GERENCIAMENTO DE REDES DE COMPUTADORES UTILIZANDO INTELIGÊNCIA ARTIFICIAL DISTRIBUÍDA — UMA ABORDAGEM OPERACIONAL

Fernando Luiz Koch, MSc.
koch@lrg.ufsc.br

Carlos Becker Westphall, PhD.
westphal@lrg.ufsc.br

Universidade Federal de Santa Catarina - UFSC
Centro de Tecnologia e Computação - CTC
Laboratório de Redes e Gerência - LRG
CEP : 88.040-900 Florianópolis SC Brasil

Resumo

As abordagens centralizadas para o Gerenciamento de Redes de Computadores têm demostrado uma clara inadequação para o gerenciamento eficiente de redes de computadores com grande número de nós ou com grande diversidade de equipamentos. Existe, no meio comercial e acadêmico, um enorme rol de pesquisas sendo realizadas objetivando abordagens descentralizadas para o problema do gerenciamento. Este trabalho apresenta as pesquisas sobre o uso de sistemas de Inteligência Artificial Distribuídos aplicados ao gerenciamento de redes de computadores. O objetivo final será a definição e implementação de um conjunto de aplicações distribuídas, denominadas *agentes autônomos*, com características de integração e adaptabilidade para ambiente de gerenciamento de redes heterogêneos e integrados ao ambiente de gerenciamento SNMP.

Abstract

Centralized approaches for Computer Network Management have demonstrated a clear inadequacy for efficient management of large and/or heterogeneous computer networks. Many researches are being carried out on decentralized approaches for the management problem. This work presents the researches on the Distributed Artificial Intelligence area applied for computer network management. The final goal is define and implement a set of applications called autonomous agents *with adaptability features for heterogeneous computer network management environments, integrated with the SNMP management systems.*

1. INTRODUÇÃO

O gerenciamento de redes de computadores esta se tornando uma atividade cada vez mais complexa [FEI95]. As soluções amplamente utilizadas no meio comercial são baseadas no uso de estruturas *centralizadas*, onde as aplicações são separadas das origens dos dados e dos dispositivos que serão controlados. Uma abordagem alternativa é a *hierárquica*, onde existe o conceito de "Gerente de Gerentes" e gerenciamento por domínio [KAH97]. Por fim, uma outra alternativa é a arquitetura *distribuída*, que é a proposta deste trabalho.

O conceito que será utilizado é o do *Gerenciamento Indireto*, que é um estilo complementar de interação implementado com o uso de novas técnicas do campo da Inteligência Artificial, em particular dos chamados "agentes autônomos" [KAY90][MAE94b]. Ao invés de uma interação iniciada pelo usuário, via comandos ou manipulação direta, ele é engajado num processo cooperativo em que homem e agentes de computador iniciam as comunicações conjuntamente, monitoram eventos e realizam tarefas.

A metáfora usada é do *assistente pessoal* que está colaborando com o usuário no mesmo ambiente de trabalho. Este assistente torna-se gradualmente mais eficiente à medida que

aprende com os interesses dos usuários, seus hábitos e preferências e, também, com o comportamento do ambiente e comunidade na qual está inserido.

A idéia de empregar agentes em ambientes computadorizados para delegar algumas tarefas baseadas em computador, foi introduzida por visionários como Nicholas Negroponte [NEG70] e Alan Kay [KAY84]. Mais recentemente, muitos fabricantes de computadores têm adotado esta idéia para ilustrar sua visão de programação e interface do futuro, especialmente em ambiente de redes de computadores, como pesquisas de empresas renomeadas como a *Hewlett Packard* e a *Sun Microsystems*.

1.1 OBJETIVOS

O objetivo final deste trabalho é criar um modelo para implementação de agentes autônomos, que permita um alto grau de adaptabilidade, que tenha conceitos de reusabilidade de módulos e, por fim, que agilize ao máximo a tarefa de criação de uma comunidade de agentes para a gerência de redes de computadores.

As características desejadas da comunidade são :

1. Os agentes devem ser *autônomos*. Ou seja, um agente deve ter controle sobre os seus atos. Isto é desejado, pois quando um usuário delega alguma tarefa para o seu agente, espera que ele tenha independência para trabalhar sobre a sua requisição, não importa o que esteja acontecendo [GIL97];
2. os agentes devem ser *orientados a tarefas*;
3. os agentes devem *ter uma base de conhecimentos* que contenha todas as informações sobre como realizar determinados objetivos na forma de regras de produção. As rotinas básicas para a execução de um objetivo devem ser programadas no agente como regras fundamentais [KAU94];
4. os agentes devem ser *capazes de aprender novas regras de produção* e armazená-las em sua base de conhecimento sempre que forem necessárias para a realização de determinada tarefa. Este mecanismo de aprendizagem de novos conhecimentos implementará as caraterísticas adaptativas e de aumento da eficiência propostos pela Inteligência Artificial Distribuída;
5. os agentes devem coletar dados dos dispositivos gerenciados através do *protocolo SNMP*;
6. deve ser possível a *análise dos dados coletados*, através da consolidação em relatórios gerenciais, por agentes especializados nesta tarefa;
7. os agentes devem *trabalhar como uma coletividade*, através dos processo de comunicação e cooperação entre eles. Ou seja, agentes são *sociais* e comunicam-se por meios padronizados, tal como a linguagem de comunicação KQML (*Knowledge Query and Manipulation Language*) [LAB96];
8. deve ser criada uma *estrutura genérica*, que permita a rápida definição de um novo agente, com as caraterísticas descritas, com total reusabilidade, tornando o sistema prático e facilmente configurável para novos objetivos do sistema de gerenciamento;
9. o sistema deve prover uma *interface amigável*, segura e acessível aos usuários e gerentes da rede, através de aplicações para o ambiente Internet/Intranet, que permita interação, criação de agentes e visualização de resultados através de páginas HTML comerciais.

2. AGENTES AUTONOMOS

Os agentes autônomos, ou agentes de software ("*software agents*"), ou agentes inteligentes, dentre as diversas nomenclaturas dadas para o componente básico da área de pesquisa da Inteligência Artificial Distribuída (IAD), são uma tecnologia computacional relativamente nova e ainda não bem definida. A introdução desta tecnologia no ambiente de gerência de redes de computadores trás uma série de vantagens e novas características, tais como : diminuição do tráfego entre o agente e o nó gerenciável, maior abstração dos objetos gerenciáveis pelos gerentes, maior agilidade na tomada de decisões e maior adaptabilidade do sistema.

