UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Um ambiente para a interação de agentes na Internet

por

ROBERTO DUARTE FONTES

Dissertação submetida à avaliação, como requisito parcial para
a obtenção do grau de Mestre em
Ciência da Computação

Prof. Dr. Luís Otávio Campos Alvares
Orientador

Porto Alegre, janeiro de 2000.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Fontes, Roberto Duarte

Um ambiente para a interação de agentes na Internet / por Roberto Duarte Fontes – Porto Alegre: PPGC da UFRGS, 2000.

113 f.: il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2000. Orientador: Alvares, Luís Otávio Campos.

1. Inteligência Artificial Distribuída. 2. Agentes Inteligentes. 3. Sistemas Multiagentes. 4. Internet. 5. Redes de Computadores. I. Alvares, Luís Otávio Campos. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profa. Wrana Panizzi
Pró-Reitor de Pós-Graduação: Prof. Franz Rainer Semmelmann
Diretor do Instituto de Informática: Prof. Philippe O. A. Navaux
Coordenadora do PPGC: Profa. Carla Maria Dal Sasso Freitas
Bibliotecária-Chefe do Instituto de Informática: Beatriz Haro
Bibliotecária responsável pela normalização de documentos: Ida Rossi

Em memória ao meu Pai

Adão de Paiva Fontes

Agradecimentos

Gostaria de agradecer a muitas pessoas que neste tempo de mestrado, deram-me muito auxílio e confiança. Citar nomes pode parecer injusto, pois corro o risco de esquecer alguns, no entanto, algumas pessoas foram tão importantes que não agradecer explicitamente a elas seria muito mais grave.

A minha mãe Joana, que me acompanha nos momentos mais difíceis, importantes e alegres da vida. Sempre com palavras de carinho, apoio e estímulo. Quem dera que todas as mães fossem que nem ela.

A todos os colegas que com o passar do tempo se tornaram meus grandes amigos, dentre eles: Luciano Ferreira, André Schneider e André Puppin (parcerias de apartamento), Miguel Feldens e Silvio Cazella, que através do espírito de companheirismo tornaram mais alegres os momentos dentro e fora do Instituto de Informática. A todas as demais pessoas que conheci durante este período.

A todos os funcionários do Instituto de Informática e da Biblioteca pelo esmero e disposição que sempre colocaram em seus serviços; em especial, as funcionárias Eliane Iranco e Ida Rossi.

Ao meu professor orientador, Luis Otávio Alvares, pelo companheirismo, orientação e grande dedicação.

Agradeço a todas as pessoas que ajudaram, diretas ou indiretamente, durante a realização deste curso de mestrado.

Por fim, agradeço a Deus, sem o qual nada seria possível, pela vida, pela esperança e pelas bênçãos.

Sumário

Lista de Abreviaturas

ACC	Canal de comunicação de agentes *(Agent Communication Channel*)
AMS	Sistema de gerenciamento de agentes (*Agent Management System*)
ANS	Servidor de nomes de agentes (*Agent Name Server*)
AR	Roteador de agentes (*Agent Router*)
BNF	Notação para descrição de gramáticas livres de contexto (*Backus-Naur Form*)
BSA	Agente Básico de *Software* (*Basic Software Agent*)
DF	Módulo para pesquisa de agentes (*Directory Facilitator*)
DNS	Sistema de Nomes de Referência (*Domain Name System*)
FIPA	Fundação que estuda agentes inteligentes (*Foundation for Intelligent Physical Agents*)
FTP	Protocolo de transferência de arquivos (*File Transfer Protocol*)
HTML	Linguagem de marcação de hipertexto (*Hyper Text Markup Language*)
IA	Inteligência Artificial
IAD	Inteligência Artificial Distribuída
KQML	Linguagem de Consulta e Manipulação do Conhecimento (*Knowledge Query and Manipulation Language*)
LAN	Rede de computadores com abrangência local (*Local Area Network*)
MAN	Rede de computadores com abrangência metropolitana (*Metropolitan Area Network*)
NIC	Centro de informações sobre rede (*Network Information Center*)
OSI	Modelo de referência para a interconexão de sistemas abertos (*Open Systems Interconnection*)
PDU	Unidade protocolar de dados (*Protocol Data Unit*)
QOS	Qualidade de serviço (*Quality Of Service*)

RFC	Conjunto informal de artigos sobre a Internet (*Request For Comments*)
SAP	Ponto de acesso de serviço (*Service Access Point*)
SDL	Linguagem de especificação e descrição (*Specification and Description Language*)
SMA	Sistemas Multiagentes
SMTP	Protocolo de transferência de e-mail (*Simple Mail Transfer Protocol*)
TCP/IP	Protocolo de controle de Transmissão / Protocolo da Internet (*Transmission Control Protocol / Internet Protocol)*
URL	Localizador Uniforme de Recurso (*Uniform Resource Locator*)
UTF-8	Formato de transformação do padrão Unicode (*Unicode Transformation Format*)
WAN	Rede de computadores com abrangência mundial (*Wide Area Network*)

Lista de Figuras

Lista de Tabelas

Resumo

O grande salto tecnológico ocorrido nos últimos decênios, em áreas como a informática e as telecomunicações, aliado à heterogeneidade de arquiteturas de *hardware* e sistemas operacionais, faz da Internet um ambiente propício para a utilização de agentes.

Este trabalho tem por objetivo principal apresentar a especificação de um modelo destinado à interação de agentes autônomos no âmbito da Internet, de forma a possibilitar, entre outros, a localização de agentes e a conexão entre os agentes, permitindo a troca de mensagens entre eles.

Para tanto, inicialmente apresentam-se os conceitos básicos relacionados a agentes, sistemas multiagentes e uma visão geral da estrutura e protocolos utilizados na Internet. A seguir é apresentado informalmente um modelo de ambiente para facilitar a interação de agentes na Internet.

Tal modelo é depois especificado detalhadamente e materializado na forma de um protótipo implementado e que provê o serviço proposto. Finalmente, expõem-se as conclusões do trabalho em termo das principais características, limitações e futuras extensões.

Palavras-chave: Inteligência Artificial Distribuída, Sistemas Multiagentes, Internet, Rede de Computadores.

Title: "An environment for agents' interaction on the Internet".

Abstract

Recent major technological progress in areas such as information technology and telecommunications, together with hardware architecture and operating system hybridity make the Internet a favorable environment for agents' use.

This study aims principally to present specifications for a model geared towards autonomous agent interaction on the Internet in order to make possible agent location and connection, among other functions. This will thus allow for message exchange.

In light of the above, basic concepts related to agents, multi-agent systems and a general overview of the structure and protocols used on the Internet are presented initially. To follow, an environment model for facilitating agent interaction on the Internet is summarily described.

This model is then described in further detail and materialized in the form of an applied prototype that provides the functions proposed. Finally, this study's conclusions are outlined in terms of its main characteristics, limitations and future extensions.

Keywords: Distributed Artificial Intelligence, Multi-Agent Systems, Internet, Network Computers.

1 Introdução

A Inteligência Artificial Distribuída (IAD) é uma nova concepção dentro do escopo geral da Inteligência Artificial (IA), que acompanha o avanço da tecnologia de desenvolvimento de máquinas paralelas e a difusão em larga escala de sistemas computacionais distribuídos.

As pesquisas que determinaram a especialização da IAD têm-se tornado um domínio estabelecido nos últimos anos. Como oposto as ferramentas de IA clássica, em que a metáfora de inteligência é baseada no comportamento humano individual e a ênfase está na representação do conhecimento e métodos de inferência, a metáfora usada em IAD fundamenta-se no comportamento social, enfatizando ações e interações [SIC 92].

De acordo com Boisser e Demazeau [BOI 92] a IAD constitui-se em uma das áreas de pesquisa da IA tradicional. No entanto, a IAD não é apenas uma questão de implementar sistemas de IA tradicional em uma plataforma distribuída. A principal diferença entre a IAD e a área de Processamento Distribuído (PD) reside no nível de abstração: enquanto os conceitos técnicos de PD referem-se à distribuição e fluxo de dados entre diferentes elementos de processamento fisicamente distintos, em IAD o foco de atenção volta-se para aspectos de interação, cooperação, ou seja, distribuição dinâmica do controle e das ações para o fluxo de conhecimento entre unidades logicamente distintas.

O estudo do comportamento computacionalmente inteligente, resultante da interação de múltiplas entidades - dotadas de um certo grau de autonomia e chamadas de agentes - e do sistema como um todo denominado sociedade, pode ser conceituado como IAD [OLI 96]. A autonomia diz respeito à habilidade, concernente ao agente, com a finalidade de tomar iniciativa e exercer um grau de controle sobre suas ações, requerendo inteligência devido à necessidade de sobrevivência em um ambiente real, dinâmico e nem sempre benigno.

Existem inúmeras razões para a utilização da IAD. A seguir são explicitadas algumas:

- necessidade de poder computacional, pelo fato de que a resolução de problemas de forma centralizada exige *hardware* de alto custo;

- maior segurança e tolerância a falhas, pois quando um problema é proposto, esse é analisado por vários sistemas em paralelo, de forma a chegar-se à mesma solução através de formas e métodos diferentes;

- aproveitamento da tecnologia existente para obter soluções de problemas que, com os sistemas tradicionais, não poderiam ser resolvidos.

O termo agente é muito utilizado, tendo-se tornado uma marca de qualidade em termos mercadológicos. Praticamente, no mercado de hoje, todo o programa é dito como incluindo *agent technology*, como, por exemplo, processadores de textos que possuem flexibilidade, programas que verificam quando algo é digitado, alertando quando uma palavra está mal escrita. Esses programas são agentes? Na verdade, deve-se tomar algum cuidado ao definir-se se um determinado *software* é ou não um agente, por quanto o mundo dos *softwares* que hoje vivem sobre a rubrica dos agentes é multifacetado. Mas os agentes possuem algumas características próprias que os diferenciam dos demais *software*.

Conceitualmente, agentes são entidades capazes de reagir e/ou agir a favor de algo ou alguém, influenciando seu ambiente. Este ambiente, necessariamente, inclui outros agentes. Interagir com este ambiente, ou, mais especificamente, com outros agentes é o elemento chave de um Sistema Multiagente (SMA).

A Internet é um ambiente privilegiado para a utilização de agentes em virtude de suas características. Dentre as principais, podem-se citar a arquitetura, a topologia, a heterogeneidade de *hardware* e *software*, além da abrangência.

A motivação maior na realização deste trabalho foi a de facilitar a interação de agentes autônomos em um ambiente altamente dinâmico e heterogêneo, como a Internet.

Outro aspecto motivador foi a possibilidade de fundir várias tecnologias em um único trabalho: agentes, sistemas multiagentes, arquitetura, protocolos e serviços Internet, programação para redes, orientação a objetos e multiprogramação.

Os objetivos desta dissertação são o refinamento e implementação da arquitetura para a coordenação de agentes na Internet, proposta por Cazella [CAZ 97], a definição e especificação de um protocolo que torne a comunicação entre os agentes transparente ao usuário e a agregação deste protocolo à arquitetura. Em outras palavras, os objetivos são a criação de um ambiente que permita a implementação de sistemas multiagentes na Internet.

Para se atingir tais objetivos, foram traçadas algumas diretrizes básicas para o trabalho. Entre estas se podem citar o estudo de outras arquiteturas existentes, o estudo da linguagem Java - a utilizada para a implementação protótipo do ambiente, de alguns tópicos estruturais e de alguns protocolos utilizados na Internet.

A seguir, apresenta-se uma descrição da organização deste trabalho, destacando-se os pontos importantes de cada capítulo:

O Capítulo 2 introduz os conceitos básicos relacionados a agentes e sistemas multiagentes;

No Capítulo 3, expõem-se a arquitetura, protocolos e conceitos básicos relativos à Internet;

Já a arquitetura geral proposta para facilitar a interação de agentes na Internet é apresentada no Capítulo 4;

No Capítulo 5, materializa-se o modelo introduzido no Capítulo 4 através da especificação em linguagem coloquial e por intermédio de diagramas;

A forma como se realizou a análise das propriedades estáticas e dinâmicas do Protocolo, é explicitada no Capítulo 6.

O Capítulo 7 descreve detalhes em nível de implementação do ambiente, procurando-se mostrar as estruturas internas básicas dos componentes implementados e suas respectivas funcionalidades;

No Capítulo 8, são apresentadas, comparações com outras propostas desenvolvidas também para dar suporte a implementações de agentes na Internet.

Finalmente, as conclusões do trabalho em termos de contribuições, limitações e trabalhos futuros compõem o Capítulo 9.

2 Agentes

Tendo em vista a crescente utilização das redes de computadores e a necessidade de agentes independentes e autônomos que possam realizar tarefas em grupo, através da interação e colaboração de outros agentes, os SMA estão-se proliferando rapidamente. A problemática sempre presente nas pesquisas era como fazer com que programas baseados no conhecimento pudessem comunicar-se em uma rede para solucionar problemas complexos.

A solução, então, foi a utilização de agentes. A palavra Agente vem do latim “agaens”, que significa produtor de resultados, uma pessoa ou coisa que realiza uma ação. Agentes são assuntos freqüentes na IA, mas, apesar disso, ainda não se tem uma definição precisa deles. Pesquisadores têm proposto várias definições, desde um nível elementar a um mais elaborado, sendo que a maioria deles definem agentes como entidades, sejam de *software* ou de *hardware*, as quais executam um conjunto de operações, beneficiando um usuário ou outro programa, com algum grau de independência e autonomia, utilizando algum conhecimento dos objetivos e desejos dos usuários.

Berthet, em [BER 92], os define como entidades capazes de receber estímulos, inferir e agir ou reagir, tendo, em seu comportamento, um objetivo definido.

Um agente pode ser considerado um meio que produz um certo número de ações a partir dos conhecimentos e mecanismos internos que lhe são próprios [BIT 96].

Agentes são programas que travam diálogos, negociam e coordenam transferência de informações [COE 94].

Agente pode ser definido como alguém ou alguma coisa que atua como um representante para outro partido, com o propósito expresso de desempenhar ações que são benéficas para a parte representada. É diferenciado de outras aplicações por suas dimensões e capacidade de interagir independentemente da presença do usuário [HEI 95].

Os agentes apresentam conceitos de habilidade para execução autônoma e habilidade para executar raciocínio orientado ao domínio [VIR 95].

Agente é uma entidade persistente dedicada a um propósito específico. 'Persistente' distingue agente de sub-rotinas; agentes têm suas próprias idéias sobre como realizar tarefas, suas próprias agendas. 'Propósito específico' distingue-se de toda aplicação multifunção; agentes são tipicamente mais inteligentes [FRA 96].

Uma definição mais geral encontrada na literatura é a de Demazeau [DEM 92], e refere-se a agentes da seguinte forma: uma entidade real ou virtual que está inserido num ambiente onde pode tomar algumas ações, que é capaz de perceber e representar parcialmente esse ambiente, que é capaz de comunicar-se com outros agentes e que possui um comportamento autônomo - uma conseqüência de sua observação, seu conhecimento e suas interações com outros agentes.

As definições apresentadas tentam exprimir o que cada autor entende por agente, baseado nas características que seu agente possui. Pelo fato de cada autor, na maioria das definições encontradas, envolver o problema que o agente busca solucionar, encontram-se as diferentes interpretações do termo.

Fundamentando-se nessas afirmações, pode-se concluir que os agentes são entidades, que se comportam em um ambiente de acordo com seus próprios objetivos, suas observações, seu estado corrente de conhecimento e interações com outros agentes visando a atingir seus objetivos propostos através da execução de suas ações.

A figura 2.1, baseada em Demazeau [DEM 90], apresenta um modelo de agente, no qual suas capacidades de comunicação, percepção do ambiente em que está inserido, conhecimento adquirido, metas e capacidade de raciocínio fazem com que o agente tome suas decisões e, com base nelas, execute suas ações. Um modelo de arquitetura de agentes que leva em conta as crenças, desejos, intenções e expectativas, influenciando a capacidade interativa do agente, é apresentado em [COR 93].

As ações dos agentes são intencionais [HÜB 95]. De acordo com essa afirmação, pode-se dizer que o agente deliberativo planeja suas ações com base nas crenças e desejos mentais que possui, satisfazendo, com isso, uma determinada intenção. Portanto, um agente:

- tem conhecimento sobre o problema. Pode-se obter este conhecimento perguntando ao usuário, percebendo variáveis do ambiente que o cerca, ou comunicando-se com outros agentes;
- tem metas que deve almejar no transcorrer do tempo;
- é capaz de analisar um conjunto de possíveis soluções para essas metas, dependendo de sua capacidade de raciocínio;
- tem capacidade de tomar decisões.

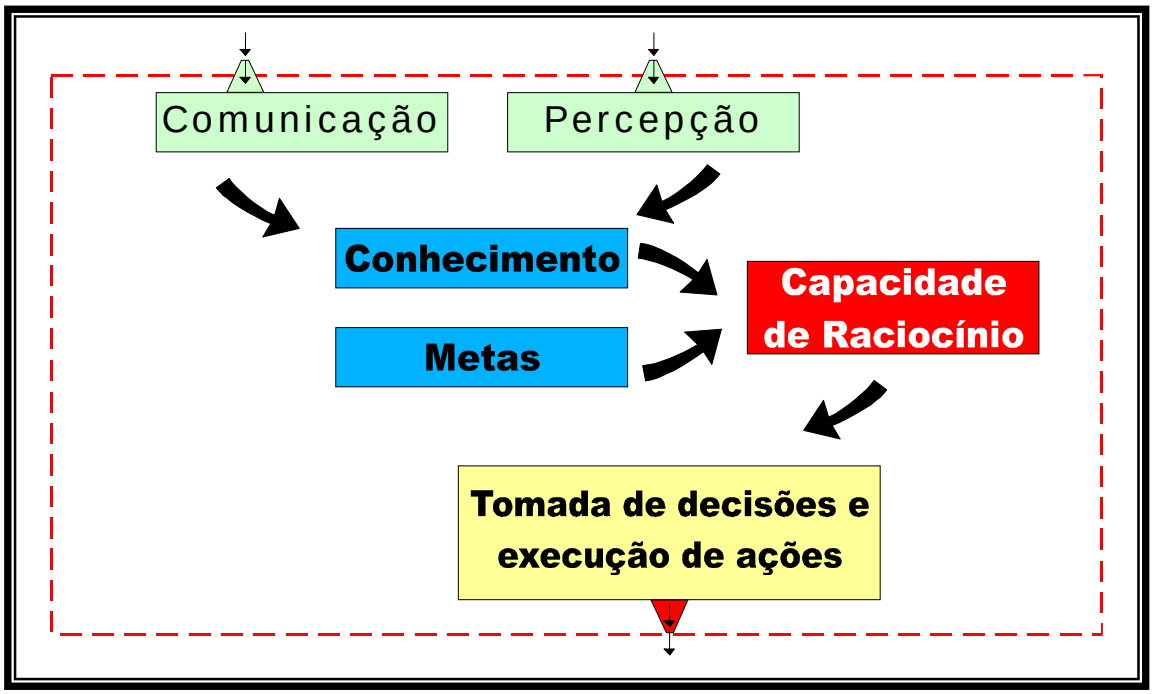

FIGURA 2.1 - Modelo de agente

2.1 Propriedades de Agentes

As propriedades que um agente deve possuir segundo Ferber [FER 91], são:

- capacidade de percepção: um agente deve perceber ou compreender seu ambiente e as mudanças que podem ocorrer nele em conseqüência de ação de outros agentes;
- capacidade de comunicação: um agente deve ser capaz de comunicar-se com outros agentes, podendo, durante a comunicação, haver troca de conhecimento e de planos;
- capacidade de ação: um agente deve ser capaz de realizar ações em seu ambiente a fim de resolver problemas;
- raciocínio social: um agente deve ser capaz de raciocinar sobre as atividades de outros agentes devido à representação interna dos membros da sociedade;
- estrutura de controle: um agente deve ser capaz de decidir quando compreender, comunicar, planejar e atuar; assim, esta estrutura deve ter condições de possibilitar a ativação das características anteriormente citadas.

2.2 Classificação de Agentes

A capacidade de resolução de problemas por parte do agente e a arquitetura do agente são fatores que identificam três grandes categorias de agentes conforme figura 2.2 [SIC 92].

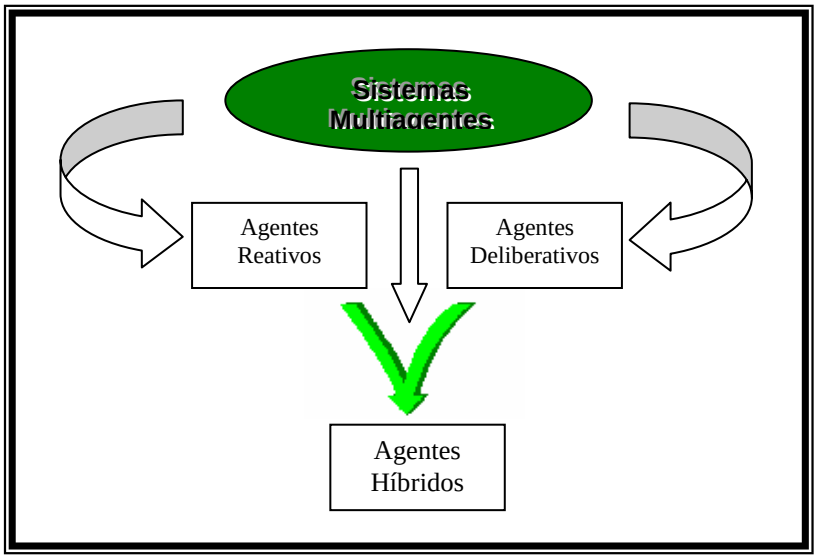

FIGURA 2.2 - Classificação de agentes

2.2.1 Agentes Reativos

Estes agentes não possuem uma representação explícita do ambiente, ou seja, seus conhecimentos se manifestam através de seus comportamentos. Dessa forma, eles não sabem da existência dos outros agentes inseridos em seu ambiente, apoiando suas ações somente nas percepções que têm do mundo, não as utilizando no planejamento de suas ações futuras.

Cada agente, analisando-o individualmente, não possui inteligência, mas sim em coletividade [DEM 91]. Sociedades de agentes reativos reúnem um elevado número de membros que se comunicam através de propagação de sinais, apresentando comportamento simples, do tipo estímulo-resposta. Assim, como as respostas são obtidas de forma rápida, os sistemas reativos podem ser vantajosos pelo baixo tempo de resposta.

Os agentes reativos são tidos como agentes não deliberativos, ou seja, seus objetivos não são explícitos, considerando-se o seu comportamento emergente. Definidos também como agentes basicamente comportamentais,

isto é, a partir de uma ação, sofrem uma reação, são, por isso, denominados estímulos-respostas.

A seguir, suas principais características são explicitadas:

- baseiam-se em modelos de organização etológica ou biológica – relacionados ao comportamento dos animais, ex: sociedade de formigas, já citada anteriormente;

- possuem uma representação implícita do conhecimento sobre o ambiente e sobre os outros agentes – (o conhecimento é representado pelo comportamento dos agentes reativos no sistema multiagente);

- são agentes baseados em comportamento – cada agente, individualmente, exibe comportamentos de acordo com a situação na qual se encontram no ambiente de solução de algum problema. (Quando o ambiente se altera, os agentes reativos mudam seu comportamento);

- o seu modo de funcionamento é por estímulo-resposta – a relação do agente reativo com seu ambiente ocorre através de respostas a estímulos recebidos. O agente executa uma determinada ação quando uma certa condição for satisfeita;

- possuem capacidade de percepção e comunicação – a comunicação ocorre através do ambiente, o qual é utilizado como meio de transmissão de mensagens;

- não possuem raciocínio – os agentes reativos não possuem memória das suas ações executadas no passado nem qualquer previsão das ações a serem executadas no futuro. Como não têm capacidade de raciocínio, não planejam suas ações, mas reagem a estímulos;

- apresentam grande quantidade de elementos – a sociedade de agentes reativos reúne, geralmente, um grande número de membros.

2.2.2 Agentes Deliberativos

Os agentes deliberativos possuem representação explícita do ambiente e de outros agentes, podendo raciocinar e decidir sobre seus objetivos, escolher quais planos seguir e quais ações devem ser executadas para isso. Dispõem de uma memória de ações executadas e, com base nisso, podem planejar suas ações futuras através da criação de planos. Assim como as organizações humanas são sociedades compostas por um pequeno número

de membros que utilizam, como forma de comunicação, a troca de mensagens - envio e recebimento.

As características destes agentes, segundo Sichman [SIC 93], constituem-se em:

- capacidade de percepção – um agente deve perceber seu ambiente, sendo capaz de implementar mudanças. O ambiente muda devido às intervenções ou devido às ações tomadas por outros agentes;

- capacidade de resolução de problemas – a quantidade de conhecimento, em um Sistema Baseado em Conhecimento, pode ser considerada como uma enorme base de conhecimento, a qual influi numa maior capacidade de resolução de problemas;

- capacidade de ação – um agente deve ser capaz de agir em seu ambiente, influenciando na resolução final do problema;

- capacidade de comunicação – um agente deve ser capaz de comunicar-se com outros agentes. (O termo comunicação é usado em um sentido vasto, significando que agentes podem trocar não somente dados, mas "peças" de conhecimento, bem como planos);

- sociabilidade – um agente deve ter capacidade de reagir a atividades de outros agentes. Este fato consuma-se por meio de uma representação interna dos atributos de outros agentes;

- estrutura de multicamada – um agente deve decidir quando perceber, comunicar-se, planejar e agir. Portanto, uma estrutura de controle complexa deve ser fornecida à medida que ativa cada uma das diferentes atividades.

2.2.3 Agentes Híbridos

Os agentes híbridos são aqueles que, em seu funcionamento, dispõem da combinação de duas ou mais capacidades pertencentes aos tipos anteriormente explicados.

É raro haver um agente somente reativo, que só reage aos estímulos do ambiente, ou um agente só deliberativo, que possui uma representação simbólica de seu ambiente. A diferença entre as duas abordagens é apenas teórica, acentua Sichman [SIC 95], acreditando que, na prática, existe um compromisso entre a eficácia e a complexidade.

Um sistema híbrido é representado, na maioria das vezes, por uma grande quantidade de agentes reativos e uma pequena de agentes

deliberativo. Este fato se deve à simplicidade dos agentes reativos em relação aos deliberativos.

Um dado importante que se deve levar em conta na escolha do modelo do agente, é a necessidade de responder às restrições de tempo real. Isso obriga o agente a ter respostas rápidas (aspecto reativo), e a dispor de uma planificação das tarefas a mais longo prazo (aspecto cognitivo).

2.3 Sistemas Multiagentes

A fundamentação dos Sistemas Multiagentes é baseada na interação social de indivíduos que convivem entre si e interagem mutuamente a fim de atingir objetivos comuns e individuais. Desse modo, um agente é concebido como um indivíduo autônomo, com funções que lhe são inerentes para o desempenho de suas funções e o alcance dos seus objetivos. Estes agentes, no entanto, dividem um ambiente comum, e cada um possui diferentes objetivos e pontos de vista, gerando, muitas vezes, alguns conflitos.

Em uma sociedade humana, um problema complexo pode ser resolvido de forma eficiente por um grupo de indivíduos competentes e cuja organização evolui dinamicamente e de maneira autônoma, a fim de elaborar a resolução do problema. Criar uma organização que evolui dinamicamente e de maneira autônoma equivale a elaborar a auto-organização de um grupo de indivíduos.

A auto-organização de uma sociedade de agentes pode ser tomada como a modificação da topologia dos agentes, a qual ocorre de forma autônoma, permitindo-lhes adaptarem-se ao seu ambiente. Esta reorganização dinâmica da sociedade surge em virtude das mudanças nas interações entre os agentes e o ambiente, ou seja, no estado interno dos mesmos.

Uma sociedade de agentes é composta pelo ambiente no qual os agentes estão inseridos, por um método de organização da sociedade, e pelas interações entre os agentes e seu ambiente, envolvendo formas de comunicação entre os mesmos. (Ver figura 2.3)

Segundo Oliveira [OLI 96], as sociedades de agentes classificam-se como:

- homogêneas ou heterogêneas – homogêneas quando os agentes são todos do mesmo tipo (mesma arquitetura); heterogêneas, em caso contrário;

- fechadas ou abertas – sempre que os agentes são fixos no ambiente, é considerada fechada; sempre que há possibilidade de migração (entrada/saída de agentes), aberta;

- baseadas em normas ou não – toda vez que existem regras explícitas de comportamento - válidas para toda a sociedade, - é considerada baseada em normas; caso contrário, não.

