Tabela 5.2: Valores médios de indicadores de tempo para o segundo caso de teste

Com respeito ao indicador tempo de resposta, os valores pertencentes às colunas segunda, terceira e quarta são maiores que os da primeira conforme motivos esclarecidos anteriormente.

O cálculo do tempo de resposta para a primeira coluna engloba uma migração (sempre para o mesmo nó) seguida da realização da tarefa local. O cálculo do tempo de resposta para as colunas segunda, terceira e quarta engloba três migrações e a realização de três tarefas locais, uma em cada nó do itinerário. Portanto, espera-se que tempos de resposta para os nós 1, 3 e 4 sejam aproximadamente três vezes o tempo de resposta para o nó 2.

A partir da tabela 5.2, pode-se observar que a média dos tempos de migração alcançou a marca de 0,832 segundos. Com respeito aos tempos de resposta médios, os valores para as colunas segunda, terceira e quarta apresentam o comportamento esperado, isto é, são aproximadamente três vezes o valor da primeira coluna.

5.3.3. Terceiro caso de teste

O terceiro caso de teste também se baseia na configuração ilustrada pela figura 5.6. Porém os nós correspondem a diferentes estações de trabalho com respeito ao segundo caso de teste. As tabelas A.5 e A.6 (presentes no Apêndice A) apresentam dados obtidos a partir da execução do SBAM desenvolvido.

A tabela A.5 contém, em cada linha, valores para o indicador tempo de migração. Colunas representam nós para os quais agentes monitores de contrato migram. Estes nós são destinos de fluxos específicos.

A tabela A.6 mostra, em cada linha, os valores obtidos para o indicador tempo de resposta. Colunas representam destinos de um fluxo. A partir dos quais os valores de parâmetros de QoS são enviados.

A tabela 5.3 sumariza os dados relativos ao terceiro caso de teste, contendo os tempos médios de migração e resposta por nó.

Nó / Tempo	2 (fluxo1)	1 (fluxo2)	3 (fluxo2)	4 (fluxo2)
Migração médio (segundos)	0,955	0,805	0,675	0,890
Resposta médio (segundos)	1,996	4,982	4,987	4,834

Tabela 5.3: Valores médios de indicadores de tempo para o terceiro caso de teste

Com respeito ao indicador tempo de resposta, os valores pertencentes às colunas segunda, terceira e quarta são maiores que os da primeira conforme motivos esclarecidos anteriormente.

O cálculo do tempo de resposta para a primeira coluna engloba uma migração (sempre para o mesmo nó) seguida da realização da tarefa local. O cálculo do tempo de resposta para as colunas segunda, terceira e quarta engloba três migrações e a realização de três tarefas locais, uma em cada nó do itinerário. Portanto, espera-se que tempos de resposta para os nós 1, 3 e 4 sejam aproximadamente três vezes o tempo de resposta para o nó 2.

A partir da tabela 5.3, pode-se observar que a média dos tempos de migração médios alcançou a marca de 0,831 segundos. Com respeito aos tempos de resposta médios, os valores para as colunas segunda, terceira e quarta apresentam o comportamento esperado, isto é, se aproximam de três vezes o valor médio da primeira coluna.

A próxima seção deste capítulo trata de uma análise detalhada dos três casos de teste descritos anteriormente.

5.4. Análise dos casos de teste realizados

O SBAM desenvolvido apresentou comportamento robusto e realizou a tarefa de monitoramento conforme esperado. A integração entre as infra-estruturas de computação distribuída aberta e suporte a sistemas baseados em agentes móveis ocorreu de forma transparente com o modo de segurança desabilitado no Aglets Workbench.

Os dados apresentados no Apêndice A para os indicadores tempo de migração e de resposta permitem que uma série de considerações seja feita. Tais considerações são descritas na seqüência.

Inicialmente, pode-se observar que as duas primeiras linhas das tabelas de A.1 a A.6 geralmente contêm valores bem mais altos do que os respectivos valores médios. Esta observação está relacionada ao fato de que a infra-estrutura Aglets Workbench contém um Carregador de Classes Aglet[11] responsável por carregar dinamicamente, durante migrações, as definições de classes de agentes e de objetos por eles manipulados. Tal carregador trabalha com o mecanismo de memória *"cache"*, isto é, definições de classes mais freqüentemente carregadas a partir de nós remotos são armazenadas localmente para acesso rápido.

Assim sendo, é possível inferir que, para as configurações dos casos de teste, após a terceira migração tanto a definição do agente monitor de contrato como de objetos que ele manipula se tornam disponíveis localmente via memória *"cache"*. Desta forma, os valores das duas primeiras linhas das tabelas de A.1 a A.6 não foram levados em consideração para o cálculo dos valores médios.