Um agente autônomo é um software que auxilia as pessoas e agem em seu favor. Estes agentes funcionam permitindo que os usuários deleguem trabalhos que eles gostariam de ter prontos automaticamente. Os agentes podem, tal como os assistentes, automatizar tarefas repetitivas, relembrar coisas que o usuário esqueceu, sintetizar inteligentemente os dados complexos, aprender a partir do usuário e, até mesmo, fazer recomendações [GIL97].

O objetivo final de criar uma comunidade de agentes para o ambiente de gerenciamento de redes de computadores é automatizar o máximo possível a esta tarefa. O sistema automático precisa ter características de adaptabilidade e se ajustar a situações especiais, que podem ser obtidas até mesmo pelo mais simples sistema de Inteligência Artificial Distribuída. Estes sistemas são extremamente úteis em modelos onde o problema é imprevisível.

2.1 ARQUITETURA GENÉRICA PARA AGENTES

O objetivo da criação do conceito de *Agente Genérico* é definir as estruturas, mecanismos e habilidades básicas, que garantem um comportamento mínimo, interno e externo de qualquer sistema agente. Os agentes reais, ou seja, aqueles operacionais, serão produzidos por "clonagem" de agentes genéricos e reprogramação de seus objetivos.

As características discutidas até o momento, podem ser transformadas quase unicamente em um estrutura composta por três módulos, conforme dito em [BUR92a] e mostrado na Figura 1 :

Figura 1 - Estrutura do *Agente Genérico*

1. *COGNIÇÃO*, responsável pelas ações cognitivas do agente;
2. *INTERAÇÃO*, responsável pelas interações de entrada e saída de dados com o ambiente;

3. *COMUNICAÇÃO,* responsável pela comunicação entre os vários tipos de agentes em uma comunidade, num ambiente de agentes distribuídos;

Para proporcionar as características de *sociabilidade* para os agentes, estes devem interagir ou comunicar com outros agentes. Esta comunicação poderia se feita por meios proprietários ou por modelos de comunicação padrão como a linguagem KQML (*Knowledge Query Manipulation Language*), da ARPA [FIN95]. O sistema constituído inteiramente por agentes que se comunicam e cooperam pode ser chamado um *Sistema Multi-Agente* [CAS92][FER92].

O problema da comunicação entre agente não é um problema *bem definido*. A maioria dos autores que apresentam soluções ou referem problemas nesta área o fazem para resolver problemas específicos de seus trabalhos sem se preocupar com o problema global. Talvez, por causa destas soluções isoladas não exista um consenso ou padrão para o problema da comunicação entre agentes. Porém, os esforços feitos por grupos como o ARPA, com o KQML, ou pesquisadores, como [DUR90], com o *Contract-Net* estão caminhando para se tornar soluções de aceitação padrão.

Deve constar, por fim, a possibilidade de uso, agora em fase de pesquisa por este grupo, de soluções hoje em destaque no mercado e centros de pesquisa, como o sistema de comunicação para objetos distribuídos, como, por exemplo, o CORBA. Salientamos que é reconhecida suas características úteis para o ambiente de comunicação entre agentes, porém estas soluções encontram-se, agora, em fase de avaliação.

3. GERENCIAMENTO DE REDES BASEADO EM AGENTES AUTONOMOS

Um sistema de gerenciamento de redes de computadores usual tem um fluxo de informações como o apresentado na Figura 2. Os dados são coletados do dispositivo gerenciado através de um protocolo de gerenciamento qualquer (SNMP, CMIP ou proprietário), são analisados e consolidados em informações gerenciais a partir das quais serão decididas e executadas as ações de gerenciamento a serem efetuadas sobre o dispositivo, se necessárias.

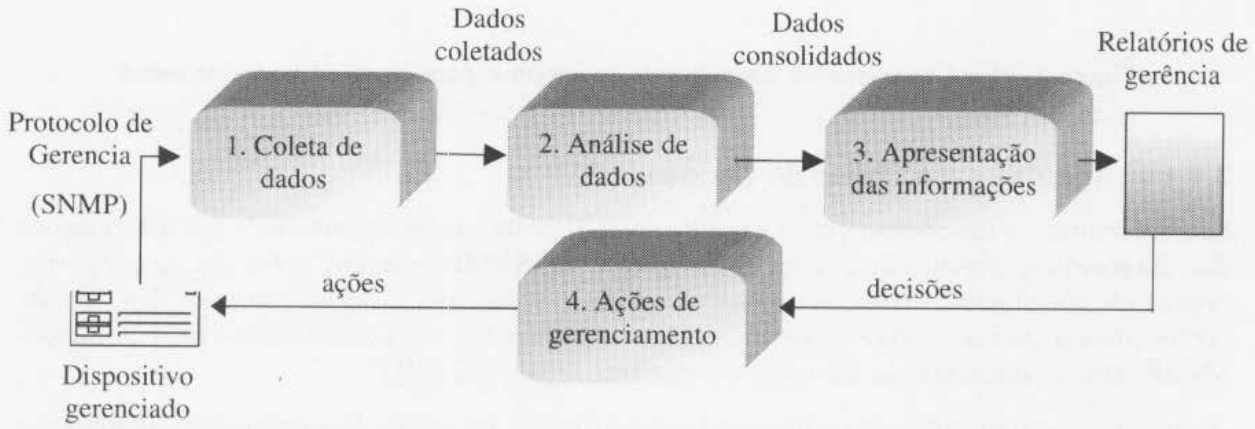

Figura 2 - Fluxo de informações em um sistema de gerenciamento

Em um sistema usual de gerenciamento de redes o gerente humano tem uma participação de preparação e supervisão na tarefa (1) e é o elemento principal na execução das tarefas (2), (3) e (4), cabendo a ele a análise e consolidação dos dados formando os relatórios gerenciais e a responsabilidade completa sobre da decisão das ações de gerenciamento a serem executadas. Eventualmente, pode haver algum tipo de sistema de ajuda computadorizado, mas a intenção deve ser do gerente da rede.

3.1.1 Tipos de Agentes Autônomos

Da adaptação do fluxo de informações apresentado na Figura 2 para a tecnologia de inteligência artificial distribuída, foi feita uma divisão de tarefas com a especialização de três tipos distintos de agentes : (1) *agentes para coleta de dados*,(2) *agentes para gerenciamento* e (3) *agentes para interface*. Estas classes de agentes serão explicadas a seguir.