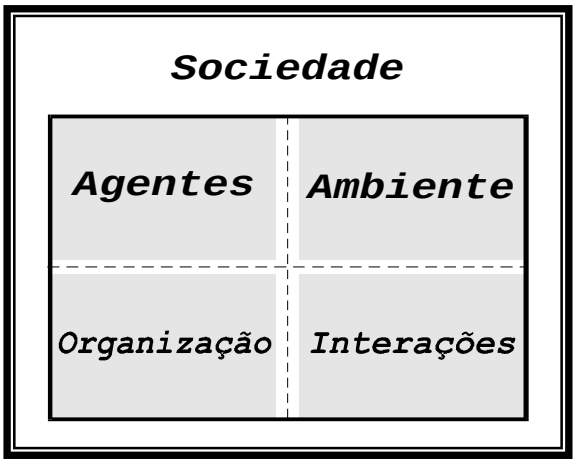

FIGURA 2.3 - Sociedade de Agentes

2.4 Comportamentos de Agentes

De acordo com Sichman e Demazeau [SIC 92], existem dois critérios para a classificação dos comportamentos apresentados por agentes em um ambiente.

- *Localização da tarefa* – local ou global:

  uma tarefa local engloba somente um agente, sendo que a tarefa global abrange todos os agentes.

- *Capacidade de execução de tarefas* – capaz ou incapaz:

  um agente é capaz de executar uma tarefa se ele tem as ferramentas necessárias para realizá-la.

Segundo esses critérios, podem ocorrer os seguintes comportamentos:

- *Cooperação* – para a execução de uma tarefa, um agente terá de cooperar com outros, ou porque não é capaz de realizá-la sozinho (soluções possivelmente restritas), ou porque os

outros podem realizá-la mais eficientemente (em um menor período de tempo);

- *Colaboração* – algumas metas globais podem envolver todos os agentes e podem ser realizadas individualmente por vários agentes. (O principal problema é selecionar esses agentes);

- *Distribuição* – algumas metas globais apenas podem ser realizadas por vários agentes coletivamente. (O principal problema é dividir e distribuir a tarefa entre os vários agentes);

- *Co-habitação* – o agente tem de desincumbir-se de uma tarefa com sucesso e ser capaz de fazê-la sozinho. Tal fato é típico de problemas clássicos de IA em um ambiente monoagente.

2.5 Comunicação entre Agentes

O comportamento dos agentes baseia-se fortemente na capacidade de comunicação que possuem. Os processos de alocação de tarefas, cooperação, transmissão de resultados, resolução de conflitos, negociação, entre outros, sucedem-se no âmbito das interações. Evidentemente, estas, por sua vez, apóiam-se na definição dos processos de comunicação disponíveis.

Um agente também pode perceber o ambiente por meio de sensores, tais como sensores visuais. A percepção e a comunicação podem ser vistas como ações a serem planejadas e, através da aquisição de conhecimento sobre outros agentes, um agente pode ser capaz de reconhecer essas ações.

Podem-se distinguir três tipos de trocas de informações entre os agentes:

- conhecimento do agente – usualmente é um modelo do ambiente, mas pode incluir um modelo do próprio agente. Dois agentes podem ter descrições complementares ou conflitantes de uma situação compartilhada, devido às diferenças na percepção do ambiente por ambos. Uma descrição é conflitante quando surgem descrições contraditórias sobre uma mesma situação do ambiente;

- planos ou caminhos possíveis para uma solução – a troca de soluções possíveis ocorre quando dois ou mais agentes concordam com uma solução ou plano comum. Tal processo pode ser feito encontrando-se a intersecção das soluções de cada agente. A intersecção pode ser vazia, devido às diferentes capacidades de raciocínio dos agentes;

- escolha de um plano – após ser encontrado um conjunto de possíveis soluções, uma delas precisa ser de escolha comum, quando a cooperação for necessária. Para realizar essa escolha, duas possibilidades são oferecidas: escolher a primeira solução emitida da intersecção do conjunto de possíveis soluções ou exigir raciocínio sobre a capacidade de decisão de cada agente.

A necessidade de escolha aparece, também, quando um agente requisita a outro uma tarefa. No entanto, quando uma escolha não é aceita pelo agente, a negociação pode ocorrer.

Em qualquer comunicação, alguma discrepância é esperada entre o significado do que o remetente pretendia expressar e aquele que o destinatário entendeu. Quanto menor esta discrepância, mais acurada é a compreensão das partes e mais efetiva a troca de informação. Esta efetividade é aumentada substancialmente quando o remetente e o destinatário compartilham a mesma linguagem, das mesmas bases culturais, educacionais, dos mesmos pontos de referência e da familiaridade com o assunto em questão, portanto, o processo de comunicação torna-se um problema se as partes não estão cientes da possível diferença entre o intencionado e o percebido [STE 95].

Se a comunicação for descrita em termos de envio e recebimento de mensagens, cada parte integrante desta deve ser capaz de inferir no que o remetente intencionava quando a proferiu, como por exemplo, se estava solicitando alguma informação ou comandando algum ato. Na inexistência de estruturação nas mensagens, esta inferência torna-se um processo ineficiente. Sendo assim, as mensagens devem estar contidas por restrições formais e estruturadas para a facilidade de sua interpretação, por exemplo: distinguindo tipos de mensagens nas quais a intenção do remetente possa ser imediatamente reconhecida pela mensagem em si.

A simples estruturação pode não ser suficiente, já que os agentes devem saber como reagir às mensagens recebidas ou o que esperar após as enviadas. Estes requerimentos devem ser tratados como uma estrutura única, denominada de “diálogos estruturados entre protocolos de cooperação de agentes” [STE 95].

Sendo a comunicação utilizada para a troca de conhecimento. Um esquema de comunicação entre agentes controlados por protocolos – no sentido de diplomacia, comportamento da comunicação, entre outros – é uma maneira estruturada de simplificar o controle da comunicação e de diminuir sua sobrecarga [CAM 90].

Werner [WER 87] vê nos protocolos mais do que uma especificação de quem pode dizer o que, para quem e quais as possíveis reações do outro participante ao que é dito. Para ele, numa especificação de um protocolo está implícita uma especificação da arquitetura global do sistema.

O controle da comunicação sempre foi um problema crucial em sistemas distribuídos. A capacidade limitada dos canais de comunicação representa um gargalo para o desempenho de sistemas. Na IAD o problema é ampliado pela necessidade de interação entre os agentes e por não haver um consenso sobre como a comunicação deve ser tratada. As soluções propostas sempre tentaram a comunicação implícita por uma memória compartilhada, ou, se os agentes são autônomos, pela aplicação de estratégias de controle especializadas. Ambas as abordagens utilizam soluções diferentes do comportamento humano em situações similares [BOR 95].

A comunicação entre os agentes de uma sociedade tem duas formas:

- comunicação direta – existe quando os agentes de uma sociedade conhecem uns aos outros e trocam informações. Neste tipo de comunicação, encontram-se as arquiteturas baseadas em trocas de mensagens;

- comunicação indireta – há quando os agentes se comunicam sem conhecer uns aos outros, pelo envio e recebimento de informações, de acordo com características pré-definidas. Neste tipo de comunicação, encontram-se as arquiteturas de quadro-negro (*blackboard*).

2.5.1 Arquitetura de Quadro-negro

Na arquitetura de quadro-negro, a comunicação dá-se através de uma estrutura de conhecimento compartilhada, chamada quadro-negro (*blackboard*), a qual é geralmente dividida em regiões ou níveis. Os agentes podem escrever ou ler em um ou mais níveis, sob a supervisão de um mecanismo global de escalonamento. Não existe comunicação propriamente dita entre os agentes, todas as interações ocorrem pelo quadro-negro (fig. 2.4), informações adicionais sobre esta arquitetura podem ser encontradas em [ROT 84].

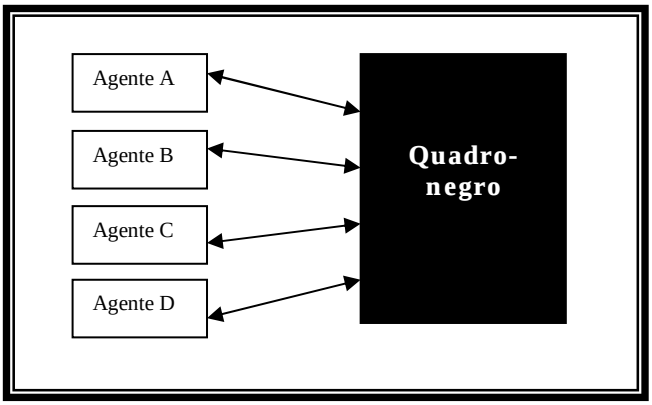

FIGURA 2.4 - Arquitetura de quadro-negro

2.5.2 Troca de Mensagens

Na comunicação baseada em mensagens, os agentes comunicam-se pela troca de mensagens. Para isso, os nomes dos agentes devem ser explicitamente conhecidos. Tipicamente, cada agente possuirá uma representação dos outros agentes, incluindo suas capacidades, seus objetivos e, possivelmente, seus conhecimentos/crenças.

Evidentemente, as potencialidades de interação, no caso de sistemas de troca de mensagens, são mais ricas do que nas arquiteturas de quadro-negro. Em contrapartida, a regulamentação dessas interações torna-se bem mais complexa. Usualmente, a organização das interações é feita com base em protocolos que definem os passos de diálogo a serem executados pelos agentes, para cada tipo de interação possível na sociedade. Os protocolos e os formalismos para a representação de mensagens podem ser bastante variados, dependendo do domínio da aplicação. Como exemplos, têm-se o protocolo de interação, requisição até satisfação, redes de contrato, negociação, cooperação, entre outros [DEM 95].

2.5.3 Linguagem de Comunicação

As linguagens de comunicação entre agentes devem definir como [AMA 96]:

- transferir conhecimento;
- realizar solicitações;
- aceitar ou rejeitar uma solicitação;

- negociar introduzindo outras propostas;
- informar falhas;
- informar se está ou não satisfeito com a colaboração.

Segundo vários autores, pode-se afirmar que uma linguagem, para a comunicação entre agentes, deve conter as seguintes características, quanto à sua forma:

- ser declarativa;
- ser legível (para as pessoas);
- ser sintaticamente simples, e extensível, já que a linguagem será integrada a uma ampla variedade de sistemas;
- ser de fácil análise e de geração;
- ser concisa;
- ser linear ou facilmente traduzível em uma estrutura linear, já que para transmissão de mensagens entre agentes, a linguagem deve ser transformada, pela camada inferior do mecanismo de comunicação, em uma cadeia de bits.

Para tal, uma linguagem de comunicação deve ser estruturada em camadas, de forma a ter uma conectividade com outros sistemas. Em particular, uma distinção deve ser feita entre - linguagem de comunicação, que expressa atos de comunicação -, e o conteúdo da linguagem, que expressa fatos sobre o domínio. Esta divisão em camadas facilita a integração: linguagem–aplicação, provendo uma estrutura para a decodificação da linguagem.

Um tópico freqüentemente negligenciado, durante o projeto de linguagens de comunicação, é a semântica [SEA 69]. Essa negligencia, resulta diretamente na obscuridade nos objetivos e nas características desejadas, em uma linguagem de comunicação entre agentes.

A semântica, de uma linguagem de comunicação, deve exibir algumas propriedades esperadas na semântica de qualquer outra linguagem, sendo estas:

- ser baseada em teoria;
- não apresentar ambigüidades;
- ter uma forma canônica (similaridade entre o significado e a representação);
- distinguir tempo e local, sendo que é projetada para interações entre aplicações dispersas espacialmente.

Um outro aspecto importante em uma linguagem de comunicação entre agentes é a implementação. Segundo Mccarthy [MCC 99], a implementação de uma linguagem de comunicação, deve apresentar algumas características especiais:

- eficiência tanto em velocidade como em ocupação do canal de comunicação;
- ter uma interface intuitiva;
- estar de acordo com a tecnologia de *software* existente;
- detalhes em nível de comunicação (camadas e primitivas do sistema de comunicação) devem ficar transparentes ao usuário;
- a linguagem deve ser maleável para uma implementação parcial, já que a implementação de agentes simples necessitam apenas de um subconjunto das primitivas de comunicação.

Em esforços concentrados, uma linguagem mostra-se como um grande atrator de admirações e concentração de esforços de pesquisas e desenvolvimentos, esta é chamada de KQML ou Linguagem de Consulta e Manipulação do Conhecimento (*Knowledge Query and Manipulation Language*).

KQML é uma linguagem para troca de informações e conhecimento. Sendo tanto um formato de mensagem, como um protocolo para manipulação de mensagens que suporta, em tempo de execução, ou compartilhamento de conhecimento entre agentes [FIN 98].

É uma linguagem para os programas utilizarem para comunicar atitudes sobre informações, como questionamento, afirmações, crenças, pedidos, concordância, e oferta.

Esta pode ser utilizada, como uma linguagem, para um programa de aplicação interagir com um sistema inteligente ou com dois ou mais sistemas inteligentes, que compartilham conhecimento na solução cooperativa de um problema.

Enfoca um amplo conjunto de primitivas, as quais definem as operações possíveis de serem realizadas pelos agentes, sobre o conhecimento e metas armazenadas de outros agentes. As atuações compreendem o desenvolvimento de modelos em alto-nível de interação entre agentes, como redes de contrato e negociações. Além disto, o KQML fornece uma arquitetura básica para compartilhamento de conhecimento, através de uma classe especial de agente, chamada facilitador de comunicação, o qual coordena a interação com outros agentes.

Como resumo, pode se dizer que o KQML é uma linguagem de comunicação que foi projetada para facilitar a cooperação e interoperação em alto-nível, entre agentes.

Atualmente a linguagem KQML está sendo utilizada em projetos onde é necessário o estabelecimento de comunicação entre agentes de autônomos, como por exemplo: *Java Agent Template* (JAT), desenvolvido por Rob Frost da Universidade de Stanford [JAT 96].

3 Internet

A interligação de redes independentes que colaboram entre si para a prestação de serviços de informação, abrangendo mais de 150 países em todos os continentes, é denominada Internet.

A Internet é formada por pequenas redes locais (LANs - *Local Area Networks*), redes de abrangência urbana (MANs - *Metropolitan Area Networks*) e grandes redes remotas (WANs - *Wide Area Networks*), as quais interconectam computadores no mundo inteiro. Esses computadores apresentam as mais variadas arquiteturas de *hardware* e sistemas operacionais, proporcionando, através de tal interconexão, um ambiente heterogêneo.

Os computadores na Internet comunicam-se entre si enviando e recebendo pacotes de informações. Esses pacotes contêm porções de dados, informações especiais de controle e endereçamento, necessários para levar os pacotes aos seus destinos e remontá-los em dados úteis. Tudo isso é realizado pelo *Transmission Control Protocol* e *Internet Protocol* (ou Protocolo de Controle de Transmissões e Protocolo da Internet), também conhecidos por TCP/IP [EDD 94].

Ninguém detém o direito de propriedade sobre a tecnologia TCP/IP, bem como nenhum órgão governamental ou comercial é responsável pelos padrões e artigos publicados. Organizações voluntárias encarregam-se do controle de usuários e dos artigos publicados na Internet. Eis algumas organizações:

- IAB – a IAB (*Internet Advisory Board*) é constituída de várias organizações, e seu objetivo principal é coordenar a organização geral da Internet;

- InterNIC – a InterNIC (*Internet Network Information Center*) é a organização que fornece informações sobre serviços e documentação de protocolos. Além disso, ela é responsável pelo registro de endereços IP e nomes de domínios;

- IRTF – o IRTF (*Internet Research Task Force*) é um dos comitês que constituem a IAB. Ele é responsável por várias atividades em nível de pesquisa, como o desenvolvimento de protocolos;

- IETF – o IETF (*Internet Engeneering Task Force*) é um subcomitê da IAB, o qual realiza a manutenção de problemas construtivos e também a implementação de novas tecnologias;

- FNC – a FNC (*Federal Networking Council*) é um comitê que exerce a parte informativa da Internet. A FNC realiza o intermédio entre a IAB e as instituições governamentais, além de prestar suporte a agências no uso da Internet.

Os padrões sobre a arquitetura Internet não são definidos por entidades internacionais de padronização, como a ISO, por exemplo. A maior parte das informações sobre a Internet, incluindo sua arquitetura, protocolos e história, pode ser encontrada em uma série de artigos denominados de RFC (*Request For Comments*). As RFCs são um conjunto informal de artigos elaborados e distribuídos pelo IAB (*Internet Activities Board*). Normalmente, as RFCs são a síntese do pensamento e de propostas de grupos de pesquisadores nas áreas de ponta ou com problemas a resolver, sem maiores elaborações teóricas, mas com a preocupação de disseminar uma discussão sobre um dado tema [COM 98].

Por serem disponíveis eletronicamente, suas informações são disseminadas rapidamente pela Internet.

As RFCs podem ser divididas em cinco grupos:

1. obrigatórias;
2. sugeridas;
3. direcionais;
4. informativas;
5. obsoletas.

As RFCs podem ser obtidas eletronicamente através de muitos outros endereços na Internet.

3.1 Arquitetura Internet

A arquitetura Internet é amplamente utilizada na interconexão e na interoperação de sistemas computacionais heterogêneos.

A organização da arquitetura Internet é feita em camadas, não existindo, contudo, uma estruturação formal como a definida para o Modelo OSI [1] (*Open Systems Interconnection*), que é o modelo de referência para a interconexão de sistemas abertos [CAR 94].

[1] Conjunto de padrões ISO, o qual define a estrutura para a implementação de protocolo em sete camadas.

A figura 3.1 ilustra a arquitetura básica do TCP/IP, que, fazendo relação ao modelo de referência OSI, corresponde às camadas de Rede, Transporte e Aplicação [TAN 97].

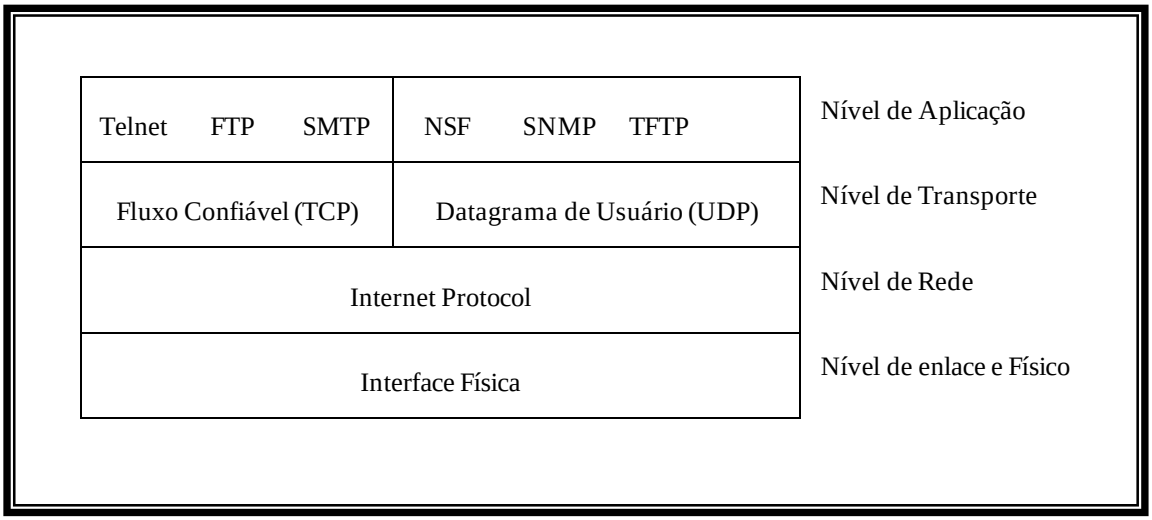

FIGURA 3.1 - Camadas da arquitetura Internet

Nível de Enlace e Físico – na arquitetura Internet, os níveis 2 e 1 (Enlace e Físico) do modelo OSI da ISO são tratados unificadamente, sendo que esta camada também é denominada Interface de Rede. O nível físico trata da transmissão de bits brutos por um canal de comunicação. No nível de enlace, a principal tarefa é transformar esses bits brutos em uma linha que pareça livre de erros de transmissão não detectados na camada de rede.

Nível de Rede – é o responsável pelo provimento da interconexão de diversas redes, com seus diferentes meios de transmissão, topologias e protocolos de acesso ao meio.

Nível de Transporte – este nível tem, como principal função, prover uma forma de comunicação *end-to-end* entre aplicações. Ele é responsável por fornecer às aplicações total transparência quanto aos níveis inferiores.

Nível de Aplicações – este é o nível mais alto desta arquitetura, desempenhando a função de efetuar o interfaceamento com as aplicações que desejam utilizar os meios de comunicação de dados.

Na figura 3.2, é mostrado um exemplo de troca de dados na conexão entre duas aplicações HTTP localizadas em 2 *hosts* distintos.

---

[2] Qualquer sistema de computador de usuário final que se conecta a uma rede.

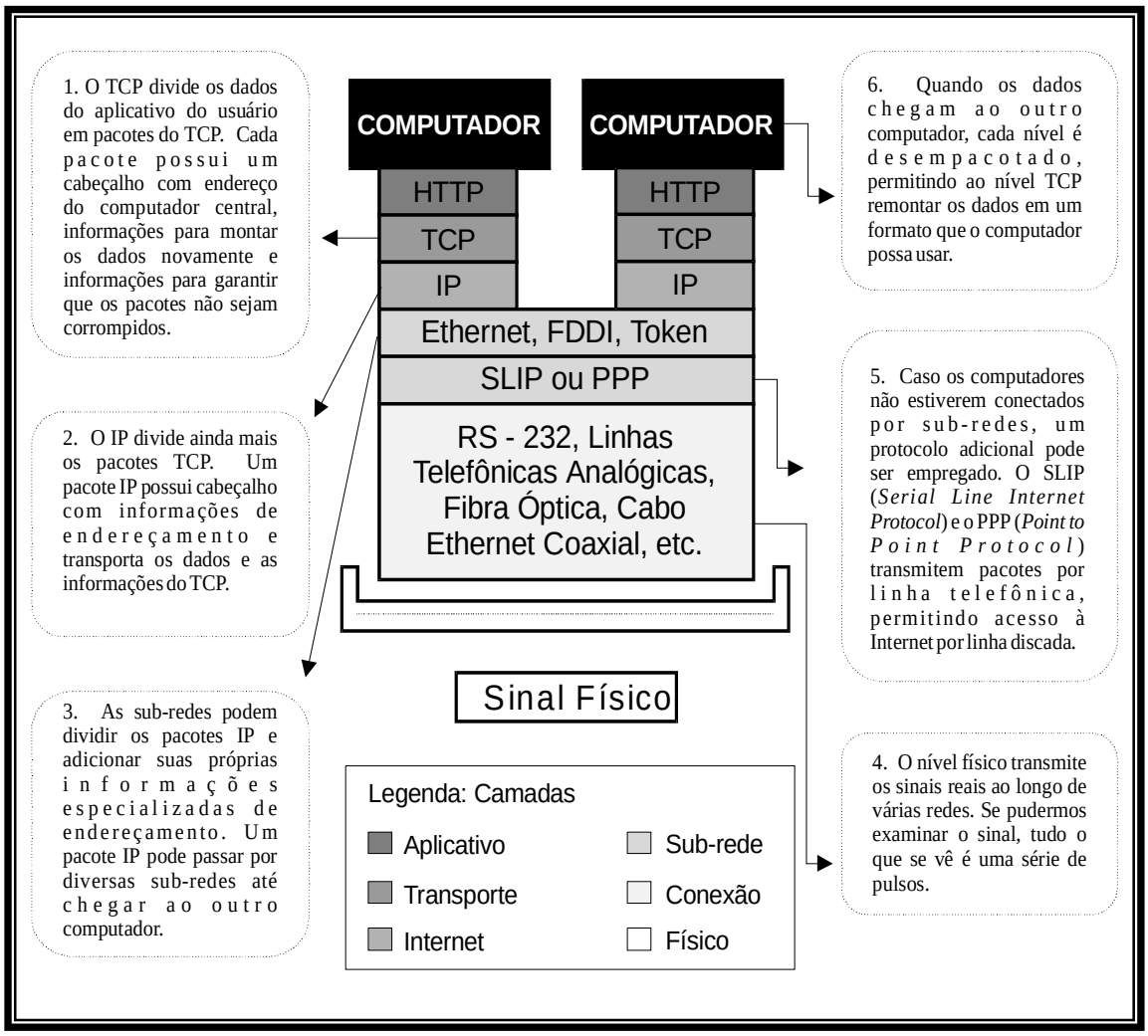

FIGURA 3.2 - Exemplo de troca de dados em uma conexão

3.2 Conceitos básicos de rede

Endereçamento - é o conceito fundamental da interligação em rede. Em um endereçamento em rede, o endereço de um dispositivo é sua identificação exclusiva. Normalmente, os endereços são numéricos e seguem um formato padrão bem definido (cada formato é definido no próprio documento de especificação). É preciso atribuir a todos os dispositivos de uma rede um identificador exclusivo que satisfaça ao formato padrão. Esse identificador é o endereço do dispositivo.

Pacotes - na maior parte das redes, tais como as redes TCP/IP, as informações transmitidas são desmembradas em partes denominadas

pacotes (ou datagramas). Isso se dá principalmente para que haja o compartilhamento de recursos e a detecção e correção de erros. A maioria dos pacotes tem um formato que inclui um cabeçalho e um corpo. Geralmente, o cabeçalho contém informações, como o endereço de origem e o de destino, o comprimento do pacote e algum tipo de indicador que permita ao receptor decodificar o corpo.

Protocolos – um protocolo é um conjunto de regras que definem o formato dos pacotes e a semântica de utilização desses pacotes. Cada pacote é um fluxo de bytes; portanto, para que uma comunicação seja estabelecida, esses bytes devem conter significados associados a eles. Cada significado é fornecido pela especificação do protocolo.

As conexões físicas da Internet estão fundamentadas em alguns componentes básicos, dos quais podemos destacar:

Roteador – consiste no equipamento responsável pelo processo de escolha entre vários caminhos possíveis de se enviar uma mensagem (roteamento). Esse componente pode ser caracterizado por um simples circuito de *hardware* ou por um computador dedicado com um *software* de roteamento. Ele possui diversas portas conectadas a diferentes redes e sabe qual porta pertence a cada rede. Quando ele recebe um pacote de informações, verifica o endereço de destino e procura a porta equivalente, baseando-se, para isso, no endereçamento em nível de rede, que, no caso da arquitetura TCP/IP, é o endereçamento IP. Também é de sua responsabilidade a tentativa de redução do percurso por onde o pacote trafegará através do envio do pacote para redes que sejam mais próximas da rede destino.

Gateway – trata-se de um componente que pode conectar redes distintas - convertendo diferentes níveis de protocolo - e, às vezes, duas topologias diferentes.

Ponte (*Bridge*) - este componente é muito utilizado para separar o tráfego em uma rede movimentada. Uma ponte acompanha os endereços de *hardware* dos dispositivos para cada rede a que está diretamente conectada. Dessa forma, só repassa pacotes de uma rede A para uma rede B se eles forem endereçados à rede B. Portanto, segmenta o tráfego de redes, criando um só domínio de colisão quando for necessário [TAN 97].

3.3 Tipos de serviços e protocolos

Na camada de rede, são oferecidos dois tipos básicos de serviço [TAN 97]:

- não orientados à conexão - o modo de transmissão não orientado à conexão conserva poucas informações durante uma conexão. A origem e o destino necessitam ter um prévio acerto sobre como será a comunicação e as características da qualidade de serviço - QOS (*Quality of Service*), as quais já

devem estar definidas. Esse modo de transmissão associa a cada pacote um endereço global, que identifica a origem e o destino do pacote. Nesse modo de transmissão, os sucessivos pacotes transferidos podem não ter qualquer relação, pois são considerados independentes uns dos outros. O modo de transmissão não orientado à conexão não dá importância a controles de fluxo nem realiza qualquer reconhecimento ou reenvio de pacotes. Através desse modo pode-se fazer uma comunicação com qualquer máquina, sem haver necessidade de realizar-se uma conexão, sendo assim mais rápido. Em compensação, não se tem à certeza do sucesso da transmissão, e o acompanhamento do processo deve ser mais eficaz.

- orientados à conexão – este serviço é baseado no sistema telefônico. São necessários que se faça uma chamada e uma conexão antes de cada transmissão. Nesse tipo de transmissão, os pacotes não precisam possuir *overheads*, como ocorre nas transmissões não orientadas à conexão. Isso é possível, pois, logo no início da conexão, a origem e o destino trocam todas as informações necessárias à transmissão. O modo de transmissão orientado à conexão é mais seguro porque possui mecanismos de reenvio de pacotes mal transmitidos, bem como mensagens de reconhecimento.

3.4 TCP/IP

TCP/IP é o nome dado a toda a família de protocolos utilizados na Internet. Este conjunto de protocolos foi desenvolvido a fim de permitir aos computadores compartilharem recursos numa rede. Toda a família de protocolos inclui um conjunto de padrões que especificam os detalhes de comunicação, assim como convenções para interconectar redes e rotear o tráfego [ARN 96].