[11] *"Aglet Class Loader"*

A análise de desempenho do SBAM desenvolvido, a partir dos valores levantados para os indicadores de tempo, deve ser feita de forma comparativa para que conclusões relevantes sejam tecidas.

Uma outra série de testes foi então estabelecida, utilizando o sistema distribuído de manipulação de áudio e as mesmas configurações de caso de teste ilustradas pelas figuras 5.5 e 5.6. Entretanto, uma abordagem diferente de desenvolvimento de sistemas foi empregada: objetos distribuídos interoperáveis através do padrão CORBA.

Conforme descrito anteriormente, sensores locais presentes no SD de manipulação de áudio foram implementados como objetos (servidores) interoperáveis CORBA. Uma aplicação cliente também baseada no padrão CORBA foi desenvolvida em Java para coleta de valores de parâmetros de QoS. Esta coleta é feita através do envio de requisições remotas para sensores presentes em nós destinos de fluxos de um contrato específico. OrbixWeb foi utilizado como implementação do padrão CORBA para comunicação entre os sensores locais e a aplicação cliente desenvolvida.

O indicador tempo de resposta foi estabelecido para análise de desempenho da abordagem CORBA. Este indicador é definido como o intervalo de tempo em segundos transcorrido entre uma requisição de valores de parâmetros de QoS para um dado nó destino de um fluxo específico, e a sua resposta.

As tabelas B.1 e B.2 (presentes no Apêndice B) mostram valores obtidos para o indicador tempo de resposta, para dois casos de teste com configurações diferentes. Cada coluna representa um nó destino de um fluxo específico para os quais as requisições foram enviadas. O primeiro caso de teste se baseia na configuração ilustrada pela figura 5.5.

Para cada coluna da tabela B.1, os valores médios dos tempos de resposta são respectivamente: 0,0183 s, 0,0233 s e 0,0222 s. A média destes valores alcançou a marca de 0,0212 segundos.

O segundo caso de teste se baseia na configuração ilustrada pela figura 5.6. Para cada coluna da tabela B.2, os valores médios dos tempos de resposta são respectivamente: 0,0185 s, 0,0212 s, 0,214 s e 0,0193 s. A média destes valores alcançou a marca de 0,0201 segundos.

O tempo de resposta médio dos casos de teste da abordagem CORBA (t_{CORBA}) é de 0,0206 segundos. Pode-se observar que nas tabelas B1 e B2 a primeira linha contém

valores bem mais altos que os respectivos valores médios. Esta observação permite inferir que a primeira interação via OrbixWeb tem tratamento diferenciado. Os dados das primeiras linhas não foram levados em consideração para o cálculo dos valores médios.

A tabela 5.4 mostra um quadro geral dos indicadores de tempo para os casos de teste do SBAM desenvolvido, conforme dados apresentados na seção anterior. Para a comparação de desempenho com a abordagem CORBA, apenas os tempos de resposta médios para itinerários com um nó foram considerados.

Caso de teste / Tempo	1	2	3
Migração médio (segundos)	0,753	0,832	0,831
Resposta médio (segundos)	1,867	1,776	1,996

Tabela 5.4: Quadro geral dos indicadores de tempo para os casos de teste do SBAM desenvolvido

O tempo de migração médio dos casos de teste do SBAM (t_{mig}) é 0,805 s, o que equivale a 39,07 vezes t_{CORBA}. O menor tempo de migração observado nos casos de teste foi de 0,512 s, o que corresponde a 24,85 vezes t_{CORBA}, enquanto que o maior foi de 2,596 s, o que corresponde a 143,49 t_{CORBA}.

O tempo de resposta médio dos casos de teste do SBAM (t_{resp}) é 1,879 s, o que equivale a 91,21 vezes t_{CORBA}. O menor tempo de resposta observado nos casos de teste foi de 1,347 s, o que corresponde a 63,39 vezes t_{CORBA}, enquanto que o maior foi de 3,575 s, o que corresponde a 173,54 vezes t_{CORBA}. O tempo de migração médio corresponde a 42,8% do tempo de resposta médio.

Os maiores tempos de migração e resposta do SBAM desenvolvido foram obtidos no terceiro caso de teste para o nó 2 (destino do fluxo1). Estes tempos dizem respeito à mesma observação (linha 5), onde o tempo de migração elevado influenciou o tempo de resposta. Trata-se provavelmente de uma situação ímpar, ocasionada por condições particulares de utilização de recursos como unidade central de processamento e tráfego de rede.

Embora os tempos médios de migração e de resposta para os casos de teste do SBAM desenvolvido sejam bem elevados (quando comparados com o tempo de resposta

médio dos casos de teste da abordagem CORBA), a utilização da abordagem baseada em agentes móveis não se torna inviável.