Cabe ressaltar que todos os agentes, de qualquer uma das classes, segue o padrão de *Agente Genérico*, especificado em *2.1 - Arquitetura Genérica para Agentes*.

A Figura 3 apresenta o fluxo de dados na comunidade de agentes para gerenciamento de redes, onde o papel do agente de coleta de dados aparece como sendo a interface entre os dispositivos gerenciados e a comunidade de agentes.

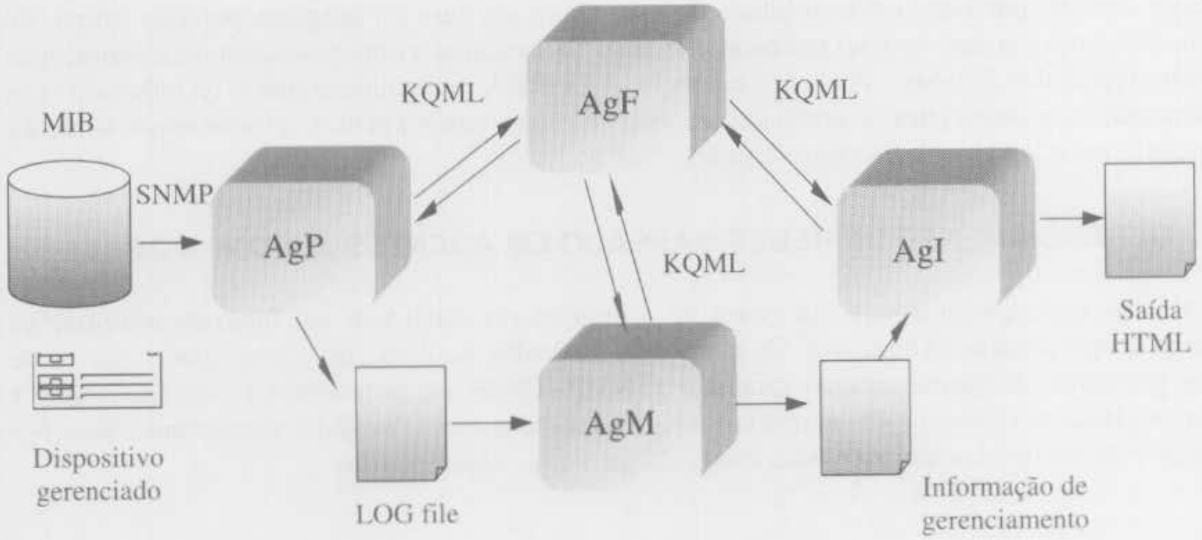

Figura 3 - Fluxo de dados na comunidade de agentes para gerenciamento de redes

3.1.1.1 Agentes para Coleta de Dados (AgP)

São denominados de *agentes probes* (AgP) neste trabalho, estes agentes coletam informações dos dispositivos gerenciados, através do protocolo SNMP e agem sobre os dispositivos, orientados por objetivos internos ou ordens oriundas de *agentes de gerenciamento*. Na base de conhecimento destes agentes existem regras especializadas em coletar dados pelo protocolo SNMP (através das diretivas *get-request* e *get-next-reques* [FEI95]).

As informações coletadas são armazenadas em arquivos de *log* de funcionamento, juntamente com o indicador do tempo em que foi coletada, o *OID* da MIB do atributo coletado, o endereço IP do dispositivo gerenciado e o valor coletado. Este arquivo de *log* servirá para a análise posterior. Um único agente *probe* pode ter vários objetivos diferentes de coleta e interação. Estes objetivos são atribuídos no momento da criação do agente ou durante a sua operação, podendo ser alteradas e retiradas a qualquer momento através de mensagens pelo sistema de comunicação. Um exemplo da estrutura de um arquivo de *log* pode ser visto na Figura 4.

Cada regra tem um temporizador associado a ela, o qual indica o intervalo de tempo e quantas vezes ela deve ser executada. Este intervalo pode assumir um valor de 0 (imediatamente) até várias horas e de uma única vez até infinitas vezes. Assim, podemos dar como objetivo para um agente coletar um valor de determinado atributo na árvore da MIB de um dispositivo de um endereço IP, a cada 5 minutos, ou coletar o valor da variável *sysDescr*, da MIB do dispositivo

gerenciado de todos os dispositivos entre os endereços IP *x* e *y*, apenas uma vez, ou ambos os objetivos.

Na resolução deste objetivo, o agente pode se deparar com um sub-objetivo (regra de produção) desconhecido. Neste caso, sua atuação é questionar à comunidade sobre este objetivo e armazenar a resposta na base de conhecimento. Assim, o agente aprende um novo conhecimento sempre que necessário, dando a característica de adaptação do sistema autônomo.

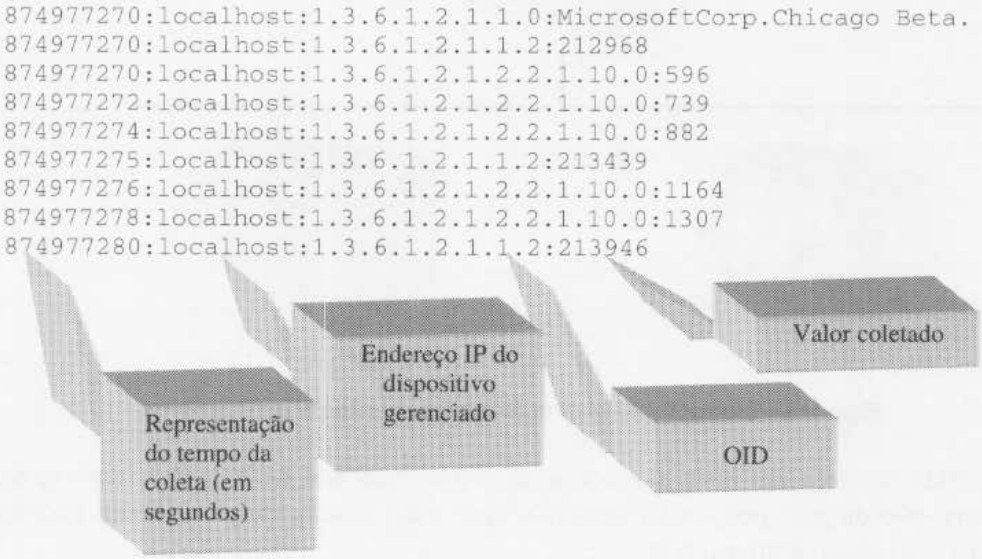

Figura 4 - Estrutura de um arquivo de LOG

3.1.1.2 Agentes para gerenciamento (AgM, AgF)

Os *agentes para gerenciamento* (chamados de *AgM*) são responsáveis pelas tarefas de coordenação das interoperações dentro da comunidade, pela análise e consolidação dos dados coletados pelos agentes e pelo gerenciamento em geral.