De forma oficial, esta família de protocolos é chamada Protocolo Internet TCP/IP, comumente referenciada como TCP/IP, devido a seus dois protocolos mais importantes (TCP: *Transport Control Protocol* e IP: *Internet Protocol*).

O protocolo IP é responsável pelo encaminhamento de pacotes de dados entre diversas sub-redes desde a origem até o seu destino, e o TCP tem, por função, o transporte (*host-to-host*) confiável de mensagens de dados entre dois sistemas [CAR 94].

3.4.1 Protocolo IP

O protocolo IP define mecanismos de transporte de blocos de dados sem conexão - denominados datagramas - da sub-rede de origem à sub-rede de destino.

O IP define três pontos importantes [STE 94]:

1. a unidade protocolar de dados (PDU) a ser transferida na Internet;

2. o *software* de IP que executa a função de roteamento, escolhendo um caminho sobre o qual os dados serão enviados;

3. a inclusão de um conjunto de regras que envolvem a expedição de pacotes não confiáveis. Essas regras indicam como os *hosts* ou *gateways* podem processar os pacotes; como e quando as mensagens de erros podem ser geradas; e as condições em que os pacotes podem ser descartados.

3.4.1.1 Endereçamento IP

Um sistema de comunicação é dito universal quando ele permite que um nó se comunique com qualquer outro conectado à rede. Para tanto, é indispensável que se estabeleça uma metodologia consensual de identificação dos computadores [COM 98].

A cada um dos computadores que compõe uma rede TCP/IP é associado um endereço lógico chamado IP. Esse endereço tem 32 bits e normalmente é escrito em decimal como quatro bytes separados por um ponto (por exemplo: 143.54.8.170). No esquema de endereçamento Internet, os inteiros são cuidadosamente escolhidos para um roteamento eficiente. Especificamente, um endereço IP define o identificador da rede ao qual o *host* está conectado e também a identificação deste *host*, sendo única na rede.

Conceitualmente, cada endereço é um par (*netid, hostid*), em que *netid* identifica a rede, e *hostid* identifica um *host* naquela rede [COM 98]. Na prática, cada endereço IP deve ter um formato específico. Na figura 3.3 a seguir, é apresentado o formato dos endereços IP.

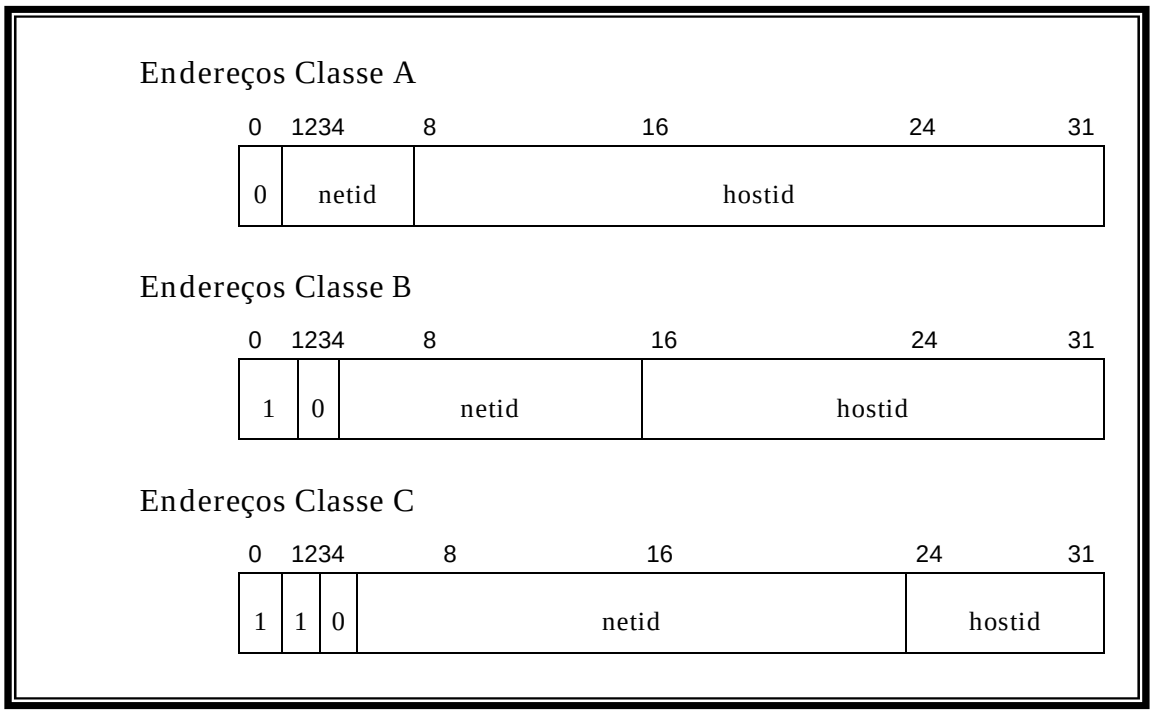

FIGURA 3.3 - Formato dos endereços IP

Os endereços da classe A, identificados pelo primeiro bit (0), permitem o endereçamento de 224 *hosts*, reservando 7 bits para *netid* e 24 bits para o *hostid*. É utilizada para redes com um número elevado de *hosts*. Os endereços da classe B são empregados em redes de tamanho médio, sendo identificados pelos dois primeiros bits (10), permitem o endereçamento de 216 *hosts*, reservando 14 bits para o *netid* e 16 bits para o *hostid*. Finalmente, redes da classe C, que possuem menos de 254 *hosts*, alocam 21 bits para o *netid* e, somente, 8 bits para o *hostid*.

O endereço IP foi definido de tal forma que seja possível extrair as partes do *netid* ou do *hostid* rapidamente. Os roteadores, que usam a parte *netid* de um endereço ao decidir qual o destino de um pacote, dependem de uma extração eficiente a fim de obter um bom desempenho.

3.4.1.2 Protocolo TCP

O protocolo TCP embora seja apresentado como parte de uma pilha de protocolos TCP/IP da Internet ele é um protocolo independente e de finalidade geral, podendo ser adaptado para utilização com outros sistemas de transmissão [COM 98].

O TCP foi idealizado com a finalidade de proporcionar uma forma segura de transferência de dados, implementando mecanismos de recuperação de dados perdidos, danificados ou recebidos fora de seqüência e minimizando o atraso de trânsito para transmissão de dados [CAR 94].

Trata-se de um protocolo orientado à conexão, o que indica que, neste nível, vão ser solucionados todos os problemas de erros que não forem solucionados no nível IP, já que este último é um protocolo sem conexão.

Baseando-se no modelo de referência OSI, o protocolo TCP é um protocolo que reside na camada de transporte, conforme mostra a figura 3.4.

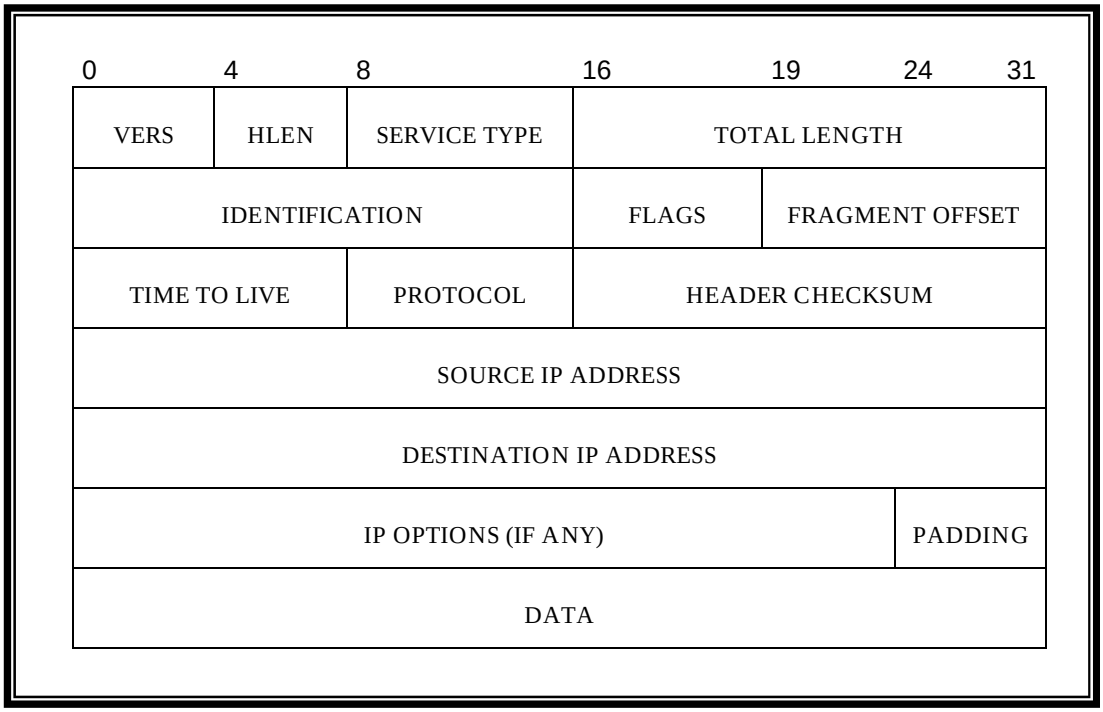

FIGURA 3.4 - Formato das mensagens TCP

O TCP permite que múltiplos programas de aplicação numa determinada máquina se comuniquem concorrentemente, encarregando-se em demultiplexar o tráfego TCP entrante entre os programas de aplicação.

Números de portas são utilizados a fim de identificar o destino final numa máquina. A cada porta é associado um número inteiro com três dígitos, para identificá-la.

O protocolo TCP foi estruturado sobre a abstração de conexão, na qual os objetos a serem identificados são conexões de circuitos virtuais e não portas individuais. As conexões são identificadas por um par de pontos terminais (*endpoints*). Uma conexão consiste em um circuito virtual entre dois programas de aplicações, em que o TCP define um *endpoint* como um par de inteiros *(host,port*), no qual host é o endereço IP para um computador, e *port* é uma porta TCP nesse computador.

Exemplo: o *endpoint* 143.54.8.170,25 especifica a porta TCP número 25 na máquina como o endereço IP 143.54.8.170.

Uma conexão está definida por dois *endpoints*. Assim, para que haja uma conexão entre as máquinas 129.88.30.10 (*no Institut D'Informatique Et de Mathematiques Appliquees de Grenoble*, França) e 143.54.8.172 (na Universidade Federal do Rio Grande do Sul, Brasil), a conexão deve ser definida pelos seguintes *endpoints*:

(129.88.30.10,1069) e (143.54.8.172,25).

Uma vez que o TCP identifica uma conexão por um par de *endpoints*, um número de porta pode ser compartilhado por múltiplas conexões na mesma máquina.

O TCP enxerga o fluxo (*stream*) de dados como uma seqüência de bytes, a qual ele divide em segmentos para a transmissão. Usualmente, cada segmento trafega, através da Internet, com um único datagrama IP.

É utilizado, no TCP, um mecanismo de janela deslizante para resolver dois problemas relevantes [GAS 93]:

- transmissão eficiente;
- controle de fluxo.

O mecanismo de janelas do TCP torna possível enviar múltiplos segmentos antes da chegada de um reconhecimento. O fato de TCP ser um protocolo janela deslizante também resolve o problema de controle de fluxo *end-to-end*, permitindo que o receptor restrinja a transmissão até haver suficiente espaço de buffer para acomodar mais dados. Este mecanismo de janelas opera ao nível de bytes e não de segmentos ou pacotes, ou seja, o fluxo de dados é numerado seqüencialmente, e um remetente (transmissor) mantém três ponteiros associados com cada conexão.

É importante entender que, pelo fato de as conexões em TCP serem *full-duplex*, duas transferências ocorrem simultaneamente numa conexão: uma em cada direção. As duas transferências são completamente independentes, já que em qualquer momento, os dados podem fluir através da conexão, numa direção ou em ambas. Assim, o *software* TCP, em cada extremo da conexão, mantém duas janelas por conexão (para um total de quatro): uma janela para os dados que estão sendo enviados, e outra janela para os dados recebidos. O tamanho da janela pode ser definido, e, no caso de TCP, este permite que o tamanho da janela varie no tempo.

3.5 DNS (*Domain Name System*)

O mecanismo que implementa em um domínio universal, como a Internet, uma forma organizada, confiável, eficiente e distribuída, para mapear os nomes em endereços, baseado em uma estrutura de árvore, é denominado DNS [CAR 94].

O DNS é composto por um conjunto de servidores cooperativos ditos servidores de nomes ou *nameservers*. Um servidor de nomes é um programa que fornece a tradução de domínios para endereços IP, podendo ser executados em máquinas dedicadas ou não, o que depende do seu porte. O *software* cliente, chamado resolvedor ou *resolver*; pode usar um ou mais servidores quando da tradução de um nome.

O DNS estabelece a sintaxe de nomes e regras para delegação de autoridade sobre os nomes, além de implementar um algoritmo computacional eficiente para mapear nomes em endereços.

Conceitualmente, todos os servidores de nomes dos domínios Internet estão estruturados em árvore, de acordo com a estrutura hierárquica de nomes [ARN 96]. O topo da árvore é um servidor, chamado de raiz (*root*), que corresponde ao nível mais alto na hierarquia, cujo domínio de autoridade é o próprio NIC (*Network Information Center*). A partir daí, no próximo nível da árvore, existe um conjunto de servidores de nome que reconhecem cada um dos domínios de topo da Internet:

- ARPA - (ARPANET) Advanced Research Projests Agency network;
- COM - Organizações comerciais;
- EDU - Instituições educacionais;
- GOV - Órgãos governamentais;
- MIL - Grupos militares;
- NET - Centros de suporte a redes, como os fornecedores de serviços da Internet;
- INT - Organizações internacionais;
- ORG - Organizações não lucrativas;
- Código do país - Códigos do país com duas letras (conforme definido no X.500 na ISO-3166);

Os nomes de domínio de segundo nível são únicos e devem ser registrados no NIC antes que possam ser conectados à Internet e conhecidos pelos servidores de mapeamento de nomes de endereços IP. Sem o registro do nome, os outros usuários da Internet não são capazes de se comunicar com esse nome; eles precisarão utilizar o endereço IP. Os próximos níveis da árvore contem os servidores que conhecem cada subdivisão do segundo nível, e assim, sucessivamente, até o fim da estrutura hierárquica de nomes.

A figura 3.5 ilustra a estrutura de servidores de nomes, onde cada servidor conhece os nomes constantes de todos os servidores na hierarquia abaixo dele.

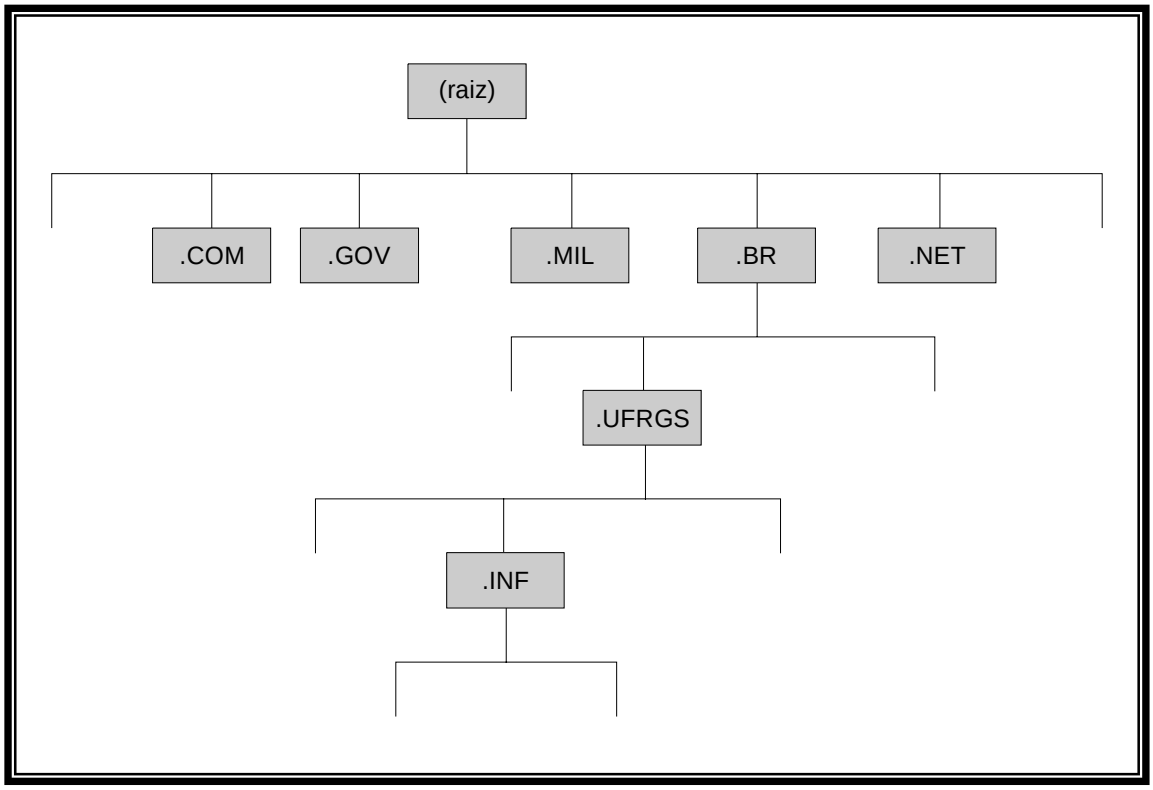

FIGURA 3.5 - Exemplo de árvore de hierarquia de nome de domínio da Internet

O mecanismo para resolução de nomes percorre a árvore de cima para baixo até chegar às folhas, de duas maneiras:

1. contactando servidores de nomes um de cada vez;

2. solicitando ao sistema do servidor que efetue o trabalho completo de resolução de nome.

Quando um servidor de nomes recebe um pedido de tradução, ele verifica se o nome pertence ao subdomínio que controla e resolve a tradução do nome para o endereço IP segundo uma tabela própria, devolvendo esta resposta ao cliente; se não for o caso, e a solicitação é para a resolução completa, o servidor contacta o domínio que pode resolver, retornando, depois, a resposta ao solicitante; se o nome não está no seu domínio e a solicitação não é para resolução completa, ele informa qual domínio pode resolver aquela tradução solicitada. Isso é possível porque cada servidor de domínio conhece os endereços dos servidores abaixo dele, na árvore.

Para ver como funciona exatamente o DNS, será utilizado, como exemplo, o endereço Internet do Instituto de Informática da UFRGS: inf.ufrgs.br. A referência principal responsável por este nome é br, ou Brasil (indicando que esta rede está localizada, fisicamente, no Brasil). A referência seguinte, ufrgs, indica qual instituição é responsável por esta rede. A referência inf indica uma das redes mantidas pela UFRGS.

3.6 *Sockets*

Um *socket* representa uma comunicação TCP entre duas aplicações executadas em uma rede TCP/IP. Uma conexão *socket* é especificada através do endereço IP e de um número de identificação, chamado de porta [TAN 92].

A maioria dos sistemas operacionais trabalha com multiprogramação. Sendo assim, a comunicação não é mais feita entre computadores, mas sim entre aplicativos. Dessa maneira, algo que identifique a aplicação destinatária deve vir junto com o pacote. Isso é feito por meio de uma abstração denominado porta lógica. Um determinado computador pode estar com uma ou várias portas lógicas abertas ao mesmo tempo.

Cada porta deve ser referenciada através de um número com 16 bits, podendo as conexões chegar até 65000. As portas abaixo de 1024 são reservadas para aplicações padrões, como é o caso da porta lógica de número 80, que atende a todas as aplicações HTTP; o de número 25 que atende às aplicações de e-mail; e a porta de número 13, que atende às aplicações de Telnet.

Assim, para uma aplicação mandar um pacote para outra, tem de saber não só o endereço IP do outro computador, mas também o número da porta lógica a que esta aplicação está conectada.

A comunicação entre duas máquinas é, geralmente realizada através de um *socket* "*stream*" ou de datagrama. A comunicação por *stream* implica uma conexão confiável, livre de erros, sem nenhum limite de mensagem. O protocolo que implementa tal estilo retransmite mensagens recebidas com erros. Mensagens de erros também são retornadas caso alguém tente enviar uma mensagem depois que a conexão for rompida.

Na comunicação por datagrama, não existem conexões. Cada mensagem é endereçada individualmente; se o endereço está correto, possivelmente será recebida, embora não seja garantido. Geralmente datagramas são usados para pedidos que requeiram uma resposta do destinatário, e, se nenhuma resposta em um intervalo de tempo for recebida, o pedido é repetido. Os datagramas individuais são mantidos em separado quando lidos, isto é, limites de mensagens são preservados.

4 Um ambiente para a interação de agentes na Internet

Um sistema multiagente baseia-se na interação de agentes, a fim de alcançar objetivos individuais ou coletivos.

Abaixo são apresentados alguns quesitos que um ambiente de interação entre agentes deve apresentar:

- a localização de agentes;
- estabelecimento de contato entre os agentes;
- a identificação dos agentes (nomes de agentes, função e endereço desses em uma rede) sem que ocorra a coincidência na identificação;
- a entrada e saída de agentes na rede;
- a troca de mensagens entre os agentes;
- a procura de agentes com determinadas características.

Em Cazella [CAZ 97], apresenta-se uma arquitetura que possibilita a implementação de agentes inteligentes no âmbito da Internet, permitindo e facilitando a interação entre eles.

Este trabalho teve, como primeiro objetivo, o refinamento da arquitetura de coordenação de agentes na Internet, proposta por Cazella, que é sua referência inicial. O refinamento desta arquitetura foi obtido pelo estudo de alguns ambientes de desenvolvimento de agentes, de alguns tópicos estruturais da Internet e de protocolos em nível de aplicação mais utilizados (FTP, SMTP e HTTP).

4.1 Arquitetura

A visão geral da arquitetura de coordenação de agentes para a Internet, proposta por Cazella [CAZ 97], é apresentada na figura 4.1. Nesta figura, podem-se identificar dois domínios de rede (com suas máquinas locais designadas por c1..cn), conectadas às respectivas máquinas "*agent*", comunicando-se através da Internet. Cada máquina pode ter um ou mais

agentes, independente da função ou origem desses (representados pelos hexágonos 1..n).

Esta arquitetura torna a comunicação entre agentes transparente para o usuário. Um agente deve conhecer apenas a identificação do agente com o qual quer interagir, não importando a sua localização. Os agentes podem estar localizados na mesma máquina, no mesmo domínio ou em domínios distintos.

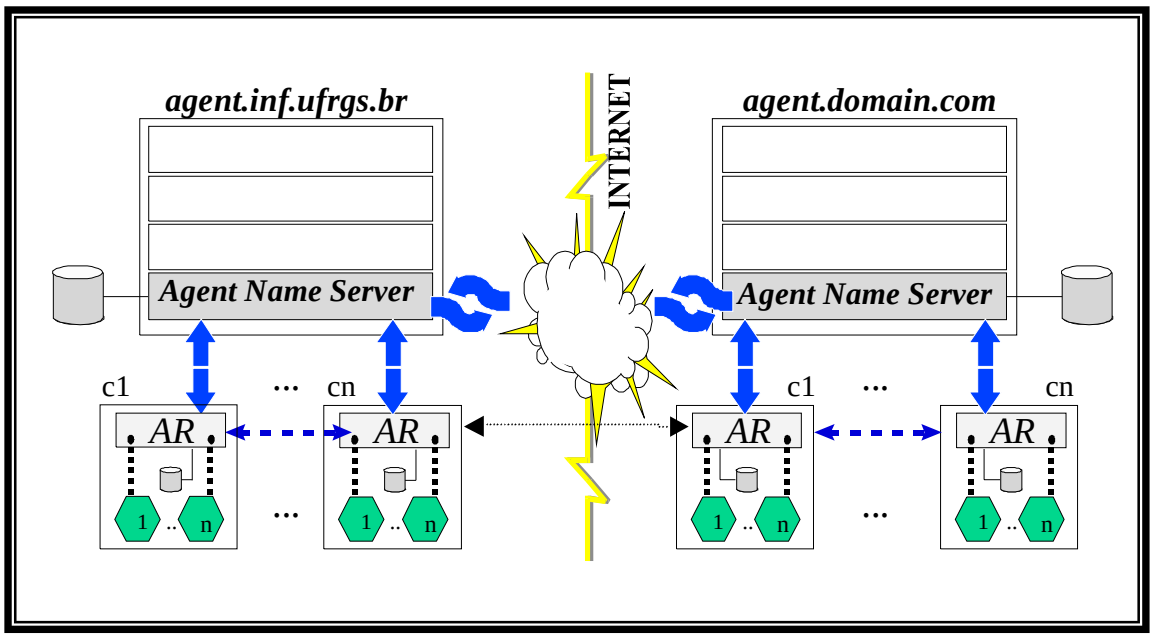

FIGURA 4.1 - Visão geral da arquitetura de interação de agentes para a Internet

As principais características desta arquitetura são:

- a existência de um servidor central – existe um servidor de nomes de agentes (ANS - *Agent Name Server*) por domínio, a qual desempenha a função de armazenar (em um arquivo local) e fornecer (quando solicitado) os endereços dos agentes localizados no domínio e suas descrições;

- roteadores de agentes - os processos de comunicação entre os agentes são feitos por intermédio dos roteadores de agentes (AR - *Agent Router*) que desempenham os papéis de intermediadores na comunicação entre os agentes.

Cada um dos serviços em uma rede (e-mail, FTP, Telnet, HTTP, etc..) faz uso de uma porta lógica específica. O ANS e os ARs também utilizam portas lógicas próprias (dedicadas) para que os agentes façam conexões com eles.

A arquitetura mostrada acima é independente do tipo de sociedade em questão, tendo um funcionamento que supre as necessidades de integração entre os agentes, independente do fato de a sociedade ser aberta ou fechada. O suprimento de tais necessidades é feito através do ANS, que é responsável por manter uma lista contendo as identificações de todos os agentes cadastrados no domínio referido.

Para haver uma maior agilidade no serviço prestado pelo ANS, ele é instalado numa máquina que possui, como endereço, o nome "*agent*", podendo este ser virtual. Como ilustração, tem-se a máquina "agent.inf.ufrgs.br". Para que ocorra a resolução de nomes de agentes, basta uma requisição a esse servidor.

Cada máquina, em uma rede, pode conter um ou mais agentes específicos, independente de função e origem.

Nas máquinas de uma rede, onde os agentes estão localizados, além desses, ter-se-á um roteador denominado AR. O AR é um processo que está ininterruptamente em execução, aguardando por conexões, a fim de executar, principalmente, três funções de fundamental importância para a comunicação:

- distinguir, através de mecanismos internos, de qual e para qual agente na máquina onde está em execução, é destinada a informação que está chegando;

- intermediar a comunicação dos agentes. Em outras palavras, desempenhar o papel de meio de comunicação entre os agentes;

- gerenciar um arquivo de *cache*, onde são armazenadas as últimas requisições de identificação feitas pelos agentes ao ANS, por intermédio do AR.

A utilização de um arquivo de *cache*, local a cada AR, proporciona um melhor desempenho e uma menor centralização em relação ao ANS.

A centralização da comunicação dos agentes nos ARs não tira a flexibilidade da arquitetura, sendo esta necessária devido aos seguintes aspectos:

- integridade das máquinas – algumas verificações nas informações são feitas nos ARs para averiguação da segurança;

- consumo de recursos - seria inviável uma arquitetura em que todos os agentes ficassem em execução, no aguardo do momento de agirem, comprometendo assim o desempenho. Apenas os ARs ficam em execução e, no momento em que for necessária a ativação de um determinado agente, este será ativado pelo AR, a fim de que possa agir.

4.1.1 Identificação de agentes

A identificação do agente, figura 4.2, é formada a partir do e-mail do proprietário naquele domínio ao qual é cadastrado e do nome do agente.

FIGURA 4.2 - Identificação de agentes

Para exemplificar a identificação de um agente, é utilizado um agente de nome agenda, pertencente ao domínio “inf.ufrgs.br”. Sendo a identificação de um agente na arquitetura o e-mail do seu proprietário, concatenado com seu nome, tem-se como exemplo: fontes@inf.ufrgs.br/agenda.

Essa identificação de agentes proporciona à arquitetura:

- facilidade na identificação de agentes - se houver mais de um agente de funções semelhantes em atividade na mesma máquina (por exemplo, várias agendas do mesmo usuário ou de diferentes usuários), será possível identificar para qual desses agentes a informação é destinada;

- facilidade de roteamento - esta identificação do agente propicia uma facilidade para a determinação do domínio de rede em que se encontra o agente;

- identificação do proprietário do agente - devido à utilização do e-mail na composição da identificação do agente fica fácil à determinação do proprietário deste. Em caso de um determinado agente1 tentar estabelecer conexão com um outro agente2 e, após um certo número de tentativas, essa conexão não for possível, define-se que o agente1 poderá enviar um e-mail informando ao responsável (proprietário), pelo agente2, que algum tipo de irregularidade está ocorrendo com o agente2.

