UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

Grids de Agentes Processadores para Gerência de Redes de Computadores e Telecomunicações

AUTORA: Maria Tereza Nagel

ORIENTADOR: Carlos Becker Westphall, Dr.

BANCA EXAMINADORA: Marcos Dias de Assunção, Msc.

Mário Dantas, PhD.

PALAVRAS-CHAVE: agentes processadores, *grids* de agentes, gerência de redes.

Florianópolis, 08 de julho de 2004.

AGRADECIMENTOS

Agradeço ao meu professor e orientador Dr. Carlos Becker Westphall pelo apoio e ajuda neste trabalho de conclusão de curso. Agradeço também ao mestre Marcos Dias de Assunção pela grande paciência que teve comigo e auxílio na condução deste trabalho. E sem esquecer do professor Dr. Mário Dantas, obrigada pela participação na banca.

Quero agradecer meus pais e minha irmã por sempre me incentivarem a continuar e lutar mesmo nos momentos difíceis. Obrigada também pelo carinho e educação que vocês me deram.

Ao meu namorado, obrigada pelas correções, apoio emocional e por estar ao meu lado em todos os momentos.

Aproveito para agradecer também meus amigos e colegas da faculdade, em especial ao CCO002, pelos momentos de descontrações, estudos e companhia. Vocês são muito especiais para mim e estarão pra sempre guardados no meu coração.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

API: *Application Programming Interface*
CIM: *Common Information Model*
CoABS: *Control of Agent-Based Systems*
DARPA: *Defense Advanced Research Projects Agency*
DLL: *Dynamic Link Library*
DMTF: *Distributed Management Task Force*
FIPA: *Foundation for Intelligent Physical Agents*
HTTP: *Hypertext Transfer Protocol*
IA: Inteligência Artificial
IAD: Inteligência Artificial Distribuída
ISO: *International Organization for Standardization*
MAGE: *Multi-AGent Environment*
OSI: *Open Systems Interconnection*
P2P: *peer-to-peer*
PAD: Processamento de Alto Desempenho
RDP: Resolução Distribuída de Problemas
SLA: *Service Level Agreement*
SMA: Sistemas Multi-Agentes
SMTP: *Simple Network Management Protocol*
VO: Organização Virtual
WBEM: *Web-Based Enterprise Management*
WMI: *Windows Management Instrumentation*
XML: *Extensible Markup Language*

RESUMO

O uso de gerenciamento centralizado de redes de computadores e telecomunicações pode levar a situações em que um grande volume de dados precisa ser analisado por uma única estação de gerenciamento. Uma possibilidade para a solução deste problema é a utilização de uma arquitetura baseada em *grids* de agentes para a gerência de redes. Neste trabalho é apresentada uma arquitetura com esta abordagem. Ela utiliza agentes para a construção do *grid.* Desta maneira, as funcionalidades que compõem a gerência podem ser descentralizadas, solucionando problemas como a manipulação e análise das informações.

Uma das etapas mais importantes no gerenciamento de redes e sistemas computacionais é a análise das informações coletadas, o foco deste trabalho. Para isto, utiliza-se agentes de processamento que são responsáveis por realizar as inferências sobre os dados armazenados em busca de problemas que possam estar ocorrendo na rede gerenciada. O resultado da análise é composto por informações de gerência que darão origem aos relatórios e alertas para serem enviados aos agentes de interface. Os resultados indicam que a abordagem atinge com sucesso os objetivos a que se propõe.

Palavras-Chaves: agentes processadores, *grids* de agentes, gerência de redes.

ABSTRACT

The management centralized use of computer and telecommunications networks can take to situations where great amounts of data has to be analyzed by a single management station. A possibility to solve this problem is the use of an architecture based on grids of agents. This approach uses agents of software to build the grid. Though, the features that compounds the management can be decentralized solving problems like manipulation and information analysis.

One of the most important steps in network and computational systems management is the information analysis. In order to that, it is utilized processors agents that are responsible to make inferences over stored data searching for problems that might be ocurring at managed network. The analysis' result is composed by management informations that will create reports and alerts to be sent to interfaces' agents. Results show that this approach have success in their objectives.