Agentes facilitadores de comunicação

Um tipo especial de agente de gerenciamento necessário para a existência de uma comunidade é o *agente facilitador de comunicação* (AgF). Ele é responsável pela coordenação do processo de comunicação de dados e troca de mensagens entre os agentes, sendo um gerenciador de quadro-negro quando da necessidade de troca de informações entre agentes.

Estes tipos de agentes perfazem vários serviços de comunicação úteis, como por exemplo, manter o registro dos serviços de nomes de agentes, rotear mensagens baseado em seu conteúdo, prover mediação e serviço de tradução [FIN95].

A Figura 5 apresenta um exemplo de comunicação entre um agente A e outro agente B, através de um agente facilitador F. O agente A pode solicitar ao agente F que monitore as mudanças na sua base de conhecimento. O agente B informa um novo conhecimento a F que repassa, de imediato, para o agente A.

Agentes para análise de dados

Os *agentes de gerenciamento para análise de dados* são responsáveis pela consolidação dos dados coletados na forma de informações gerenciais. Estes agentes são criados e programados para cruzar dados de diferentes dispositivos e tomar decisões que são apenas possíveis quando se tem um visão mais ampla sobre o funcionamento do sistema que está sendo gerenciado.

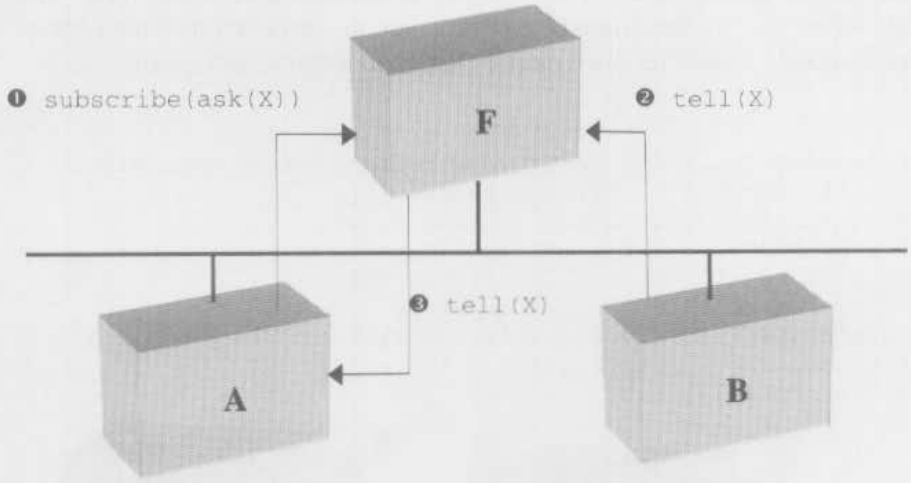

Figura 5 - Exemplo de interação entre agentes e facilitadores

Para fazer estas análises em sistemas aonde as regras não são bem conhecidas ou definidas, criou-se uma abordagem por *redes neurais*, que são usadas para resolver problemas de *classificação* e *predição* entre outros.

Assim, através de um sistema neural, pode-se treinar agentes de gerenciamento para reconhecer padrões de comportamento de dispositivos e predizer valores futuros que estes possam atingir, tendo subsídios para tentar evita-los, entre outras possibilidades que um analisador neural pode oferecer.

A abordagem da implementação de agentes de gerenciamento por redes neurais foi inicialmente deixada em segundo plano devido ao grande tempo e capacidade de processamento desperdiçadas na tarefa de seu treinamento. Porém, os resultados teoricamente atingíveis com o uso destes sistemas são incomparáveis e o estudo aprofundado de seu uso é recomendado.

Alguns exemplos de agente de gerenciamento :

- *para analisar dados* : analisa os dados provenientes de vários agentes de coletas de dados (através de seus arquivos de *log*), sobre o tráfego de pacotes IP nas interfaces de dispositivos concentradores (*hubs*) e verifica quais os pontos de acesso à rede que têm maior volume de informações e em quais tempos. Assim, pode-se programar o sistema para fornecer maior largura de banda (se possível) para os usuários que utilizam o recurso com maior freqüência em determinado horário.
- *para consolidar dados* : agrupa os dados provenientes de um agente coletor, consolidando-os em amostras diárias com agrupamento horário. Facilita a conclusão de quais os horários de maior utilização da rede e permite criar gráficos de barra diários que ilustrem estas estatísticas.
- *para prever valores futuros* : através do uso de redes neurais para predição de séries temporais, é possível criar agentes de gerenciamento que analisam os dados coletados e prevêem os valores futuros. Assim, caso estes valores previstos indiquem a quebra de determinado limite máximo, o sistema pode tomar uma ação para evitar algum problema maior.
- *para classificar valores* : também através do uso de redes neurais, permite ao sistema interpretar um dado de tráfego, por exemplo, como ALTO, NORMAL ou BAIXO, através da classificação do conjunto de valores históricos analisados

3.1.1.3 Agentes de Interface (AgI)

Os agentes de Interface (AgI) são responsáveis por criar a interface entre o gerente humano e a comunidade de agentes. Através destes agentes o usuário pode criar novos agentes de coletas de dados ou modificar sua tabela de objetivos, programar novos relatórios de consolidação de dados para agentes de interface, ter informações sobre o comportamento e atuação dos agentes da comunidade e obter dados gerenciais em forma gráfica ou tabular.

A interface é feita através de linguagem HTML. Isto torna possível a interação com uma comunidade através de qualquer paginador de Internet/Intranet convencional, em qualquer ponto de acesso à Internet ou Intranet, tornando o sistema amigável e facilmente acessível.

Um exemplo de página HTML gerada por um agente Interface pode ser visto na Figura 6. Nele é apresentado um gráfico de barras com o número de acessos a um servidor HTTP, com levantamento diário e agrupamento horário. São apresentados, também, valores numéricos dos dados obtidos.