Na arquitetura, existe um controle de cadastramento de agentes, mantido pelo ANS, para que não seja possível o cadastramento de agentes com identificações já existentes, fazendo com que sejam únicas no domínio Internet.

4.1.2 Processo de comunicação entre agentes

O processo de comunicação entre os agentes pode ocorrer de três formas: comunicação entre agentes localizados na mesma máquina, em máquinas diferentes do mesmo domínio e em domínios distintos.

4.1.2.1 Na mesma máquina

Para a comunicação entre agentes localizados na mesma máquina é imprescindível a participação do AR local, isto é, o agente interessado em enviar informações para algum outro agente que se localiza na mesma máquina, deverá solicitar os serviços do AR.

A figura 4.3 apresenta uma visão dos agentes em uma máquina interligada à rede.

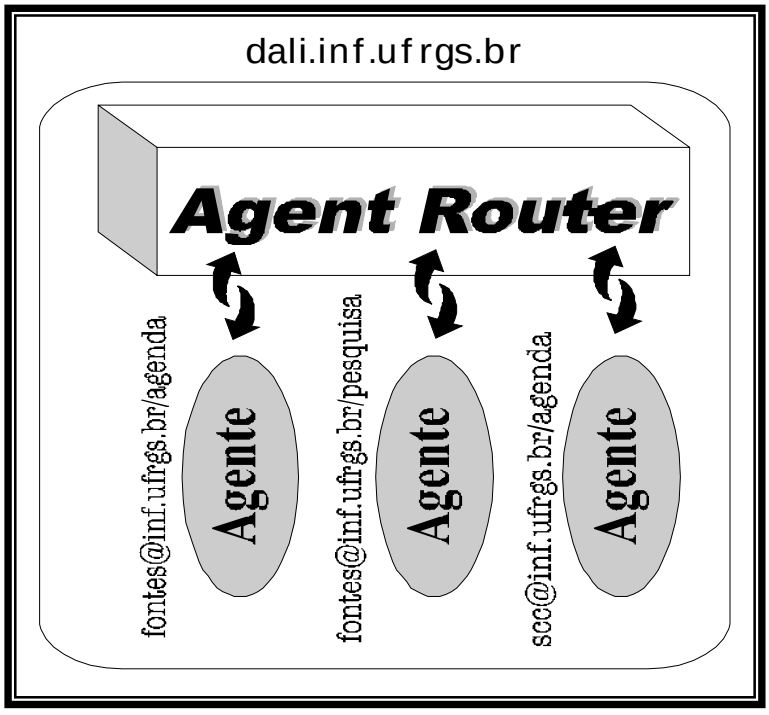

FIGURA 4.3 - Agentes em uma máquina

A fim de iniciar o processo de comunicação, o agente solicita os serviços do AR, através do envio da identificação do agente ao qual deseja comunicar-se; o AR, de posse dessa identificação, primeiramente faz a interpretação e a avaliação, em seguida realiza uma pesquisa ao seu arquivo de agentes locais, para verificar se o agente está localizado na sua máquina. Se a localização do agente destinatário for à mesma, a do remetente (mesma

máquina), o AR faz a ativação do agente destinatário e, então, a troca de informações entre os agentes pode se suceder, tendo o AR como intermediador.

4.1.2.2 Em máquinas diferentes no mesmo domínio

Entre agentes pertencentes a máquinas diferentes no mesmo domínio (figura 4.4), o processo é idêntico ao anterior (4.1.2.1 Na mesma máquina), até a consulta ao arquivo de agentes. Na consulta ao arquivo de agentes, o agente, como não faz parte da máquina, não é encontrado; então, o AR submete uma pesquisa a seu arquivo de *cache*. Supondo que a consulta ao arquivo de *cache* resulte nula, uma requisição é feita ao ANS do domínio.

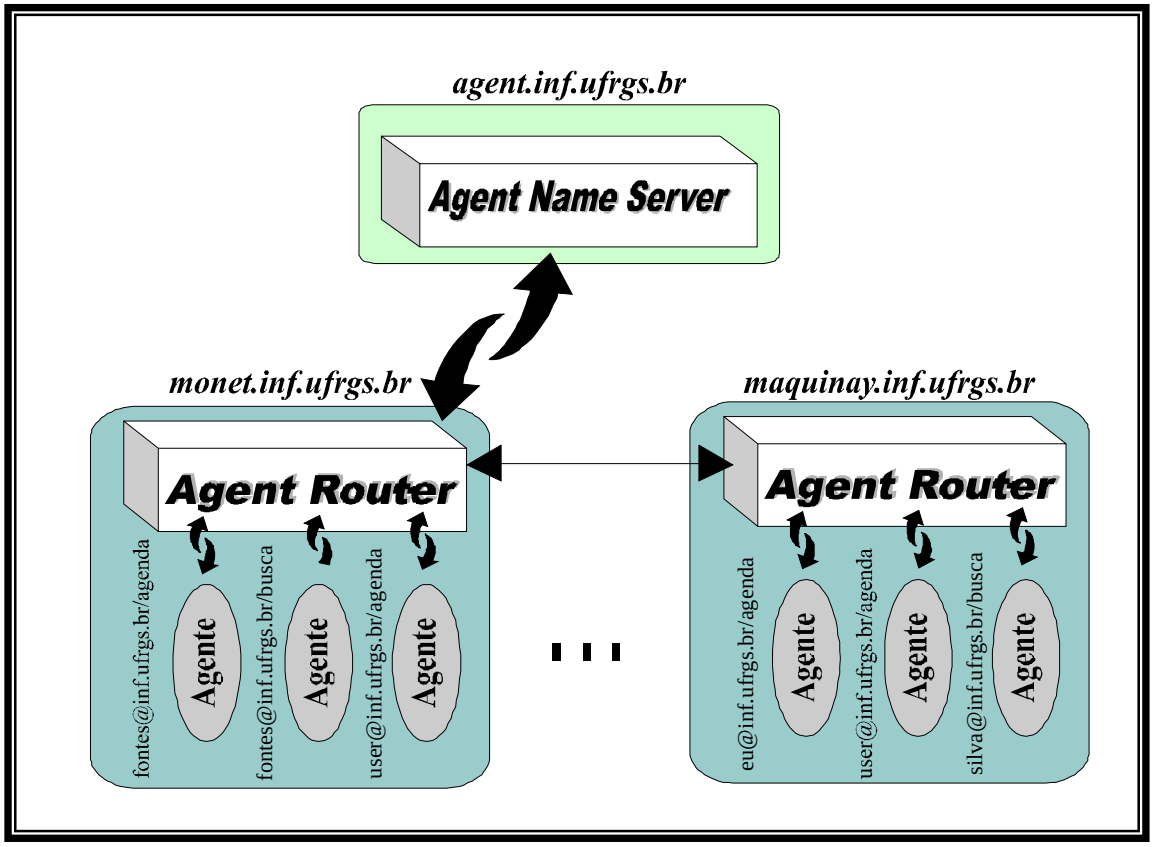

FIGURA 4.4 - Agentes em máquina diferentes no mesmo domínio

O ANS procede à resolução de nomes, retornando o endereço da máquina onde está localizado o agente de destino da mensagem, ou uma mensagem indicando que tal agente não existe no domínio. Sendo positiva a resolução de nomes, o AR local reporta-se ao AR do endereço retornado, e solicita os serviços deste para a comunicação com o agente requisitado. O procedimento é, então, exatamente igual ao na comunicação entre agentes

localizados na mesma máquina, como descrito anteriormente (4.1.2.1 Na mesma máquina).

Após o término da comunicação o AR local insere, em seu arquivo de *cache*, os dados do agente não cadastrado, agilizando, assim, futuras comunicações com esse, por parte dos agentes cadastrados na máquina.

4.1.2.3 Em domínios diferentes

Na comunicação entre agentes localizados em domínios distintos (figura 4.5), todo o processo ocorre, basicamente, de forma idêntica à descrita anteriormente (4.1.2.2 Em máquinas diferentes no mesmo domínio), tendo como diferença a resolução de nomes.

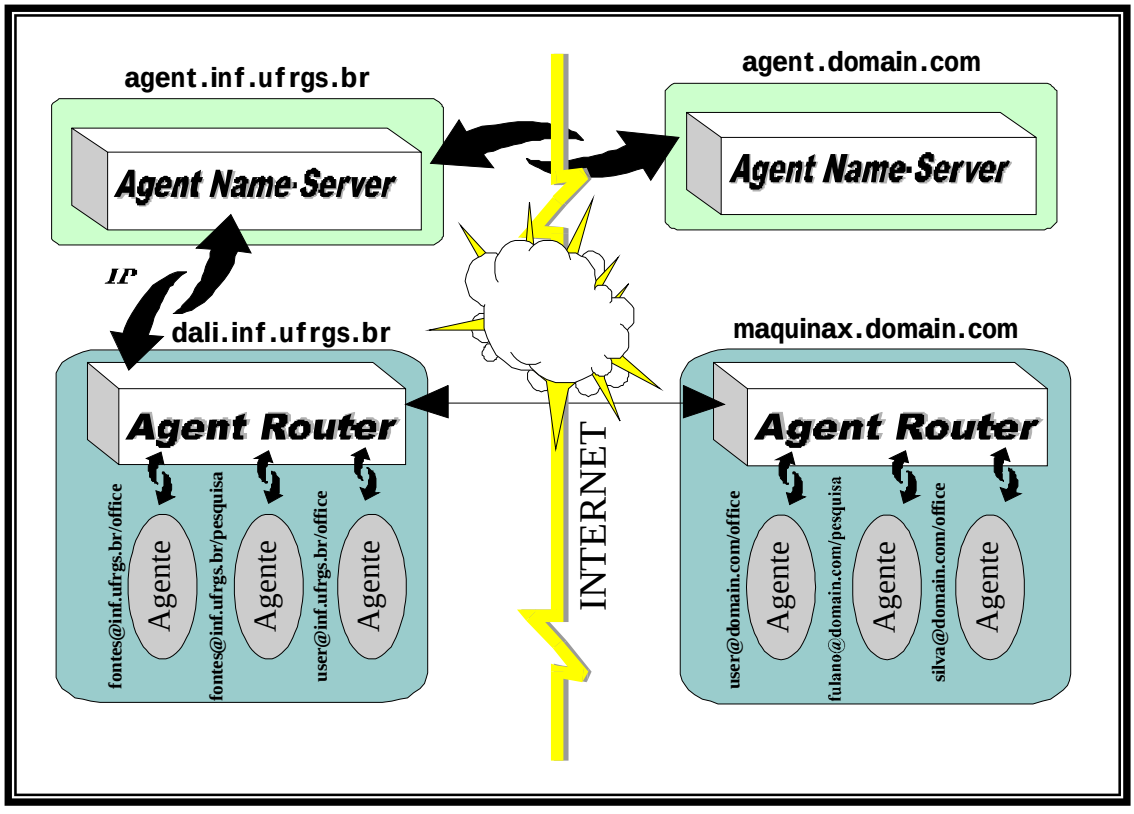

FIGURA 4.5 - Agentes em domínios diferentes

Na resolução de nomes, realizada pelo ANS, para uma requisição de comunicação no mesmo domínio, é pesquisada a identificação do agente no arquivo global de agentes, enquanto que, para domínios diferentes, é feita, em primeiro lugar, uma consulta ao DNS, em busca da máquina "*agent*" do domínio do qual o agente destinatário faz parte, e, em segundo, uma pesquisa junto ao ANS do domínio retornado pela consulta.

4.1.3 Exemplo de interação entre agentes

Esta seção mostra um exemplo de interação entre agentes de domínios distintos, utilizando-se, como ilustração, o esquema apresentado na figura 4.6.

1) Numa primeira etapa do exemplo, supõe-se que o AGENTE2 e o AGENTE3 desempenham a mesma função de agentes agenda. O dono do AGENTE2 solicita a marcação de uma reunião com o proprietário do AGENTE3.

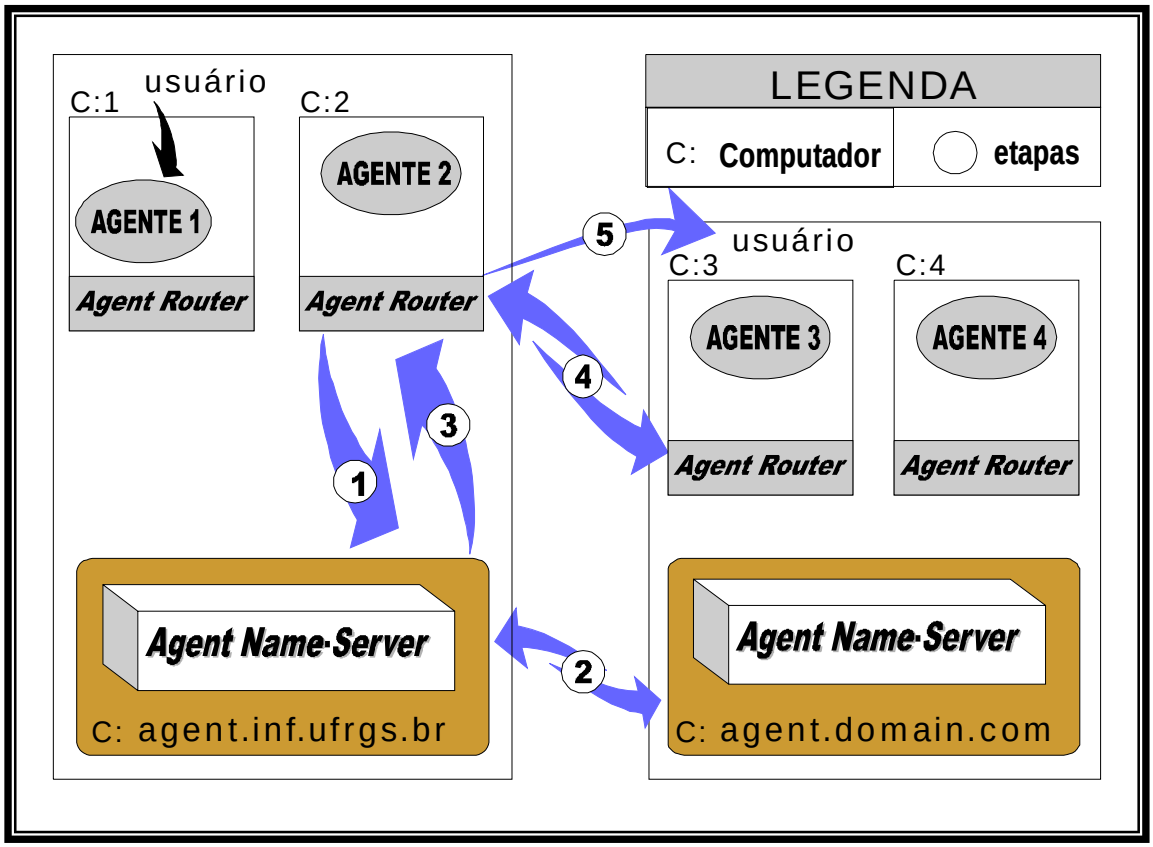

FIGURA 4.6 - Exemplo de uma interação na Arquitetura

Para poder efetivar a marcação da reunião com o agente desejado, são dados vários passos:

I - O AGENTE2 inicia o processo de requisição através da procura do endereço do agente destinatário no arquivo de agentes local (função realizada na verdade pelo AR). Não será encontrado, pois, neste exemplo, o agente está localizado em um outro domínio de rede;

II - Uma pesquisa é então realizada ao arquivo de *cache* mantido pelo AR. Supondo-se, esta ser a primeira vez que o agente destinatário é

requisitado por algum agente cadastrado na máquina (C:2), ele não será encontrado na execução da pesquisa;

III - Após tentativas frustradas de localizar o endereço do agente desejado, tanto no arquivo local de agentes quanto no arquivo de *cache*, o agente entra em contato com o ANS da sua rede (domínio) e solicita o endereço do agente desejado (item ① do esquema).

Neste exemplo, é apresenta a comunicação entre os agentes como se fosse direta, uma vez que fica transparente a participação dos ARs nessa atividade.

Como o agente desejado é gerenciado pelo ANS de outro domínio está contido em outro domínio gerenciado por outro ANS

Como o agente desejado não faz parte da rede onde está o ANS responsável pelo AGENTE2, este terá de estabelecer uma comunicação com a máquina que possui o domínio referenciado pelo nome do agente, de entrar em contado com o ANS desta outra rede (item ② do esquema), e, finalmente, solicitar o endereço requerido. De posse desse endereço, o AR repassa-o para o AGENTE2 (item ③), e este agora pode entrar em contato com o AGENTE3.

Deve se observar que o acesso ao outro ANS ocorreu pelo fato de o agente em questão não estar na mesma rede do AGENTE2.

2) Na segunda etapa (item ④), o AGENTE2 estabelece uma comunicação com o AGENTE3 (no esquema, a conexão é apresentada de maneira direta ag2 <-> ag3, os detalhes são abordados no capítulo 6), sendo feita, a partir deste momento, toda a interação exigida por um agente agenda.

Após o estabelecimento da conexão, depois de toda a negociação ter terminado, a conexão é desfeita, e o endereço do AGENTE3 é armazenado no arquivo de *cache* do AR, residente na máquina do AGENTE2.

Supondo que, mesmo de posse do endereço completo do AGENTE3, o AGENTE2 não tenha conseguido estabelecer comunicação, após de algumas tentativas frustradas, o AGENTE2 envia um *e-mail* informando o ocorrido para o usuário responsável pelo agente requerido (item ⑤).

5 Especificação detalhada do modelo

Neste capítulo, são apresentadas as especificações detalhadas da arquitetura geral descrita no capítulo 4.

A especificação terá fundamentação nos três componentes principais explicitados anteriormente:

- Agente – é a função específica do agente acrescida de um módulo responsável pelas funções de comunicação do protocolo;
- AR – sua principal função consiste em intermediar a comunicação entre os agentes;
- ANS – é um servidor responsável por localizar os agentes na Internet.

As seções seguintes expõem, os principais aspectos do modelo.

5.1 Agente

O Agente é formado por um módulo de aplicação o qual executa as funções da aplicação específica, e por um módulo responsável por toda à parte da comunicação, denominado cliente de comunicação. A figura 5.1 ilustra esses módulos.

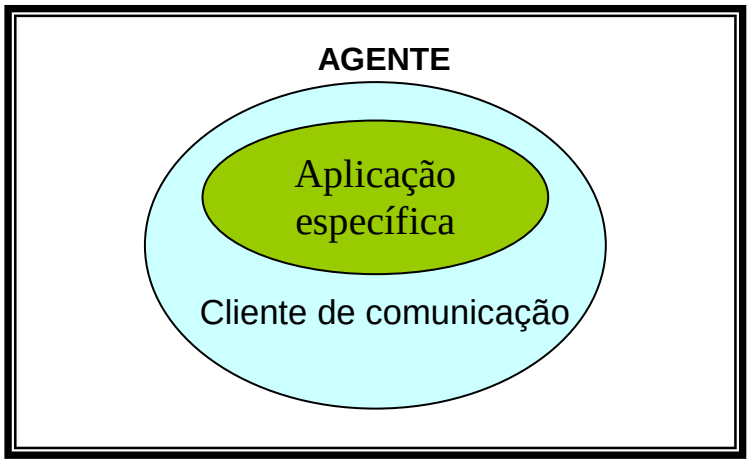

FIGURA 5.1 - Módulos do agente

O cliente de comunicação do agente serve de interface entre a aplicação específica e o AR.

A figura 5.2 apresenta a estrutura interna do cliente de comunicação do agente, que é formado pelos seguintes módulos:

- assinatura – é responsável pela assinatura da aplicação específica. A assinatura é uma string, que as mensagens entre os agentes devem conter, destinada a autenticação das mensagens;

- login/requisição – o *login* do agente junto ao AR e a requisição de comunicação são responsabilidades deste módulo;

- endereçamento – este módulo é o encarregado pela adição à mensagem de comunicação de dois *headers* (um para a identificação do destinatário e outro para o controle da consistência da mensagem), também pela obtenção da identificação do remetente, para o envio de resposta;

- protocolo – o tratamento em nível de protocolo é feito por este módulo.

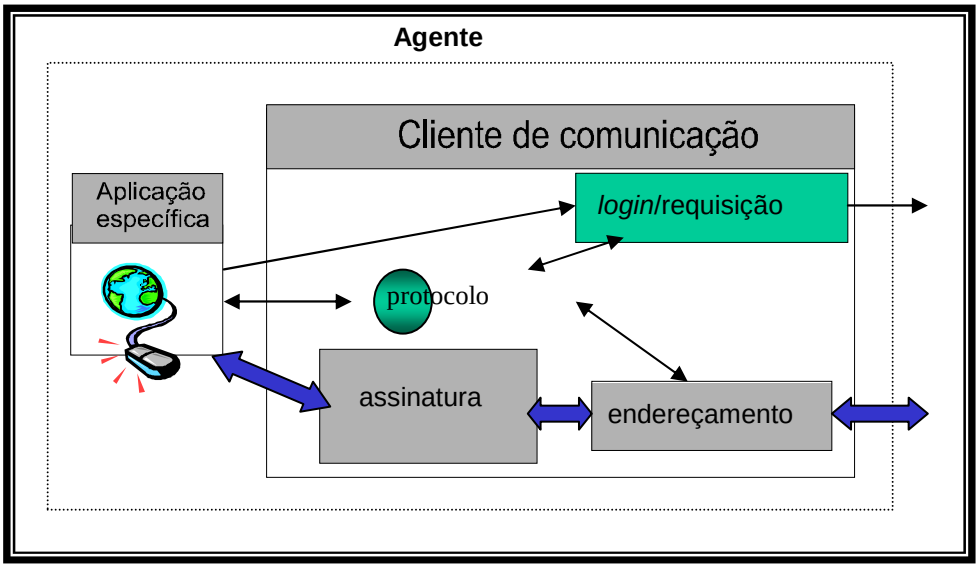

FIGURA 5.2 - Módulo cliente

Para dar início ao processo de comunicação, o agente requisitante precisa conectar-se à porta lógica, que é monitorada pelo AR local, e enviar a sua identificação para o *login*.

O registro do agente junto ao AR, denominado processo de *login*, é responsável pela consistência no roteamento das mensagens. Neste processo, uma variável booleana é retornada para o cliente de comunicação do

agente, indicando que a conexão foi registrada com sucesso ou refutada. Na não aceitação do *login* do cliente pelo AR, uma mensagem indicando que já existe algum agente ativo, com a mesma identificação utilizada por aquele cliente, é informada ao usuário.

Ocorrendo alguma falha na requisição de comunicação ou na conexão do agente ao AR, um código é retornado ao agente e, então, informado ao usuário. Em erros de requisição, o AR é responsável pelo envio de um código ao agente. Em falhas de conexão, é retornado pelo protocolo TCP e, transparentemente, mapeado para o protocolo de agentes (ver 5.4.3 Mensagem de protocolo).

Depois de haver decorrido, com sucesso, o processo de *login*, inicia-se o processo de requisição de comunicação, que é responsável pela localização do agente destinatário e pela ativação deste se não estiver ativo no momento. Este processo começa a partir do envio da identificação do agente destinatário ao AR.

Após o envio da identificação do agente destinatário - na requisição de comunicação - é aguardado o código “200” do protocolo, indicando que o destinatário está ativo e espera o início da comunicação.

Na comunicação, emprega-se o conceito de pacote de mensagens, isto é, juntamente com a mensagem são empacotados *headers*, destinados ao controle e transporte (ver 5.4.2 Mensagem de comunicação).

O módulo cliente é responsável, na transmissão das mensagens, pela adição de *headers*, referentes à assinatura do módulo aplicativo, à lista de destinatários e a sua consistência; na recepção, pela decodificação das mensagens destinada à averiguação da consistência, da assinatura e determinação do agente remetente.

Para o término da comunicação entre os agentes, o código “600” do protocolo é enviado de um agente para o outro e, após o recebimento da confirmação, que consiste no mesmo código “600”, o módulo de aplicação do agente decide se faz a desconexão ou aproveita a condição de ativo a fim de realizar outra requisição.

A identificação dos agentes, nessa arquitetura, é uma string, formada a partir do e-mail do seu proprietário, no domínio em que ele é cadastrado, e de seu nome, conforme é mostrado a seguir. Esta identificação, associada ao controle de cadastramento de agentes, faz com que seja única no âmbito da Internet (ver 4.1.1 Identificação de agentes). A seguir na figura 5.3, é apresentada em notação *Backus-Naur Form* (BNF), a gramática que especifica o conjunto de sentenças, integrantes da identificação dos agentes.

```
identificacao_agentes          ::= username "@" dominio "/" nome_agente;

caracteres <TERMINAL>   ::= '[a-zA-Z]+';

numerico  <TERMINAL>   ::= '[0-9]+';

alfanumerico                    ::= {caracteres | numerico};

identificador <TERMINAL> ::= '[a-zA-Z_]+[a-zA-Z0-9_]*';

composicao_de_nomes <SPACE = ""> ::= {identificador | alfanumerico | "-" | "_" |"."};

username                        ::= composicao_de_nomes;

dominio <SPACE = "">       ::= alfanumerico "." alfanumerico [{"." alfanumerico}];

nome_agente                     ::= composicao_de_nomes;
```

FIGURA 5.3 - BNF da identificação de agentes

5.2 AR

O AR é, ao mesmo tempo, um cliente e um servidor, sendo responsável pelo tratamento de todas as conexões provenientes dos agentes, exercendo o papel de intermediário na comunicação entre eles.

A composição básica do AR é ilustrada na figura 5.4. Os módulos componentes são:

- gerenciador de conexões – tem, como função geral, o tratamento das conexões provenientes dos agentes;
- roteador - faz o roteamento das mensagens entre os agentes. Este módulo também é encarregado da decodificação dos *headers* das mensagens provenientes dos agentes, além de adicionar os *headers* – o de identificação do remetente e o de controle de consistência – às mensagens destinadas aos agentes;
- parser - é responsável pelo *parser* da identificação dos agentes em todo o processo de comunicação (*login*, requisição e comunicação) e também pela verificação de erros sintáticos;
- gerBD - faz o gerenciamento do arquivo local dos agentes, onde se encontram armazenadas as localizações (*PATH*) dos agentes no disco

local, e também do arquivo de *cache*, onde são armazenados os últimos números IPs das máquinas, onde se encontram os ARs acessados;

- cliente – trata-se do módulo responsável pelas conexões *socket* entre ARs, destinadas à troca de mensagens entre os agentes localizados em máquinas diferentes, também pelas conexões com o ANS;

- segurança – forma uma espécie de *firewall*, que faz a verificação do endereço IP da máquina de onde se origina a conexão, para determinar, segundo uma lista de acessos, a aceitação ou não dessa para a realização do *login*;

- login - é destinado ao *login* das conexões dos agentes, a partir da identificação desses, a fim de ser utilizado como informação pelo módulo de roteamento;

- protocolo - o tratamento em nível de protocolo é realizado neste módulo;

- ativador – encarregado pela ativação do agente;

- servidor - é o servidor de conexões responsável pelas conexões *sockets* provenientes dos agentes e pelas transferências para o gerenciador de conexões, com a finalidade de que sejam tratadas.

Após a conexão de um agente à porta lógica onde o AR está ativo e depois do envio de sua identificação para que seja processado seu *login*, é retornado, como resposta, para o *login* do agente junto ao AR, um valor do tipo lógico; se for retornado (FALSO), já existe algum agente conectado com essa identificação; então, o agente é desconectado. Se (VERDADEIRO), é realizado o *login* do agente.

Depois da realização do *login* do agente junto ao AR, para iniciar-se o processo de requisição de comunicação, basta ao agente enviar a *string* com a identificação do agente destinatário (requisição). Se, na localização deste agente, ocorrer alguma falha, é retornada uma mensagem do protocolo, informando-a ao agente.