A abordagem de SBAMs é emergente e a infra-estrutura Aglets Workbench está em fase de amadurecimento. Correntemente tal infra-estrutura se encontra em versão alfa teste. A IBM reconheceu recentemente, na lista de discussão eletrônica da AWB, que o seu protocolo para migração e comunicação de agentes (ATP) ainda é bastante ineficiente. Melhorias significativas em seu desempenho são esperadas para próximas versões.

A abordagem CORBA, por outro lado, vem sendo objeto de esforços de pesquisa e desenvolvimento desde 1989 quando foi estabelecido o OMG. A família de produtos Orbix, da qual OrbixWeb faz parte, teve sua primeira versão comercial lançada em meados de 1993 estando portanto bem madura.

A abordagem de SBAMs deve continuar a ser investigada visto que agentes móveis podem carregar conhecimento consigo visando a realização de tarefas de forma eficiente. Em interações entre componentes no modelo Cliente/Servidor, como é o caso da abordagem CORBA, apenas dados são migrados, cabendo normalmente ao cliente tratar estes dados após todo o seu volume ter sido transportado de um nó remoto.

A abordagem CORBA, para o estudo de caso aplicado, é bem mais rápida que o SBAM desenvolvido porque o volume de dados transferido é pequeno e tais dados correspondem a tipos básicos (inteiro e ponto flutuante), não sofrendo nenhum tipo de tratamento.

Para a abordagem baseada em agentes, entretanto, em média despende-se 42,8% do tempo de resposta para a migração do agente monitor de contrato, que é um processo elaborado onde código e dados são transportados. Além disso, antes da realização da tarefa de monitoramento, o agente é restaurado para execução ao chegar na agência destino de migração. Transcorre-se então mais um intervalo de tempo de preparação que tem impacto no tempo de resposta.

A partir destas considerações, recomenda-se o uso de SBAMs em domínios em que agentes móveis possam aplicar conhecimento para a realização de tarefas de forma eficiente, tratando dados antes de enviá-los. Desta forma, o intervalo de tempo despendido com a migração pode ser recompensado através de uma diminuição no tempo de transferência de dados de resposta (pois espera-se que um volume menor de dados seja transferido), o que tem como conseqüência a otimização da utilização de recursos de

comunicação e a redução de custos. Além disso, o nó de onde o agente foi despachado tem sua carga de processamento diminuída pois o tratamento de dados é realizado nos nós para os quais o agente migra.

O próximo capítulo tece conclusões sobre o emprego da abordagem de sistemas baseados em agentes móveis e lista um conjunto de trabalhos que podem ser realizados como extensão desta dissertação.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1. Conclusões

A abordagem de sistemas baseados em agentes móveis (SBAMs) é potencialmente singular como paradigma para desenvolvimento de sistemas distribuídos. Entretanto, trata-se de uma abordagem emergente e de recente concentração de esforços de pesquisa e desenvolvimento.

Infra-estruturas para suporte a SBAMs ainda estão em fase inicial de desenvolvimento e precisam alcançar maior maturidade para que sistemas baseados em agentes móveis possam ser desenvolvidos com robustez e empregados em larga escala. Há uma grande expectativa de que, num futuro muito próximo, uma série de infra-estruturas estarão disponíveis possibilitando assim a popularização de SBAMs. Desta forma poder-se-á fazer uma avaliação mais consistente e abrangente desta abordagem, e será possível verificar se suas potencialidades são traduzidas em realidade.

O maior desafio para a consolidação de sistemas baseados em agentes móveis é o atendimento completo de requisitos de segurança. Visto que agentes podem migrar livremente numa rede, nós devem ser protegidos contra ações nocivas por parte de agentes. Por outro lado, agentes móveis não devem ser adulterados nos nós que os recebem. Limitações na atual tecnologia de computação não permitem que estes requisitos de segurança sejam tratados em sua plenitude.

O estudo de caso empregado neste trabalho (qualidade de serviço em sistemas multimídia distribuídos) possibilitou que fossem examinados os seguintes aspectos relativos a SBAMs:

- de que maneira esta abordagem atendeu aos requisitos de paradigmas de desenvolvimento de sistemas distribuídos (apresentados na seção 1.1)?
- que vantagens da abordagem de SBAM se mostraram verdadeiras na prática?

Com respeito ao primeiro aspecto, o requisito de independência de plataforma é intrínseco a agentes móveis. Facilidade de uso poderia ser explorada no estudo de caso,

através do agente de interface que é o único a interagir diretamente com seres humanos. Entretanto, tal requisito não é o foco desta dissertação e, portanto, não foi priorizado.