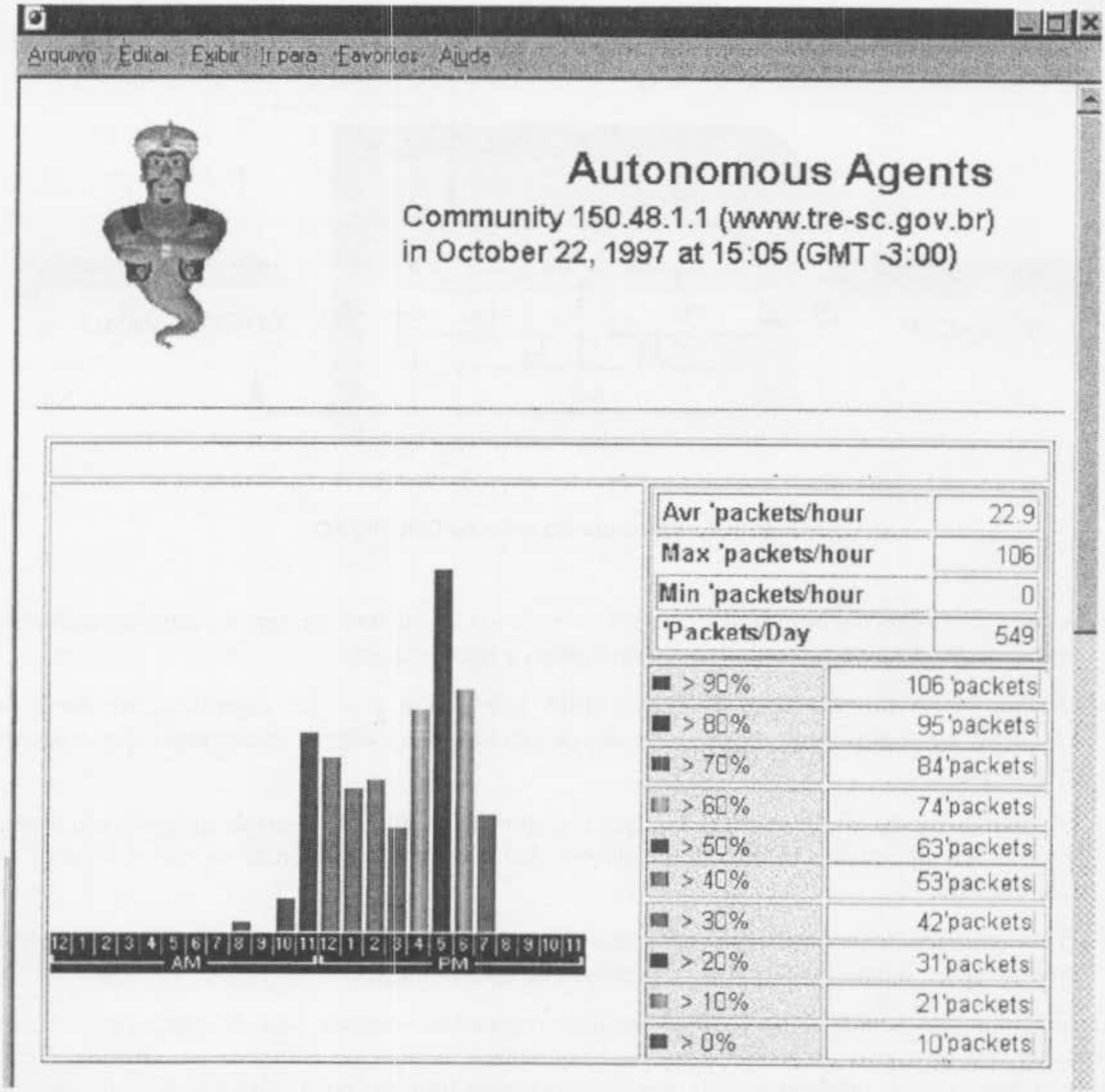

Figura 6 - Exemplo de saída de agente interface

4. IMPLEMENTAÇÃO

Esta seção apresentará as estruturas de dados básicas e o código fonte para a implementação do código crítico de execução de um agente autônomo. Como a listagem de código fonte de um agente chega a atingir 40.000 linhas de código C, torna-se impossível apresentá-la e comentá-la por inteiro.

O projeto de implementação do núcleo de vida de qualquer tipo de agente é o mesmo. De fato, o que varia entre um agente de coleta de dados e um agente gerente são os objetivos de vida e a biblioteca de funções básicas de vida com as quais os sistemas são criados.

Os agentes herdam em suas implementações similaridades nas estruturas de dados, no sistema de ciclo de vida, na máquina de inferência, no módulo de aprendizado de novas regras e no sistema de comunicação.

4.1 MÓDULO DE COGNIÇÃO E ESTRUTURAS DE DADOS

Os sub-componentes do módulo COGNIÇÃO são descritos a seguir e apresentados na Figura 7.

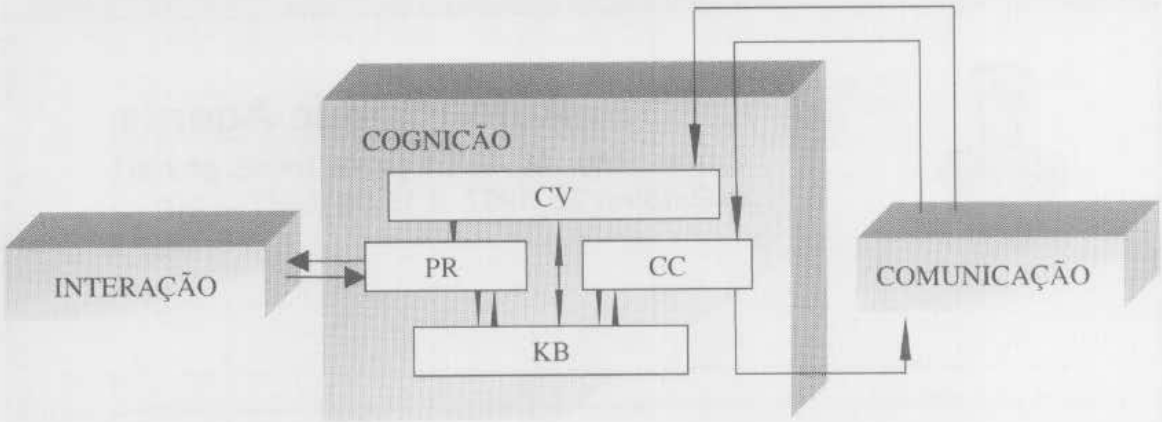

Figura 7 - Estrutura do módulo COGNIÇÃO

- A *base de conhecimentos* (KB), contém os recursos cognitivos do agente, representados por regras de produção e por objetivos permanentes e temporários;
- O *componente de resolução de regras* (PR), que perfaz as ações cognitivas por meio de execução das regras de produção, que são designadas pelos objetivos correntes que o agente precisa atingir;
- O *componente de cooperação* (CC), que é responsável pelo processo de aprendizado e pela troca de conhecimento (regras de produção, dados e objetivos) dentre os vários agentes da comunidade;
- O *componente de ciclo de vida* (CV), que prepara a lista de objetivos imediatos que o agente deve atingir e reage (ou estimula) o processo de comunicação;