Keywords: processors agents, grids of agents, network management.

1 – Introdução

As redes de computadores estão se tornando indispensáveis nas instituições. Na tentativa de se minimizar a ocorrência de falhas e diminuir o tempo de parada dos serviços, busca-se um gerenciamento eficiente sobre as mesmas. Porém, com o crescente aumento no tamanho e complexidade destas redes, as soluções de gerência centralizada tornam-se incapazes de proporcionar um bom desempenho porque aumentam também o volume de dados coletados e que precisam ser analisados. Transformar estes dados em informações e relatórios que demonstrem as condições de operação da rede e seus serviços, ou que possam indicar eventuais problemas, pode se tornar uma tarefa intensiva, exigindo um grande poder de processamento das estações envolvidas no gerenciamento da rede.

Tomando a gerência de redes como uma aplicação em *grid*, onde existe um grande volume de dados que precisa ser transformado em informações de gerência de alto nível, encontra-se um cenário onde *grids* de agentes podem ser aplicados.

Computação em *grid*, ou *grids* computacionais, são uma forma de acesso universal, facilitado, confiável e seguro a agrupamentos dinâmicos de recursos computacionais heterogêneos dispersos geograficamente [Foster e Kesselman 1999]. *Grids* de agentes são, basicamente, *grids* que utilizam como infra-estrutura sistemas de agentes de software. O sucesso de uma arquitetura de *grid* está relacionado à forma com que ela gerencia os recursos computacionais

disponíveis. Mais especificamente à forma como é realizado o balanceamento das cargas de trabalho aplicadas ao *grid*.

Assunção [2003] propôs, utilizando *grids* de agentes, uma arquitetura para o gerenciamento de redes, que através da distribuição das tarefas de manipulação e análise dos dados proporciona maior extensibilidade e adaptabilidade que os modelos centralizados tradicionalmente utilizados. Esta utilização de *grids* de agentes se mostra interessante na gerência, pois além de distribuir tarefas ligadas ao gerenciamento, possibilita uma distribuição da análise e do armazenamento das informações, conforme descrito em simulações [ASSUNÇÃO 2003] que demonstram os resultados esperados da arquitetura.

1.1 – Caracterização do Problema

Levando em consideração o fluxo de trabalho tradicional do gerenciamento, algumas atividades foram identificadas: coleta de dados dos dispositivos da rede, classificação e armazenamento destes dados, análise dos dados a fim de transformá-los em informações de gerência e apresentação destas informações.

Na situação onde os dados extraídos precisam ser transformados em informações de gerência é exigido um alto poder de processamento. Com a utilização de uma arquitetura de *grids* baseada em agentes, é possível efetuar uma distribuição e um balanceamento de carga da análise dos dados coletados, baseando-se na capacidade dos recursos do *grid* e utilizando os recursos ociosos. Dessa forma, um grande volume de dados pode ser analisado de maneira confiável, mesmo quando não se possui recurso financeiro para uma atualização constante do hardware.

A principal dificuldade é proporcionar uma divisão das atividades de análise no *grid*. Distribuir esta análise sem que as informações percam a sua essência é uma tarefa complicada. A divisão do conjunto de dados “problema” de

aplicações distribuídas tem sido alvo de vários estudos e, na gerência de redes, esta divisão pode ser um pouco complicada, devendo ser tratada para não proporcionar uma perda de significado das informações. Daí a existência do *grid* classificador, responsável por preparar as informações para que as tarefas de análise possam ser distribuídas mais facilmente. A raiz do *grid* de processamento coordena esta distribuição, funcionando como uma espécie de *broker*, no sistema.

O *grid* de processamento ou análise é a parte mais importante da arquitetura e onde estão localizados os maiores desafios de desenvolvimento. É responsável pela consolidação dos dados coletados em relatórios de gerência. Os principais problemas que podem surgir dizem respeito à divisão das atividades de análise, controle de recursos, balanceamento de carga e tolerância à falhas. O resultado da análise é composto por informações de gerência que dão origem aos relatórios e alertas enviados ao *grid* de interface.