A abordagem de agentes se mostrou na prática uma arquitetura flexível para computação distribuída. Encapsulamento e modularidade são realmente enfatizados pois agentes são entidades autocontidas com interfaces e comportamento bem definidos. Isto torna evoluções em SBAMs uma tarefa com impactos reduzidos sobre o sistema como um todo. A infra-estrutura Aglets Workbench, utilizada nos casos de teste, contempla estes conceitos. As interfaces de agentes são estabelecidas através de protocolos de mensagens.

Atualmente não existem ainda ambientes de desenvolvimento de sistemas baseados em agentes móveis. O requisito de rápido desenvolvimento e rápida evolução pode ser atendido quando surgirem ambientes fundamentados em estratégias como Desenvolvimento Rápido de Aplicações. O requisito de facilidade de integração poderá ser atendido num futuro próximo quando padrões abertos para SBAs se consolidarem.

Com respeito ao segundo aspecto listado anteriormente, a abordagem baseada em agentes se mostrou bastante intuitiva para a estruturação de sistemas distribuídos. Agentes são entidades autônomas que realizam tarefas a elas delegadas. Esta concepção é muito similar ao comportamento de seres humanos. O modelo proposto por Oliveira et al. [Oliv97] para negociação e gerência de qualidade de serviço em sistemas multimídia distribuídos comprova tal argumento, pois é bastante intuitivo e de fácil compreensão.

A computação assíncrona é uma vantagem que foi explorada na implementação do SBAM para monitoramento de qualidade de serviço em sistemas multimídia distribuídos. A comunicação entre agentes, na maioria dos casos, se dá através de troca de mensagens assíncronas. O agente de interface, por exemplo, interage com seres humanos enquanto realiza outras tarefas, tais como: tratar os dados de qualidade de serviço recebidos pela agência onde reside. Esta capacidade é uma conseqüência da característica de *"multithreading"* da linguagem Java.

Uma vantagem potencial que não foi examinada na prática diz respeito ao suporte a computadores móveis e dispositivos leves e também o suporte à operação desconectada (este bastante relacionado à primeira). Para tanto, hardware específico deve ser utilizado.

A redução de custos de comunicação não foi avaliada através do estudo de caso. Um domínio de aplicação mais apropriado deveria ser utilizado, envolvendo sistemas distribuídos com grande transferência de dados entre nós para uma verificação consistente desta vantagem.

Os indicadores de tempo analisados comparativamente na seção 5.4 mostram que o requisito de independência de plataforma traz penalidades no que diz respeito a desempenho. Isto porque ambientes de execução de agentes são baseados em interpretação de código. Entretanto, os dados apresentados não inviabilizam a abordagem de SBAMs pois apenas uma infra-estrutura (Aglets Workbench) em fase alfa teste foi empregada. Além disso, a IBM reconheceu que o seu protocolo para migração e comunicação de agentes ainda é bastante ineficiente.

Os resultados obtidos neste trabalho englobam:

- capacitação na tecnologia Java;
- integração da infra-estrutura de computação distribuída aberta OrbixWeb (baseada no padrão CORBA[1]) com a infra-estrutura para suporte a SBAMs Aglets Workbench;
- revisão bibliográfica abrangente sobre sistemas baseados em agentes, com ênfase em agentes móveis;
- avaliação de infra-estruturas existentes para suporte a SBAMs;
- desenvolvimento de um SBAM para monitoramento de qualidade de serviço em sistemas multimídia distribuídos;
- análise comparativa para indicadores de tempo entre o SBAM desenvolvido (baseado em Aglets Workbench) e a abordagem CORBA (através da infra-estrutura OrbixWeb).

[1] *"Common Object Request Broker"*

6.2. Trabalhos futuros

As seguintes ações são sugeridas como extensão do trabalho desta dissertação:

- ampliar o escopo do SBAM desenvolvido para a etapa de monitoramento, de forma que este sistema contemple todo o modelo proposto por Oliveira et al. [Oliv97]. Esta ação visa avaliar o comportamento de SBAMs compostos por um grande número de agentes, num domínio de aplicação amplo;

- explorar aspectos de inteligência em agentes móveis (representação e manipulação de conhecimento versus mobilidade). Para tanto, as etapas de negociação e renegociação de qualidade de serviço em sistemas multimídia distribuídos podem ser utilizadas como estudo de caso. A negociação envolve gerência de conflitos (disputa por recursos), requerendo capacidades de raciocínio e aprendizado por parte de agentes. Entretanto, a utilização das etapas de negociação e renegociação depende da disponibilidade de recursos que contenham capacidades de reserva e alocação dinâmica;

- efetuar casos de teste e comparar indicadores de tempo com respeito a outras infra-estruturas para suporte a SBAMs como, por exemplo: Voyager e Odyssey. Tais infra-estruturas são promissoras, porém no momento em que este trabalho foi realizado ainda estavam em fase inicial de desenvolvimento;

- realizar casos de teste em domínios de aplicação que envolvam computadores móveis e dispositivos leves para verificar se SBAMs se mostram na realidade uma abordagem vantajosa para estes domínios;

- realizar casos de teste de forma a avaliar se SBAMs realmente reduzem custos de comunicação comparativamente com outras abordagens. Domínios de aplicação envolvendo grande transferência de dados podem ser utilizados.

