UNIVERSIDADE DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

TITULO: AGENTES COLETORES PARA GRIDS DE AGENTES NA GERENCIA DE REDES DE COMPUTADORES

AUTOR: **EDUARDO ERLÊ DOS SANTOS**
ORIENTADOR: **CARLOS BECKER WESTPHALL**
CO-ORIENTADOR: **FERNANDO LUIZ KOCH**
BANCA EXAMINADORA: **CARLA MERKLE WESTPHALL**
FERNANDO AUGUSTO DA SILVA CRUZ

PALAVRAS-CHAVE: Agentes Inteligentes, Computação Distribuída, Gerência de Redes de Computadores.

Florianópolis, 11 de julho de 2003.

DEDICATÓRIA

Dedico este trabalho primeiramente a minha família, porque lutaram muito e se privaram de muitas coisas para que eu pudesse concluir a minha graduação culminando neste trabalho. Dedico aos companheiros do LRG que foram fontes preciosas do conhecimento que se encontram espalhados nestas páginas. Dedico ao meu orientador e co-orientador que apontaram a direção em tantas vezes que estava perdido. A todos que foram parte deste trabalho direta ou indiretamente, dando força, conhecimento, e orientação dedico este que não poderia ter sido feito sem vocês.

AGRADECIMENTO

Agradeço aos meus amigos, porque acreditaram em mim quando eu achava que não podia. Agradeço-os porque nesta capital onde não conhecia ninguém encontrei uma família, e os tenho nas boas horas e nas ruins. Agradeço porque me acolheram em suas casas, por ajudar a conseguir os estágios que tive, por permanecer os finais de semana nos laboratórios estudando comigo, e por partilhar daquela cerveja quando podíamos finalmente descansar.

Além da dedicatória, deixo o meu agradecimento a Teresa Maria dos Santos e a Erlê dos Santos, meus pais, por tornarem possível que pela primeira vez uma pessoa entre toda a minha família obtenha um curso de graduação, conseguido com certeza pelo seu esforço profissional e pedagógico. Agradeço pela sua preocupação e dedicação com seus filhos.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SÍMBOLOS E ABREVIATURAS

CMIP	Common Management Information Protocol
FIPA	Foundation for Intelligent Physical Agents
ISO	International Organization for Standardization
OSI	Open System Interconnection
SNMP	Simple Network Management Protocol
XML	Extensible Markup Language

RESUMO

Ambientes interligados em rede possuem, em sua maioria, uma grande variedade de equipamentos, e de sistemas de gerência independentes para esses equipamentos. Uma possibilidade para a gerência de uma rede de computadores é o uso de Grids de Agentes, onde um grid é uma arquitetura distribuída que compartilha os recursos de cada máquina dando a impressão ao usuário de um grande computador virtual com muitos recursos. Essa arquitetura pode ser implementada por meio de agentes, que descentralizam as funcionalidades que compõem a gerência da rede e se valem de técnicas de inteligência artificial para conquistar autonomia em sua tarefa. O primeiro passo para a gerência é a coleta dos dados necessários para avaliar a performance da rede. Para isso buscamos utilizar agentes coletores para a obtenção dos dados, abstração do sistema do computador alvo e para a formatação da informação padronizando e facilitando o ato da gerência.

ABSTRACT

Computer network systems are composed by several different equipments that have their own management system associated. One possibility for the management task is to use Grids of Agents where the 'grid' is a distributed architecture that aggregates the resources from several machines forming a resourceful virtual system. This architecture can be implement through autonomous agents systems that distribute the management activities and make use of artificial intelligence concepts to reach autonomy. The first step of a management system is collecting the data set from the network's devices. In this work we present a solution by using 'collector agents' that act closer to the managed devices by extracting data and formatting into an abstract representation language, thus distributing the management task.

SUMÁRIO

1. INTRODUÇÃO

O aumento da complexidade das redes de computadores pelo seu crescimento numérico, ou pela diversidade de seus componentes, tem dificultado a atividade de gerência do sistema. Uma das dificuldades está em isolar e identificar os problemas das redes, devido à diferença de nível entre o pessoal envolvido, tais como: técnicos de manutenção, operadores de rede, gerentes de sistema de informações e gerentes de comunicação. Também há dificuldade pela diversidade de formas de controle e monitoração, porque embora os produtos envolvidos na rede se tornem gradativamente mais complexos, cada fornecedor oferece ferramentas próprias de controle para monitorar seus produtos.

Esse cenário exige um grande esforço da equipe encarregada de administrar uma rede de computadores, gerando uma busca pela automatização de processos, tentando diminuir esforço humano o máximo possível e da maneira mais segura e confiável, mantendo o funcionamento adequado da rede e evitando o custo de uma paralisação no sistema.

Uma das possibilidades para modelagem desse sistema de gerência de redes é o uso de um grid de agentes, que consiste em um sistema formado por comunidades de agentes distribuídos pela rede, cada um com a sua tarefa específica. Essas tarefas podem ser coletar os dados de um computador conectado a essa rede, organizar os dados coletados dos vários computadores de uma rede e gerenciá-los, ou apresentá-los por meio de uma interface amigável para as pessoas encarregadas da gerencia da rede.

1.1 Objetivo Geral

Esse trabalho é motivado pela necessidade de conhecer formas de gerenciar de redes de computadores, buscando obter uma melhor performance com o menor esforço humano, baseando-se em teorias de agentes autônomos. Esses agentes podem ser capazes de coletar dados dos dispositivos, interpretar os dados, tomar decisões sobre o estado da rede, e aprender em novas situações, aumentando a confiabilidade do sistema e

buscando prevenir problemas, em vez de pagar o alto custo de remediá-los posteriormente.

1.2 Objetivos Específicos

Como objetivos específicos, destacam-se as atividades de implementar um agente coletor utilizando os dados de linha de comando do sistema operacional do Computador, implementar um sistema reduzido de coordenação, com um agente gerente, para analisar os dados coletados e criar uma interface gráfica reduzida para demonstrar os dados coletados e a análise feita pelo agente gerente.

1.3 Organização Do Trabalho

O conteúdo deste documento está organizado em 7 capítulos, obedecendo a seguinte ordem:

No capítulo 2 são apresentadas algumas definições sobre gerenciamento de redes de computadores com o objetivo de apresentar alguns recursos que podem ser gerenciados.

No capítulo 3 é discutido o contexto e a necessidade do uso de grids para a gerência da rede. Essa noção é útil para compreendermos uma arquitetura de utilização de agentes, distribuídos em um ambiente de rede e atuando na sua gerência com o intuito de otimizar os recursos da mesma.

No capítulo 4 é apresentada uma pequena introdução sobre agentes, com a discriminação de alguns tipos de agentes e as estruturas que dão suporte a eles.

No capítulo 5, são apresentadas as especificações dos agentes utilizados no projeto, o seu funcionamento interno, a representação de conhecimento e as estruturas que dão suporte a conhecimentos específicos.

No capítulo 6 é discutido o porque de utilizar agentes coletores, quais as vantagens desse sistema de coleta de informações. Características da aplicação são apresentadas neste capítulo bem como a estrutura interna dos

agentes, aprendizado das regras para coleta de dados, formatação dos dados coletados, composição e envio da mensagem para o agente gerente.

No capítulo 7 é apresentada uma conclusão formada a partir do estudo feito aplicado na prática. Também são apresentadas possíveis melhorias neste trabalho e mencionados alguns trabalhos que poderão ser desenvolvidos a partir deste.

2. GERÊNCIA DE REDES DE COMPUTADORES

A gerência de redes de computadores busca garantir a qualidade dos serviços para as aplicações utilizadas na rede, o que a torna uma prática vital para a operação de redes. Para suprir a necessidade de gerência, vários fabricantes desenvolvem ferramentas de gerenciamento, mas que não interagem entre si, e não possuem uma forma de manuseio integrado, tornando o ato de gerenciar a rede ineficiente e caro.

Buscando o desenvolvimento de ferramentas de gerenciamento que fossem comuns a todos os fabricantes a ISO (*international Organization for Standardization*) desenvolveu padrões de gerenciamento que permitem a interoperabilidade de múltiplos e diversificados sistemas computacionais contidos em uma rede de comunicação. O objetivo deste trabalho não é discorrer sobre as normas da ISO, nem sobre o seu modelo de gerência, o OSI (Open System Interconnection), e sim apresentar uma forma de dividir o problema da gerência de redes em partes distintas, no caso da ISO, em áreas funcionais, facilitando a identificação do escopo de um problema na rede e definindo quais as ações serão tomadas para saná-lo.

2.1 Áreas Funcionais

O Gerenciamento de uma rede busca resolver problemas relativos à sua configuração, falha de componentes, níveis de desempenho alcançados, segurança e contabilização de sua utilização. Estas diferentes partes do problema de Gerenciamento de Redes são denominadas áreas funcionais. Baseado em BRISA (1993), temos as seguintes áreas funcionais:

- **Gerenciamento de Configuração:** tem por função a manutenção e monitoração da estrutura física e lógica da rede, incluindo a existência de componentes e sua interconectividade. Corresponde ao conjunto de facilidades que exercem o controle sobre os objetos gerenciados, identificando-os, coletando e provendo dados sobre os mesmos para dar suporte a funções de:

- Atribuição de valores iniciais aos parâmetros do sistema;
- Início e encerramento de operações sobre objetos gerenciados;
- Alteração da configuração do sistema;
- Associação de nomes a conjuntos de objetos gerenciados.

- Gerenciamento de Falhas: é responsável pela manutenção e monitoração do estado de cada um dos objetos gerenciados e pelas ações necessárias ao restabelecimento das unidades com problemas. As informações coletadas podem ser usadas em conjunto com o mapa da rede, para indicar quais elementos da rede estão funcionando, quais operam precariamente ou quais permanecem fora de operação. O Gerenciamento de Falhas também pode prover um registro das ocorrências, um diagnóstico de falhas e uma correlação entre os resultados do diagnóstico e as subseqüentes ações de reparo.