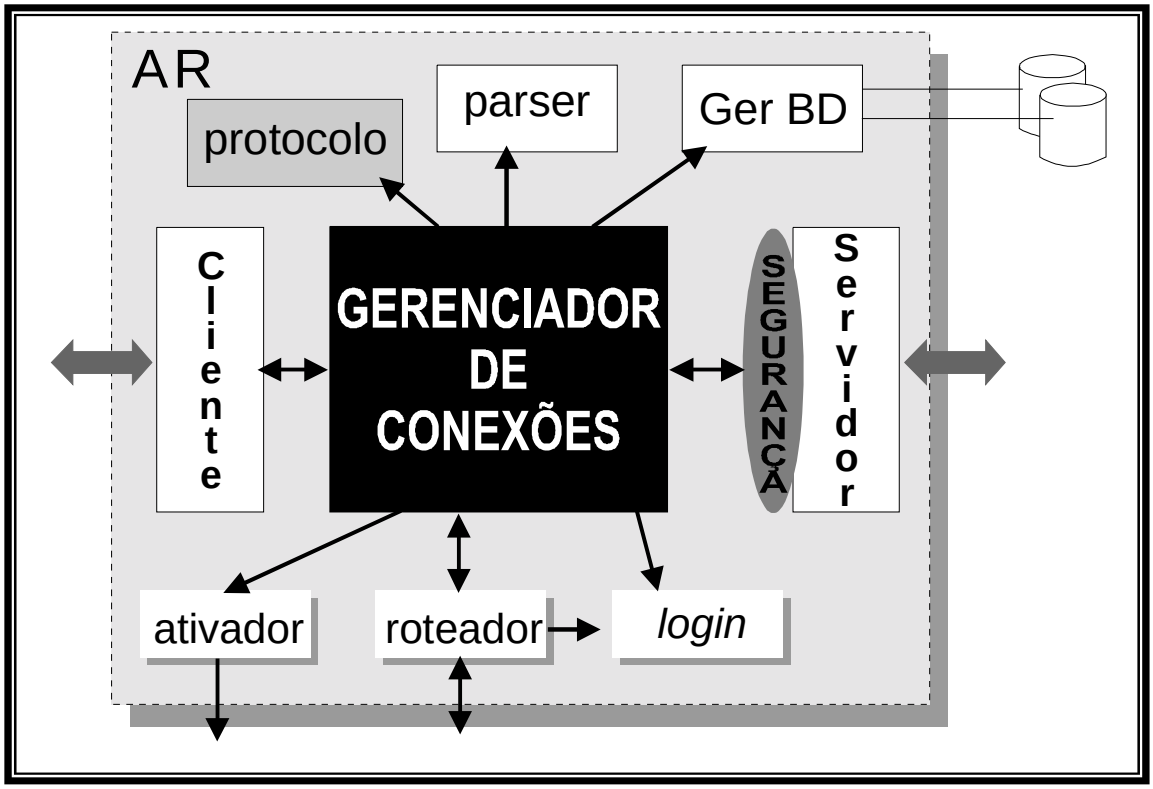

FIGURA 5.4 - Módulo AR

A *string* de identificação é avaliada tanto no processo de *login* quanto na requisição, a fim de verificar a não existência de erros sintáticos.

De posse da localização do agente destinatário, o AR procede, se na mesma máquina, à execução desse agente; se, em máquina ou domínio diferente, a execução é realizada pelo AR da máquina onde o agente está localizado.

Na execução de agentes, que não fazem parte da máquina onde está o agente requisitante da comunicação, é acionada uma conexão *socket* para o AR pertencente ao endereço retornado, da consulta ao ANS. Em tal conexão, pode haver erros, que são mapeados para o protocolo e, a seguir, enviados ao agente com a finalidade de que este os processe.

A comunicação entre ARs é quase idêntica à realizada entre agente e AR, incluindo todo o processo de *login*. A única diferença nesta conexão é o envio do código “200” do protocolo ao agente requisitante da comunicação, pelo AR destinatário, após estar o agente devidamente conectado e registrado ao AR. A mensagem do protocolo informa que a requisição foi aceita e que o agente está autorizado a iniciar a troca de mensagens com o agente destinatário. O envio do código não é feito diretamente pelo AR destinatário ao agente, mas sim roteado via AR remetente para o agente.

Quando do encerramento de qualquer conexão ao AR, seja esse término intencional ou por falha, é feito o tratamento das referências mantidas para a conexão, ou seja, o *logout*.

5.3 ANS

O ANS é o servidor de endereços de agentes, o qual fica sempre aguardando conexões *socket* para a sua porta lógica, para, então, proceder à resolução de nomes.

A constituição interna básica do ANS é, como ilustrada na figura 5.5, formada pelos seguintes módulos:

- protocolo – módulo responsável pelo tratamento em nível de protocolo;
- gerBD – encarrega-se do gerenciamento do arquivo global de agentes por domínio. Nesse arquivo encontram-se armazenados os endereços IPs das máquinas onde estão localizados os agentes que pertencem ao domínio e suas descrições;
- servidor – é o servidor de conexões *socket*, ativo em uma determinada porta lógica, diferente da dos ARs, aguardando novas conexões;
- cliente – é o módulo responsável pelas consultas ao DNS e também pelas conexões, para requisições a outros ANSs;
- segurança – é responsável pela verificação das permissões dos agentes. O objetivo dessa verificação é a manutenção de uma hierarquia de acessos à resolução de nomes;
- gerenciador – o módulo gerenciador é responsável pela administração da distribuição das tarefas entre os módulos internos.

Na resolução de nomes, primeiramente é feita uma verificação para determinar, a não existência de erros sintáticos na requisição de comunicação do agente. Ocorrendo algum erro na requisição, é retornado para o AR e, posteriormente, repassado ao agente o código “400”.

Não havendo erro sintático na requisição do agente, inicia-se o processo de resolução de nomes com a verificação, em nível de domínio, para qual destino é a requisição. Esta verificação determina se o agente requisitado faz parte do próprio domínio ou é pertencente a algum outro na Internet. A verificação é realizada através da avaliação da *string* de identificação do agente destinatário (requisição), enviada pelo agente remetente, através do AR local.

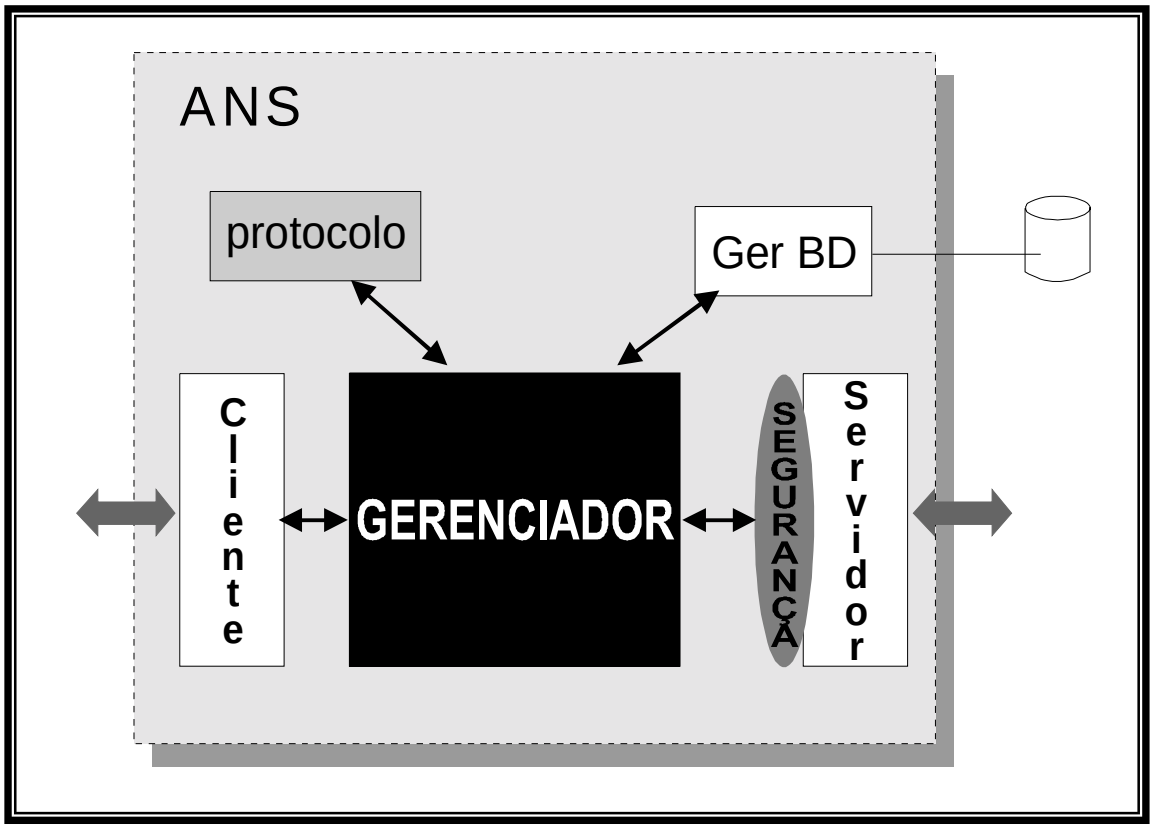

FIGURA 5.5 - Módulo ANS

É feita uma consulta ao arquivo global de agentes, quando a requisição é para o próprio domínio, resultando numa *string*, com a informação que representa o endereço IP da máquina onde este agente está localizado, conforme exemplo: “143.54.8.170”; ou a *string* vazia “ ” significando que tal agente não existe. O número IP é enviado ao AR local, ou, se o agente não existir, o código “404” é repassado ao agente.

Uma consulta ao DNS é acionada, se a requisição é para outro domínio. Essa consulta deve retornar em um vetor de números inteiros que representam o endereço IP da máquina “*agent*” do domínio requisitado. Após ser retornado o endereço IP da consulta ao DNS, faz-se uma conexão *socket* ao ANS deste endereço, e envia-se a identificação do agente destinatário. O ANS do destinatário faz todo o tratamento da requisição, conforme foi mencionado, e retorna o endereço IP da máquina, onde está localizado o agente destinatário, ou mensagem informando algum erro, segundo o protocolo.

Na consulta ao DNS, se um vetor preenchido com zeros for retornado, indica que a máquina “*agent*” não existe, sendo retornado ao agente o código “505”.

Após a resolução de nomes os clientes (ARs ou ANSs), são automaticamente desconectados.

5.4 Mensagens

Os protocolos TCP e *sockets* provêem canais de comunicação baseado em fluxos (*streams*); dados são tratados como fluxos contínuos de bytes. O protocolo que aqui se apresenta é caracterizado pela expressão da informação em uma unidade auto-suficiente - a mensagem. Pelo fato de que clientes não conseguem decodificar misturas arbitrárias de dados provenientes de diversas mensagens, utiliza-se o conceito de pacote de mensagem. Os benefícios desse conceito, através da utilização de encapsulamento, são [CAR 94]:

- mensagens independentemente tratadas – as mensagens, em uma conexão, são distintamente transmitidas por meio de um único canal de comunicação, podendo ser independentemente tratadas. Sem a utilização de encapsulamento, o servidor deve conhecer o formato de toda a mensagem, de forma a assegurar sincronização na transmissão de mensagens distintas. Isso será um problema se o cliente continuamente sofrer atualizações;

- decodificação distinta – o encapsulamento permite a decodificação separada das mensagens. Se uma mensagem está incorretamente formada, a decodificação distinta das mensagens irá fazer com que as mensagens subseqüentes não sejam atingidas por essa falha;

- consistência das mensagens – se acontecer alguma falha de rede durante a transmissão da mensagem, esta não será entregue parcialmente, a parte afetada será descartada, deixando o tratamento de falhas para o protocolo TCP.

Todos os valores transmitidos através de *streams*, no protocolo aqui apresentado, são expressos em ordem de bytes de rede, isto é, bytes significativos, em primeiro lugar. Essa é uma particularidade importante para o interfaceamento com clientes e servidores implementados em diversas linguagens. A ordem de bytes de rede é amplamente utilizada em comunicações na Internet.

Foram adotadas as seguintes padronizações no protocolo, para a representação em bytes, dos tipos expressos nas conexões de rede:

- *boolean* – os valores lógicos são representados por um byte;

- *int* – os números inteiros têm a sua representação feita por uma série de quatro bytes, do mais significativo para o menos significativo;

- *string* – é representado pelo formato UTF-8, que é um formato de transformação do padrão Unicode[3] [NOE 94] [GOO 98], definido no RFC2044 [YER 98].

No formato UTF-8, mostrado na figura 5.6, *strings* são codificados utilizando-se uma seqüência de dois bytes, na qual os dois primeiros bytes especificam o tamanho, seguido pelos caracteres codificados. O tamanho é expresso do bit mais significativo para o menos significativo.

tamanho		caracteres codificados
len hi	*len lo*	*...*

FIGURA 5.6 - Codificação de *strings* em UTF-8

Codificação de caracteres em UTF-8:

Os caracteres individuais são codificados de acordo com as seguintes tabelas abaixo: caracteres ASCII - com um único byte; caracteres gregos, hebreus e árabes - com dois bytes, e todos os outros caracteres são codificados como três bytes. Foi empregada uma variante do padrão UTF-8: o caractere '\u0000' é codificado em dois bytes, para que nenhum caractere seja codificado com zero byte.

Caractere '\u0000' é codificado com dois bytes:

byte 0				byte 1		
1	*1*	*1*	*00000*	*1*	*1*	*000000*

FIGURA 5.7 - Representação do caractere '*u0000*'

Os caracteres pertencentes à faixa de '*u0001*' a '*u007f* ' são codificados com um simples byte.

[3] Unicode é um padrão internacional para a representação de caracteres multilínguas através do set de caracteres de 16 bits do padrão ISO/IEC 10646.

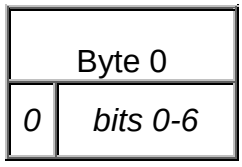

FIGURA 5.8 - Representação dos caracteres de '\u0001' a '\u007f'

Os caracteres da faixa de '\u0080' até '\u07ff ' são codificados com dois bytes.

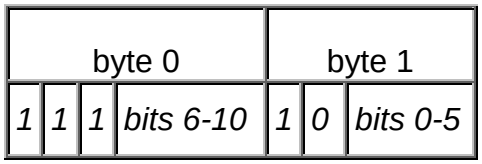

FIGURA 5.9 - Representação dos caracteres de '\u0080' até '\u07ff'

Os caracteres da faixa de '\u0800' a '\uffff ' são codificados com três bytes.

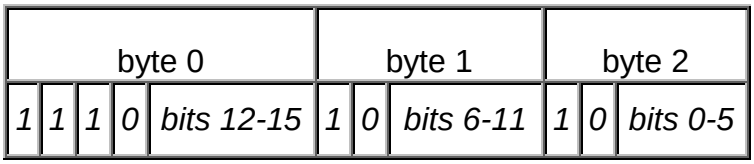

FIGURA 5.10 - Representação dos caracteres de '\u0800' a '\uffff'

Existem três diferentes tipos de mensagens utilizadas no protocolo:

1. de requisição;
2. de comunicação;
3. de protocolo.

5.4.1 Mensagem de requisição

A requisição é a solicitação para que se ative o agente destinatário. Em todo o processo de requisição, é utilizado, para transmissão e recepção, o tipo *string*, isto é, a identificação enviada, do destinatário para o AR, do AR para o AR destinatário, sendo em máquinas diferentes, e da

recepção do número IP, com a localização da máquina onde está localizado o agente, é feita em *string*.

Para uma consulta ao ANS, é enviada pelo AR uma mensagem, que é a identificação do agente destinatário, encapsulada juntamente com a identificação do cliente requisitante (remetente), ambas do tipo *string*. Essas identificações são precedidas por um *header* que indica, através de um número inteiro, o tamanho da mensagem. A figura 5.11 apresenta o formato da mensagem de requisição ao ANS. A resposta à consulta, que é o número IP da máquina "*agent*" do domínio requisitado, faz-se, também, em formato *string*.

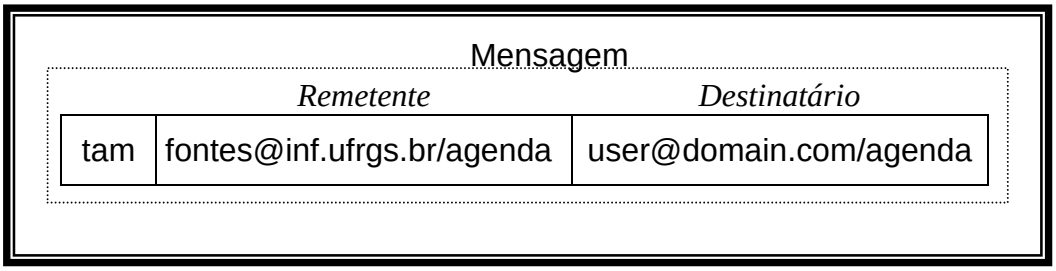

FIGURA 5.11 - Mensagem de requisição ao ANS

A incorporação da identificação do agente remetente, na mensagem, é destinada a futuros incrementos das funcionalidades do ANS. Esse incremento é proporcionado pela implementação de módulos de restrição de acessos às informações do arquivo de agentes global, mantido pelo ANS.

5.4.2 Mensagem de comunicação

Na comunicação entre os agentes com o conceito de encapsulamento, cada mensagem pode ter tamanho e formato arbitrários. O formato pode ser de qualquer tipo, proporcionando, assim, um elevado nível à comunicação.

Juntamente à mensagem, são encapsulados *headers*, destinados ao controle do fluxo, da integridade e da segurança. Estes *headers* são:

- assinatura – o módulo de aplicação do agente é assinado, isto é, só é permitida a comunicação entre agentes que tenham a mesma assinatura. Esta serve para impedir que agentes não autorizados intervenham por engano ou maliciosamente na comunicação dos agentes. Para a assinatura das mensagens, utiliza-se o formato *string*;

- destinatários – a comunicação de um agente pode ocorrer sob forma de *multicast* ou de *broadcast*. Esse *header* é composto por um número inteiro, maior que zero, que indica o número de destinatários, sucedido pelo mesmo número de *strings* representando a identificação desses destinatários. Para

a transmissão em *broadcast,* emprega-se, como *header,* somente o número –1;

- tamanho da mensagem – a fim de manter-se um controle da consistência da mensagem, usam-se números inteiros. Estes números referem-se ao tamanho da mensagem. O controle da consistência da mensagem existe para que se possa verificar problemas na mensagem, ocorridos por alguma falha na camada de transporte de rede;

- identificação do remetente – a identificação do remetente da mensagem serve para que o agente destinatário possa encaminhar ao remetente da comunicação uma resposta. Utiliza-se o tipo *string* para a identificação da mensagem.

Na figura 5.12 são apresentados os estágios de encapsulamento das mensagens entre os agentes na mesma máquina.

O formato da mensagem de comunicação pode ser qualquer um, desde que, ambos os módulos de aplicação dos agentes integrantes da comunicação o conheçam, não importando esse formato para o AR nem também para o ANS.

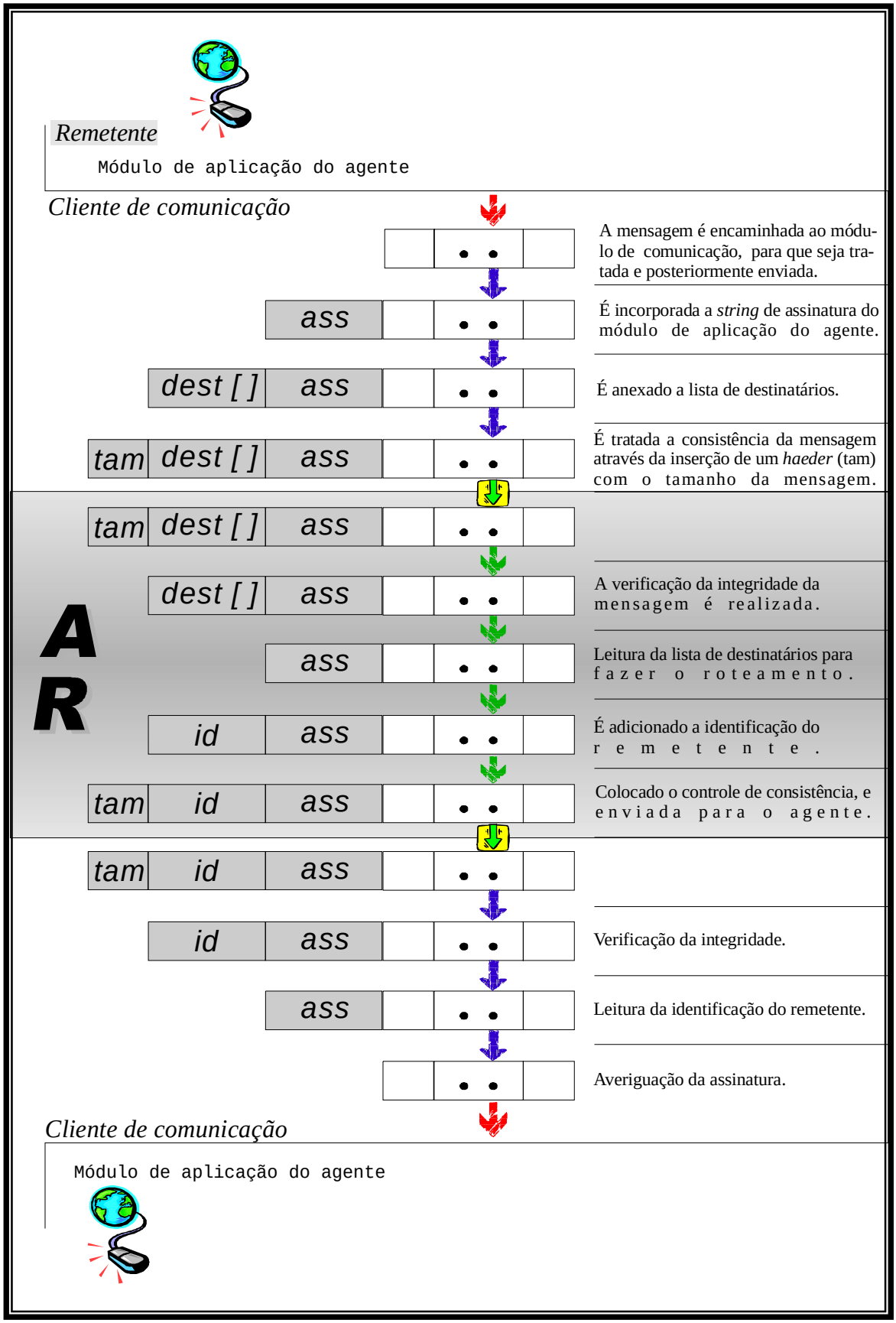

FIGURA 5.12 - Estágios de encapsulamento de mensagens entre agentes na mesma máquina

No roteamento das mensagens entre agentes localizados em máquinas diferentes, como o ilustrado na figura 5.13, as mensagens que chegam ao AR responsável pelo agente remetente (AR1) são retransmitidas, sem nenhuma alteração para o AR responsável pelo agente destinatário (AR2), e vice-versa.

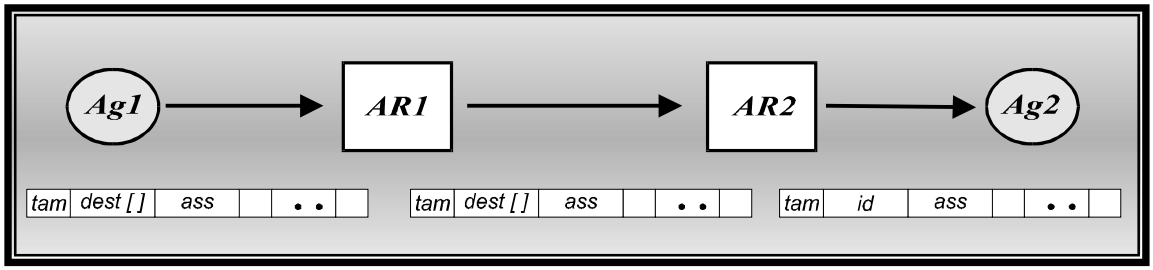

FIGURA 5.13 - Formato das mensagens na comunicação de agentes localizados em máquinas diferentes

5.4.3 Mensagem de protocolo

As mensagens de protocolo são tratadas independentemente das mensagens de comunicação, isso, como já visto, é uma característica da utilização do conceito de empacotamento de mensagens.

Podem ser enviadas mensagens de protocolo tanto:

- do agente para o AR;
- de um agente para outro(s), por intermédio do(s) AR(s);
- do ANS para o AR;
- quanto do AR para os agentes.

O que diferencia uma mensagem de comunicação de uma de protocolo é, além da informação que são os códigos do protocolo transmitidos como números inteiros, a assinatura.

A assinatura é igualmente uma *string*, porém diferente das mensagens de comunicação, que apenas os clientes de comunicação dos agentes precisam conhecer, podendo ser modificada pelos usuários, para ter-se alguma segurança em nível de restrição de acessos a agentes. Ela tende ser conhecida por todos os módulos do sistema.

O protocolo toma, como base, os códigos de *status* do protocolo HTTP.

Os códigos de *status* são formados por três dígitos inteiros. O primeiro dígito do código de *status* define a classe de resposta. Os últimos dois dígitos não são categorizados. Existem oito valores para o primeiro dígito, conforme mostrado na tabela 5.1, a seguir.

TABELA 5.1 – Classes de resposta do protocolo

000	Erro desconhecido	Indica que algum fato desconhecido ocorreu;
1xx	Informacional	Não utilizado, mas reservado para futuro uso;
2xx	Sucesso	A ação foi realizada com sucesso;
3xx	Redirecionamento	Não utilizado, mas reservado para futuro uso;
4xx	Erro em cliente	A requisição apresenta erro de sintaxe ou erro no agente requerido;
5xx	Erro em servidor	O servidor falhou na requisição ou na comunicação;
6xx	Específico agente	São as ações que os agentes podem desempenhar;
7xx	Operações em arquivos	Representam operações de entrada e saída em arquivos.

A seguir, na tabela 5.2, são apresentados os códigos de resposta do protocolo de agentes.

TABELA 5.2 – Códigos de resposta do protocolo

Código	**Significado**
000	*Unknown error* – Erro desconhecido.
200	*OK* - Sucesso.
400	*Bad Request* - Erro de sintático na requisição.
401	*Unauthorized* - Não autorizado (requer autenticação).
404	*Not Found* – Agente não encontrado.

408	*Request Timeout* – Tempo de requisição esgotado.
409	*Rejected requisition* – Requisição rejeitada.
500	*Internal Server Error* - Erro interno no servidor.
503	*Service Unavailable* - Servidor temporariamente sobrecarregado ou em manutenção.
505	*Service Doesn't Exist* - Este serviço não existe no domínio requerido.
507	*Off Line* – Desligado ou fora de linha .
600	*Disconnect* – Desconectar.
700	*Recorder* – Cadastrar agente.
701	*Remove* – Remover agente.
702	*Consult* – Consultar dados do(s) agente(s).

Os códigos de *status* são extensíveis, mas somente os da classe 6xx não precisam de nenhum tratamento nos ARs e ANS, pois são de escopo, apenas, do módulo de comunicação dos agentes.

5.4.3.1 Códigos das mensagens de protocolo

Cada código de *status* do protocolo de agentes é descrito em detalhes abaixo:

- Erro desconhecido 000

É o código de *status* representando erro não mapeado para o protocolo.

- Informacional 1xx

Essa classe de códigos de *status* indica uma série de respostas a requisições de consulta ao AR e ANS. Não é definido nenhum código de *status* 1xx por não ser válido para a atual versão do protocolo. Entretanto, tal classe deve ser útil a aplicações experimentais fora do escopo dessa especificação.

- Sucesso 2xx

Essa classe de códigos de *status* indica que ações foram realizadas com sucesso.

- 200 *OK*

  Agente ativo. A requisição foi realizada com êxito, e o agente destinatário está ativo, aguardando pela comunicação;

- Redirecionamento 3xx

Essa classe de códigos de *status* indica que, em futuras requisições de comunicação, será necessária a utilização de outro endereço (identificação), porque o agente foi, temporariamente ou definitivamente, transferido de local.

- Erro em cliente 4xx

A classe 4xx de códigos de *status* é destinada a erros no cliente (módulo de comunicação do agente).

- 400 *Bad Request*

  A requisição não pode ser processada pelo AR, devido a erros na sintaxe da identificação do agente. Depois de o AR enviar esta mensagem, o agente é desconectado;

- 401 *Unauthorized*

  Após o agente destinatário estar conectado ao AR do agente remetente, e já ter sido enviada, pelo remetente, a primeira mensagem, é requerida pelo agente destinatário a assinatura (ou por causa de a assinatura contida como *header* da mensagem de comunicação estar em branco ou por haver sido modificada). Após este código ser enviado, é utilizado um tempo padrão de espera para a resposta: se ultrapassado este tempo, ambos os agentes são desconectados;

- 404 *Not Found*

  Este código de *status* é enviado ao agente requisitante da comunicação informando que o destinatário não foi encontrado. É desconectado após o envio desse código;

- 408 *Request Timeout*

  Este código informa ao agente requisitante da comunicação, que o tempo máximo para o processamento da requisição de comunicação expirou. Depois disso, o cliente é desconectado;

- 409 *Rejected Requisition*

  Comunica o agente que a requisição foi rejeitada pelo destinatário.