Todo agente tem uma *base de conhecimento de regras* (ou somente base de regras) onde ficam armazenadas as regras de produção que implementam as metas e submetas do sistema. Nesta base de regras estão também as entradas para as regras básicas (ou fundamentais) programadas no momento da criação dos agentes. Justamente o conjunto estas regras básicas fazem a diferença entre os agentes do tipo coletor de dados e gerentes. O conteúdo desta base de

conhecimento pode ser algo como mostrado na Figura 8. Os valores que apontam para endereços físicos, são conhecimentos básicos do agente, atribuídos no momento da sua criação.

```
seta(%fato,%valor)->  4287:0398.

pega(%fato,?valor)->  4287:0404.

getrequest(%host,%var,?valor)->  443F:06AA.

getnextrequest(%host,%var,?valor)->  443F:06F2.

salvaSNMP(%host,%var,%str)->  443F:004A.

coletaSNMP(%host,%SNMPvar)->
   getnextrequest(%host,%SNMPvar,?valor)

salvaSNMP(%host,%SNMPvar,?va
```

Figura 8 - Conteúdo de uma base de regras

Além desta estrutura, existe uma *base de fatos*, onde ficam armazenados os valores de fatos apurados pelo sistema em sua vida, através da execução de regras que os atribuem (exemplo de base de fatos apresentada na Figura 9). Contudo, alguns fatos são atribuídos nesta estrutura no momento da criação do agente, como por exemplo o seu identificador único (nome), o tempo de início da execução e um fato chamado VIVER que enquanto tiver o valor 1 mantém o agente vivo. Para "matar" o agente, basta criar uma regra que atribua o valor "0" para o fato viver, como : `matar():- seta("VIVER",0)`, para o agente com a base de conhecimento acima apresentada.

```
coletaSNMP("localhost","1.3.6.1
.2.1.1.0"),1,1
coletaSNMP("localhost","1.3.6.1
```

Figura 9 - Exemplo de base de objetivos permanente

O primeiro objetivo da base apresentada na Figura 9 diz para o agente coletar na máquina local, o conteúdo da variável MIB referenciada pelo OID 1.3.6.1.2.1.1.0 (*sysDescr*), através de uma instrução *get-next-request*, a cada 1 segundo e apenas uma vez. A execução desta produção também salvará o valor coletado no arquivo e LOG, pela regra `salvaSNMP(%host,%SNMPvar,?valor)`, de *coletaSNMP()*. Este objetivo pega o valor da variável *sysDescr* da árvore MIB do dispositivo gerenciado, que contém uma descrição textual que deve incluir informação sobre o hardware, sistema operacional e software de rede.

O segundo objetivo diz para o agente coletar na máquina local o conteúdo da variável MIB referenciada pelo OID 1.3.6.1.2.1.1.2 (*sysUpTime*). Observe-se que, como o método de coleta é uma chamada *get-next-request*, o valor do OID da variável lida é sempre um a mais do que

aquele especificado numericamente. Este valor será coletado a cada 5 segundos e durante todo o tempo de vida do agente (-1).

Por fim, o terceiro objetivo diz para o agente coletar na máquina local o conteúdo da variável MIB referenciada pelo OID 1.3.6.1.2.1.2.2.1.10.0 (*ifInOctets[0]*). Considere a mesma observação do item anterior sobre o valor do OID quando do uso de um pacote *get-next-request*. Este objetivo é executado a cada 2 segundos, durante todo o tempo de vida do agente.

O arquivo de log resultante pelo agentes com esta base de conhecimentos e objetivos é aquele apresentado na Figura 4.

4.1.1 Componente de resolução de regras (PR)

O componente de resolução de regras implementa a máquina de inferência, que, em última analise, é o cérebro do agente autônomo. Ela é responsável por, dado um objetivo para resolver, levantar e executar todos os seus sub-objetivos, recursivamente, até todas as regras básicas relacionadas tiverem sido executadas. Ou seja, no sistema recursivo que foi criado para a implementação do desenvolvedor de regras, o ponto de corte (limite) ocorre quando se atingem as regras básicas.

Normalmente, estas regras de produção são construídas sobre instruções que coletam dados procedentes do sistema SENSOR, manipulam estes dados e geram saídas para serem armazenadas ou reações para o sistema ATUADOR. No caso de agentes para o gerenciamento de redes, os sistemas SENSOR e ATUADOR são construídos sobre rotinas que coletam ou atribuem valores através do protocolo SNMP ou analisam arquivos de LOG gerados por outros sistemas (por exemplo, servidores HTTP, LOG de acesso dos usuários do UNIX entre outros).

O funcionamento do Componente de resolução de regras (PR) consiste em estabelecer a regra de produção a ser executada e submete-la para a máquina de produção. Quando uma regra desconhecida é submetida para execução, o PR ordena ao Componente de Cooperação questione a comunidade de agentes pelas sub-produções que compõe a nova regra [BUR92a].

O fluxo de dados do Componente de Resolução de Regras (PR) pode ser visto na Figura 10.

4.2 INTEGRAÇÃO COM SISTEMAS SNMP

As regras básicas para a integração como o protocolo SNMP formam uma biblioteca de, aproximadamente, 10.000 linha de código C e não poderia ser apresentada integralmente neste trabalho. Porém, sua utilização é muito fácil , graças a sua interface muito simplificada. A implementação é a mesma para os pacotes *get-request* e *get-next-request*, alterando apenas o pacote enviado pela função *SNMPsend()*.

A Figura 11 apresenta os *prototypes* das funções escritas em C que implementam a biblioteca de integração com o protocolo SNMP, usada pelos agentes coletores da dados para recolher e salvar dados dos dispositivos gerenciados. O código original foi escrito com comentários na língua inglesa.