- Gerenciamento de Desempenho: preocupa-se com o desempenho corrente da rede, incluindo parâmetros estatísticos tais como atrasos, vazão, disponibilidade e número de retransmissões. Consiste em um conjunto de funções responsáveis por manter e examinar registros com um histórico dos estados do sistema para fins de planejamento e análise.

- Gerenciamento de Segurança: aborda os aspectos de segurança essenciais para operar uma rede corretamente e proteger os objetos gerenciados. O sistema de gerenciamento deve providenciar alarmes para o gerente da rede quando eventos de segurança forem detectados. Como alarmes devemos entender uma notificação ou evento específico para representar uma situação de risco de segurança na rede.

- Gerenciamento de Contabilização: preocupa-se com a manutenção e monitoração de quais recursos e de quanto desses recursos estão sendo utilizados. Estas informações podem ser utilizadas para estatísticas ou para “bilhetagem” (faturamento).

2.2 Motivação para o uso de Agentes

A gerência de redes de computadores é um processo complicado, detalhista, e de difícil análise, por isso, necessita de ferramentas que auxiliem a equipe responsável pela rede, minimizando o custo de manter a rede e maximizando a performance da mesma.

O domínio da rede de computadores pode ser dividido em áreas funcionais que auxiliarão na criação dessas ferramentas, no caso deste trabalho, especificando qual a função dos agentes dentro dessa rede, quais aspectos devem ser monitorados, quais dados devem ser coletados, quais ações os agentes gerentes devem realizar para manter o funcionamento dessa rede, e como formatar os relatórios para o usuário de forma a destacar essas funcionalidades dentro da rede.

3. GRIDS

Um grid é uma arquitetura que distribui os recursos computacionais de uma rede, buscando extrapolar o limite de um computador, possibilitando que um computador que tenha a necessidade de mais recursos, possa recorrer a algum computador da rede que possua esses recursos disponíveis. Esse comportamento faz com que um grid passe a impressão de um grande computador virtual com tantos recursos quantos os que estão disponíveis nesta rede.

Um exemplo simples de um grid pode ser a utilização dos recursos de processamento de uma máquina ociosa na rede por uma maquina saturada. O processo, que não pode ser executado na máquina que está ocupada, é enviado à máquina ociosa que realiza o trabalho. Outro exemplo é o compartilhamento de dados. Quando um dado é gravado ele é espelhado em outros pontos da rede, permitindo um acesso mais eficiente aos dados para todas as estações da rede, quando este acesso é gerenciado pelo grid, e ainda permitindo um sistema de recuperação, para que, quando ocorrer perda de um dado armazenado, este possa ser recuperado de outro ponto qualquer da rede.

Através do compartilhamento de recursos dos computadores conectados a uma rede, um grid cria a ilusão de que cada computador possui muito mais recursos do que realmente há disponível para serem utilizados, desde que recursos disponíveis para serem cedidos existam nesta rede, ou seja, um grid provê uma maneira de explorar os recursos subutilizados em uma rede criando a possibilidade de um substancial aumento na eficiência do uso de recursos disponíveis. Esses recursos não são somente processamento ou espaço em disco, também pertencem ao conjunto de recursos compartilháveis em um grid, as impressoras, ou um dispositivo específico, como um telescópio conectado a um computador, para coleta de dados espaciais, ou seja, qualquer dispositivo que faça parte da rede que implementa o grid.

Uma das dificuldades para implementação do grid é a característica heterogênea de um sistema de rede, que geralmente possui equipamentos e programas de várias empresas distintas, necessitando de uma busca pela padronização e integração do ambiente de rede.

3.1 Utilização de Grids de Agentes

Uma possibilidade para a implementação de um grid é a realização de um Grid de Agentes, que distribui os agentes no ambiente de rede, descentralizando as funcionalidades que compõem a gerência da rede, e utilizando técnicas de inteligência artificial para conquistar autonomia em sua tarefa. Esses agentes podem ser flexíveis o suficiente para se adaptar a sistemas operacionais específicos de computadores, ou a qualquer programa de controle de um equipamento que faça parte desta rede, e pode gerenciar esse recurso de forma inteligente.

Segundo Assunção, Westphal e Koch (2003), são identificados os serviços necessários para o grid. Esses serviços agrupam funcionalidades e metas para serem alcançadas pelos agentes, e formam grids com funções específicas. Este trabalho classifica os grids de agentes em *grids de agentes coletores* (GC), responsáveis por coletar os dados necessários para a gerência da rede, *grids de agentes classificadores e de armazenamento* (GCL), que são responsáveis pelas atividades de classificação, indexação e armazenamento dos dados coletados; *grids de agentes analisadores* (GP), preocupados com a análise das informações da rede, e *grid de interface* (GI), responsável pela interação com usuário ou com outros sistemas de gerência.

O foco deste trabalho é nos agentes de coleta de dados do grid, ou seja, no funcionamento do grid de agentes coletores. Nos capítulos seguintes focaremos o trabalho sobre esses temas justificando o seu uso e demonstrando uma forma de implementá-los.

4. AGENTES

Um agente é algo que tem a capacidade de perceber seu ambiente através de sensores, e agir nesse ambiente através de atuadores. Um agente humano, por exemplo, possui olhos e ouvidos que recebem estímulos do ambiente, percebem o ambiente, ou seja, são os sensores do agente. Esse mesmo agente também possui boca, mãos, pés, que interagem nesse ambiente, atuam nele, por isso, são denominados atuadores. Esta mesma metáfora pode ser utilizada para modelagem de agentes em hardware ou software, como um robô ou um agente de busca na Internet.

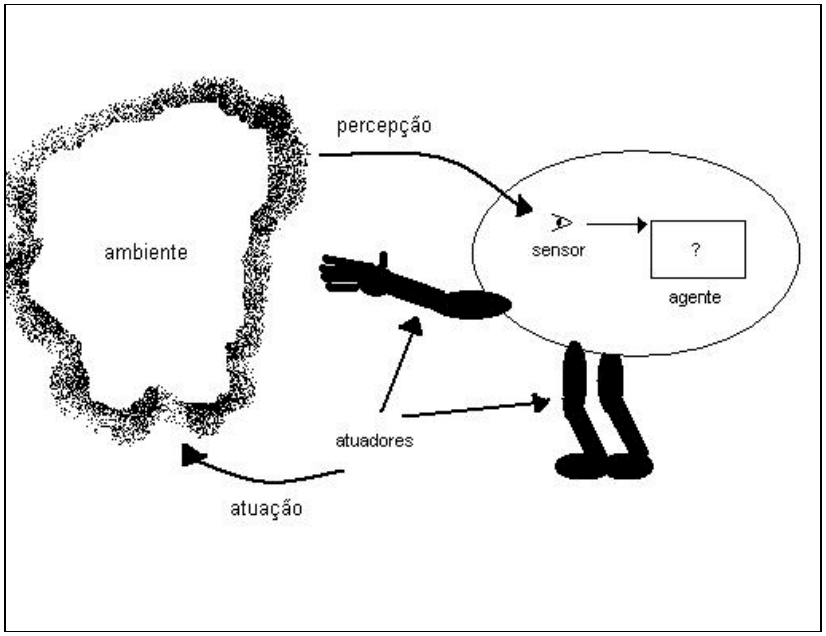

Figura 4 – Interação do agente com o ambiente por meio de sensores e atuadores.

Os agentes podem ser considerados como uma das formas de modelar um sistema, não necessariamente de implementá-lo, um compilador, por exemplo, pode ser encarado como um agente, pois possui uma percepção do ambiente, no caso, uma linguagem fonte, e atua no sistema, gerando um resultado em linguagem objeto. Este exemplo não tem nenhuma técnica de Inteligência Artificial, ou modelagem de agente, mas pode ser definido como uma forma de agente. Da mesma maneira, podemos modelar um sistema pensando em Orientação a Objetos e implementá-lo sem utilizar

uma única classe. Não é o caso deste trabalho, mas é praticamente possível, o que nos mostra que a modelagem de agentes é uma forma de pensar um sistema, projetando este para que possa resolver um conjunto de problemas de uma forma mais eficiente.

4.1 Agentes Inteligentes

Pela definição de Russell e Norvig (1995), um agente inteligente é aquele capaz de “fazer a coisa certa”, ou seja, ele é capaz de realizar a ação de forma a obter o máximo de sucesso possível. O princípio básico para utilização de agentes, é que eles “saibam das coisas”, ou seja, que tenham um conhecimento básico de como agir de acordo com as percepções do ambiente.

4.2 Agentes Inteligentes Autônomos

Agentes autônomos são sistemas computacionais que operam em ambientes dinâmicos e imprevisíveis. O grau de ‘autonomia’ de um agente está relacionado à capacidade de decidir, por si só, como relacionar os dados dos sensores com os comandos dos atuadores, enquanto realiza seus esforços para atingir seus objetivos.

O comportamento deste agente é baseado nos conhecimentos que possui e em sua capacidade de adquirir novos conhecimentos, ou seja, de aprender.

O nível de autonomia de um agente aumenta com a sua experiência, semelhante a um animal, que nasce somente com os instintos e com o ato reflexo, e através da experiência vai adquirindo conhecimentos por si mesmo para que possa continuar vivo.

4.3 Estrutura de Agentes

De acordo com o problema e com o ambiente no qual os agentes serão aplicados, serão necessárias estruturas de suporte as ações do agente, para que ele possa agir com 'inteligência' na busca da melhor solução.