- Erro em servidor 5xx

São representados por essa classe de códigos de status do protocolo, iniciadas pelo dígito “5”, os erros que podem ocorrer caso o servidor (AR/ANS) seja incapaz de processar a requisição do agente ou tenha ocorrido algum erro na conexão;

- 500 *Internal Server Error*

  É devolvido pelo servidor esse código, se foi encontrada alguma condição não estabelecida no processo de comunicação;

- 503 *Service Unavailable*

  É enviado ao agente este código, representando a condição de impossibilidade temporária, na aceitação de requisições. Indica que o servidor ou está temporariamente sobrecarregado ou em manutenção;

- 505 *Service Doesn’t Exist*

  Este código de número “505” indica ao agente requisitante da comunicação que esse serviço não existe no domínio requerido;

- 507 *Off Line*

  O código “507” é informado ao agente pelo AR local, indicando que a máquina onde se encontra o agente destinatário, está desligada ou fora da rede. Esse código, além de representar a situação das máquinas onde estão os ARs, denota também a situação do ANS;

- Específico agente 6xx

Essa é a única classe do protocolo que aceita incremento nos códigos de *status*, sem precisar de modificações tanto nos ARs quanto nos ANSs. Seu escopo é limitado somente ao conhecimento dos agentes; devido a isso, esta classe é também útil a aplicações experimentais, em nível de agente;

- 600 *Disconnect*

  Para os agentes encerrarem formalmente a comunicação, esse código deve ser enviado de um agenteX para o agenteY e confirmado pelo agenteY, através do envio, como resposta para o agenteX, o mesmo código, ou vice-versa;

- Operações em arquivos 7xx

São representadas por essa classe de códigos de *status* as operações que podem ser realizadas em arquivos (arquivos locais e globais de agentes);

- 700 *Record*

  Esse código indica tanto ao AR quanto ao ANS que a operação a ser realizada é o cadastramento de um novo agente;

- 701 *Remove*

  Para um agente ser retirado do domínio, é necessário que AR e ANS recebam esse código seguido pela identificação do agente;

- 702 *Consult*

  Para se fazer uma consulta ao arquivo global de agentes em busca de algum agente com determinadas características.

5.4.4 Primitivas de serviço - PDUs (*Protocol Data Unit*)

A comunicação entre entidades de camadas adjacentes de um mesmo sistema aberto ocorre através de pontos de acesso de serviço (SAP – *Service Access Points*). Cada nível da hierarquia constitui uma camada de serviço. Em cada camada existe uma *interface* com a camada superior e outra com a inferior. Uma camada de um nível n é referenciada camada (n). O objetivo de uma camada (n) é prover certos e bem definidos serviços para a camada (n+1) e superiores, escondendo os detalhes de implementação. A camada (n) é implementada baseando-se nos serviços oferecidos pela camada (n-1) [NOG 91].

A seqüência de eventos que ocorre na *interface* entre duas camadas adjacentes, através dos SAP (n), é descrita pelas primitivas de serviço. Cada primitiva de serviço tem parâmetros de entrada e saída. No protocolo, são definidos quatro tipos de primitivas de serviço:

- Primitiva de serviço de pedido (*Request*) – é usada por uma entidade (n+1), ou usuário (n), para solicitar ou ativar um determinado serviço prestado pela camada (n);

- Primitiva de serviço de indicação (*Indication*) – é emitida pela camada (n), prestadora de serviços (n), para informar uma entidade (n+1) sobre a ocorrência de um determinado evento de serviço;

- Primitiva de serviço de resposta (*Response*) – é utilizada por uma entidade (n+1) para responder a uma primitiva de serviço de indicação recebida anteriormente da camada (n);

- Primitiva de serviço de confirmação (*Confirmation*) – é empregada pela camada (n) para informar à entidade (n+1) que o serviço solicitado por meio de uma primitiva de serviço de pedido foi completado

Abaixo, são mostradas as tabelas com as PDUs, para cada módulo da arquitetura, contendo as suas respectivas descrições:

Legenda:

id:	*identificação do agente*
id-s :	*login do sistema (identificação)*
id-r :	*login do remetente (identificação)*
id-d :	*login do destinatário (identificação)*
tam :	*tamanho da mensagem*
id-d[] :	*lista de destinatários (identificações dos destinatários)*
ass :	*assinatura do aplicativo*
msg :	*mensagem*
resp:	*resposta (conteúdo)*
path:	*localização em disco do agente (path)*
ip:	*endereço IP da máquina*
coment:	*comentário (dados adicionais do agente)*

TABELA 5.3 – PDUs do AR

AR-CONNECT.request (*endereço,porta*)	Solicita a abertura de conexão ao ANS ou a outro AR, identificando o endereço e porta *socket*.
AR-CONNECT.indication ( )	Indica a conexão de um agente ou AR.
AR-LOGIN.indication (*id*)	Mostra que a solicitação de *login* de um agente ou outro AR chega ao AR.
AR-LOGIN.response (*resp*)	Responde com a aceitação ou não, à solicitação de *login*.

AR-COMMUNICATION.indication (*id-d,id-r*)	Indica pedido de comunicação do agente remetente para com o agente destinatário.
AR-COMMUNICATION.response (*resp*)	Responde ao agente solicitante da comunicação o *status* de ativação do agente destinatário.
AR-DATA.indication (*tam,id-d[],ass,msg,id-r*)	Indica a chegada de mensagem do agente (*id-r*).
AR-DATA.request (*tam,id-d[],ass,msg,id-s*)	Denota a requisição de envio de mensagem de sistema (AR) para o(s) destinatário(s).
AR-DATA.response (*tam,id-d[],ass,msg,id-r*)	Roteia a mensagem proveniente do agente remetente para o(s) destinatário(s).
AR-CONSULT-ANS.request (*id*)	Indica pedido de consulta ao ANS, para o agente identificado por (*id*).
AR-CONSULT-ANS.indication (*ip*)	Designa notificação da resposta ao pedido de consulta feito ao ANS.
AR-RECORD-ARQ.indication (*id,ip,path,coment)*	Indica chegada de pedido de cadastramento de novo agente.
AR-RECORD-ARQ-ANS.request (*id,ip,coment*)	Denota pedido de cadastramento de agente ao ANS.
AR-RECORD-ARQ-ANS.indication (*resp*)	Representa a resposta do ANS sobre o pedido de cadastramento do novo agente.
AR-RECORD-ARQ.response (*resp*)	Indica *status* de resposta, ao *frame* de gerenciamento, sobre o pedido de cadastramento do novo agente.

AR-CONSULT-ARQ.indication (*id*)	Indica chegada de pedido de consulta ao arquivo local de agentes.
AR-CONSULT-ARQ.response (*path,coment*)	Representa a resposta, contendo os dados referentes à consulta.
AR-REMOVE-ARQ.indication (*id*)	Indica pedido de remoção do agente identificado por (*id*).
AR-REMOVE-ARQ-ANS.request (*id*)	Designa pedido de remoção do agente, identificado por (*id*), para o ANS.
AR-REMOVE-ARQ-ANS.indication (*resp*)	Contém a resposta do ANS, ao pedido de remoção de agente.
AR-REMOVE-ARQ.response (*resp*)	Indica confirmação positiva ou negativa do pedido de remoção de agente ao *frame* de gerenciamento.

TABELA 5.4 – PDUs do ANS

ANS-CONNECT.indication ( )	Indica a chegada de conexão de um AR ou outro ANS.
ANS-CONNECT.request (*endereço,porta*)	Solicita a abertura de conexão a outro ANS, identificando o endereço e porta *socket*.
ANS-LOGIN.indication (*id*)	Representa a solicitação de *login* de um AR ou ANS.
ANS-LOGIN.response (*resp*)	Responde com a aceitação ou não da solicitação de *login*.

ANS-CONSULT.indication (*id*)	Denota chegada de pedido de consulta do endereço IP do agente referenciado por (*id*).
ANS-CONSULT.response (*ip*)	Indica que a resposta é enviada, contendo os dados referentes à consulta ao agente.
ANS-RECORD-ARQ.indication (*id,ip,coment*)	Designa pedido de cadastramento de agente feito através do AR.
ANS-RECORD-ARQ.response (*resp*)	Notifica ao AR sobre o pedido de cadastramento de agente.
ANS-REMOVE-ARQ.indication (*id*)	Significa requisição de remoção do agente identificado por (*id*).
ANS-REMOVE-ARQ.response (*resp*)	Indica confirmação positiva ou negativa do pedido de remoção de agente.

TABELA 5.5 – PDUs do agente

AG-CONNECT.request (*endereço,porta*)	Designa pedido de conexão ao AR localizado no endereço e porta identificados pelos parâ-metros (*endereço,porta*).
AG-LOGIN.request (*id*)	Indica solicitação de *login* ao AR, tendo como identificação o parâmetro (*id*).
AG-LOGIN.indication (*resp*)	Denota resposta do AR à solicitação de *login.*

AG-COMMUNICATION.request (*id*)	Designa requisição de ativação do agente identificado por (*id*).
AG-COMMUNICATION.indication (*resp*)	Indica reposta para o pedido de ativação do agente destinatário.
AG-DATA.request (*tam,id-d[],ass,msg*)	Denota requisição de envio de mensagem.
AG-DATA.indication (*tam,id-r,ass,msg*)	Indica chegada de mensagem.

TABELA 5.6 – PDUs do *frame* de gerenciamento

G-CONNECT.request (*endereço,porta*)	Significa pedido de conexão *socket* ao AR localizado no endereço e porta identificados pelos parâmetros (*endereço,porta*).
G-LOGIN.request (*id*)	Indica solicitação de *login* ao AR, tendo como identificação o parâmetro (*id*).
G-LOGIN.indication (*resp*)	Significa resposta do AR à Solicitação de *login.*
G-RECORD-ARQ.request (*id,ip,path,coment*)	Designa pedido de cadastramento de novo agente.
G-RECORD-ARQ.indication (*resp*)	Resposta indicando a aceitação ou não da solicitação de cadas-tramento do novo agente.

G-CONSULT-ARQ.request (id)	Indica requisição de consulta ao agente identificado por (*id*) no arquivo de agentes mantido pelo AR.
G-CONSULT-ARQ.indication (*path*)	Designa resposta, contendo a localização em disco (*path*), do agente requerido pela consulta.
G-REMOVE-ARQ.request (*id*)	Indica pedido de remoção do agente do domínio, tendo como identificação o parâmetro (*id*).
G-REMOVE-ARQ.indication (*resp*)	Representa a resposta ao pedido de remoção do agente.

5.4.5 Diagramas

A seguir nesta seção, são apresentados, diagramas contendo os modelos de transição de estados do Agente, do AR e do ANS, correspondendo as figuras 5.14, 5.15 e 5.16, e também diagramas de eventos, representando os processos de requisição de ativação e de comunicação (na mesma máquina, em máquinas diferentes do mesmo domínio e em máquinas pertencentes a domínios diferentes), figuras 5.17, 5.18, 5.19, 5.20, 5.21 e 5.22.

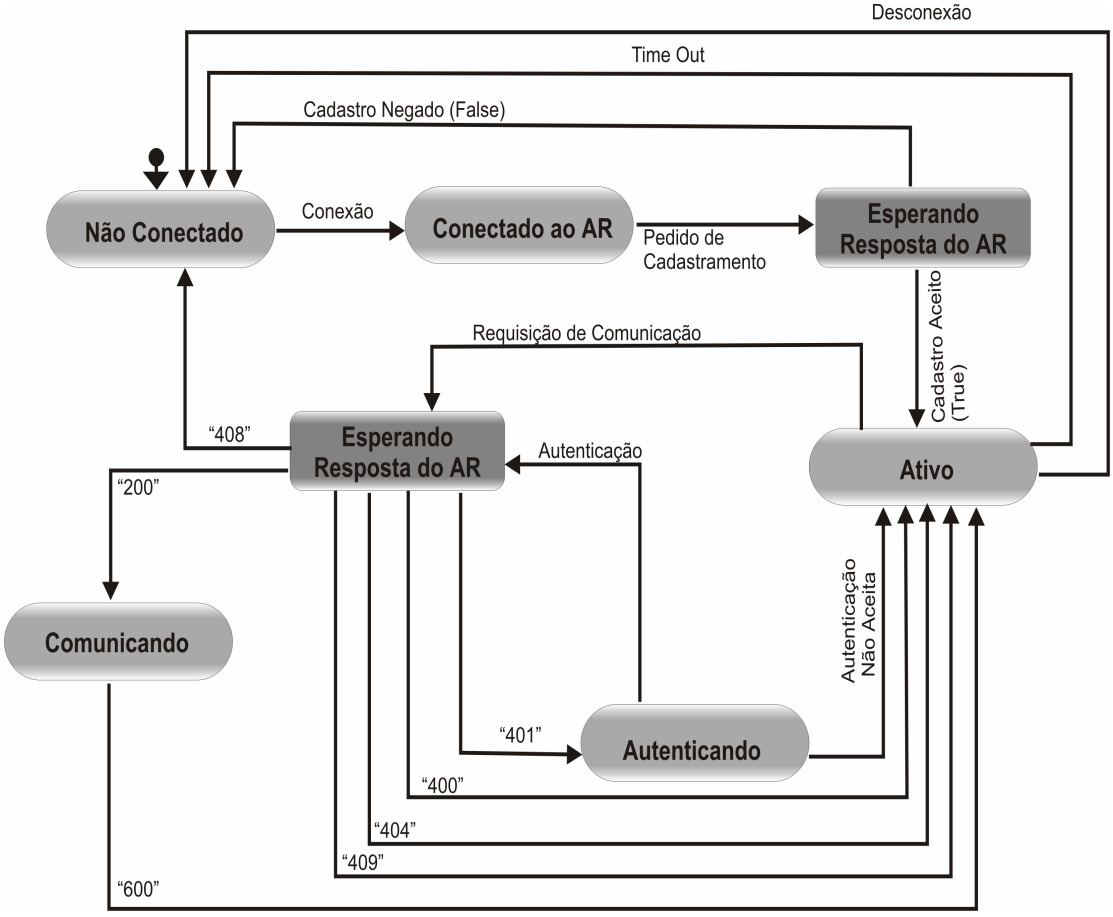

FIGURA 5.14 - Diagrama de estados do Agente

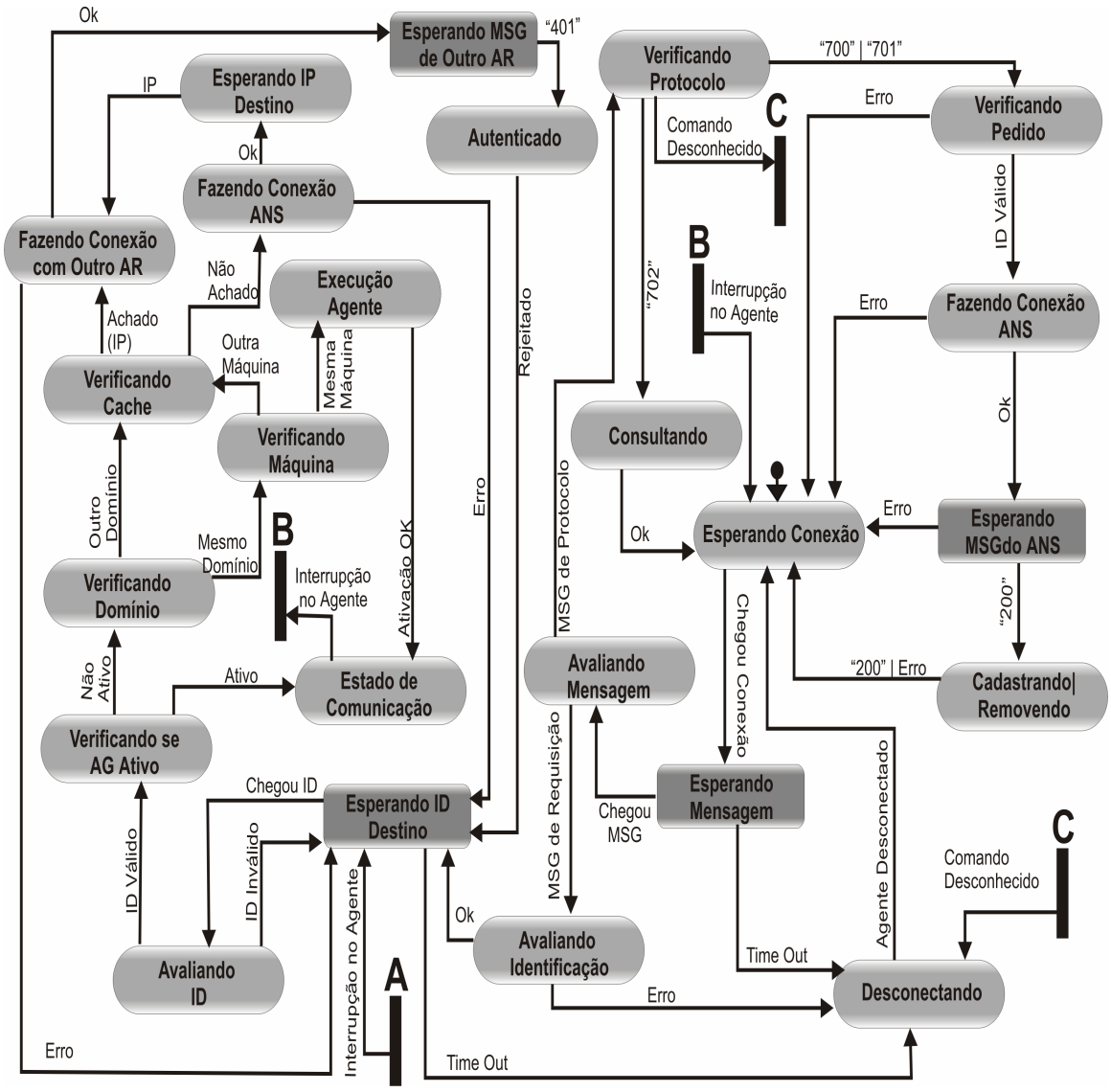

FIGURA 5.15 - Diagrama de estados do AR

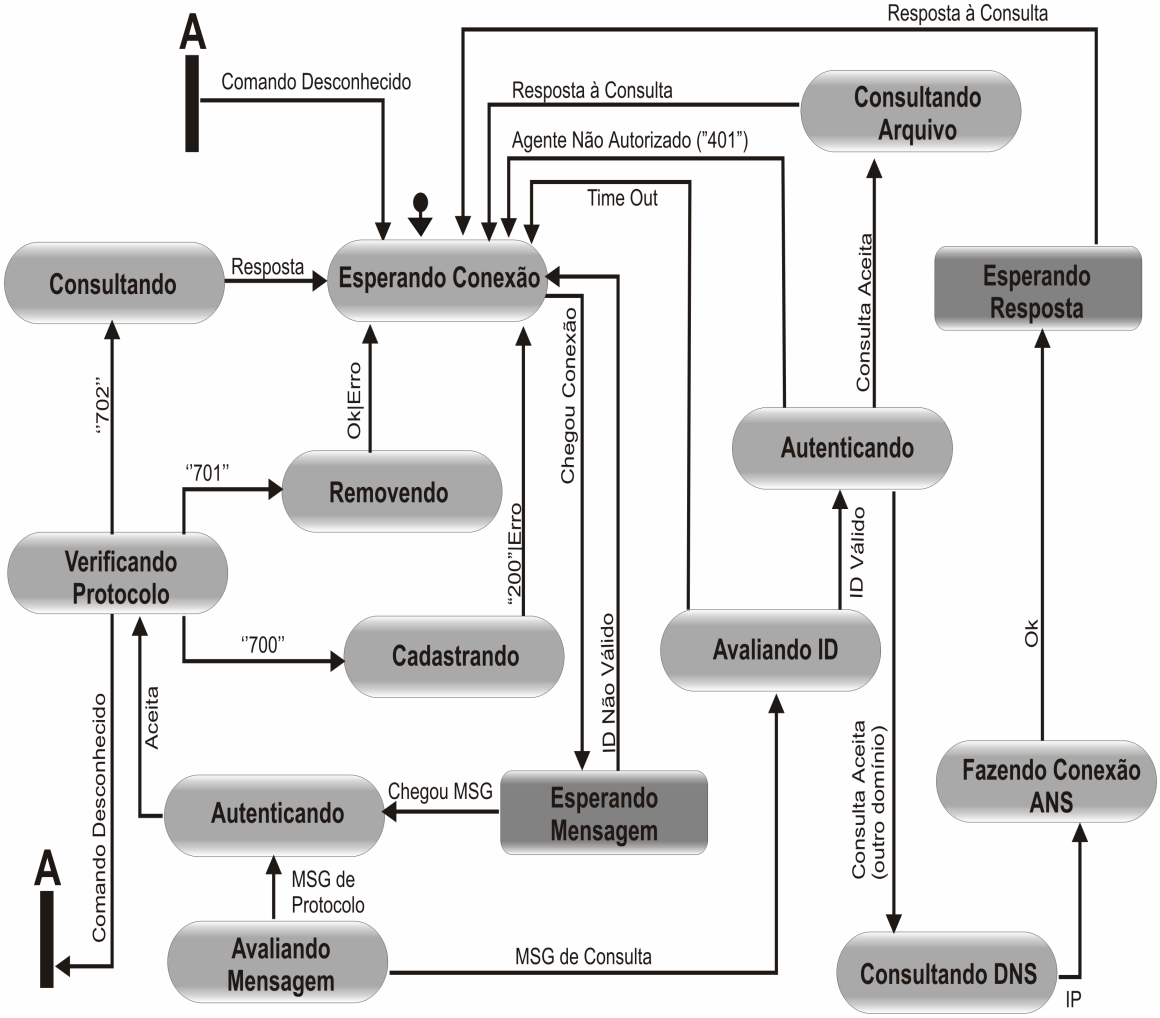

FIGURA 5.16 - Diagrama de estados do ANS

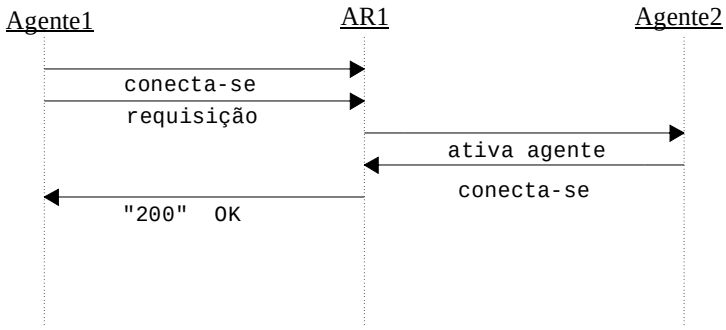

FIGURA 5.17 - Diagrama de eventos (processo de requisição para agente na mesma máquina)

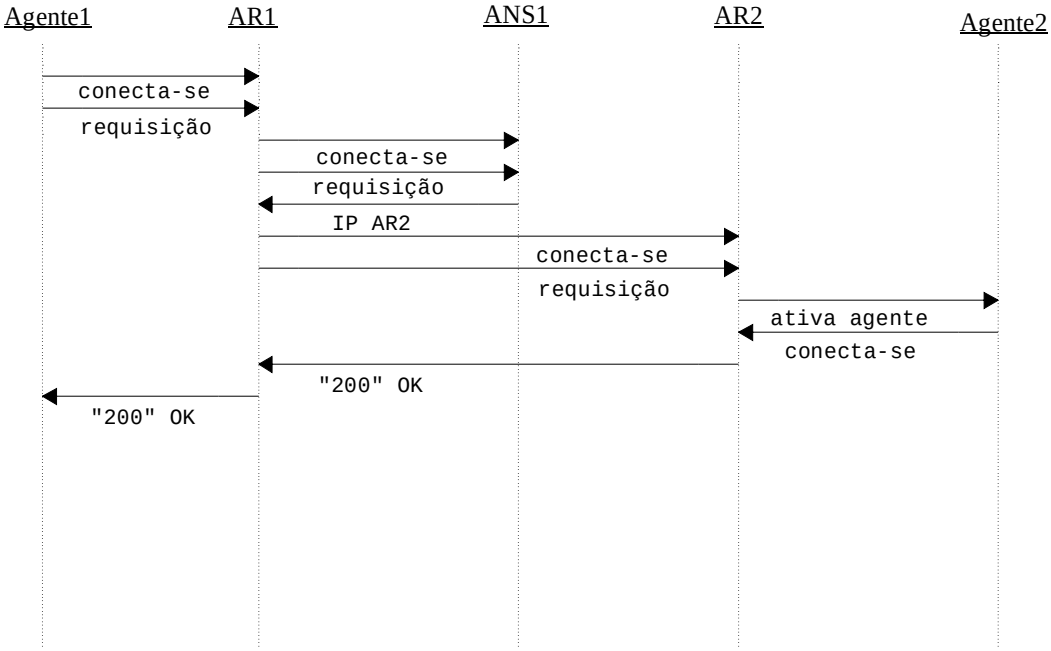

FIGURA 5.18 - Diagrama de eventos (processo de requisição para agente em máquina diferente)

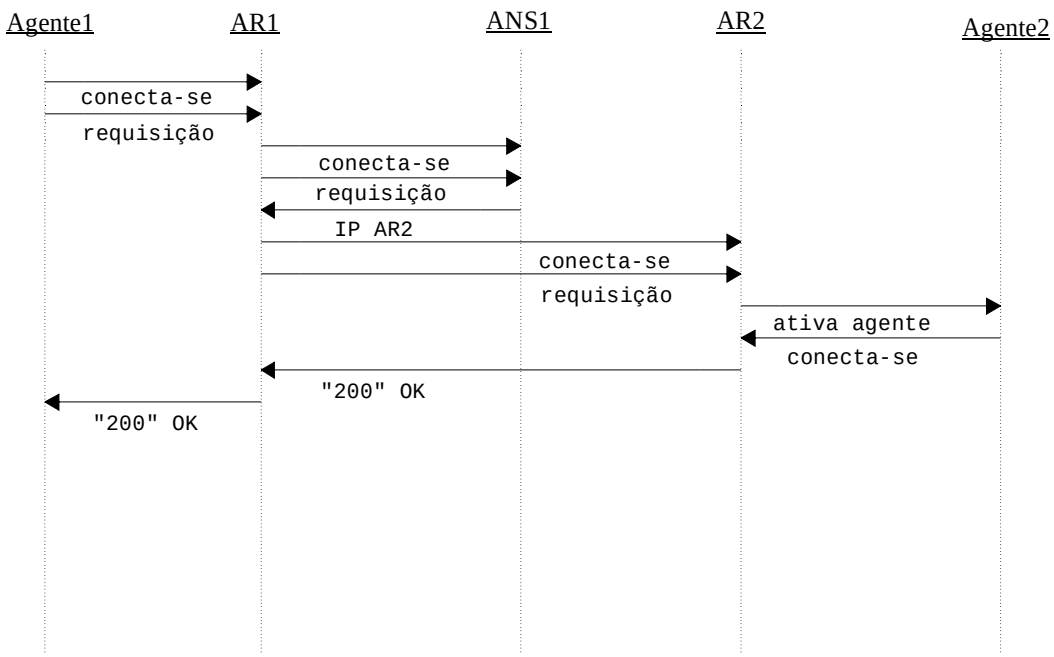

FIGURA 5.19 - Diagrama de eventos (processo de requisição para agente em máquinas de domínios diferente)

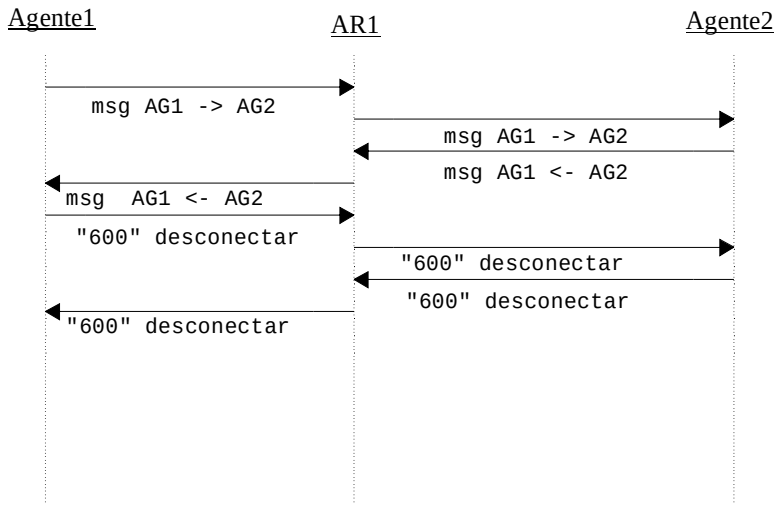

FIGURA 5.20 - Diagrama de eventos (processo de comunicação na mesma máquina)