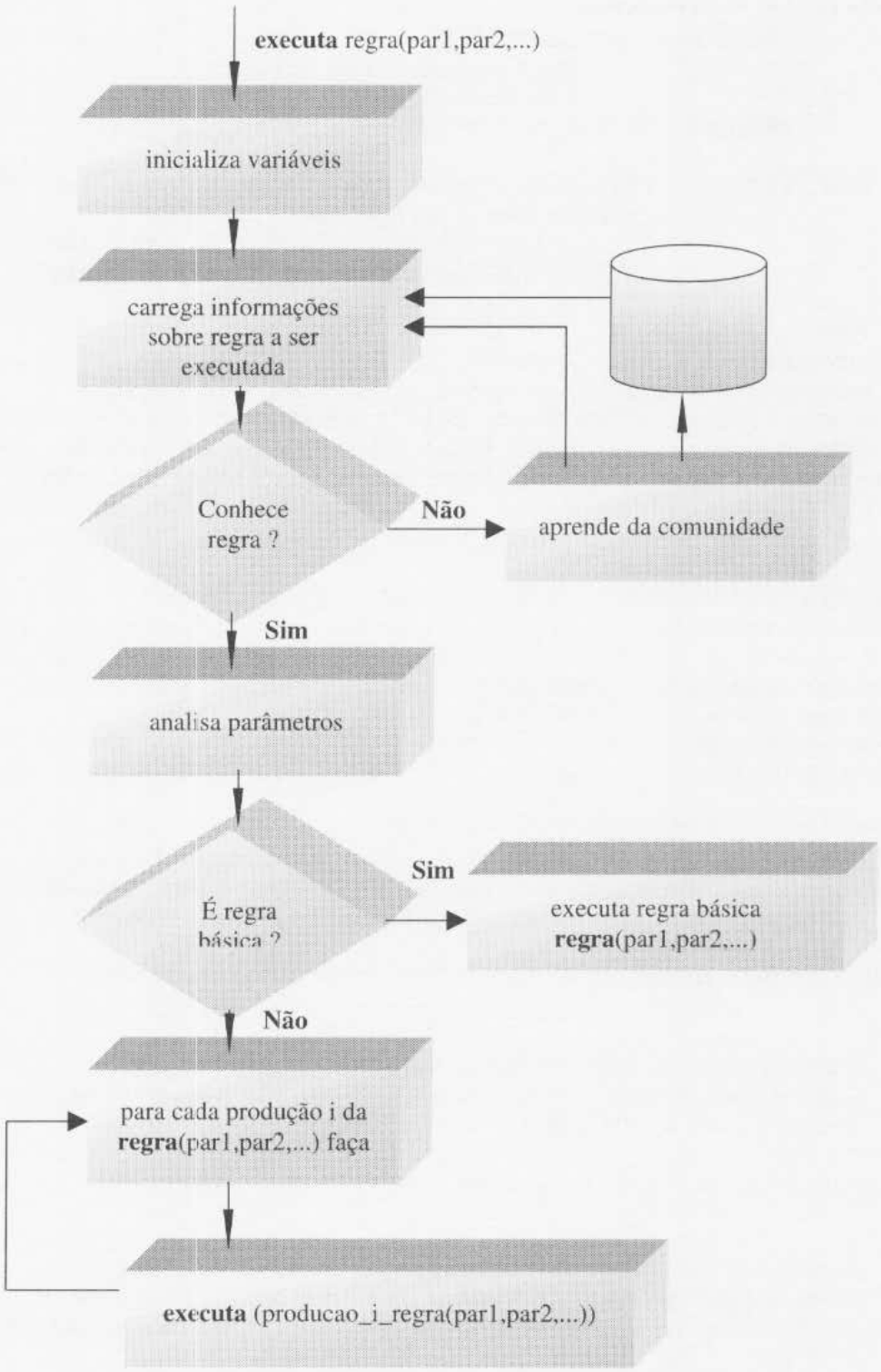

Figura 10 – Fluxo de dados no Componente de resolução de regras

```
/*************************************************************************
** MODULE INFORMATION
**********************
**      FILE NAME:      SNMPlib.h
**      PART NAME:      SNMP integration library
**      AUTHOR:         Fernando Koch
**      VERSION:        1.0
**
** DESCRIPTION : This module presents the prototypes for the SNMP
**               integration library functions.
**               It was implemented based on the Tricklet application,
**               inheriting functions and data structures.
**
** PUBLIC :
** SNMPstart()
** SNMPsend(int tipo,char *host,char *var,char *valor)
** SNMPgetrequest(char *host,char *var,char *valor)
** SNMPgetnextrequest(char *host,char *var,char *valor)
*************************************************************************
*/

#include <stdlib.h>
#include <stdio.h>
#include <snmp.h>

#define community   "public"
#define timeout     1000000L
#define retries     3
#define port        161

/***
   SNMPgetnextrequest(char *host,char *var,char *valor)
   Send a get-next-request packet to a SNMP agent requesting the
   contents of a <variavel> and returning the <valor>
***/
SNMPgetnextrequest(char *host,char *var,char *valor);

/***
   SNMPgetrequest(char *host,char *var,char *valor)
   Send a getrequest packet to a SNMP agent requesting the
   contents of a <variavel> and returning the <valor>
***/
SNMPgetrequest(char *host,char *var,char *valor);

/***
   SNMPsend(int tipo,char *host,char *var,char *valor)
   Send a SNMP packet of type <tipo> to host <host> with
   OBJECTID <var>. The string converted returning value is
   attributed in <valor>.
   Returns 0 if success or -1 if failure
***/
SNMPsend(int tipo,char *host,char *var,char *valor);
```

Figura 11 - Pedaços do código fonte da biblioteca de integração SNMP

5. CONCLUSÃO

O objetivo final proposto por este trabalho foi atingido, com a criação de um modelo de agentes autônomos flexível e adaptável para uma larga gama de ambientes e serviços que podem estar disponíveis. A integração com o ambiente SNMP foi essencial para a prática do sistema em ambientes comerciais.

Outra característica de destaque é a possibilidade de interação humano/agente através do protocolo HTTP/CGI, que permite o fácil acesso do gerente humano à comunidade de agentes. Já os sistemas de análise de informações através de redes neurais, apesar de poderosos em seus resultados, mostram dificuldades de performance no período de treinamento.

As maiores dificuldades de pesquisa foram aquelas a respeito dos agentes autônomos. Por ser um campo de pesquisa recente, existe pouco material disponível nesta área e nem mesmo há um consenso sobre o que vem a ser um agente autônomo, muito menos a forma como deve ser implementado.