Novamente é valido o princípio de que o agente tem que 'saber das coisas', e para que isso seja possível, ele tem que possuir todas as informações e regras necessárias, para que possa tomar as decisões corretas, e a escolha sobre que conhecimentos ficarão embutidos no agente, é essencial para que o agente possa atingir suas metas de maneira eficiente.

Em Russell e Norvig (1995), de acordo com a sua estrutura interna, os agentes podem ser divididos em quatro grupos, que são: *Agentes Reflexivos*, *Agentes com Estados Internos*, *Agentes Baseados em Metas* e *Agentes Baseados em Utilidade* apresentados a seguir.

4.3.1 Agente Reflexivo

O agente mais simples é o agente reflexivo. Para esse agente não existe passado nem futuro, pois as suas ações são baseadas nas informações colhidas pelo sensor naquele instante, atuando no meio através das regras contidas na sua base de conhecimentos. Por isso, quando cessa a percepção do ambiente cessa a ação.

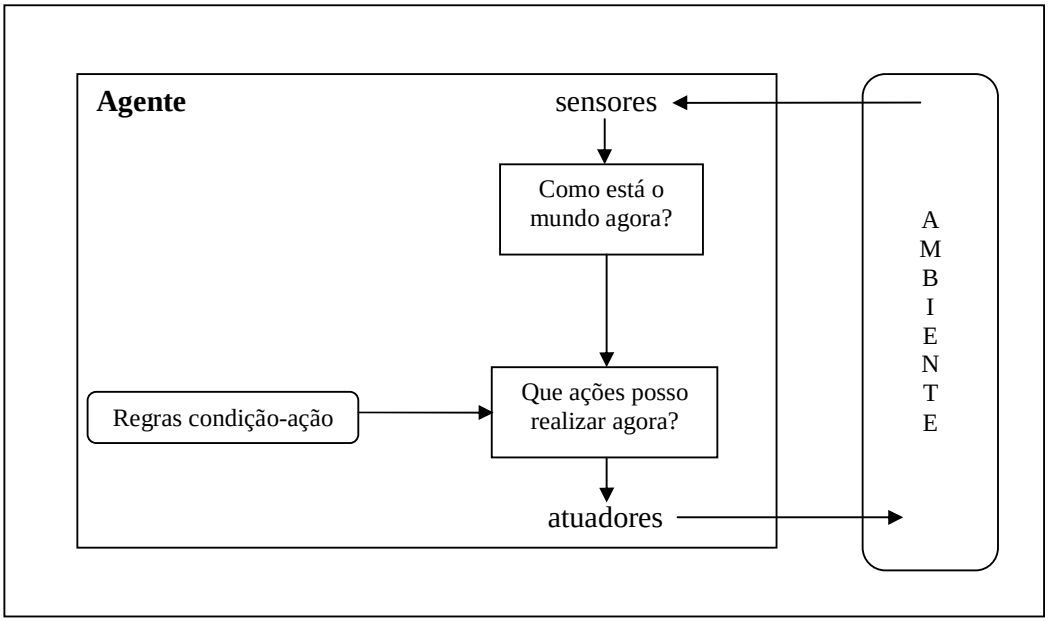

Figura 4.3.1 – Diagrama esquemático de um agente reflexivo.

4.3.2 Agente com Estados Internos

Este agente armazena informações que não são percebidas no momento como o estado interno, a forma que o mundo evolui e as conseqüências das ações no mundo. Como possui memória pode agir conforme a evolução do sistema, alternando os estados internos para se adaptar a essa evolução.

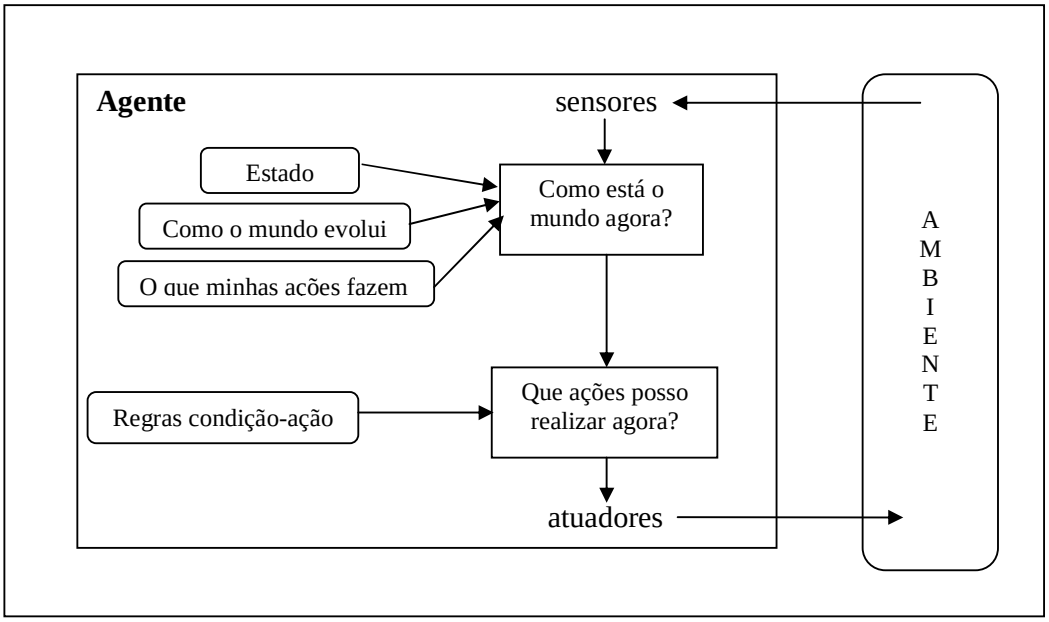

Figura 4.3.2 – Diagrama esquemático de um agente com estados internos

4.3.3 Agente Baseado em Metas

Este agente se vale de metas pré-definidas para escolher qual a melhor forma de atuar no ambiente. Ele tem a capacidade de refletir sobre o resultado das ações que pode realizar no sistema e de escolher o que mais próximo o deixa de sua meta.

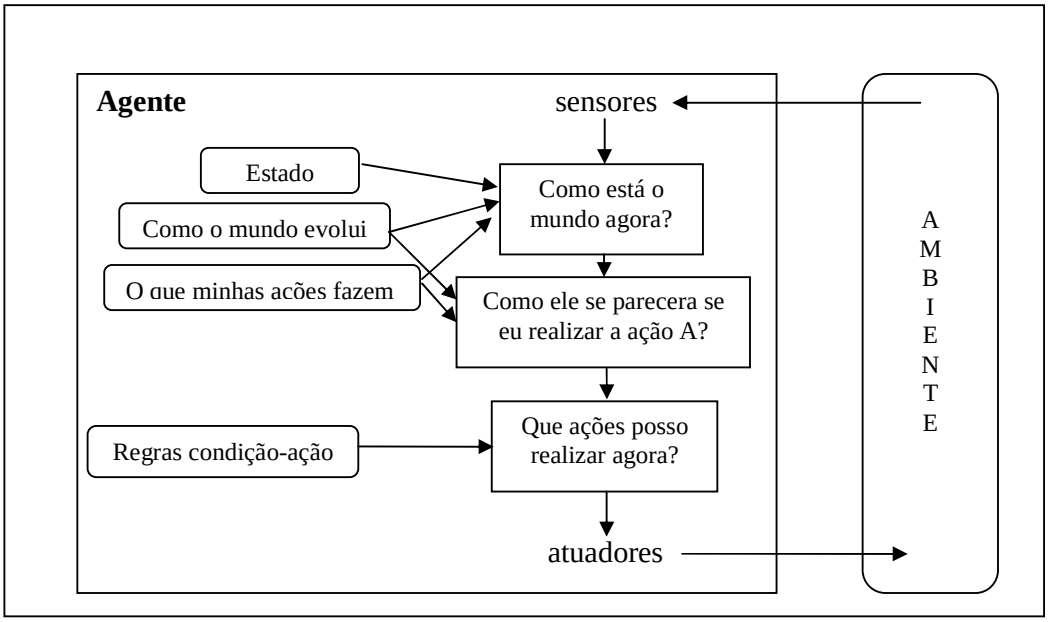

Figura 4.3.3 – Diagrama esquemático de um agente baseado em metas.

4.3.4 Agente Baseado em Utilidade

A característica que difere este agente é um atributo chamado 'utilidade'. A utilidade é uma função que mapeia um estado em um número real que descreve o 'grau de felicidade' associado ao estado. A utilidade permite tomar decisões racionais em casos em que a meta é conflitante, como, por exemplo, no caso de existirem várias metas, ou uma das metas apresenta dois caminhos de custos semelhantes para ser atingida.