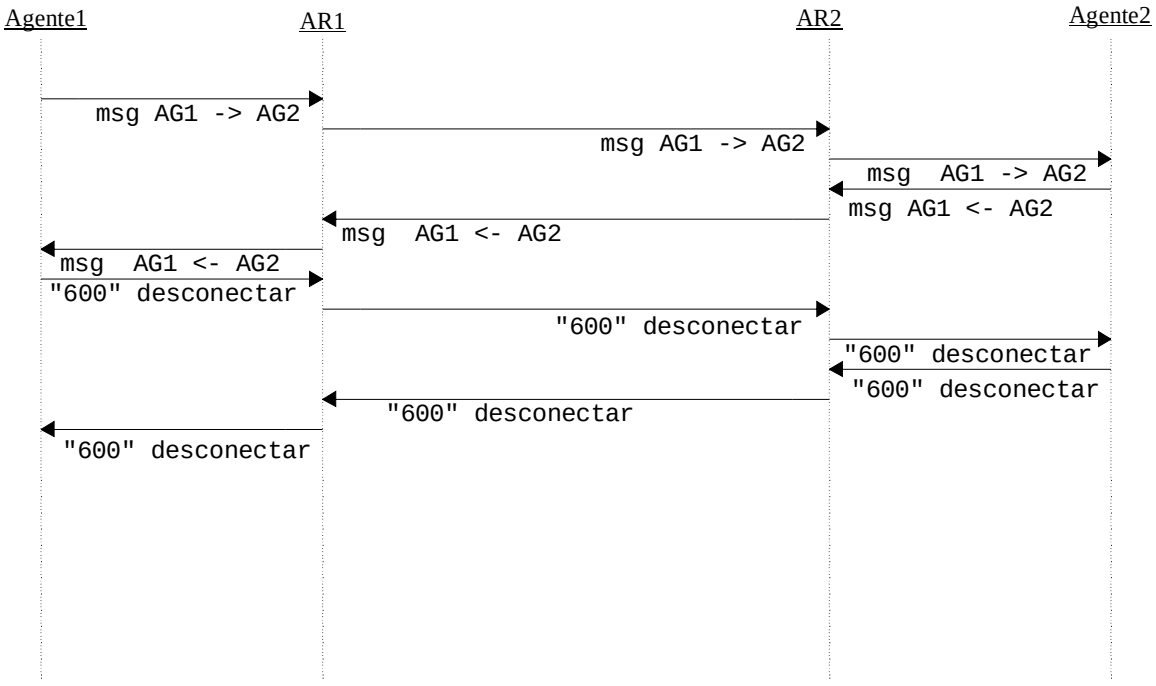

FIGURA 5.21 - Diagrama de eventos (processo de comunicação em máquinas diferentes)

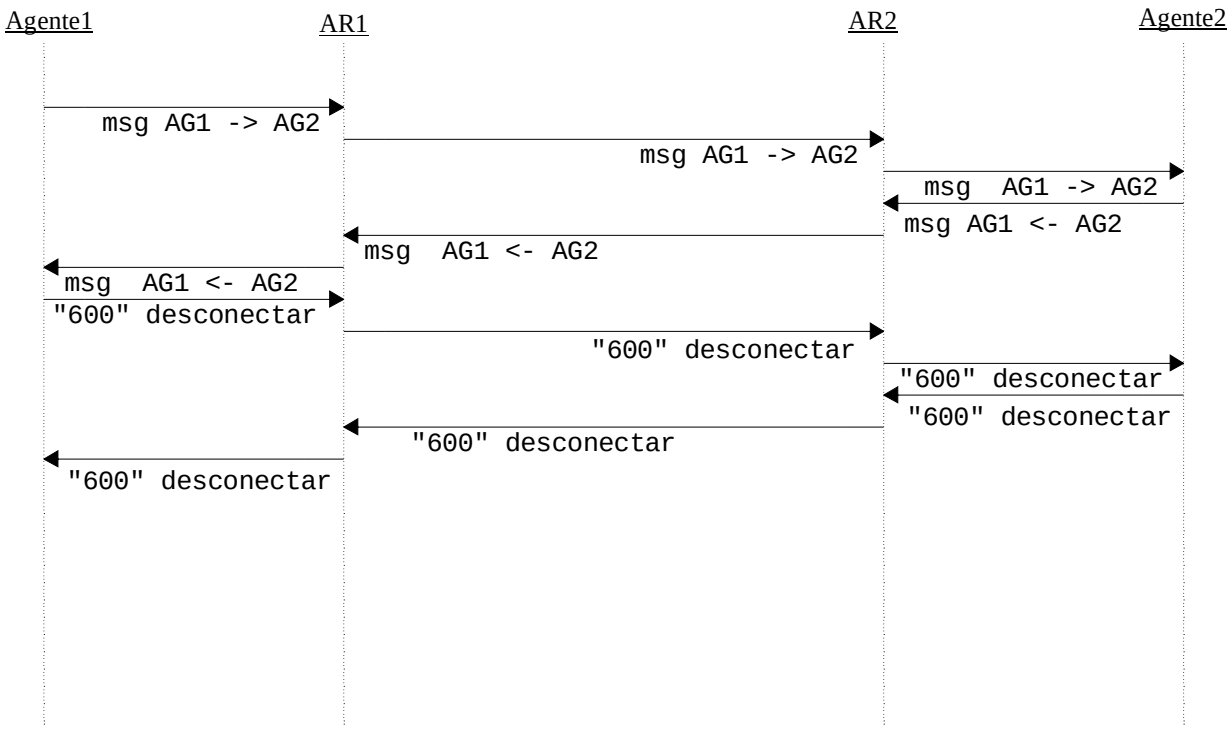

FIGURA 5.22 - Diagrama de eventos (processo de comunicação em domínios diferentes)

6 Validação do Protocolo

De forma a propiciar uma simulação do seu comportamento, objetivando-se uma especificação clara, concisa e sem ambigüidades, uma maior eficiência e um suporte direto para a implementação é exposto, neste capítulo, a forma como se realizou a análise das propriedades estáticas e dinâmicas do Protocolo.

6.1 Linguagem SDL

Para a descrição precisa das funções e *interfaces* entre módulos de *software* definidos durante a fase de desenvolvimento de um projeto, se faz necessário à utilização de linguagens de especificação.

A linguagem SDL (*Specification and Description Language*) foi padronizada pelo CCITT (*Comité Consultatif Internacional de Télégraphique at Téléphonique*) para especificação de sistemas de telecomunicações [CCI 88]. A SDL é constituída com fundamentação nos seguintes conceitos:

- estrutura – descrita hierarquicamente por elementos, chamados de sistema, bloco, canal, processos, rotas de sinais e sinais;
- comportamento – descrito utilizando o conceito de máquinas de estados finitos estendidas;
- dados – definidos como tipos abstratos de dados através da álgebra de tipos;
- comunicação – assíncrona via canais e filas infinitas.

Uma especificação SDL consiste em um determinado número de módulos interconectados. O comportamento de cada módulo é modelado por um ou mais processos. A comunicação entre processos é assíncrona via filas infinitas.

Em SDL, o comportamento é definido por um conjunto de processos executados em paralelo. A cada processo é associado exatamente uma fila de entrada, na qual são colocados todos os sinais de entrada vindos de diferentes rotas.

Um processo SDL é descrito por uma máquina de estados finitos estendida, que consiste basicamente em estados e transições. Para cada

estado existe um conjunto de símbolos de entrada associado a um conjunto de transições. Quando em um estado, um processo está suspenso aguardando um sinal de entrada, sinais somente podem ser recebidos quando o processo está em um estado, e a recepção de um sinal é o único evento que pode gerar uma transição para um novo estado.

Uma descrição mais detalhada pode ser encontrada em [TRI 91].

6.2 Métodos de Validação

Existem várias técnicas para análise de projetos de protocolos. Os principais, segundo BOCHMANN [TRI 94], são:

- Análise de alcançabilidade - utilizada principalmente no caso de especificações na forma de máquinas de estados finitos;
- Derivação de invariantes - empregada para determinar os estados alcançáveis baseados na estrutura de especificações, dadas na forma de Redes de Petri;
- Prova de programas - utilizada quando um protocolo é especificado em uma linguagem de programação;
- Execução simbólica - similar à prova de programas. Também trata de protocolos sob a forma de programas, e analisa todas as possíveis transições a serem executadas.

Uma vez que todos os modelos de transição de estados já haviam sido esboçados e que o modelo da linguagem SDL é baseado no conceito de máquinas de estados finitos, foram utilizados neste projeto os métodos de análise de alcançabilidade e de execução simbólica.

Com o objetivo de se gerar uma especificação clara, concisa e sem ambigüidade, uma base para análise de critérios (eficiência e correção) e também um suporte direto para a implementação, utilizou-se as seguintes técnicas de verificação [TRI 94]:

- Análise de Alcançabilidade – consiste na exploração de todas as possíveis interações entre entidades de um protocolo, através da construção de uma árvore de alcançabilidade a partir do estado inicial global. Cada novo estado global (ou seja, o estado de ambas as entidades e os canais) é gerado como resposta a um possível evento;
- Análise Enumerativa – consiste na exploração da árvore de classes de estados. Tal árvore é obtida através do exame de todas as possíveis classes de estados, começando da classe inicial. Uma classe de estados é definida como sendo o

conjunto de estados alcançáveis a partir do estado inicial, passíveis de serem escalonados em uma execução;

- Indução Estrutural – assemelha-se à indução finita matemática. É utilizada para verificação de propriedades básicas através da prova que esta válida para todos os objetos de um sistema. Também, através da demonstração que esta é válida para todos os objetos diretamente criados. Com isto, é possível provar que se esta propriedade é válida para um objeto, então ele ainda é válido quando os objetos de um tipo são modificados;

- Prova de Assertivas – consiste na introdução de assertivas em certos pontos do conjunto de programas que implementam a especificação formal do protocolo de comunicação. Tais assertivas são verificadas a partir de outras que são derivadas da especificação do serviço, expressando as propriedades desejadas do protocolo;

- Lógica Temporal – é uma extensão da lógica de predicados baseada em seqüências de eventos totalmente ordenados no tempo.

6.3 Derivação do Resultado

Para derivar um resultado na verificação do protocolo foi necessário a utilização de vários programas. Tais programas são citados a seguir.

Toda a modelagem do sistema foi realizada com a metodologia Orientada a Objetos, para isso utilizou-se a ferramenta CASE *With Class* 98. A partir dessa modelagem foi utilizado, para a geração da especificação em SDL, o *software Visual Parser ++.* Esse *software* que trabalha em conjunto com o *With Class* fazendo a tradução dos diagramas de estados em linguagem SDL.

Com a SDL gerada, foi necessário para a realização da Análise Enumerativa, da Lógica Temporal e da Indução Estrutural, o *software CS VERILOG ObjectGEODE*^{TM} que trabalha com máquina de inferência PROLOG e de REGRAS (sintáticas e semânticas), para a exploração da árvore de classes de estados em SDL. Com estas análises concluídas pôde-se suprimir os *deadlocks* dinâmicos.

Para a efetuação da Análise da alcançabilidade com o objetivo de validar as propriedades sintáticas (sinais nunca recebidos, ambigüidades de estados e *deadlocks* estáticos) foi utilizado o *software Programmar* que consiste da realização da análise de especificações BNF. Assertivas foram

introduzidas manualmente em certos pontos do programa, visando à realização da Prova de Assertivas.

A partir dos procedimentos, descritos anteriormente, derivou-se um resultado operacional que apresenta uma especificação sem ambigüidades, de descrição clara e concisa. Um suporte direto para a implementação que agilizou a fase de desenvolvimento. E uma certeza de eficiência livre de erros.

7 Implementação

Nesse capítulo, é apresentado em uma visão básica, o protótipo implementado, incluindo o ambiente de desenvolvimento e respectivos componentes.

7.1 Ambiente de desenvolvimento

O protótipo implementado contendo 9700 linhas de código foi desenvolvido em Java, linguagem utilizada, devido a suas características. Dentre as quais destaca-se a independência de arquitetura, possibilitando executar, compilar e compartilhar códigos em qualquer arquitetura, desde que, nela, exista um ambiente de execução Java.

Além disso, o sistema Java oferece uma extensa biblioteca de classes requeridas para a transmissão de conteúdos avançados e dinâmicos em ambientes heterogêneos de rede.

7.2 Componentes do sistema

São apresentados, a seguir, os principais componentes do protótipo implementado.

7.2.1 Servidor de Nome de Agentes – (ANS - *Agent Name Server*)

O ANS, apresentado na figura 7.1, é um servidor *multithread* [4] de endereços de agentes. Este servidor é encarregado da resolução de nomes de agentes, isto é, para uma requisição de algum agente, é retornado o número IP da máquina correspondente à consulta, ou um código de erro de acordo com o protocolo. É composto basicamente por duas classes:

[4] Java permite que um mesmo programa tenha vários segmentos sendo executados concorrentemente.

- ***ANServer*** - é a classe responsável pela função de servidor e por criar uma nova *thread* - (*SystemLookup*) - para o tratamento de cada conexão;

- ***SystemLookup*** - é a classe *thread* responsável pela resolução de nomes.

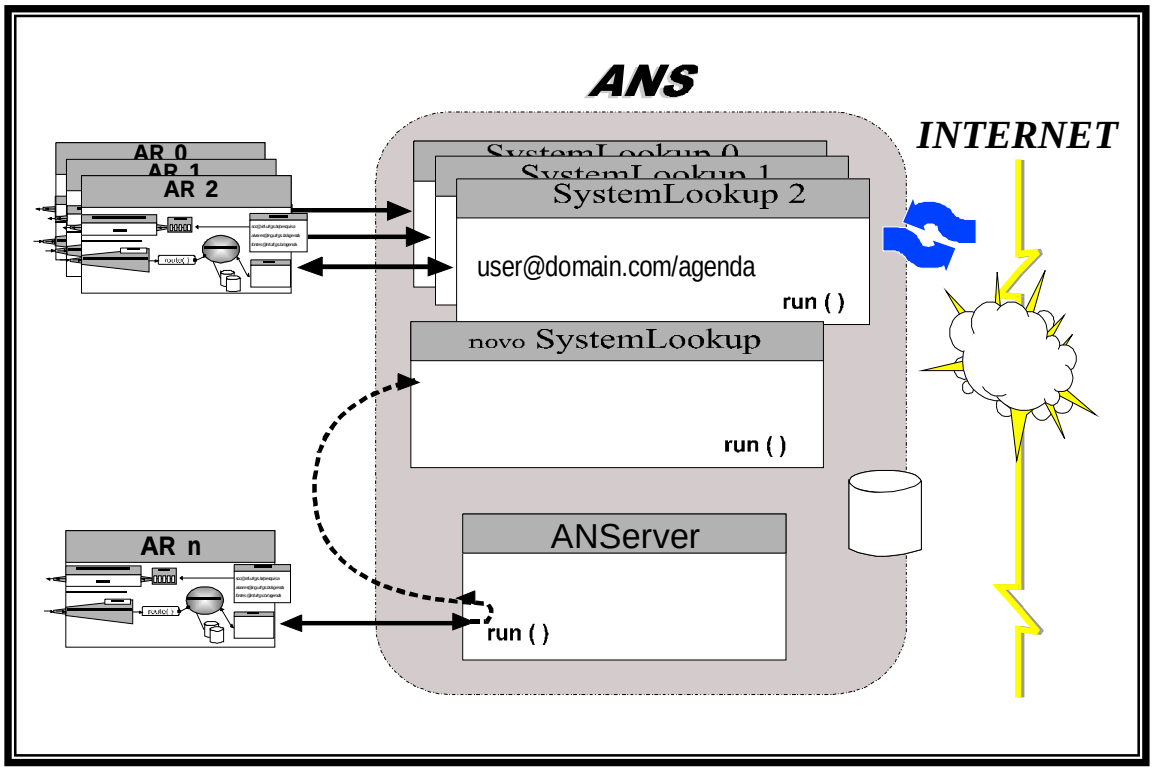

FIGURA 7.1 - *Framework* ANS

7.2.1.1 ANServer

A classe *ANServer* é uma *thread* responsável por aceitar novas conexões, provenientes dos ARs e de outros ANSs, e por ativar *threads* (*SystemLookup*) para o tratamento dessas conexões.

Essa classe é uma aplicação *standalone*, que provê um método construtor simples, que ativa *threads* auxiliares para atenderem a cada conexão. O servidor fica ativo em uma determinada porta, aguardando por novas conexões *socket*. Para cada conexão é criada uma nova instância da classe *SystemLookup*, passando o novo *socket* como parâmetro. Depois de ser criada a nova instância, esta é ativada através do método *run*().

7.2.1.2 SystemLookup

A classe *SystemLookup* é destinada ao tratamento individual das conexões. Para cada instância desta, é recebida uma mensagem contendo a identificação do agente, para que seja processada a resolução de nomes, ou seja, retornado o número IP correspondente ao endereço de sua localização ou uma mensagem do protocolo, se na resolução ocorrer algum erro.

Para realizar esse processo de resolução de nomes, primeiramente, nesta classe, é realizado um *parser* na mensagem entrante, para que sejam feitas: a verificação sintática e a separação do conteúdo da mensagem (*e-mail*, domínio, nome, etc.). Com a obtenção do domínio a partir da operação de *parser*, é verificado, se a resolução é destinada para o próprio domínio ou para algum outro. Na resolução para o próprio domínio, a classe *ManipArq*, destinada à tratamentos em arquivos, é acionada para fazer uma consulta em busca da identificação do agente requerido.

Sendo a resolução para algum outro domínio, uma consulta ao DNS, para a localização da máquina "*agent*" daquele domínio, é feita. Uma operação de *ping* é então acionada para verificar a disponibilidade do serviço naquele domínio. Após a verificação da disponibilidade, é repassado o endereço do ANS do domínio requerido para o AR requisitante.

Uma implementação simples em nível de segurança existe nesta classe, sendo essa através da verificação da assinatura, contida no *header* da mensagem, e do endereço retornado pelo método *getAddress*(), para determinar a origem da conexão.

Após a consulta o AR ou ANS é desconectado automaticamente.

7.2.2 Roteador de Agentes – (AR - *Agent Router*)

O *Agent Router*, mostrado na figura 7.2, segue o mesmo modelo *multithreading* que o ANS. O AR é composto basicamente por duas classes:

- ***Agent_RouterServer*** - é a classe responsável por aceitar as novas conexões provenientes dos clientes (agentes) e criar uma nova *thread* - (*ConexSystemHandler*) - para o tratamento de cada conexão. E também é responsável pela manutenção de uma lista com todas as conexões correntes;

- ***ConexSystemHandler*** - é a classe que faz o trabalho de processamento das conexões dos clientes.

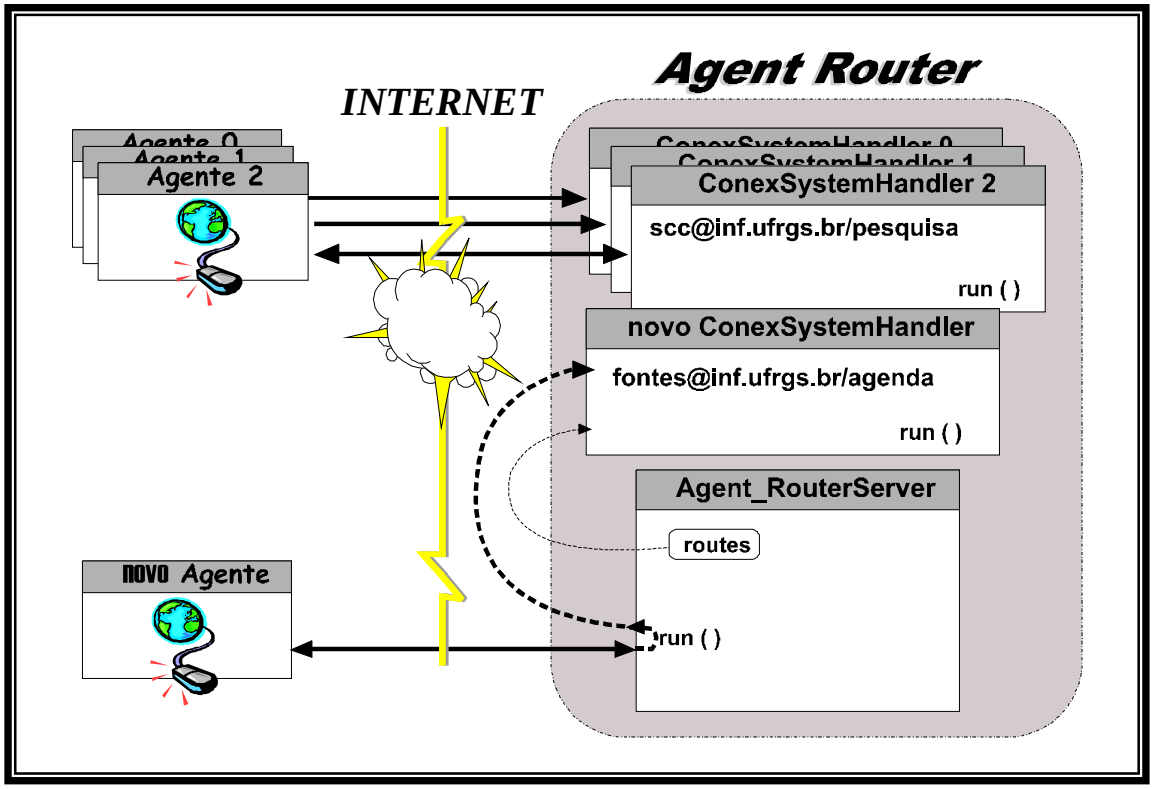

FIGURA 7.2 - *Framework* AR

7.2.2.1 Agent_RouterServer

A classe *Agent_RouterServer* é uma *thread* responsável por aceitar novas conexões de clientes.

O método construtor dessa classe é extremamente simples. É chamado o construtor da superclasse com um nome apropriado para a *thread*, criando-se assim um servidor que fica ativo escutando em uma porta específica, e uma tabela para que sejam processadas as operações de roteamento.

Esta classe, a partir da referencia das conexões *sockets*, abre uma InputStream e uma OutputStream, correspondentes aos canais de entrada e saída. Esses *streams* são buferizados, provendo assim, um eficiente fluxo de entrada e saída e também métodos para a comunicação em alto nível.

A tabela de roteamento criada (*Hashtable routes*) é usada pelos gerenciadores de clientes (*ConexSystemHandler*) para manter um registro dos agentes conectados ao servidor.

7.2.2.2 ConexSystemHandler

A classe *ConexSystemHandler* é responsável pelo tratamento das conexões dos clientes. Quando um cliente se conecta ao servidor, uma nova instância *ConexSystemHandler* é iniciada para processar o cliente (figura 6.3). O cliente, inicialmente, transmite a sua identificação e, então, procede à comunicação.

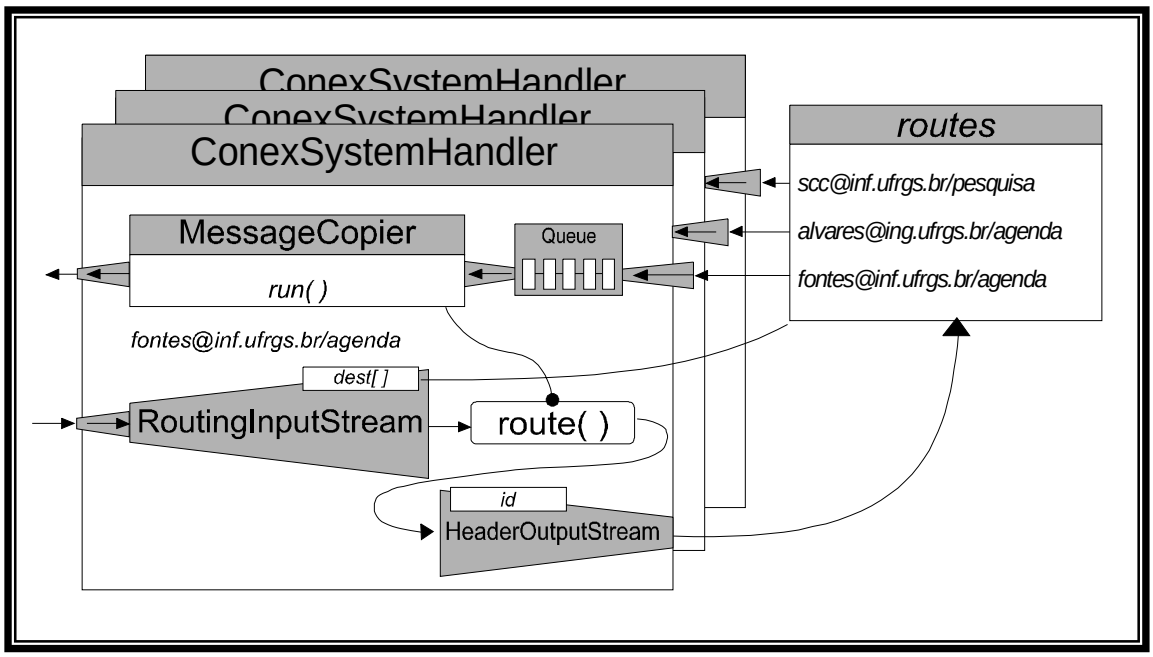

FIGURA 7.3 - Classe ConexSystemHandler

Como primeiro passo na manipulação de conexões, verifica-se a identificação do agente, para ver se ainda não está em uso, não permitindo, assim, acesso simultâneo pelo mesmo agente. Após aceitar a verificação, o agente é efetivamente registrado no servidor.

Uma fila é criada para o roteamento das mensagens destinadas aos clientes. Estes irão escrever mensagens nessa fila, e uma *thread* específica será a responsável pela entrega (roteamento) dessas mensagens para o cliente.

A utilização de uma fila para o roteamento das mensagens deve-se à prevenção de prováveis problemas de rede, isolando o cliente remetente do atual canal de comunicação. Cada cliente é responsável pelo envio das próprias mensagens, da fila para o canal de comunicação. Assim, se ocorrer algum problema na conexão de rede, somente aquele cliente é afetado.

A classe denominada *MessageCopier* é utilizada para retransmitir mensagens da fila para o canal de comunicação e vice-versa, não possibilitando, dessa forma, que os clientes interajam diretamente com o canal

de comunicação, protegendo a si próprios. Nesta classe, é implementada uma simples capacidade de tolerância a falhas, a partir de uma referência do construtor para uma outra *thread* que é responsável, caso ocorrer alguma exceção na retransmissão das mensagens, pela suspensão da *thread ConexSystemHandler*, fazendo com que o roteamento seja finalizado e, então, uma apropriada finalização seja efetuada.

O *route*( ) é um método responsável pelo processamento das mensagens para o cliente. Consiste em determinar se a mensagem é de *broadcast*, para todos os clientes ou de *multicast* para uma lista selecionada. A *RoutingInputStream* é uma classe concatenada ao método *route*( ), que fica constantemente aguardando pela chegada de mensagens dos clientes, para então proceder à verificação da lista de roteamento dessas mensagens.

A identificação do remetente da mensagem é realizada pela classe *HeaderOutpuStream*, que tem como função a adição de um *header* (id), que se constitui da identificação do remetente, à mensagem.

7.2.3 Agente

Como já mencionado anteriormente, o agente é formado por dois módulos: um módulo de aplicação, que é a aplicação específica, e um módulo responsável por toda a comunicação, que é denominado cliente de comunicação.

O cliente de comunicação, representado na figura 6.4, desempenha o papel de intermediador (*interface*) entre a aplicação específica do Agente e o AR (*Agent Router*).

Esta classe cliente abre um canal de comunicação *socket* com o servidor e tenta proceder ao *login* do cliente. Se a conexão é concretizada, então o agente pode ser “assinado” e obter o fluxo do canal para a comunicação. Mensagens de entrada são encaminhadas através da *interface* do cliente. Uma estrutura de dados, em formato de fila, é usada para servir como *buffer*, prevenindo assim problemas em comunicações extensas. Se algum problema acontecer com a comunicação, somente o módulo de transferência de mensagens (*MessageCopier*) será afetado, fazendo, assim, com que o agente possa ser finalizado normalmente.

O cliente é responsável pela assinatura das mensagens, através da classe *HeaderOutputStream*, para obter uma relativa segurança na comunicação ,e também, é de sua responsabilidade a notificação à aplicação específica da quebra de conexão.

A aplicação específica deve ser registrada junto ao cliente, através do método *register*(), dando entrada na tabela de *Hash*, para que somente mensagens entrantes, com a assinatura idêntica à contida na tabela, sejam repassadas para essa aplicação, sendo as outras desprezadas.