1.2 – Objetivos

Este trabalho tem como objetivo geral a criação de agentes processadores para o *grid* de gerência proposto pelo Laboratório de Redes de Gerência da Universidade Federal de Santa Catarina [ASSUNÇÃO 2003].

Como objetivos específicos, pode-se citar:

- Levantamento teórico e aprendizado de conceitos relacionados à computação em *grid,* agentes autônomos, computação baseada em agentes, gerência de redes e lógica de primeira ordem.
- Conhecimento da plataforma Agentlight [AGENTLIGHT], na qual serão desenvolvidos os protótipos de *grids* de agentes. Esta atividade tem por objetivo propiciar um contato com a plataforma, facilitando o desenvolvimento dos protótipos.
- Desenvolvimento das regras de análise baseadas em estudos de casos de gerência. O objetivo é criar regras para tratar os dados

coletados dos equipamentos. Estas regras visam procurar por problemas nos equipamentos e dão origem a relatórios e alertas que são apresentados aos usuários pelos agentes de interface.

1.3 – Estado da Arte

As aplicações citadas por alguns como as primeiras tentativas de computação em *grid* são a análise de sinais de rádio na busca de vida extraterrestre inteligente (SETI@Home) [SETI@HOME], o superprocessador virtual proporcionado pela *United Devices* e o *Distributed.Net* [DISTRIBUTED.NET]. Estes três exemplos de aplicações utilizam o mesmo princípio, que consiste em decompor um conjunto de dados referente ao problema a ser resolvido em arquivos que podem ser baixados por uma aplicação instalada no computador do usuário, geralmente sob a forma de um protetor de tela, e após serem processados durante os períodos de ociosidade do computador do usuário, são devolvidos sob a forma de arquivos de resultados. Embora atualmente estas aplicações sejam consideradas aplicações *peer-to-peer* (P2P) por alguns, elas são citadas por muitos como exemplos de *grid* e também como tecnologias precursoras.

Uma segunda corrente de desenvolvimento do *grid* iniciou com a tentativa de criar *middlewares* que facilitem a computação em larga escala, ou em *grid*, proporcionando gerenciamento de recursos e APIs para desenvolvimento de aplicações. Alguns trabalhos pioneiros são o *Legion* [LEGION] e o *Globus* [FOSTER, KESSELMAN 1997]. O *Globus* é um *middleware* que proporciona uma pilha de quatro camadas para controle de hardware, comunicação, compartilhamento de recursos e coordenação de tarefas.

1.3.1 – Estado da Arte em *Grids* de Agentes

Ao se encontrar o termo *grid* de agentes relacionado em algum trabalho, percebe-se que existem duas situações distintas, um *grid* de agentes como um sistema, e um *grid* como uma espécie de *middleware* que proporciona a interoperabilidade entre diferentes plataformas de agentes [ASSUNÇÃO 2004].

Um dos trabalhos mais expressivos sobre *grids* de agentes é parte do projeto CoABS (*Control of Agent-Based Systems*) [COABS]. O CoABS é um programa do DARPA (*Defense Advanced Research Projects Agency*) e do *US Air Force Rome Labs.* O programa tem por objetivo investigar o uso da tecnologia de agentes para melhorar operações militares de comando, controle, comunicação, acúmulo e acesso à informações importantes no planejamento de ações militares. O *grid* CoABS também é responsável por facilitar a integração dinâmica dos vários sistemas desenvolvidos pelos pesquisadores do projeto CoABS no estabelecimento de coalizões construídas em campos de batalha.

Zhongzhi e outros [ASSUNÇÃO 2004] apresentam um modelo de *grid* baseado em agentes, fazendo uso da plataforma *Multi-AGent Environment* (MAGE), usada para a construção dos agentes. O modelo de computação em *grid* baseada em agentes apresentada neste trabalho se baseia nos padrões FIPA [FIPA] para garantir a interoperabilidade com outras plataformas. O MAGE é citado como a primeira plataforma de agentes desenvolvida por um grupo asiático a se conectar ao projeto Agentcities [WILLMOTT 2001]. São citados dois exemplos de aplicações usando o modelo de *grid* proposto: uma aplicação de *e-Business* e uma segunda aplicação envolvendo uma cadeia de fornecimento de petróleo.