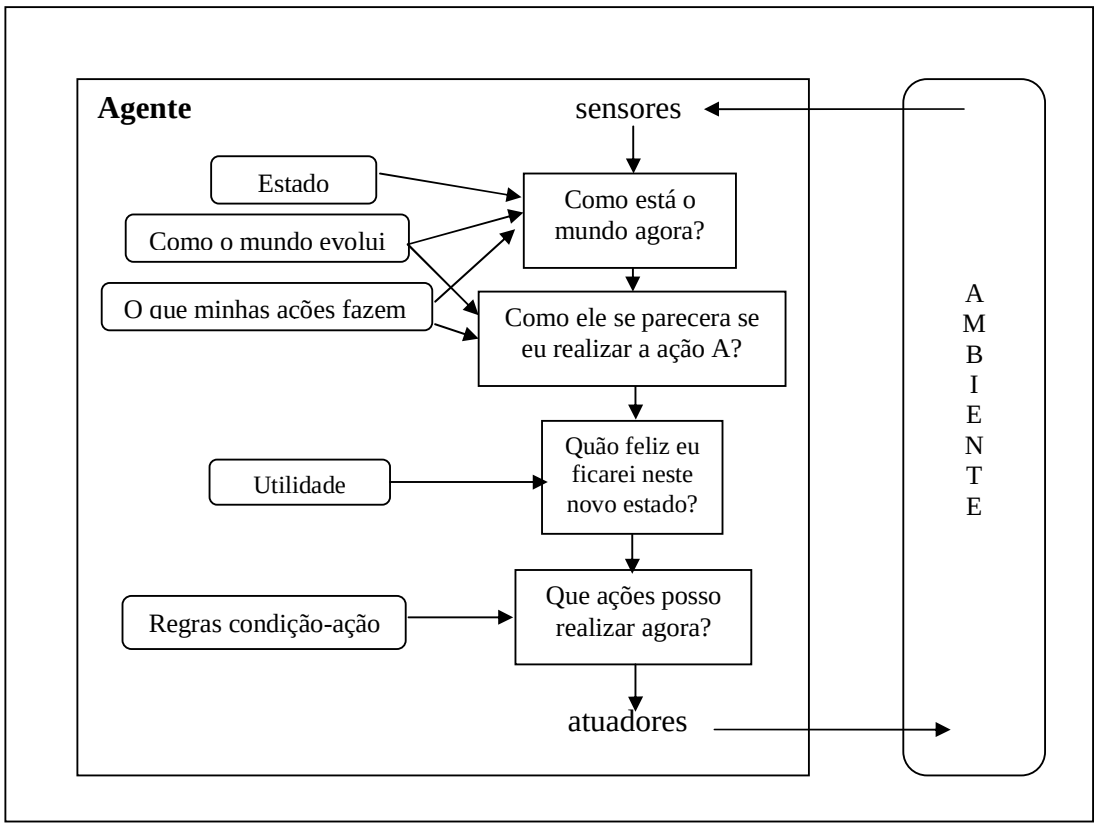

Figura 4.3.4 – Diagrama esquemático de um agente baseado em utilidade

4.4 Definições de Agentes Adotadas Para Este Trabalho

Algumas definições de agentes baseadas em trabalhos correlatos como em Koch (1997), ou Jennings e Wooldridge (1998), alguns destes desenvolvidos no Laboratório de Redes e Gerência da Universidade Federal de Santa Catarina foram aqui adotadas.

a. **Agentes devem ser realmente autônomos:** Os agentes devem ser capazes de controlar a si mesmo e a suas ações. Eles devem ser capazes de trabalhar independente do que aconteça com outros agentes, tendo total liberdade para adaptar-se a possíveis mudanças no ambiente. Sem esse item se obterá apenas um sistema usual onde qualquer coisa que aconteça demandará atenção total do Administrador do Sistema para poder se adaptar as novas condições do ambiente e configurar os novos parâmetros de operação.

b. **Agentes podem ser guiados por metas e baseados em regras:** Um conjunto de regras constitui a base de conhecimento para qualquer sistema de administração – incluindo os humanos. A base de conhecimento pode ser constituída igualmente por metas, que também são definidas através de regras, e que direcionam os trabalhos do agente. Definimos metas estáticas como parâmetros de configuração fixados na criação do agente e metas dinâmicas aquelas adicionadas ao comportamento inicial do agente e que foram derivadas de mudanças do ambiente ou de mensagens recebidas.

c. **Uma aplicação de agentes deve se comportar como uma 'base de dados globais':** onde cada dado ou informação, armazenada por um agente é compartilhado por todo o sistema (outros agentes). A comunicação torna a aplicação mais robusta, favorecendo a intercomunicação entre os agentes e facilitando a troca de conhecimentos, ou seja, facilitando o caminho para que cada agente tenha acesso a todas as informações necessárias quando precisar tomar uma decisão qualquer. Igualmente, no sistema 'global' – entenda-se por isso um sistema mundial, como a internet – um agente pode aprender uma nova regra de outra comunidade e partilhar informação com a sua própria, aumentando a base de conhecimentos de sua comunidade.

d. **Agentes devem ser capazes de aprender novas habilidades (skills):** Onde essas regras podem ser aprendidas por meio de outros agentes, da comunidade de agentes, da comunidade global ou, em ultimo caso, do ser humano. Essas novas regras podem ser armazenadas dinamicamente nas bases locais de conhecimento dos agentes. Outra

possibilidade é inferir novas regras a partir do conhecimento atual do ambiente.

e. Especificamente para Agentes Gerentes de Rede, **estes podem interagir com soluções comerciais usando protocolos padronizados.** Isso significa que Agentes podem saber, por meio de regras em suas bases de conhecimento, como interagir com protocolos de domínio público como o SNMP (Simple Network Management Protocol) e o CMIP (Common Management Information Protocol), como resgatar informações dos arquivos de registro (*log)* e também como interagir com outros sistemas de gerencia existentes. Tornar o seu sistema compatível significa facilidade de aceitação no mercado.

f. Finalmente, mas não menos importante, Agentes **podem ter uma interface humana amigável.** Uma interface amigável possível de ser utilizada entre homens e máquinas é um protocolo definido por um subgrupo definido da linguagem natural humana.

5. ARQUITETURA DE AGENTES

Neste capítulo será apresentada uma visão da implementação do agente, de sua arquitetura interna, das particularidades dos módulos do agente e da forma como ele coleta os dados das estações da rede.

A plataforma adotada foi a plataforma AgentLight de Koch e Meyer (2002), que é descrita a seguir.

5.1 Plataforma AgentLight

A arquitetura de agentes descrita a seguir é baseada na plataforma AgentLight, utilizada para implementar as regras que formam o agente coletor foco deste trabalho.

A plataforma AgentLight foi desenvolvida tendo em vista a necessidade de expandir a tecnologia de agentes para equipamentos com limitados recursos de memória e processamento, como celulares e palms. Essa plataforma foi desenvolvida em Java, permitindo que todo equipamento para o qual exista uma máquina virtual Java possa executar as classes que formam o AgentLight. Isso proporciona a independência da plataforma de hardware utilizada, bem como também proporciona reusabilidade de código, que são pontos fortes para a aplicação desta plataforma de agentes.

Alguns princípios que devem ser conhecidos para a utilização da plataforma AgentLight são descritos a seguir:

5.1.1 Containers

Containers são utilizados para executar um conjunto de agentes que realizam uma única tarefa. O container é responsável por encapsular os agentes que fazem parte da aplicação. Necessariamente existe interno ao container o agente gerente que é responsável pela atividade dentro deste e demais agentes são criados de acordo com as regras passadas ao container.

O container também é responsável por oferecer um canal de comunicação entre agentes pertencentes a um container, e um canal de comunicação entre containers distintos. O container também disponibiliza para os agentes internos a ele uma base de conhecimento global bem como um motor de inferência.

5.1.2 Agentes Genéricos

A arquitetura de agentes adotada se apropria do conceito de *Agentes Genéricos*, que segundo Koch e Westphall (2001), serve para definir a sua estrutura básica, ou sejam, os mecanismos e as habilidades necessárias para garantir um mínimo de comportamento padrão, internamente e externamente, para qualquer agente que for criado. Essa definição serve como modelo para a criação de novos agentes pelo sistema, que poderão receber novos conhecimentos se ajustando ao ambiente de destino.

O artigo citado apresenta três módulos básicos para a implementação de agentes:

- **Módulo de Inferência**: Para agentes baseados em regras o módulo de inferência é o que implementa a máquina de inferência para resolução das regras e também é responsável para armazenar o conhecimento do agente.

- **Módulo de Interação**: Este módulo é responsável pela comunicação com o ambiente, por isso é um módulo que atua de acordo com as características deste ambiente.

- **Módulo de Comunicação**: É o módulo responsável pela troca de mensagens entre os agentes. Para o formato das mensagens podemos utilizar padrões de domínio público como o da FIPA (Foundation for Intelligent Physical Agents), disponível em FIPA (2001).

5.1.3 Funcionamento Interno do Agente

A estrutura do agente utilizado é formada pelos módulos básicos que compõem um agente genérico.

O módulo de inferência dos agentes utiliza as regras de uma linguagem baseada em Prolog, qualquer conhecimento específico para resolução do problema deve ser construído a partir das regras básicas ou deve ser implementado como uma especialização.

5.1.3.1 Prolog

Para compreensão do conhecimento contido nas bases dos agentes e das regras utilizadas por eles é necessário conhecer a sintaxe do Prolog.

Os tipos no Prolog são:

- O tipo numérico, por exemplo, 1 ou 4.5;
- O tipo palavra, por exemplo, 'gato' representado entre apóstrofes;
- O tipo ***átomo***, por exemplo, gato sem apóstrofes.

O tipo átomo representa a menor unidade de conhecimento no Prolog, ou seja, se inserimos na base de conhecimento do agente o átomo:

gato.

Significa que gato é verdadeiro no contexto atual sempre que alguma regra perguntar o valor de verdade de *gato*.

O conhecimento também pode ser representado por meio de ***fatos***, que são representados da forma:

cor(gato, preto).

Chamamos de functor o átomo que agrupa funcionalmente os parâmetros entre os parênteses, no caso do exemplo acima *cor*.

Uma palavra escrita começando com letra maiúscula é uma variável. Utilizando o exemplo anterior, se após ensinarmos o fato da cor do gato perguntássemos:

cor(gato,X).

A resposta seria “X = ‘preto’”.

Uma lista é representada da forma:

[elemento1, elemento2, ... , elementoN]

Podendo esta conter átomos, palavras ou números.

Por último a representação de uma regra em Prolog, que é formada por uma lista de fatos e átomos, como a seguir, utilizando-se dos exemplos anteriores:

cor_gato(COR) :- gato, cor(gato,COR).