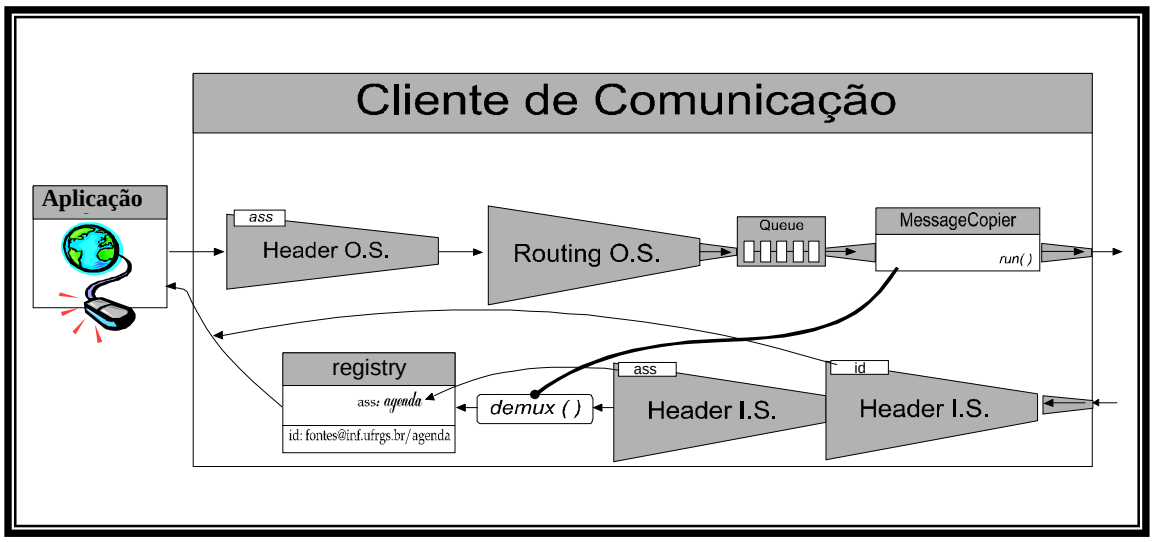

FIGURA 7.4 - Classe Cliente

Um método denominado *connect*(), é utilizado para se fazer a conexão com o AR específico; se o AR não responder a tentativa de conexão, uma mensagem é apresentada ao usuário, informando que o servidor não foi encontrado.

Para o processo de *login* junto ao AR, um método chamado *logon*() é utilizado. Esse método utiliza para isto, a identificação contida na tabela de *Hash* do cliente.

Uma *thread*, através do método *demux*(), é responsável pela decodificação das mensagens vindas do servidor.

7.4 Gerenciamento de agentes no domínio

O gerenciamento de agentes em um domínio dá-se através da inclusão ou exclusão de informações nos arquivos de agentes mantidos pelo ANS e ARs locais.

Para tanto, deve-se fazer uso de um programa em Java (um *frame* [5]), mostrado na figura 7.5.

[5] Os programas para cadastrar agentes e retirar agentes da sociedade são feitos utilizando a linguagem Java. Utiliza-se o termo *frame*, pois estes programas são construídos utilizando uma classe denominada em Java de *frame*.

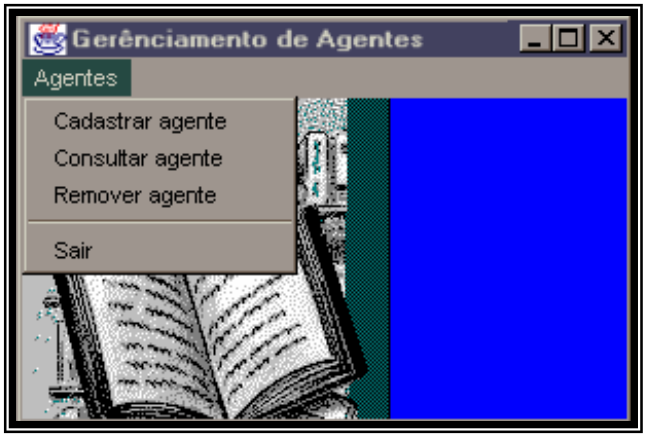

FIGURA 7.5 - Gerenciamento de agentes

7.4.1 Entrada de agentes no domínio

Para formalizar a entrada de um agente no domínio, o usuário deve cadastrá-lo junto ao ANS e ao AR de sua máquina por meio de um programa específico. Ao executar esse programa, serão solicitadas algumas informações sobre o novo agente que está sendo cadastrado. Tendo-se preenchido todos os campos (sem exceção), o programa irá inscrever os dados desse novo agente junto ao arquivo de agentes do AR local e, posteriormente, no arquivo global do domínio junto ao ANS, através de conexões *socket*. Operações de cadastramento concluídas, o agente estará apto a fazer parte do domínio. A figura 7.6 demonstra a *interface* do *frame* para cadastro de agentes.

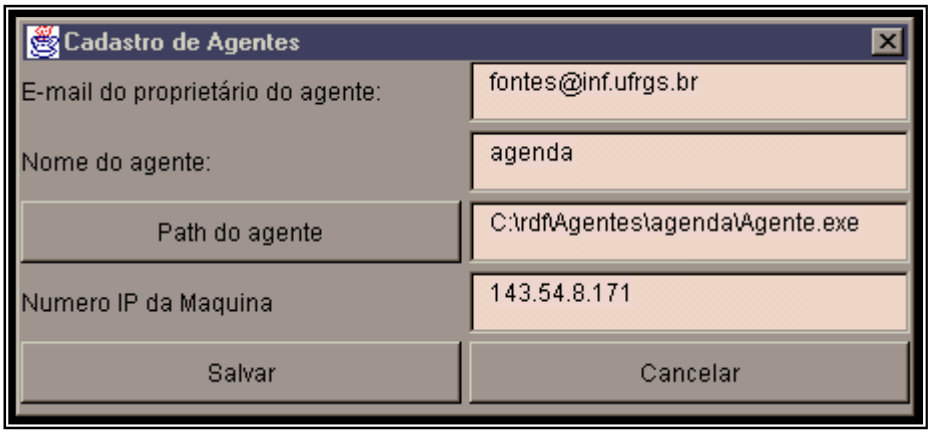

FIGURA 7.6 - Cadastro de agentes

7.4.2 Saída de agentes do domínio

Na retirada do agente do domínio (para desativá-lo), o proprietário deverá também executar o *frame* de gerenciamento, descadastrando, assim, o agente. Esse programa pedirá que sejam fornecidos o *e-mail* do proprietário e o nome do agente. De posse dessas informações, o programa irá eliminar o agente do arquivo de agentes da máquina em questão, e posteriormente fará uma conexão com o ANS e pedirá que este seja retirado do arquivo de agentes do domínio. A figura 7.7 mostra a *interface* do *frame* para retirada de agentes.

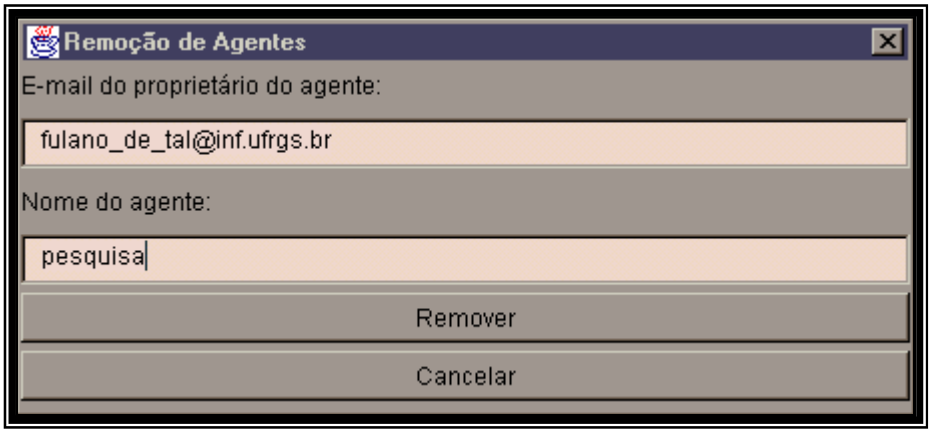

FIGURA 7.7 - Remoção de agentes

8 Comparação com outros modelos

Neste capítulo, expõem-se comparações com outras propostas – SodaBot [COE 94][CAZ 97], JAT (*Java Agent Template*) [JAT 96][CAZ 97] e FIPA 98 [FIP 98] -, desenvolvidas também para dar suporte a implementações de agentes na Internet.

8.1 Modelo especificado x SodaBot

Uma semelhança existente entre o SodaBot e a especificação apresentada é a existência de um processo que fica sempre em estado de *listening*, no caso do SodaBot este é desempenhado pelo BSA (Agente Básico de *Software*), no caso do modelo desenvolvido neste trabalho, este papel é desempenhado pelo AR.

Existem duas diferenças no aspecto desses processos. Uma é que no SodaBot existe um BSA para cada máquina em execução por usuário, no caso do modelo especificado existe um AR responsável por todos os agentes de sua máquina, independente de quem é o usuário "proprietário". A outra é que o BSA quando inativo ativa-se e desativa-se a cada 5 minutos, enquanto que o AR mantém-se sempre ativo.

O SodaBot utiliza o conceito de *time sharing,* enquanto que no ambiente implementado o de processamento paralelo (*threads*).

A linguagem utilizada para a programação de Agentes no SodaBot é uma linguagem própria o "SodaBotL", enquanto que no especificado neste trabalho a linguagem é independente do ambiente, podendo-se utilizar qualquer linguagem para a programação de agentes.

O SodaBot implementa seu próprio esquema de nomes. Cada BSA tem um nome que parece com um endereço de e-mail, mas os nomes de usuários e *host*, não necessitam corresponder a entidades endereçáveis na Internet. Na arquitetura especificada procurou-se aproveitar a designação de entidades já endereçáveis, ou seja, utilizar o e-mail como uma forma de saber a qual domínio de rede pertence um dado agente.

Os BSAs são interconectados via três camadas de servidores dispostos de maneira hierárquica, para que os agentes implementados possam se encontrar. No modelo especificado, existe em cada domínio de rede um ANS contendo um arquivo com dados dos agentes cadastrados no domínio. Em comunicações de agentes pertencentes a domínios distintos, o ANS se

encarrega de encontrar o agente através de iteração com o ANS do domínio ao qual o agente pertence, identificado pelo seu endereço (*e-mail*).

8.2 Modelo x JAT (*Java Agent Template*)

Comparando a arquitetura apresentada neste trabalho com a do JAT v3.0, podemos ter como um ponto a favor a não necessidade de a cada interação, os agentes terem de entrar em contato com o centralizador, pois o sistema pega o endereço contido no ANS e coloca no *cache* do AR solicitante. Fazendo assim com que para a próxima interação com o mesmo agente não seja necessária a solicitação do endereço ao ANS.

Todas as mensagens provenientes de agentes no JAT utilizam o padrão KQML, como um protocolo de alto-nível. Optou-se pela criação de um protocolo baseado nos códigos de *status* do protocolo HTTP. Essa opção teve como objetivos, além de uma maior eficiência, uma especificação clara, concisa e principalmente sem ambigüidades.

O JAT inclui facilidades para troca dinâmica de recursos, as quais podem incluir classes Java, interpretadores, arquivos de dados e informações embutidas nas mensagens KQML.

Uma limitação do JAT, em relação ao modelo aqui especificado, é que ele não implementa o envio de mensagens em *multicast* nem *broadcast*.

Ambas as definições utilizam mensagens distintas para dados e protocolo.

Outro ponto a favor do sistema aqui especificado em relação ao JAT é a não necessidade de se ter mais de uma porta em *listennig* para o caso da existência de múltiplos agentes em uma mesma máquina. Isto é devido ao endereço do agente ser constituído, no JAT, pelo nome do *host* mais o número da porta. Não se sobrecarregando assim a máquina.

8.3 Modelo x especificação FIPA

O modelo aqui especificado e implementado apresenta significativas similaridades com a especificação proposta posteriormente pela *Foundation for Intelligent Physical Agents* – FIPA [FIP 98].

Ambos os modelos utilizam para a localização dos agentes um esquema de endereçamento global único, tendo como composição do endereço tanto o *hostname* quanto o número IP, mais o nome do agente. O sistema construído utiliza o *e-mail* do proprietário do agente, facilitando assim a identificação deste. Não é utilizada na composição do endereço a porta de conexão como no FIPA, pois se utiliza o conceito de padronização de serviços.

Também é utilizado, por ambos, um agente para o controle da comunicação em meta-nível - sistema de roteamento de mensagens. No FIPA este se dá pelo *Agent Communication Channel* (ACC), no sistema especificado, pelo AR.

Para se efetuar uma pesquisa sobre agentes com determinadas características/endereços, ambos os modelos utilizam funcionalidades baseadas em páginas amarelas. No FIPA esta se dá pelo *Directory Facilitator* (DF), no sistema aqui especificado pelo ANS.

As duas definições se preocupam pelo gerenciamento de acessos. No FIPA tal controle é executado pelo *Agent Management System* (AMS), neste pelo AR.

Uma vantagem deste sistema em relação ao do FIPA é a facilidade para incremento de funcionalidades. Como se utilizou para a definição da arquitetura, conceitos adotados na Internet, preocupou-se em implementar um sistema com um certo nível de segurança e modularização.

Outra vantagem apresentada é que na especificação da FIPA é utilizado o conceito *First In First Out* (FIFO), enquanto o ambiente implementado utiliza processamento em paralelo (*threads*).

Não há a especificação de interações entre agentes que estão localizados na mesma máquina pela FIPA.

Outra diferença entre os modelos é que o proposto pela FIPA utiliza dois módulos distintos para a comunicação de agentes, - um para agentes localizados no mesmo domínio e outro para domínios diferentes - enquanto que neste, o mesmo módulo é utilizado.

O sistema aqui realizado utiliza o conceito de encapsulamento das mensagens, tanto para mensagens de protocolo quanto para de informações, enquanto que a especificação da FIPA não, fazendo com que as mensagens fiquem complexas e grandes.

9 Conclusão

Neste capítulo, será mostrada uma visão geral do trabalho realizado, objetivando apresentar as contribuições geradas. Posteriormente, serão comentadas algumas possibilidades de trabalhos futuros.

9.1 Visão geral da dissertação

Os objetivos gerais desta dissertação de mestrado foram o refinamento e implementação da arquitetura para a interação de agentes na Internet - proposta por Cazella [CAZ 97] -, a especificação de um protocolo que tornasse a conexão entre os agentes transparente, facilitando, assim, a implementação de agentes no ambiente da Internet.

O resultado de todo o estudo realizado foi a obtenção de um ambiente, robusto no aspecto de suporte ao agente, oferecendo toda uma infra-estrutura de comunicação, permitindo, dessa forma, o estabelecimento de comunicação direta entre agentes autônomos, construídos com diferentes linguagens de programação e executados em plataformas heterogêneas.

O ambiente apresenta significativas similaridades com a especificação proposta posteriormente pela *Foundation for Intelligente Physical Agents* (FIPA) [FIP 98].

A dissertação foi organizada em nove capítulos básicos, os quais tentaram demonstrar todo o estudo relevante realizado até chegar-se ao modelo.

Inicialmente, foram apresentadas as definições básicas dos termos agente e sistemas multiagentes, constituindo-se em um apanhado geral sobre agentes, no qual são apresentadas definições possíveis, tipos existentes e características principais.

Foram apresentados, a seguir, alguns tópicos estruturais em nível de Internet. Dentre esses tópicos, estão a arquitetura Internet e o protocolo TCP/IP, núcleo da Internet.

O ambiente descrito nos capítulos 4 e 5 surgiu a partir do estudo minucioso da proposta da arquitetura para a coordenação de agentes na Internet [CAZ 97], de alguns outros sistemas que foram tomados como referência para aquela proposta da arquitetura e de protocolos de comunicação mais utilizados na Internet. O modelo proposto foi explicitado em linguagem coloquial e através de diagramas.

Posteriormente, mostrou-se como foi realizada a análise das propriedades estáticas e dinâmicas do protocolo, derivando-se um resultado operacional eficiente e livre de erros.

Expuseram-se detalhes em nível de implementação do ambiente, procurando-se mostrar as estruturas internas básicas dos componentes implementados e suas respectivas funcionalidades.

Finalmente, são apresentadas, comparações com outras propostas desenvolvidas também para dar suporte a implementações de agentes na Internet.

9.2 Contribuições

Este ambiente (arquitetura/protocolo) certamente não é único e, possivelmente, não seja o melhor para algumas classes de aplicações, mas reúne pontos que são de grande peso a seu favor, que são o de facilitar a interação de agentes autônomos em um ambiente altamente dinâmico e heterogêneo, como a Internet, e o de possibilitar a utilização de toda a estrutura já definida para a Internet, como é o caso do DNS e dos domínios já existentes.

O ambiente propicia a implementação de sistemas multiagentes no âmbito da Internet, sem que os desenvolvedores de agentes tenham de se preocupar com uma parte dispendiosa: a conexão entre os agentes para a troca de mensagens.

Em si, o ambiente é uma arquitetura aberta, compondo-se de 9700 linhas de código, podendo ser organizadas sociedades, tanto abertas como fechadas compostas por agentes das mais diversas funcionalidades.

A maior contribuição obtida com este ambiente para a interação de agentes na Internet, acredita-se ter sido, principalmente, na área de Sistemas Multiagentes (SMA), pois, com este ambiente, torna-se mais fácil a implementação de futuros sistemas multiagentes, sem a preocupação da parte de estabelecimento de conexão entre os agentes.

9.3 Limitações

Uma limitação do modelo proposto é a dependência em relação ao ANS. Essa dependência faz com que haja, em caso de pane na máquina *agent*, onde está instalado o ANS, uma não comunicação entre agentes localizados em máquinas diferentes (desde que o AR local não tenha o endereço de destino em *cache*). Essa limitação pode ser minorada com a utilização de tolerância a falhas a partir da replicação do ANS, por exemplo.

Na atual versão do protótipo, implementado em Java devido as características favoráveis como portabilidade, independência de arquitetura, suporte a concorrência (*multithread*), suporte a rede, entre outras, existe a restrição do limite de 50 conexões por porta *socket*, imposta pela versão de Java utilizada.

9.4 Trabalhos Futuros

O presente trabalho pode ser complementado e aperfeiçoado com novos estudos. Dentre estes, podem-se salientar três, muito importantes, que podem ser realizados imediatamente:

1. uma maior preocupação com tolerância a falhas, a partir da análise em nível de segurança e de falhas - não abrangidas por esta dissertação - seguidas da incorporação ao ambiente;

2. ferramentas para a gerência de conexões, destinadas à administração remota das conexões e configurações, possibilitando-se, com estas, um controle do ambiente, por parte de administradores. Estas ferramentas seriam de grande valia se integradas à mecanismos de controle de segurança;

3. avaliação completa do sistema, através de simulações com testes mais amplos, utilizando-se uma grande variedade/quantidade de agentes e uma diversidade de domínios na Internet;

4. implementar na arquitetura suporte a agentes móveis.

Outro trabalho futuro, mais em nível de pesquisa em sistemas multiagentes, seria o de enriquecimento do modelo, de forma a tratar não apenas o aspecto de estabelecer a conexão entre os agentes, mas, realmente, o de facilitar a interação entre eles, provendo, por exemplo, uma tradução entre linguagens de interação diferentes e/ou provendo uma adaptação entre protocolos de interação distintos.

Espera-se que as idéias apresentadas ao longo deste trabalho contribuam para o desenvolvimento de vários outros trabalhos futuros - teóricos e práticos - pela riqueza do tema tratado.

Bibliografia

[AMA 96] AMANDI, Analía A. **Programação Orientada a agentes:** exame de qualificação. Porto Alegre: CPGCC da UFRGS, 1996.

[ARN 96] ARNETT, Matthew F. **Desvendando o TCP/IP**. Rio de Janeiro: Campus, c1996. 543 p.

[BER 92] BERTHET, S. et al. Knowing each other better. In: *INTERNATIONAL WORKSHOP ON DISTRIBUTED* ARTIFICIAL INTELLIGENCE, 11., 1992. **Proceedings...** [S.I.]: Glen Arbor, 1992.

[BIT 96] BITTENCOURT, Guilherme. **Inteligência Artificial-Ferramentas e Teorias**. Campinas: Instituto de Computação, UNICAMP, 1996. Trabalho apresentado na Escola de Computação, 1996, Campinas-SP.

[BOI 92] BOISSER, Oliver; DEMAZEAU, Yves. A Distributed Artificial Intelligence View on General Purpose Vision Systems. In: WERNER, Eric; DEMAZEAU, Yves (Eds.). **Decentralized A.I.3**. Amsterdam: North-Holland, 1992.

[BOR 95] BORGES, M. et al. Suporte por Computador ao Trabalho Cooperativo. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 15., 1995, Canela, BRS. **Anais...** Porto Alegre: Instituto de Informática da UFRGS, 1995.

[CAM 90] CAMPBELL, John A.; D'INVERNO, Mark P. Knowledge Interchange Protocols. In: DEMAZEAU, Yves; MÜLLER, Jean-Pierre (Eds.). **Decentralized A.I.** Amsterdam: Elsevier Science Publishers, 1990. p.63-80.

[CAR 94] CARVALHO, Tereza C. M. de B. **Arquitetura de Redes de Computadores OSI e TCP/IP**. São Paulo: Makron Books, 1994. 669 p.

[CAZ 97] CAZELLA, S. C. **Uma arquitetura para coordenar a interação de agentes na Internet**. Porto Alegre: CPGCC da UFRGS, 1997. Dissertação de mestrado.

[CCI 88] CCITT. Comité Consultatif International de Télégraphique et Téléphonique. **Recommendation Z100**: Specification and Description Language. Genebra: CCITT, 1988. 180p (Blue Book, v10, n.2).

[COE 94] COEN, Michael. **SodaBot**. MIT AI Lab Technical Report 1493, June, 1994. Disponível por WWW em: http://www.ai.mit.edu/people/sodaboat (03 ago. 1998).

[COM 98] COMER, Douglas E. **Interligando em Rede com TCP/IP**. Rio de Janeiro: Campus, 1998. v.1.

[COR 93] CORRÊA, Milton; COELHO, Helder. Around the Architectural Agent Approach to Model Conversations. In: EUROPEAN WORKSHOP ON MODELLING AUTONOMOUS AGENTS IN A MULTIAGENT WORLD, 1993. **Proceedings...** [S.l.: s.n.], 1993.

[DEM 90] DEMAZEAU, Y.; MÜLLER, J. P. Decentralized Artificial Intelligence. In: DEMAZEAU, Yves; MÜLLER, Jean Pierre (Eds.). **Decentralized A.I.** Amsterdam: North-Holland, 1990.

[DEM 91] DEMAZEAU, Y.; MÜLLER, J. P. From Reactive to Intentional Agents. In: DEMAZEAU, Yves; MÜLLER, Jean Pierre (Eds.). **Decentralized A.I.2.** Amsterdam: North-Holland, 1991.

[DEM 92] DEMAZEAU, Y.; SICHMAN, J. S.; BOISSIER, O. When can knowledge-based Systems be Called Agents? In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 12., 1992, Rio de Janeiro. **Anais...** Rio de Janeiro: SBC, 1992. p 172-185.

[DEM 95] DEMAZEAU, Yves. From Interactions to Collective Behaviour in Agent-Based Systems. In: EUROPEAN CONFERENCE ON COGNITIVE SCIENCE, 1995. **Proceedings…** Saint-Malo: [s.n.], 1995.

[EDD 94] EDDINGS, Joshua. **Como funciona a Internet**. São Paulo: Ed. Quakr, 1994.

[FER 91] FERBER, Jacques; JACOPIN, Eric. The Framework of Eco-Problem Solving. In: DEMAZEAU, Yves; MÜLLER, J. P. (Eds.). **Decentralized A.I.2.** Amsterdam: North-Holland, 1991.

[FIN 98] FININ, Tim. **Specification of the KQML**.
Disponível por WWW em: http://www.cs.umbc.edu/kqml/kqml-spec.ps (24 mar. 1998).

[FIP 98] FIPA - FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS. **FIPA 98 Specification**.
Disponível por WWW em: http:/www.fipa.org (22 dez. 1998).

[FRA 96] FRANKLIN, Stan; GRAESSER, Art. **Is it an Agent, or just a Program?**: A Taxonomy for Autonomous Agents. University of Memphis,1996. Disponível por WWW em: http://www.msci.memphis.edu/~franklin/AgentProg.html (15 dez. 1998).

[GAS 93] GASPARINI, A. F. L.; BARRELLA, F. E. **TCP/IP**. São Paulo: Érica, 1993. 309 p.

[GOO 98] GOOSSENS, Michel. **International Character Codes**. CN/ASD. Disponível por WWW em: http://consult.cern.ch/cnl/215/ node45.html (11 nov. 1998).

[HEI 95] HEILMANN, Kathryn et al. **Intelligent Agents: A Technology and Business Aplication Analysis**. November 1995. Disponível por WWW em: http://haas.berkeley.edu/ ~heilmann/agents/ index.html (05 out. 1998).

[HÜB 95] HÜBNER, Jomi F. **Migração de Agentes em Sistemas Multiagentes Abertos.** Porto Alegre: CPGCC - UFRGS, 1995.

[JAT 96] Java Agent Tamplate.
Disponível por WWW em: http:/cdr.stanford.edu/ABE/ JavaAgent.html (24 mai. 1998).

[MCC 99] MCCARTHY, John. **Elephant 2000: A Programming Language Based on Speech Acts**. Stanford University. Disponível por WWW em: http://www-formal.stanford.edu/jmc/elephant/ elephant.html (15 jan. 1999).

[NOE 94] NOERR, Peter L. Character set and UNICODE. **Koninklike Bibliotheek Albert I**. Brussel, December, 1994. Disponível por WWW em: http://www.ua.ac.be/KB/pn/pnoerr0.html (11 nov. 1998).

[NOG 91] NOGUEIRA, J. M. S. **Protocolos e Serviços de Comunicação**: Princípios, Especificação e Teste. Belo Horizonte: UFMG-DCC, 1991.

[OLI 96] OLIVEIRA, Flávio M. Inteligência Artificial Distribuída. In: ESCOLA REGIONAL DE INFORMÁTICA, 4., 1996, Canoas-RS; Itajaí-SC; Londrina-PR. **Anais...** Canoas:SBC, 1996.

[ROT 84] ROTH, B. Hayes. **A blackboard Model of control.** Stanford, CA: Stanford University, 1984. (Techincal Report HPP83-38).

[SEA 69] SEARLE, J. R. **Speech Acts**. [S.I.]: Cambridge University Press, 1969.

[SIC 92] SICHMAN, Jaime; DEMAZEAU, Yves; BOISSER, Oliver. When can Knowledge-Based Systems be called Agents? In: BRAZILIAN SYMPOSION ON ARTIFICIAL INTELLIGENCE, 9., 1992. **Proceedings...** Rio de Janeiro: [s.n.], 1992.

[SIC 93] SICHMAN, J. S.;DEMAZEAU, Y. Using Class Hierarchies to Implement Social Reasoning in MultiÁgent Systems. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 11., 1994, Fortaleza. **Anais...** Fortaleza: [s.n.], 1993.

[SIC 95] SICHMAN, J.S. **Du Raisonnement Social chez les Agents**. Grenoble: Institut National Polytechnique de Grenoble, 1995. Thesé de Doctorate en Informatique.

[STE 94] STEVENS, W. R.; WRIGHT, G. R. **TCP/IP Illustrated:** the protocols**.** [S.I.]: Addison-Wesley, 1994. v.1.

[STE 95] STEFANINI, M.-Hélène; DEMAZEAU, Yves. TALISMAN: A Multi-Agent System for Natural Language Processing. In: BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 12., 1995. **Proceedings...** Berlim: Springer Verlag, 1995. p. 312 - 322 (Lecture Notes in Artificial Intelligence,v. 991).

[VIR 95] VIRDHAGRISWARAN, Sankar. **Mobile Unstructured Business Object Technology**, 1995. Disponível por WWW em: http://www.crystaliz.com/logicware/mubot.html (02 out 1998).

[TAN 92] TANENBAUM, Andrew S. **Modern Operating Systems**. Englewood Cliffs, N.J.: Prentice-Hall, 1992. p. 242-243.

[TAN 97] TANENBAUM, Andrew S. **Redes de Computadores**. 3. ed. Rio de Janeiro: Campus, c1997. 923 p.

[TRI 91] TRINDADE, Rodrigo. S. **Estudo Comparativo das Linguagens de Especificação e Teste de Protocolos ESTELLE, LOTOS, SDL e CHILL**. Porto Alegre: CPGCC da UFRGS, 1991. 25 p.

[TRI 94] TRINDADE, Rodrigo. S. **Uma Ferramenta para Análise de Protocolos Especificados em SDL**. Porto Alegre: CPGCC da UFRGS, 1994. Dissertação de mestrado.

[YER 98] YERGEAU, F. **Request for Comments: 2044**. Network Working Group. Category: Informational, October 1996. Disponível por WWW em: http://www.faqs.org/rfcs/rfc2044.html (26 jul. 1998).

[WER 87] WERNER, Eric. Cooperating Agents: A unified Theory of Communication and Social Structure. In: GASSER, Les; HUHNS, Michael N. (Eds.). **Distributed Artificial Intelligence II**. [S.l.]: Morgan Kaufmann, 1989. p.3-36.