Apêndice A

Dados dos Casos de Teste do SBAM Desenvolvido

Nó 2 (fluxo 1)	Nó 1 (fluxo 2)	Nó 3 (fluxo2)
2,4	4,564	3,708
2,249	1,33	0,861
0,693	0,888	1,313
1,126	0,758	0,778
0,738	0,555	1,085
0,75	0,793	0,636
0,596	0,799	0,738
0,903	0,741	0,636
0,881	0,535	0,655
0,819	0,578	0,74
1,625	0,977	0,713
0,95	0,542	0,666
0,899	0,526	0,862
0,986	0,635	0,646
0,922	0,625	0,525
1,005	0,741	0,526
0,734	0,626	0,72
0,679	0,613	0,673
0,642	0,623	0,585
0,623	0,558	0,512
Média 0,865	0,673	0,722

Tabela A.1: Tempos de migração (em segundos) para o primeiro caso de teste

Nó 2 (fluxo 1)	Nó 1 (fluxo 2)	Nó 3 (fluxo2)
6,091	8,186	14,89
3,246	8,807	3,729
1,597	3,506	3,926
1,947	3,496	3,136
1,565	3,059	3,517
2,254	3,975	3,521
1,489	3,213	3,166
1,77	3,378	3,174
1,768	3,063	3,333
1,946	3,072	2,902
2,275	3,16	3,197
1,977	2,699	2,832
2,057	2,848	2,643
1,845	3,217	3,157
1,849	2,833	3,022
1,886	3,016	2,736
1,854	2,712	2,918
1,784	2,907	2,822
1,859	2,867	2,645
1,898	2,581	2,787
Média 1,867	3,085	3,079

Tabela A.2: Tempos de resposta (em segundos) para o primeiro caso de teste

Nó 2 (fluxo 1)	Nó 1 (fluxo 2)	Nó 3 (fluxo2)	Nó 4 (fluxo2)
3,943	4,477	6,068	5,303
3,818	1,704	2,489	1,493
1,078	0,588	1,122	0,845
0,611	1,26	1,153	1,043
1,122	0,943	0,782	0,774
0,951	0,562	1,141	1,217
0,717	0,577	0,811	0,817
0,811	0,653	1,081	0,894
0,723	0,496	0,884	0,763
1,282	0,59	0,684	0,925
0,762	0,787	0,993	0,791
0,601	0,665	0,819	1,077
0,82	0,639	0,876	0,964
0,582	0,688	0,604	0,957
0,883	0,576	1,133	1,048
0,912	0,716	0,764	0,741
0,734	0,61	1,011	0,792
0,743	0,749	0,894	0,792
0,75	0,77	0,658	0,634
0,761	0,661	1,034	1,051
0,824	0,696	0,913	0,896

Média

Tabela A.3: Tempos de migração (em segundos) para o segundo caso de teste

Nó 2 (fluxo 1)	Nó 1 (fluxo 2)	Nó 3 (fluxo2)	Nó 4 (fluxo2)
7,829	7,647	18,012	27,153
4,637	22,352	16,095	9,308
2,308	8,322	6,444	6,01
1,599	6,151	5,898	5,942
2,024	5,407	5,166	4,928
1,813	4,798	4,861	5,319
1,647	5,17	5,121	4,55
1,736	4,789	5,062	5,274
1,513	5,231	4,898	4,643
2,272	4,512	4,481	4,839
2,325	5,091	5,486	5,608
1,434	5,436	4,983	4,97
1,541	4,946	5,202	4,975
1,347	5,132	4,847	5,139
2,36	4,837	5,879	5,598
1,642	6,003	5,283	5,087
1,598	4,915	5,014	4,898
1,403	5,088	5,122	5,274
1,823	5,302	5,061	4,951
1,591	4,829	5,191	5,631
Média 1,776	5,331	5,222	5,202