Primeiramente, ‘,’ (vírgula) é o símbolo de conjunção, ‘;’ (ponto e vírgula) é o de disjunção e ‘:-‘ (dois pontos e traço) é o sinal de implicação, ou seja, a regra acima significa: se existir gato a cor é igual a cor do gato, e o resultado será “COR = ‘preto’”.

5.1.3.2 Utilização de Habilidades (Skills)

A definição de conhecimentos específicos que sejam necessários por algum agente é feita por meio de habilidades e ensinadas ao agente. Por exemplo, o processamento de palavras não é uma função lógica, então se existe a necessidade de que um agente saiba como efetuar operações com palavras é preciso ter a habilidade específica para resolver esse problema, habilidade esta que será ensinada para o agente por meio de regras do tipo: palavra(unir,’exem’,’plo’,Resultado), que deverá unir as palavras e retornar como resultado a palavra ‘exemplo’ na variável Resultado.

Na verdade, as habilidades são classes implementadas em Java que agem como uma extensão das regras primitivas conhecidas pelo agente e existentes no banco de regras do agente. Cada uma dessas classes é implementada como subclasse de ‘SkillInterface’ e possui um método virtual chamado ‘execute(...)’. Quando o motor de inferência da aplicação reconhece uma regra que utiliza uma dessas habilidades, ele utiliza o método ‘execute’ da classe para realizar o processamento da regra e retornar o resultado desta regra. Para se ter uma melhor noção da implementação o código abaixo

implementa a coleta de dados por linha de comando utilizando a regra CmdLineSkill(Resultado,comando, param1, ..., paramN).

Classe CmdLineSkill

```
package org.agentlight.skill;

import org.agentlight.rrengine.SkillInterface;
import org.agentlight.rrengine.Term;
import java.util.Stack;


public class CmdLineSkill implements SkillInterface {
  public boolean execute(Term[] array_, Stack stack_) {
    // pre-requisite
    if(array_.length<2) return false;
    if(! array_[0].isVariableNotBound()) return false;
    String[] _param = new String[array_.length-1];
    for(int i=1;i<array_.length;i++)
      _param[i-1] = (String) array_[i].functor();

    try{
      Process proc = Runtime.getRuntime().exec(_param);
      array_[0].bind(new Term(proc.getInputStream(),0), stack_);
    }
    catch(Exception e){
      e.printStackTrace();
      return false;
    }
    return true;
  }
}
```

6. AGENTES COLETORES

Agentes coletores são responsáveis no grid pelas atividades de coleta de informações. Esses agentes são aqueles que tem a maior interação com os objetos gerenciáveis, ou seja, com os computadores ou qualquer recurso passível de gerência que esteja conectado a rede.

Segundo Assuncao, Westphal e Koch (2003), algumas características que podem ser destacadas como justificativas para o uso de Grids de Agentes e ao uso de agentes coletores são:

- **Diminuição do tráfego entre o agente e o nó gerenciável**: A redução do tráfego de rede é uma conseqüência natural neste modelo de gerenciamento de redes, uma vez que o processo de aquisição e análise de informações é levado mais perto do (ou mesmo no próprio) local do objeto gerenciado. O agente autônomo age como um filtro das informações coletadas do dispositivo e repassadas para os gerentes do sistema de gerenciamento.

- **Maior abstração dos objetos gerenciáveis pelos gerentes:** Tendo em vista que muitas decisões podem ser tomadas diretamente pelos agentes autônomos, algumas das características e atributos do objeto gerenciado podem ser abstraídas pelos módulos gerentes ou mesmo alguns objetos gerenciados podem ser agregados em uma unidade abstrata.

- **Maior agilidade na tomada de decisões:** Sendo que as decisões estão mais próximas dos objetos gerenciados e que os processos de decisão são especializados para estes, evitando-se a necessidade de comunicação com gerente central.

- **Maior adaptabilidade do sistema:** O ideal dos agentes autônomos é estar preparado para quaisquer mudanças no ambiente onde estiver inserido e pronto para reagir positivamente a estas. Com agentes autônomos o nó gerenciável passa a ter "autonomia" com relação aos gerentes do ambiente, principalmente em questões não críticas. Desta forma, a gerência de rede torna-se mais automatizada.

6.1 Arquitetura do Agente Coletor

A estrutura de um agente coletor é talvez a mais simples do conjunto de gerência da rede. O coletor é um agente reflexivo, não tem passado nem futuro, com a única função de coletar periodicamente os dados desejados de uma estação da rede, formatá-los e enviá-los à aplicação gerente que analisará os dados recebidos do coletor.

Os conhecimentos necessários para que este agente cumpra a sua tarefa seriam sobre a forma como ele vai coletar os dados, como ele vai formatá-los e como vai enviá-los.

O conhecimento sobre a troca de mensagem entre agentes já está contido no agente, pois é fornecido pelo container do qual faz parte.

Para realizar a coleta dos dados, primeiramente é definida qual a forma a se utilizar. Como exemplos, pode se utilizar a coleta por meio de um protocolo conhecido, sendo necessário ensinar o agente a se comunicar nesse protocolo, ou por meio da interface oferecida pelo sistema operacional do computador alvo.

O método escolhido foi a coleta de dados por meio da linha de comando do sistema operacional, não necessitando do conhecimento específico de nenhum protocolo, o que não é objetivo deste trabalho, e facilitando no teste da aplicação, pois toda máquina possui um sistema operacional, mas nem toda máquina executa um programa de gerência de rede com os protocolos necessário para coleta de dados.

Para coletar os dados através de linha de comando foi utilizado uma especialização chamada *CmdLineSkill*, que possui a regra '**cmdline(comando,[lista,de,parâmetros],Resultado)**'.

Para formatar a resposta foi escolhida a linguagem XML, por ser uma linguagem atualmente bem aceita e de fácil utilização. Sendo assim, todos os dados colhidos serão transformados em um texto XML para serem transmitidos para os agentes encarregados de analisar os dados. Para formatar os dados em uma árvore XML, existe uma especialização chamada 'TemplateSkill', que gera dados formatados a partir de uma estrutura

fornecida como modelo. ‘TemplateSkill’ fornece a regra ‘**template(tree, RegExpResultados, FormatoModelo, Arvore)**’, que ensinada ao agente permite que este formate os dados para enviá-los.

Para interpretar os dados recebidos pelo sistema operacional e escrevê-los em um formato XML, utiliza-se uma especialização chamada ‘RegExpSkill’ que fornece a regra ‘**regexp(match-add, *ExpressãoRegular*, T*extoFonte*, Resultado)**’. Essa regra vai receber um ‘TextoFonte’ que é a resposta obtida do sistema operacional, e de acordo com a ‘ExpressãoRegular’ apresentada como parâmetro ela vai colher as informações e retornar como uma lista.

6.1.1 Uso de Expressão Regular

Expressões regulares auxiliam na busca de uma seqüência de caracteres específica dentro de um texto. O uso dela através de especializações tem o intuito de filtrar informação útil das respostas recebidas do sistema operacional.

Para representar seqüências de caracteres o reconhecedor de expressões utiliza símbolos que representam grupos específicos de caracteres. Abaixo uma tabela com a descrição dos mais úteis:

Símbolo	**Significado**
\w	Qualquer letra, ou seja, que pertence ao conjunto [a-z,A-Z].
\s	Espaço em branco.
\d	Um dígito numérico, ou seja, pertencente ao conjunto [0-9].
+	O caracter anterior aparece uma vez ou mais.
*	O caracter anterior aparece zero ou mais vezes.
(X)	X será guardado para posterior resgate.
[ ]	Separação em bloco.
|	Operador lógico ‘ou’.

Tabela 6.1.1 – Símbolos utilizados pelo reconhecedor de expressões regulares.

6.2 Coletando Dados

Quando a aplicação é iniciada, o grid carrega de um arquivo XML as regras básicas, que serão inseridas na base de dados global, fornecida pelo container. Para este exemplo destacamos as regras:

Essas regras ensinam as habilidades implementadas para o

```
<native>
  regexp(Cmd,Arg1,Arg2,Arg3) :-
    br.ufsc.lrg.agentgrid.skill.RegExpSkill(Cmd,Arg1,Arg2,Arg3).
</native>
<native>
  template(Cmd,Arg1,Arg2,Arg3) :-
    br.ufsc.lrg.agentgrid.skill.TemplateSkill(Cmd,Arg1,Arg2,Arg3).
</native>
<native>
  cmdline(Cmd,Arg1,Arg2) :-
  br.ufsc.lrg.agentgrid.skill.CmdLineExecuterSkill(Cmd,Arg1,Arg2).
</native>
```

container, por exemplo, ensinam a habilidade contida em br.ufsc.lrg.agentgrid.skill.CmdLineExecuterSkill(Cmd,Arg1,Arg2), para o container, que a utilizará através da regra cmdline(Cmd,Arg1,Arg2).

Para exemplificar o uso de agentes coletores, digamos que o dado a ser coletado seja a quantidade de memória do computador, então primeiro utiliza-se a regra para coleta de dados do computador, no caso, a regra ‘cmdline’, junto com um comando que retorne o dado esperado, por exemplo, ‘systeminfo’ do Windows.

Então a regra ficará:

cmdline(systeminfo,[‘/fo’,’list’],Resultado).

E o conteúdo de *Resultado* será algo como:

```
...
Memória física total:                    255 MB
Memória física disponível:               80 MB
Memória virtual: tamanho máximo:         875 MB
Memória virtual: disponível:             562 MB
Memória virtual: em uso:                 313 MB
...
```

A resposta é mais extensa, mas esses são os dados buscados. Então é necessário separar estes dados do resto do texto por meio de expressões regulares. Para conseguir o total da memória física a regra seria:

regexp(match-add, 'Memória física total:\s*(\d+\s*MB)', Resultado, MemoriaTotal).