O *Agentcities* [WILLMOTT 2001] é um projeto bastante amplo e tem por finalidade possibilitar a criação de um ambiente de agentes em larga escala, onde os pesquisadores poderão conectar suas plataformas de agentes e fazer uso dos serviços oferecidos. Esta integração será possível com a conformidade com os

padrões estabelecidos pela FIPA. O *Agentcities* faz uso de tecnologia de agentes e de *grid* para a criação deste ambiente.

A exemplo do projeto *Agentcities* percebe-se um interesse constante em fazer com que os agentes possam ser empregados em ambientes de larga escala. Muitos grupos de pesquisa europeus e americanos acreditam que nos próximos anos, a tecnologia de agentes terá um papel importante nas aplicações de larga escala, como apresenta Luck em seu *roadmap* [LUCK 2003]. O sonho de que a Internet será habitada por agentes de software tende a se concretizar. Passos importantes no sentido de garantir a interoperabilidade entre as plataformas de agentes existentes têm sido dados por instituições como a FIPA. Neste cenário de larga escala, assim como em um *grid*, um conjunto de protocolos padrão que garantam a interoperabilidade e a formação das organizações virtuais [FOSTER, KESSELMAN, TUECKE 2001] é extremamente importante.

1.4 – Organização do Trabalho

O conteúdo deste trabalho está organizado em 7 capítulos assim distribuídos:

- Neste primeiro capítulo é apresentada uma introdução sobre o trabalho, caracterização do problema e estado da arte, a fim de contextualizar o tema da pesquisa.
- No segundo capítulo são apresentadas algumas definições sobre gerenciamento de redes de computadores, suas etapas e divisão funcional. Além disso, é dada uma breve introdução ao WBEM e ao WMI devido à utilização destas tecnologias para a formulação das regras de análise.
- O terceiro capítulo apresenta alguns conceitos relacionados a agentes de software e sistemas multiagentes. São descritas características e classificações relacionadas a este tema.

- O quarto capítulo descreve a computação em *grid,* suas características, os *grids* de agentes, e também o uso de *grids* de agentes no gerenciamento de rede. Além disso, é dada uma breve introdução à arquitetura de agentes proposta para a gerência de sistemas.
- O quinto capítulo apresenta o ambiente experimental e suas características.
- No sexto capítulo é apresentado especificamente o *grid* de agentes de processamento para a gerência de redes.
- As considerações finais e trabalhos futuros são apresentados no sétimo e último capítulo.

2 – Gerência de Redes

Em uma rede de computadores a transmissão dos dados deve ser realizada de maneira confiável e eficiente para garantir a qualidade dos serviços. Deste modo, surge a necessidade de monitorar cada componente desta rede (máquinas que executam as aplicações, roteadores, sistema operacional, protocolos, entre outros), verificando se esta responde às solicitações conforme esperado. Este monitoramento é chamado gerência de redes. Segundo Subramanian [2000], o objetivo da gerência de redes é garantir que os usuários recebam os serviços de tecnologia da informação com a qualidade que eles esperam.

A idéia básica na solução de um sistema de gerência consiste na utilização de um computador interagindo com os diversos componentes da rede a serem gerenciados, para ser possível extrair desses, as informações necessárias à gerência [GUBERT 2002]. Utilizando uma ferramenta de gerência de rede o administrador pode entre outras coisas, controlar melhor o ambiente verificando o uso dos recursos computacionais, planejar um estudo de capacidade, definir medidas pró-ativas na detecção de “gargalos” encontrados no ambiente, além de ser notificado imediatamente ao ocorrer uma falha em um dos elementos gerenciados. A notificação imediata do problema contribui com a minimização do tempo de parada do serviço prestado, uma vez que a equipe de suporte pode prontamente dar início à resolução do problema no recebimento da notificação [LIMA 2003].

2.1 – Etapas do Processo de Gerenciamento

As etapas no processo de gerência de redes [STALLINGS 1999], como se pode analisar na Figura 1, são basicamente:

- Coleta de dados: processo geralmente automático que é responsável por extrair as informações de gerência dos equipamentos.
- Diagnóstico: analisa através de alguns procedimentos automáticos, ou por intermédio de um operador, os dados coletados a fim de determinar causas dos problemas apresentados.
- Ação: após determinar a causa do problema, age-se para solucioná-lo na tentativa de evitar seu reaparecimento.