Tabela A.4: Tempos de resposta (em segundos) para o segundo caso de teste

Nó 2 (fluxo 1)	Nó 1 (fluxo 2)	Nó 3 (fluxo2)	Nó 4 (fluxo2)
4,825	3,584	4,9	5,072
2,086	3,625	1,416	1,348
0,972	1,17	0,674	0,915
1,633	0,844	0,736	1,003
2,956	0,747	0,849	0,944
0,85	0,711	0,744	0,861
0,686	0,626	0,783	1,119
0,841	1,06	0,572	0,756
0,899	1,242	0,647	1,126
0,83	0,63	0,559	1,016
1,024	0,763	0,634	0,965
0,887	0,802	0,747	0,969
0,795	0,645	0,67	0,865
0,838	0,764	0,544	1,022
0,713	0,766	0,721	0,678
0,572	0,629	0,561	0,857
1,191	0,698	0,624	0,63
1,298	0,923	0,844	0,843
0,556	0,641	0,655	0,735
0,559	0,839	0,589	0,677
Média 0,955	0,805	0,675	0,890

Tabela A.5: Tempos de migração (em segundos) para o terceiro caso de teste

Nó 2 (fluxo 1)	Nó 1 (fluxo 2)	Nó 3 (fluxo2)	Nó 4 (fluxo2)
9,22	7,162	15,841	24,988
3,267	22,223	16,346	10,044
2,071	8,169	6,767	5,977
2,678	5,174	5,2	5,136
3,575	5,207	5,366	5,156
1,682	4,677	4,401	4,437
2,752	4,558	4,642	4,928
1,785	5,278	5,183	4,82
1,583	4,978	4,951	5,271
1,76	4,567	4,613	4,287
2,046	4,447	4,634	4,822
1,815	4,942	5,011	4,975
1,653	4,925	4,654	4,565
1,624	4,547	4,649	4,797
1,589	4,698	4,631	4,389
1,736	4,226	4,338	4,648
2,593	4,869	4,947	4,507
2,093	4,758	4,709	5,029
1,439	4,827	4,943	4,645
1,455	4,829	4,515	4,638
Média 1,996	4,982	4,987	4,834

Tabela A.6: Tempos de resposta (em segundos) para o terceiro caso de teste

Apêndice B

Dados dos Casos de Teste da Abordagem CORBA

	Nó 2 (fluxo 1)	Nó 1 (fluxo 2)	Nó 3 (fluxo2)
	0,118	0,185	0,152
	0,018	0,024	0,019
	0,018	0,035	0,018
	0,018	0,02	0,02
	0,018	0,019	0,02
	0,018	0,019	0,037
	0,019	0,019	0,018
	0,018	0,019	0,022
	0,02	0,019	0,029
	0,018	0,036	0,017
Média	0,0183	0,0233	0,0222

Tabela B.1: Tempos de resposta (em segundos) para primeiro caso de teste

Nó 2 (fluxo 1)	Nó 1 (fluxo 2)	Nó 3 (fluxo2)	Nó 4 (fluxo2)
2,878	0,139	0,124	0,383
0,019	O,019	0,021	0,018
0,018	0,019	0,022	0,019
0,019	0,02	0,025	0,02
0,018	0,018	0,022	0,019
0,019	0,02	0,02	0,019
0,019	0,019	0,023	0,02
0,017	0,019	0,022	0,02
0,019	0,044	0,018	0,02
0,019	0,021	0,02	0,02
0,019	0,018	0,021	0,019
0,018	0,019	0,021	0,018
0,019	0,019	0,022	0,02
Média 0,0185	0,0212	0,0214	0,0193

Tabela B.2: Tempos de resposta (em segundos) para segundo caso de teste

Referências Bibliográficas

[Agen97] The Agent Society
Agent Society Home Page
http://www.agent.org
1997

[ANSA95] Advanced Networked Systems Architecture (ANSA)
Scripts and Mobile Agents
ftp://ftp.ansa.co.uk/phase3-doc-root/bn/APM.1593.01.ps.gz
1995

[Ara97] Ara – *"Agents for Remote Action"*
The Ara Platform for Mobile Agents
Grupo de Sistemas Distribuídos
Departamento de Ciência da Computação
Universidade de Kaiserslautern
http://www.uni-kl.de/AG-Nehmer/Ara
1997

[Arau97] Araújo, D. E.; Prado, R. C. M. ; Faina, L. F.; Cardozo, E.
Implementação do Módulo DPE da Arquitetura TINA-C sobre CORBA
2.º Seminário Franco-Brasileiro em Sistemas Informáticos Distribuídos
Arquiteturas Multimídia para as Telecomunicações
Fortaleza, Brasil, 1997

[Aurr95] Aurrecoechea, C.; Campbell, A.; Hauw, L.
A Review of Quality of Service Architectures
ACM Multimedia Systems Journal, Novembro, 1995