Então a variável 'MemoriaTotal' conterá uma lista com os valores encontrados na expressão, que no exemplo é a lista que contêm o valor único ['255 MB'].

Finalmente, resta utilizar a habilidade 'TemplateSkill' para formatar o dado para saída:

template(tree, MemoriaTotal, [memoria, [total, '$1']], Arvore).

O símbolo '$1' significa que naquele local da estrutura será colocada o primeiro elemento da lista contida na variável MemoriaTotal, no exemplo o valor '255 MB'. O resultado guardado em árvore é a estrutura:

```
<memoria>
   <total>
       255 MB
   </total>
</memoria>
```

6.3 O Aprendizado do Agente

O problema da coleta de dados para um caso específico é resolvido criando uma regra de acordo com as suas características, mas essas regras só valem para um único comando de uma versão específica de um único sistema operacional, pois até em trocas de versões de um mesmo sistema acontece de haver modificações nos parâmetros ou no formato da mensagem de resposta de um comando.

Ensinar para um coletor todas as regras para que possa coletar qualquer tipo de informação não é eficiente, pois o coletor pode utilizar somente uma pequena parte delas, e sempre vai utilizar somente às regras referentes ao sistema operacional da máquina alvo. Se a necessidade de

gerência se estender a redes de computadores que possuem vários sistemas operacionais, muitas regras serão criadas que os agentes não utilizarão na coleta dos dados por pertencerem a sistemas operacionais incompatíveis.

Como solução a este problema o grid fornece aos agentes uma especialização que busca conhecimento em arquivos XML. Caso a regra para coleta de dados de um comando desejado não exista, o agente pode aprender a regra contida em um arquivo XML, obtida em um servidor conectado à Internet, possibilitando o aprendizado da regra de qualquer lugar que possua acesso à rede. Essa regra é:

grid(load,Arquivo,Agente)

ou

grid(load,Arquivo)

No exemplo acima, ‘Arquivo’ é o caminho mais o nome do arquivo no formato XML que deve ser carregado e ‘Agente’ é o nome do agente que receberá as regras, caso o nome seja suprimido, o conhecimento será guardado na base de dados global, partilhada por todos os agentes pertencentes àquele grid.

Utilizando um exemplo mais elaborado, o código a seguir recolhe informação sobre o sistema operacional e junta com a informação sobre o comando a utilizar para a coleta, carregando um arquivo cujo nome está no formato: SistemaOperacional + “ – “ + NomeDoComando + “.xml”.

```
<clause>
  learn(Command) :-
       os(OS),
       xmlPath(XMLPATH),
       agent(name,Agente),
       string(split,Arquivo,[XMLPATH,OS,'-',Command,'.xml'],''),
       __print(['carregando ',Arquivo,' no agente ',Agente,'\n']),
       grid(load,Arquivo,Agente).
</clause>
```

Ao ensinar para o container a regra xmlPath(Local), colocando em ‘Local’ o caminho onde procurar os arquivos XML, a regra acima vai recolher as informações necessárias, concatenar os strings dos resultados formando o caminho mais o nome do arquivo e vai carregá-lo no agente que executou a regra.

7. CONCLUSÃO

O ambiente de uma rede é um ambiente dinâmico. Muitas variáveis são necessárias para calcular o grau de complexidade deste ambiente, como número de usuários, recursos disponíveis, nível do trafego de dados na rede e as informações de cada computador conectado nela.

Para um ambiente complexo como este, uma solução de gerência é a aplicação de um grid de agentes, que gerenciam a rede de forma distribuída, por exemplo, na coleta de informações da rede, realizada por meio de agentes distribuídos na rede, comunicantes entre si, possuindo todo conhecimento necessário para realizar suas tarefas, e com capacidade de aprender novas tarefas quando necessário.

7.1 Considerações finais.

Como experiência prática foram coletados dados dos sistemas operacionais GNU/Linux e Windows. Os dados monitorados foram o fluxo da rede através do comando 'netstat' existente tanto no Windows como no GNU/Linux, e o volume de processos sendo executados na máquina, através dos comandos 'top' do GNU/Linux e do 'tasklist' existente no Windows. Para isso, foram feitos testes ensinando o comando na criação do agente e deixando que o agente procurasse a regra quando não tivesse uma regra que se aplicasse à necessidade do sistema.

Os dados coletados foram formatados e enviados através das habilidades implementadas e ensinadas ao agente, através da especialidade que permite o aprendizado de regras novas.

Para essas aplicações os agentes responderam eficientemente, de forma independente, nas máquinas em que foram executadas, transmitindo os dados colhidos e apresentado-os através de uma interface simples criada para acompanhar a evolução dos agentes, como mostra a figura a 7.1.

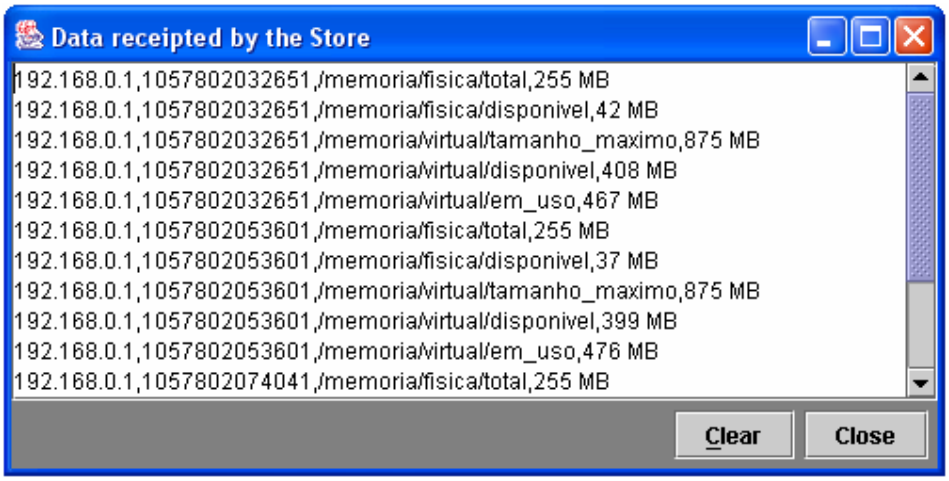

Figura 7.1 – Interface para visualização do resultado da coleta.

Na figura acima podemos distinguir quatro campos separados por vírgula, primeiro o IP da máquina na qual foi efetuada a coleta de dados, a seguir vem o horário da coleta, logo após, a posição do dado na árvore XML e por último o valor coletado. Essa interface é preenchida por meio de um agente que fica escutando uma porta da rede e recebe os dados transmitidos pelo agente coletor.

7.2 Possíveis Melhorias

Uma das possibilidades de melhoria deste trabalho é um estudo mais aprofundado, com exemplos mais complexos de coleta dos dados, bem como a utilização de mais agentes realizando outras tarefas na rede.

7.3 Sugestões para futuros trabalhos.

Uma sugestão de trabalho futuro seria a utilização de agentes para monitorar a rede de um laboratório, acompanhando a evolução dos agentes nesse ambiente durante o tempo e a apresentação dos dados colhidos em forma de análise dessa rede, analisando a performance do Grid de agentes nessa tarefa.

8. REFERÊNCIAS BIBLIOGRÁFICAS

RUSSEL, S. e NORVIG, P. **Artificial Intelligence, A Modern Approach**. USA, 1995. Editora Prentice Hall. Pag. 31-52.

BRISA - **Gerenciamento de redes - uma abordagem de sistemas abertos**, Makron books do Brasil, Brasil, 1993.

KOCH, F. L. **Agentes Autonômos para Gerenciamento de Redes de Computadores**. Florianópolis, 1997. Trabalho de Conclusão de Curso (Mestrado em Ciências da Computação). Universidade Federal de Santa Catarina.

KOCH, F. L. e WESTPHALL, C. B.. **Decentralized Network Management using Distributed Artificial Intelligence**. USA, 2001. Journal of Network and Systems Management. Plenum Publishing Corporation. p. 291-313.

KOCH, F. L.; MEYER, J-J C.. **Project AgentLight: Developing Logic-based Autonomous Agents for Small Devices**. Recife, 2002. In: I Workshop De Teses E Dissertações Em Inteligência Artificial -WTDIA'02, Proceedings.

ASSUNCAO, M.; WESTPHAL, C. B.; KOCH, F. **Arquitetura de Grids de Agentes Aplicado ao Gerenciamento de Redes de Computadores e de Telecomunicação**. Rio de Janeiro, 2003. In: Simposio Brasileiro De Redes De Computadores, Proceedings, p. 789-804.

Foundation for Intelligent Physical Agents. **FIPA ACL Message Structure Specification**. Geneva, Switzerland, 2000. http://www.fipa.org.

JENNINGS, N. R.; WOOLDRIDGE, M. **Applications of Intelligent Agents**. Springer-Verlag, 1998. In: Agent Technology: Foundations, Applications and Markets. p. 3-28.

BRANTSCHEN S.; HAAS, T. **Agents in a J2EE World**. Technical report, *Whitestein Technologies AG*, 2002.

OMG Agent Working Group. ***Agent technology green paper***. Technical Report ec/2000-03-01, Object Management Group, mar. 2000.

KAHN L. M; Cicalese, C. D. T. **CoABS Grid Scalability Experiments**, Autonomous Agents 2001 Conference. Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS. May 29, 2001

TVEIT, A. **jfipa - an Architecture for Agent-based Grid Computing**. London, 2002.In: AISB'02 Convention, Symposium on AI and Grid Computing, Proceedings.