Este trabalho baseou-se fortemente nos trabalho de Burmeister [BUR92a][BUR92b], que criou um principio de comunicação entre agentes muito eficiente, servindo visivelmente de base para a definição da linguagem KQML, apresentada com detalhes em [FIN95].

Teve, também, influência do trabalho de Maes [MAE94a][MAE94b], do MIT, que publicou trabalhos de cunho prático sobre a teoria e arquitetura para implementação de agentes. Por fim, os princípios arquiteturais dos agentes autônomos foram tomados dos trabalhos de Scalabrini e Vandenberghe [SCA96], que criaram estruturas genéricas para agentes, com o intuito de construir sistemas para gerenciamento de informações acadêmicas e *ghost-writing*.

5.1 PERSPECTIVAS FUTURAS

Apesar da implementação já realizada apresentar o sucesso ao atingir os objetivos iniciais definidos pelo sistema, algumas melhorias e implementações adicionais são propostas, com o objetivo de tornar o sistema mais seguro, eficiente e abrangente. Segue, abaixo, uma lista de objetivos futuros a serem atingidos :

1. deve ser criada uma gama de agentes de coletas de dados e de gerenciamento, com um grande número de rotinas básicas e bases de conhecimentos e regras de produção mais abrangentes para o teste completo da reusabilidade e adequação do sistema em ambientes de gerenciamento de redes heterogêneos.
2. devem ser realizados maiores estudos da utilização de redes neurais artificiais, com implementações mais eficientes, que viabilizem os processos de análise qualitativa dos dados e predição de séries temporais, criando um sistema pró-ativo seguro e eficiente.
3. deve ser criado um tipo especial de agente de gerenciamento para manipulação com grandes bases de conhecimento, onde ficarão armazenadas todas as regras de produção da comunidade e a quem os agentes se reportarão para questionar novos conhecimentos para suas bases locais. Este será denominado de *Agente Sábio* e terá o objetivo de consolidar toda a informação conhecida, a fim de tornar o processo de comunicação mais ágil e de armazenamento de regras mais seguro;
4. deve ser criado um tipo especial de agente com a habilidade de criar e colocar em execução novos agentes a partir de características desejadas de funcionamento atribuídas pelos usuários do sistema. esta agente será chamado de *Agente Criador* e terá como objetivo iniciar uma nova comunidade em qualquer ambiente que se desejar. Já existe funcional um protótipo deste agente, em forma de *proto-agente*, que foi usado para criar os agentes de demonstração deste trabalho e será usado de base para a criação deste tipo de agentes especial.

Portanto, tem-se ainda muito o que explorar dentro da tecnologia apresentada neste trabalho. Existe um número virtualmente infinito de agentes que podem ser definidos, pela criação de novas regras de produção e rotinas básicas tanto para a coleta de dados como para a emissão de relatórios gerenciais e de interface.

6. BIBLIOGRAFIA

[BUR92a] Burmeister B., Sundermeyer K. - *Cooperative Problem-Solving Guided by Intentions and Perception* - Lectures Notes In Distributed Artificial Intelligence - Anais do 11° Simpósio Brasileiro de Inteligência Artificial, SBIA'92, pp. 77-91, Elsevier Science Publishers B.V, Brasil, 1992

[BUR92b] Burmeister B., Haddadi A., Sundermeyer K. - *Generic, Configurable, Cooperation Protocols for Multi-Agent Systems* - Lectures Notes In Distributed Artificial - Anais do 11° Simpósio Brasileiro de Inteligência Artificial , SBIA'92, pp. 157-171, Elsevier Science Publishers B.V, Brasil, 1992

[CAS92] Castelfranchi,Maria M. C., Amedeo,C. - *Dependence Ralations Among Autonomous Agents*, Lectures in distributed artificial inteligence - Anais do 11° Simpósio Brasileiro de Inteligência Artificial, SBIA'92, pp.215-227, Elsevier Science Publishers B.V, Brasil, 1992

[DUR90] Durfee,E.H., Montgomery,T.A. - *A Hierarchical Protocol for Coordenating Multiagent Behavior* - Anais do AAAI'90, 1990

[FEI95] Feit, Sidnei - *SNMP : a guide to network management*, McGraw-Hill series on computer communications, USA, 1995

[FER92] Ferber J., Drogoul A. - *Using Reactive Multi-Agent Systems in Simulation and Problem Solving* - in Distributed Artificial Intelligence: Theory and Pratcis, pp. 53-80 - Kluwer Academic Publishers- USA, 1992

[FIN95] Finn, Tim ; Labrou, Yannis ; Mayfield, James - *KQML as an agent communication language*, finin@cs.umbc.edu, USA, 1995

[GIL97] Gilbert, D. - *Intelligent Agents: The Right Information at the Right Time* - http://www.networking.ibm.com/iag/iaghome.html, IBM Corporation, USA, 1997

[KAH97] Kahani, M., Beadle, P. H.W. - *Decentralized approaches for network management* – Computer Communications Review, ACM - SIGCOMM, Vol. 27, Number 3, pp. 36-47, USA, 1997

[KAU94] Kautz H.A., Selman B., Coeh M. - *Bottom-Up Design of Software Agents* -Communications of the ACM vol.37 No.7 - USA, 1994

[KAY84] Kay, A. - *Computer software*, Scientific American, Vol 251, No 3, USA, 1984

[KAY90] Kay, A. - *User interface : a personal view*, The art of human-computer interface design, Addison-Wesley, 1990

[LAB96] Labrou Y., Finin T. - *A Semantics Approach for KQML - a General Purpose Communication Language for Software Agents* - http://www.cs.umbc.edu/~jklabrou/publications.html - USA, 1996

[MAE94a] Maes, Pattie- *Modeling adaptative autonomous agents*, Artificial Life J., ed. C. Langton, Vol. 1, Nos. 1 & 2, MIT Press, pp. 199-220, New York, USA, 1994

[MAE94b] Maes, Pattie - *Agents that reduce work and information overload*, Comm. ACM, Vol. 37, No. 7, USA, 1994

[NEG70] Negroponte, N. - *The architecture machine, towards a more human environment*, MIT press, USA, 1970

[SCA96] Scalabrini E.E., Vandenberghe L., Azevedo H. de, Barthès J-P. A. - *Generic Model of Cognitive Agent Develop Open Systems - Lecture Notes in Artificial Intelligence* - Anais do 13° Simpósio Brasileiro de Inteligência Artificial - SBIA'96, pp. 61-70, Brasil, 1996