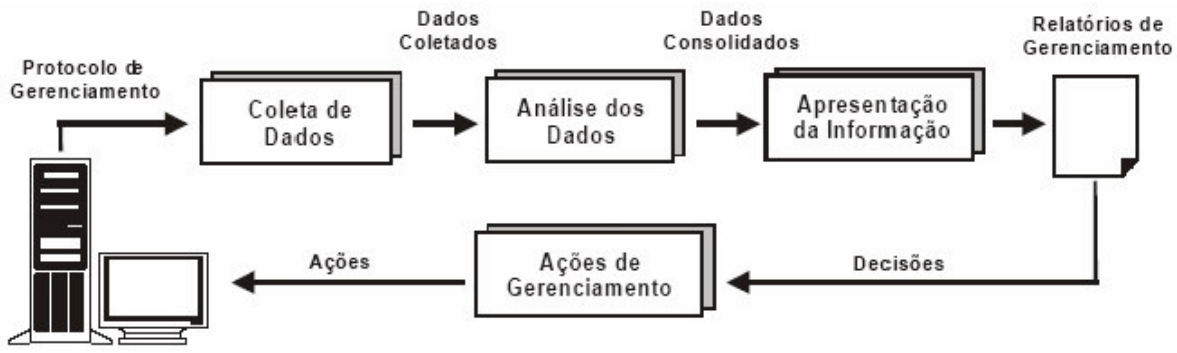


Figura 1 - Fluxo de um sistema tradicional de gerenciamento de redes [ASSUNÇÃO 2003]

2.2 – Divisão Funcional

No processo de gerenciamento uma separação funcional de necessidades foi apresentada pela *International Organization for Standardization*

(ISO), como parte de sua especificação de gerenciamento de sistemas OSI. A maioria dos fornecedores de sistemas de gerenciamento de redes adotou esta divisão funcional para descrever as necessidades de gerenciamento: falhas, desempenho, configuração, contabilização e segurança [Sampaio *apud* GUBERT 2002], [BRISA, 1993].

- Gerenciamento de Falhas: é responsável pela manutenção e monitoramento do estado dos objetos gerenciados, assim como, pelas ações necessárias ao restabelecimento das unidades com problemas. Para controlar o sistema é necessário que cada componente seja monitorado. Os problemas, em geral, ocorrem por operações incorretas ou um grande número de erros. O ideal para diminuir o impacto das falhas é a redundância e uso de rotas alternativas.
- Gerenciamento de Contabilização: preocupa-se com a monitoração e manutenção de quais recursos e de quanto desses recursos estão sendo utilizados. A utilização dos recursos deve ser administrada de modo que nenhum usuário seja lesado e para garantir o melhor desempenho da rede.
- Gerenciamento de Configuração: tem por função a manutenção e monitoramento da estrutura física e lógica da rede, incluindo a existência de componentes e sua interconectividade. Este gerenciamento se refere à utilização dos equipamentos bem como a configuração da rede em relação a desempenho, problemas de segurança, atualizações e necessidade dos usuários.
- Gerenciamento de Desempenho: este gerenciamento em uma rede implica em monitorar as atividades desta rede e controlar os recursos através, principalmente, das estatísticas de desempenho. 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 aspectos de segurança essenciais para operar uma rede corretamente. Os recursos da rede assim como as informações dos usuários devem estar protegidas por uma política de segurança robusta e efetiva.

A implantação de uma solução de gerenciamento requer esforço da equipe de suporte e às vezes, investimento da área financeira. Cabe à equipe de administração da rede, balancear o custo benefício trazido pela solução no auxílio à manutenção e prevenção dos problemas encontrados atualmente na rede [LIMA 2003].