9. Anexo

9.1 Artigo

Agentes Coletores para Grids de Agentes na Gerencia de Redes de Computadores

Eduardo Erlê dos Santos

Laboratório de Redes e Gerência

Universidade Federal de Santa Catarina

Florianópolis, SC, 88049-970, Caixa Postal 476, Brasil

erle@lrg.ufsc.br

Resumo. Ambientes interligados em rede possuem, em sua maioria, uma grande variedade de equipamentos, e de sistemas de gerência independentes para esses equipamentos. Uma possibilidade para a gerência de uma rede de computadores é o uso de Grids de Agentes, onde um grid é uma arquitetura distribuída que compartilha os recursos de cada máquina dando a impressão ao usuário de um grande computador virtual com muitos recursos. Essa arquitetura pode ser implementada por meio de agentes, que descentralizam as funcionalidades que compõem a gerência da rede e se valem de técnicas de inteligência artificial para conquistar autonomia em sua tarefa. O primeiro passo para a gerência é a coleta dos dados necessários para avaliar a performance da rede. Para isso buscamos utilizar agentes coletores para a obtenção dos dados, abstração do sistema do computador alvo e para a formatação da informação padronizando e facilitando a ato da gerência.

Abstract. Computer network systems are composed by several different equipments that have their own management system associated. One possibility for the management task is to use Grids of Agents where the 'grid' is a distributed architecture that aggregates the resources from several machines forming a resourceful virtual system. This architecture can be implement through autonomous agents systems that distribute the management activities and make use of artificial intelligence concepts to reach autonomy. The first step of a management system is collecting the data set from the network's devices. In this work we present a solution by using 'collector agents' that act closer to the managed devices by extracting data and formatting into an abstract representation language, thus distributing the management task.

Palavras-chave: Agentes Inteligentes, Computação Distribuída, Gerência de Redes de Computadores

1. Introdução

O aumento da complexidade das redes de computadores pelo seu crescimento numérico, ou pela diversidade de seus componentes, tem dificultado a atividade de gerência do sistema. Uma das dificuldades está em isolar e identificar os problemas das redes, devido à diferença de nível entre o pessoal envolvido, tais como: técnicos de manutenção, operadores de rede, gerentes de sistema de informações e gerentes de comunicação. Também há dificuldade pela diversidade de formas de controle e monitoração, porque embora os produtos envolvidos na rede se tornem gradativamente mais complexos, cada fornecedor oferece ferramentas próprias de controle para monitorar seus produtos.

Esse cenário exige um grande esforço da equipe encarregada de administrar uma rede de computadores, gerando uma busca pela automatização de processos, tentando diminuir esforço humano o máximo possível e da maneira mais segura e confiável, mantendo o funcionamento adequado da rede e evitando o custo de uma paralisação no sistema.

Uma das possibilidades para modelagem desse sistema de gerência de redes é o uso de um grid de agentes, que consiste em um sistema formado por comunidades de agentes distribuídos pela rede, cada um com a sua tarefa específica. Essas tarefas podem ser coletar os dados de um computador conectado a essa rede, organizar os dados coletados dos vários computadores de uma rede e gerenciá-los, ou apresentá-los por meio de uma interface amigável para as pessoas encarregadas da gerencia da rede.

2. Grids

Um grid é uma arquitetura que distribui os recursos computacionais de uma rede, buscando extrapolar o limite de um computador, possibilitando que um computador que tenha a necessidade de mais recursos, possa recorrer a algum computador da rede que possua esses recursos disponíveis. Esse comportamento faz com que um grid passe a impressão de um grande computador virtual com tantos recursos quantos os que estão disponíveis nesta rede.

Um grid provê uma maneira de explorar os recursos subutilizados em uma rede criando a possibilidade de um substancial aumento na eficiência do uso de recursos disponíveis. Esses recursos não são somente processamento ou espaço em disco, também pertencem ao conjunto de recursos compartilháveis em um grid qualquer dispositivo que faça parte da rede que implementa o grid.

Uma das dificuldades para implementação do grid é a característica heterogênea de um sistema de rede, que geralmente possui equipamentos e programas de várias empresas distintas, necessitando de uma busca pela padronização e integração do ambiente de rede.

2.1 Utilização de Grids de Agentes

Uma possibilidade para a implementação de um grid é a realização de um Grid de Agentes, que distribui os agentes no ambiente de rede, descentralizando as funcionalidades que compõem a gerência da rede, e utilizando técnicas de inteligência artificial para conquistar autonomia em sua tarefa. Esses agentes podem ser flexíveis o suficiente para se adaptar a sistemas operacionais específicos de computadores, ou a qualquer programa de controle de um equipamento que faça parte desta rede, e pode gerenciar esse recurso de forma inteligente.

Segundo Assunção, Westphal e Koch (2003), são identificados os serviços necessários para o grid. Esses serviços agrupam funcionalidades e metas para serem alcançadas pelos agentes, e formam grids com funções específicas. Este trabalho classifica os grids de agentes em *grids de agentes coletores* (GC), responsáveis por coletar os dados necessários para a gerência da rede, *grids de agentes classificadores e de armazenamento* (GCL), que são responsáveis pelas atividades de classificação, indexação e armazenamento dos dados coletados; *grids de agentes analisadores* (GP), preocupados com a análise das informações da rede, e *grid de interface* (GI), responsável pela interação com usuário ou com outros sistemas de gerência.

O foco deste trabalho é nos agentes de coleta de dados do grid, ou seja, no funcionamento do grid de agentes coletores. Nos capítulos seguintes focaremos o trabalho sobre esses temas justificando o seu uso e demonstrando uma forma de implementá-los.

3. Agentes Coletores

Dentro da aplicação apresentada anteriormente, existe o interesse de implementar um aplicativo para a primeira fase: a coleta de dados. Este aplicativo deve ser o mais flexivel o possível na forma como o dado é capturado e sua interpretação inicial, encapsulando a informação em um formato padrão que é usado para transferência de dados dentro do grid. As informações extraídas podem apresentar formatos bastante heterogêneos e por isso é necessário criar uma representação comum para estes dados. Esta representação pode ser feita usando XML. Neste caso, será garantida que a representação das informações coletadas da rede poderá ser interpretada corretamente pelo *grid* de agentes que irá recebê-las.

Este trabalho foi implementado utilizando-se a plataforma AgentLight, de Koch e Meyer (2002), como base de desenvolvimento para os agentes. O AgentLight foi desenvolvido em Java, rodando em ambiente Java J2SE e J2ME, utiliza uma máquina de inferência de lógica de primeira ordem – tal como PROLOG – e permite a extensão de suas funcionalidades através de criação de '*Skills*', em código Java, que podem, posteriormente, ser agregados a base de conhecimento do sistema e acessado pela máquina de inferência.

Um agente coletor é dotado de uma base de conhecimento com regras que o permitem coletar dados usando o protocolo requerido. Estes agentes podem ter um ou mais objetivos que consistem em extrair valores de objetos gerenciados de um ou mais equipamentos da rede em intervalos de tempo. Assim como um agente pode interagir com um ou mais dispositivos da rede, pode ocorrer de vários agentes extraírem dados de um único dispositivo. O *grid* coletor pode conter agentes que executam algumas análises locais destas informações. Podemos definir agentes que tenham como objetivos localizar alguns problemas mediante a análise destes dados.

3.1 Definição dos agentes

Os agentes utilizados serão agentes reflexivos, conforme descrito por Russell e Norvig (1995). Para esse agente não existe passado nem futuro, pois as suas ações são baseadas nas informações colhidas pelo sensor naquele instante e atua no meio através das regras contidas na sua base de conhecimentos. Por isso, quando cessa a percepção do ambiente cessa a ação.

Alguns outros conceitos adotados de Koch e Westphal (2001):

a. **Agentes devem ser realmente autônomos:** Os agentes devem ser capazes de controlar a si mesmo e a suas ações. Eles devem ser capazes de trabalhar independente do que aconteça com outros agentes, tendo total liberdade para adaptar-se a possíveis mudanças no ambiente.

b. **Agentes podem ser guiados por metas e baseados em regras:** Um conjunto de regras constitui a base de conhecimento para qualquer sistema de administração – incluindo os humanos. A base de conhecimento pode ser constituída igualmente por metas, que também são definidas através de regras, e que direcionam os trabalhos do agente.

c. **Uma aplicação de agentes deve se comportar como uma 'base de dados globais':** onde cada dado ou informação, armazenada por um agente é compartilhado por todo o sistema (outros agentes). A comunicação torna a aplicação mais robusta, favorecendo a intercomunicação entre os agentes e facilitando a troca de conhecimentos. Igualmente, um agente pode aprender uma nova regra de outra comunidade e partilhar informação com a sua própria, aumentando a base de conhecimentos de sua comunidade.

d. **Agentes devem ser capazes de aprender novas habilidades (skills):** Onde essas regras podem ser aprendidas por meio de outros agentes, da comunidade de agentes, da comunidade global ou, em ultimo caso, do ser humano. Essas novas regras podem ser armazenadas dinamicamente nas bases locais de conhecimento dos agentes. Outra possibilidade é inferir novas regras a partir do conhecimento atual do ambiente.

3.2 Arquitetura do agente coletor

A estrutura de um agente coletor é talvez a mais simples do conjunto de gerência da rede. O coletor é um agente reflexivo, não tem passado nem futuro, com a única função de coletar periodicamente os dados desejados de uma estação da rede, formatá-los e enviá-los à aplicação gerente que analisará os dados recebidos do coletor.

Os conhecimentos necessários para que este agente cumpra a sua tarefa seriam sobre a forma como ele vai coletar os dados, como ele vai formatá-los e como vai enviá-los. O conhecimento sobre a troca de mensagem entre agentes já está contido no agente, pois é fornecida pela plataforma sobre a qual o agente é implementado.