[Belg95] Belgrave, M.
The Unified Agent Architecture: A White Paper
http://maggie.ee.mcgill.ca/outgoing/belmarc/UAA.ps
1995

[Camp93] Campbell, A.; Coulson, G.; Hutchim, D.
A Multimedia Enhanced Transport Service in a Quality of Service Architectutre
Workshop de Suporte de Redes e Sistemas Operacionais para Áudio e Vídeo
Lancaster, Inglaterra, Novembro, 1993

[Card95] Cardelli, L.
Obliq: A language with distributed scope
Computing Systems, Volume 8, n.º 1, 1995

[Ches95] Chess, D.; Grosof, B.; Harrison, C.; Levine, D.; Paris, C.; Tsudik, G.
Itinerant Agents for Mobile Computing
Relatório de Pesquisa da IBM RC 20010
Divisão de Pesquisa da IBM, 1995

[Crys97] Crystaliz, Inc.
Crystaliz, Inc.
http://www.crystaliz.com
1997

[Fini93] Finin, T.
Specification of the KQML Agent-Communication Language plus example agent policies and architectures
Grupo de Trabalho em Interfaces Externas
Esforço de Compartilhamento de Conhecimento
Agência de Projetos de Pesquisa Avançados do Departamento de Defesa – DARPA
Relatório Técnico, Artigo de Trabalho, Junho, 1993

[FIPA97] The Foundation for Intelligent Physical Agents (FIPA)
FIPA Home Page
http://www.cselt.stet.it/fipa
1997

[Fran96] Franklin, S.; Graesser, A.
Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents
Proceedings do Terceiro Workshop Internacional sobre Teorias, Arquiteturas e Linguagens de Agentes
Springer-Verlag, 1996

[Gene92] Genesareth, M. R.; Fikes, R. E.
Knowledge Interchange Format Version 3.0 Reference Manual
Relatório Técnico Logic-92-1, Universidade de Stanford, Janeiro, 1992

[Gene94] Genesareth, M. R.; Ketchpel, S. P.
Software Agents
Communications of the ACM, 37, Julho, 1994

[Gene96] General Magic, Inc.
Telescript Technology: Mobile Agents
http://www.genmagic.com/Telescript/Whitepapers/wp4/whitepaper-4.html
1996

[Gilb96] Gilbert, D.; Janca, P.
IBM Intelligent Agents
http://www.raleigh.ibm.com/iag/iagwp1.html
1996

[GMD97] GMD FOKUS
WWW Homepage at FOKUS
http://www.fokus.gmd.de
1997

[Gray95] Gray, R. S.
Agent Tcl: A transportable agent system
Proceedings do Workshop CIKM sobre Agentes de Informação Inteligentes
CIKM 95, Baltimore, Maryland, 1995

[Gree97] Green, S.; Hurst, L.; Nangle, B; Cunningham, P.; Somers, F.; Evans, R.
Software Agents: A review
Grupo de Agentes Inteligentes
Departamento de Ciência da Computação
Faculdade Trinity – Dublin
Maio, 1997

[Guil95] Guilfoyle, C.
Vendors of Agent Technology
Seminário UNICOM sobre Agentes Inteligentes e suas Aplicações em Negócios
Londres, Inglaterra, Novembro, 1995

[Hafi95] Hafid, A.; Bochmann, G.
An Approach to Quality of Service Management for Distributed Multimedia Systems
Conferência Internacional de Processamento Distribuído Aberto, Australia, 1995, pp. 319-340

[Harr95] Harrison, C. G.; Chess, D. M.; Kershenbaum, A.
Mobile Agents: Are they a good idea?
Relatório de Pesquisa da IBM, Divisão de Pesquisa da IBM, 1995

[Hewi77] Hewitt, C.
Viewing Control Structures as Patterns of Passing Messages
Artificial Intelligence 8 (3), 323-364, 1977

[IBM96a] IBM Corporation
Laboratório de Pesquisa de Tóquio
Programming Mobile Agents in Java – A White Paper
http://www.trl.ibm.co.jp/aglets/whitepaper.htm
Setembro, 1996

[IBM96b] IBM Corporation
Centro de Competência em Agentes Inteligentes
The Role of Intelligent Agents in The Information Infrastructure
http://www.raleigh.ibm.com/iag/iagptc2.html
1996

[Iona97] Iona Technologies Ltd.
Iona Technologies: Our Products
http://www.iona.com/Products/index.html
1997

[ISO95] International Standards Organization
ISSO/IEC 10746-2 RM-ODP
ODP Reference Model Part 3: Architecture
Junho, 1995

[Janc96] Janca, P.; Gilbert, D.; Aparicio, M.; Atkinson, B.; Pritko, S; Rusnak, J.
Practical Design of Intelligent Agent Systems
IBM Corporation
Centro de Competência em Agentes Inteligentes
Junho, 1996