2.3 – Gerência de Sistemas

A gerência de redes está preocupada com o gerenciamento dos elementos necessários ao provimento de serviços de rede (*hubs, switches,* roteadores). Já o gerenciamento de sistemas envolve os sistemas existentes na rede, como estações de trabalho com seus sistemas operacionais e servidores de aplicações além de outros serviços. O gerenciamento de sistemas pode ser generalizado como "o gerenciamento de sistemas e seus recursos na rede" [SUBRAMANIAN 2000].

2.3.1 – *Web-Based Enterprise Management* (WBEM)

São apresentados alguns detalhes da arquitetura WBEM [DMTF] pelo fato de ser utilizada a implementação Microsoft desta tecnologia – *Windows Management Instrumentation* (WMI) [MICROSOFT] – para formular regras de análise de dados utilizadas pelos agentes do *grid* de processamento.

A *Web-Based Enterprise Management* (WBEM) [MICROSOFT] é uma iniciativa da indústria, que começou em 1996 por uma coleção de companhias, e que estabelece padrões de infra-estrutura de administração e fornece um caminho para combinar informações de vários tipos de hardware e sistemas de administração de softwares. WBEM especifica padrões para uma arquitetura unificada que permite acesso a dados de uma variedade de tecnologias e plataformas, e apresenta estes dados de uma maneira consistente. Aplicativos de administração podem então usar esta informação para criarem soluções que reduzam a manutenção e custos de administração de uma rede da empresa.

WBEM é baseado no esquema de *Common Information Model* (CIM), que é um padrão da indústria dirigido pela *Distributed Management Task Force* (DMTF). O CIM não determina o formato de instanciação dos dados numa aplicação de gerência, nem o formato de transmissão dos dados entre duas aplicações de gerência ou entre um dispositivo e uma aplicação de gerência. Para atender aos requisitos de interoperabilidade dos ambientes distribuídos aos quais o WBEM se propõe gerenciar, a DMTF combinou o CIM com a linguagem XML – uma forma simples, mas poderosa de representar informação para ser transportada na Web. Para completar este cenário, o WBEM propõe o uso do HTTP – protocolo universal para transporte de informações na Web – para transportar as informações do CIM em formato XML entre as aplicações de gerência e os dispositivos gerenciados, ou entre duas aplicações [Steinke *apud* SIMÕES 2002].

Segundo [MICROSOFT], WBEM assume que a administração de rede da empresa requer ferramentas que trabalhem juntas para fornecerem um modelo único e compartilhado que armazene as informações de administração. WBEM fornece este modelo e a fonte de dados, e pode ser estendida para trabalhar com os já existentes componentes de rede, ferramentas, e protocolos. Enfim, WBEM é uma *iniciativa* que propõe um conjunto de padrões para administrar a rede de empresas, como ilustrado na Figura 2.

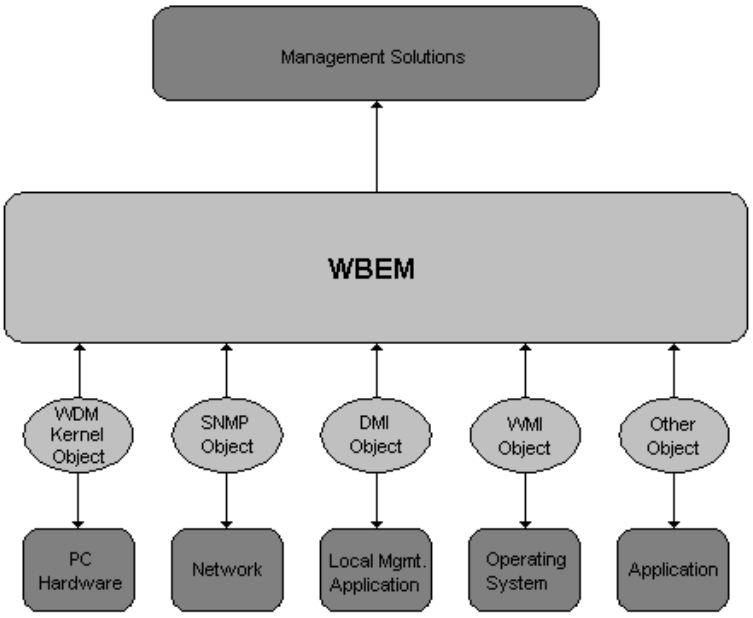


Figura 2 - Base padrão do WBEM para administração de redes de empresas [MICROSOFT]

2.3.1.1 – Arquitetura WBEM

Na arquitetura WBEM encontram-se basicamente cinco componentes, como visto na Figura 3: o cliente *Web*, o gerenciador de objetos CIM (CIMON), o esquema CIM, os provedores específicos de cada protocolo de gerenciamento e os objetos gerenciados com os agentes específicos de cada protocolo.