Para coletar os dados, primeiro é necessário definir por que meios se dará à coleta. Como exemplos podemos citar a coleta por meio de um protocolo conhecido, sendo necessário ensinar o agente a se comunicar nesse protocolo, ou por meio da interface oferecida pelo sistema operacional do computador alvo. Os exemplos são coleta por linha de comando administrativo do sistema operacional ou protocolo de gerenciamento, como o SNMP.

Como um exemplo adotado, no método de coleta de dados por meio da linha de comando do sistema operacional não é necessário o conhecimento específico de nenhum protocolo, o que não é objetivo deste trabalho, e facilita no teste da aplicação, pois toda máquina possui um sistema operacional, mas nem toda máquina executa um programa de gerência de rede com os protocolos necessários para coleta de dados.

Para coletar os dados através de linha de comando foi utilizado uma habilidade chamada *CmdLineSkill*, que possui a regra:

```
cmdline(comando,[lista,de,parâmetros],Resultado)
```

Para formatar a resposta foi escolhida a linguagem XML, por ser uma linguagem atualmente bem aceita e de fácil utilização. Sendo assim, todos os dados colhidos serão transformados em um texto XML para serem transmitidos para os agentes encarregados de analisar os dados. Para formatar os dados em uma árvore XML, existe uma habilidade chamada 'TemplateSkill', que gera dados formatados a partir de uma estrutura fornecida como modelo. 'TemplateSkill' fornece a regra:

`template(tree, RegExpResultados, FormatoModelo, Arvore)`'.

Para interpretar os dados recebidos pelo sistema operacional e escrevê-los em um formato XML, utiliza-se uma habilidade chamada 'RegExpSkill' que fornece a `regra 'regexp(match-add, Expressão Regular, Texto Fonte, Resultado)`'. Essa regra vai receber um 'Texto Fonte' que é a resposta obtida do sistema operacional, e de acordo com a 'Expressão Regular' apresentada como parâmetro ela vai colher as informações e retornar como uma lista.

3.3 Aprendizado

O problema da coleta de dados para um caso específico é resolvido criando uma regra de acordo com as suas características, mas esta ainda não é genérica, pois as regras só valem para um único comando do sistema operacional. Cada comando é chamado com seus parâmetros específicos e retorna uma mensagem com o seu formato, por isso, os parâmetros das regras do coletor devem se ajustar para cada situação.

Ensinar um coletor todas as regras para que possa coletar qualquer tipo de informação não é eficiente, pois o coletor pode utilizar somente uma pequena parte delas. Se a necessidade de gerência se estender a redes de computadores que possuem vários sistemas operacionais, muitas regras serão criadas que os agentes não utilizarão na coleta dos dados por pertencerem a sistemas operacionais incompatíveis.

Como solução a este problema o Grid fornece aos agentes uma habilidade que busca conhecimento em arquivos XML. Caso a regra para coleta de dados de um comando desejado não exista, o agente pode aprender a regra contida em um arquivo XML, obtida em um servidor conectado à Internet, possibilitando o aprendizado da regra de qualquer lugar que possua acesso a Internet.

4. Coletando Dados

Durante a inicialização do sistema as regras básicas de operação são carregadas na base de conhecimento do agente coletor. Estas regras são o próprio conhecimento de operação do agente. Como descrito anteriormente, novas regras podem ser aprendidas posteriormente, mas as regras inicializadas são suficiente para as atividades de coleta de dados via SNMP, linha de comando e interpretação (*parsing*) dos dados via expressão regular. Todas estas atividades são implementadas com '*Skills*' no sistema, em Java, por razões de performance, mas acessiveis pela máquina de inferência.

As regras básicas carregadas são:

```
regexp(Cmd,Arg1,Arg2,Arg3) :-
        br.ufsc.lrg.agentgrid.skill.RegExpSkill(Cmd,Arg1,Arg2,Arg3).
template(Cmd,Arg1,Arg2,Arg3) :-
        br.ufsc.lrg.agentgrid.skill.TemplateSkill(Cmd,Arg1,Arg2,Arg3).
cmdline(Cmd,Arg1,Arg2) :-
        br.ufsc.lrg.agentgrid.skill.CmdLineExecuterSkill(Cmd,Arg1,Arg2).
```

Apresentamos aqui um exemplo de coleta via linha de comando para demonstrar o funcionamento e a troca de mensagens no sistema.

4.1 Coleta via linha de comando

Digamos que o dado a ser coletado seja a quantidade de memória do computador, então primeiro utiliza-se à regra para coleta de dados do computador, no caso, a regra 'cmdline', junto com um comando que retorne o dado que queremos, por exemplo, 'systeminfo' do Windows. Então a regra ficará:

```
cmdline(systeminfo,['/fo','list'],Resultado).
```

E o conteúdo de *Resultado* será algo como:

```
...
Memória física total:                      255 MB
Memória física disponível:                 80 MB
Memória virtual: tamanho máximo:           875 MB
Memória virtual: disponível:               562 MB
Memória virtual: em uso:                   313 MB
...
```

A resposta é mais extensa, mas esses são os dados almejados. Então é necessário separar estes dados do resto do texto por meio de expressões regulares. Expressões regulares auxiliam na busca de

uma seqüência de caracteres específica dentro de um texto. O uso dela através de habilidades tem o intuito de filtrar informação útil das respostas recebidas do sistema operacional. Para conseguir o total da memória física a regra seria:

```
regexp(match-add, 'Memória física total: (d+\s*MB)',
              Resultado,
              MemoriaTotal).
```

Então a variável 'MemoriaTotal' conterá uma lista com os valores encontrados na expressão, que no exemplo é a lista que contêm o valor único ['255 MB']. Finalmente, utilizamos o 'TemplateSkill' para formatar o dado para saída:

```
template(tree, MemoriaTotal,
              [memoria, [total, '$1']],
              Arvore).
```

O símbolo '$1' significa que naquele local da estrutura será colocada o primeiro elemento da lista contida na variável 'MemoriaTotal', no exemplo o valor '255 MB'. O resultado guardado em árvore é a estrutura:

```
<memoria>
   <total> 255 MB </total>
</memoria>
```

Esta estrutura de árvore é então adicionada como 'galho'de uma super-árvore que formará o 'conteúdo' do pacote de comunicação. Este pacote será então enviado para o agente de armazenamento, através da estrutura de comunicação da plataforma de agentes utilizada. No grid de agentes armazenadores ela é então interpretada e armazenada apropriadamente, num processo externo ao que interessa para este trabalho.

5. Conclusão

O ambiente de uma rede é um ambiente dinâmico. Muitas variáveis são necessárias para calcular o grau de complexidade deste ambiente, como número de usuários, recursos disponíveis, nível do trafego de dados na rede e as informações de cada computador conectado nela.

Para um ambiente complexo como este, uma solução de gerência é a aplicação de um grid de agentes, que gerenciam a rede de forma distribuída, por exemplo, na coleta de informações da rede, realizada por meio de agentes distribuídos na rede, comunicantes entre si, possuindo todo conhecimento necessário para realizar suas tarefas, e com capacidade de aprender novas tarefas quando necessário.

Para comprovar a viabilidade da utilização de agentes, foram feitos testes ensinando o comando na criação do agente, e deixando que o agente procurasse a regra quando não existisse uma que se aplicasse à necessidade do sistema.

Os dados coletados foram formatados e enviados através das habilidades implementadas e ensinadas ao agente, através da especialidade que permite o aprendizado de regras novas.

Para essas aplicações os agentes responderam eficientemente, de forma independente, nas máquinas em que foram executadas, transmitindo os dados colhidos e apresentado-os através de uma interface simples criada para acompanhar a evolução dos agentes.

Com o resultado apresentado, seria uma possibilidade de melhoria deste trabalho um estudo mais aprofundado, com exemplos mais complexos de coleta dos dados, bem como a utilização de mais agentes realizando outras tarefas na rede. Igualmente, uma sugestão de trabalho futuro seria a utilização

de agentes para monitorar a rede de um laboratório, acompanhando a evolução dos agentes nesse ambiente durante o tempo e a apresentação dos dados colhidos em forma de análise dessa rede, analisando a performance do Grid de agentes nessa tarefa.

6. Referências Bibliográficas

RUSSEL, S. e NORVIG, P. **Artificial Intelligence, A Modern Approach**. USA, 1995. Editora Prentice Hall. Pag. 31-52.

BRISA - **Gerenciamento de redes - uma abordagem de sistemas abertos**, Makron books do Brasil, Brasil, 1993.

KOCH, F. L. **Agentes Autonômos para Gerenciamento de Redes de Computadores**. Florianópolis, 1997. Trabalho de Conclusão de Curso (Mestrado em Ciências da Computação). Universidade Federal de Santa Catarina.

KOCH, F. L. e WESTPHALL, C. B.. **Decentralized Network Management using Distributed Artificial Intelligence**. USA, 2001. Journal of Network and Systems Management. Plenum Publishing Corporation. p. 291-313.

KOCH, F. L.; MEYER, J-J C.. **Project AgentLight: Developing Logic-based Autonomous Agents for Small Devices**. Recife, 2002. In: I Workshop De Teses E Dissertações Em Inteligência Artificial -WTDIA'02, Proceedings.

ASSUNCAO, M.; WESTPHAL, C. B.; KOCH, F. **Arquitetura de Grids de Agentes Aplicado ao Gerenciamento de Redes de Computadores e de Telecomunicação**. Rio de Janeiro, 2003. In: Simposio Brasileiro De Redes De Computadores, Proceedings, p. 789-804.

Foundation for Intelligent Physical Agents. **FIPA ACL Message Structure Specification**. Geneva, Switzerland, 2000. http://www.fipa.org.

JENNINGS, N. R.; WOOLDRIDGE, M. **Applications of Intelligent Agents**. Springer-Verlag, 1998. In: Agent Technology: Foundations, Applications and Markets. p. 3-28.