[Java97] Sun Microsystems, Inc.
Java Home Page
http://www.javasoft.com
1997

[Joha95] Johansen, D.; Renesse, R.; Schneider, F.B.
An Introduction to the TACOMA Distributed System – Version 1.0
Relatório Técnico Ciência da Computação: 95-23
Departamento de Ciência da Computação
Universidade de Tromsø, Noruega
Junho, 1995

[Kerh94] Kerhervé, B. et al.
On Distributed Multimedia Presentational Systems: Functional and Computational Architecture and QoS Negotiation
Proceedings do 4.º Workshop Internacional sobre Protocolos para Redes de Alta Velocidade
Chapman & Hall, Londres, 1994, pp. 53-67

[Kota94] Kotay, K.; Kotz, D.
Transportable Agents
Departamento de Ciência da Computação
Faculdade Dartmouth, 1994

[Ling95] Lingnau, A.; Drobnik, O.
An Infrastructure for Mobile Agents: Requirements and Architecture
Fachbereich Informatik (Telematik)
Johann Wolfgang Goethe-Universität
1995

[Maes94] Maes, P.
Agents that reduce work and information overload
Communications of the ACM, 3, Julho, 1994

[MAFS97] Cristaliz, Inc.; General Magic, Inc.; GMD FOKUS; IBM Corporation
Joint Submission – Mobile Agent Facility Specification
http://www.omg.org/library/schedule/CF_RFP3.htm
1997

[MAGN97] Projeto MAGNA
ICE/Projects/MAGNA
GMD FOKUS
http://www.fokus.gmd.de/ice/projects/magna.html
1997

[MIT97] Instituto de Tecnologia de Massachussets (MIT)
The Scheme Programming Language
http://www-swiss.ai.mit.edu/scheme-home.html
1997

[MOA97] Projeto Objetos e Agentes Móveis (MOA)
Mobile Objects and Agents Project
http://www.opengroup.org/RI/java/moa/index.htm
1997

[Mole97] Projeto Mole
Project Mole – Mobile Agents
Grupo de Sistemas Distribuídos
Instituto de Sistemas Paralelos e de Alta Performance (IPVR)
Universidade de Stuttgart
http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole.html
1997

[Nahr95] Nahrstedt, K.; Smith, J.
The QoS Broker
IEEE Multimedia, 1996 primavera, 1995, pp. 53-67

[Nwan96] Nwana, H. S.
Software Agents: An Overview
Knowledge Engineering Review, Vol. 11, No 3, pp. 1-40, Setembro,

[Obje97] ObjectSpace, Inc.
ObjectSpace: Voyager Overview
http://www.objectspace.com/voyager
1997

[O'Har96] O'Hare, G. M. P.; Jennings, N. R.
Foundations of Distributed Artificial Intelligence
John Wiley & Sons, 1996

[Oliv97] Oliveira, L. A.; Oliveira, P. C.; Faina, L. F.; Cardozo, E.
QoS Agency: An Agent-Based Architecture for Supporting Quality of Service in Distributed Multimedia Systems
Conferência IEEE sobre Protocolos para Sistemas Multimídia – Redes Multimídia
Santiago, Chile, Novembro, 1997

[OMG97] Object Management Group (OMG)
OMG Home Page
http://www.omg.org
1997

[Perl97] Linguagem de Programação Perl
The www.perl.com Home Page
http://www.perl.com
1997

[Phyt97] Linguagem de Programação Phyton
Phyton Language Home Page
http://www.phyton.org
1997

[Plei94] Projeto Pleiades
Pleiades Project Home Page
Universidade Carnegie Mellon
http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-5/www/pleiades.html
1994

[Schu96] Schulzrinne, H.; Frederik, R; Jacobson, V.
RTP: A Transport Protocol for Real Time Systems
Computer Communications, Janeiro, 1996

[Ston96] Stone, P.; Veloso, M.
Multiagent Systems: A Survey from a Machine Learning Perspective
Submetido a IEEE Transactions on Knowledge and Data Engineering
Junho, 1996

[SunS97] Sunscript
SunScript Home Page
http://sunscript.sun.com
1997

[Visi97] Visigenic Software, Inc.
Visigenic Software - Product Overview
http://www.visigenic.com/prod
1997

[Voge95] Vogel, A. et al.
Distributed Multimedia and QoS: A Survey
IEEE Multimedia, verão, 1995, pp. 10-18

[Wool95] Wooldridge, M., Jennings, N.
Intelligent Agents: Theory and Practice
Knowledge Engineering Review, Volume 10, No 2, Junho, 1995