As aplicações podem realizar uma requisição a qualquer objeto gerenciado mantido por qualquer agente específico. Porém a requisição é sempre feita como se fosse acessado um objeto do esquema CIM e a requisição é enviada ao CIMON. O CIMON recebe as mensagens das aplicações e usa o esquema CIM para decidir qual provedor específico é capaz de prover aquela informação e envia a mensagem ao provedor. O provedor mapeia a requisição para o protocolo específico usado para obter a informação e depois de obtida devolve ao CIMON. No caso de eventos, as mensagens também são mapeadas

do protocolo específico do provedor para os objetos do esquema CIM e enviados ao CIMON.

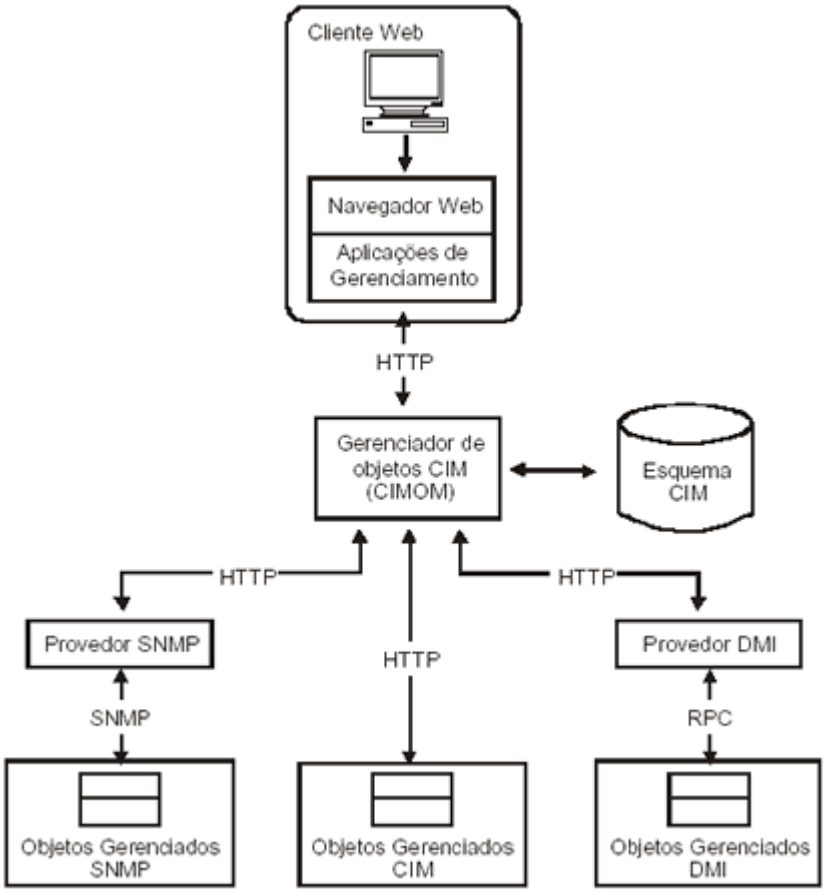


Figura 3 - Visão abstrata do modelo WBEM [SUBRAMANIAN 2000]

2.3.1.2 – WMI

A Microsoft criou sua implementação da WBEM, chamando-a de WMI (*Windows Management Instrumentation*). WMI é um componente chave dos serviços de administração do Microsoft *Windows*. Como núcleo da infra-estrutura de administração da Microsoft, o WMI ajuda a reduzir a manutenção e o custo de

administração dos componentes em uma rede da empresa baseada no *Windows* 2000. Sua arquitetura está demonstrada na Figura 4.

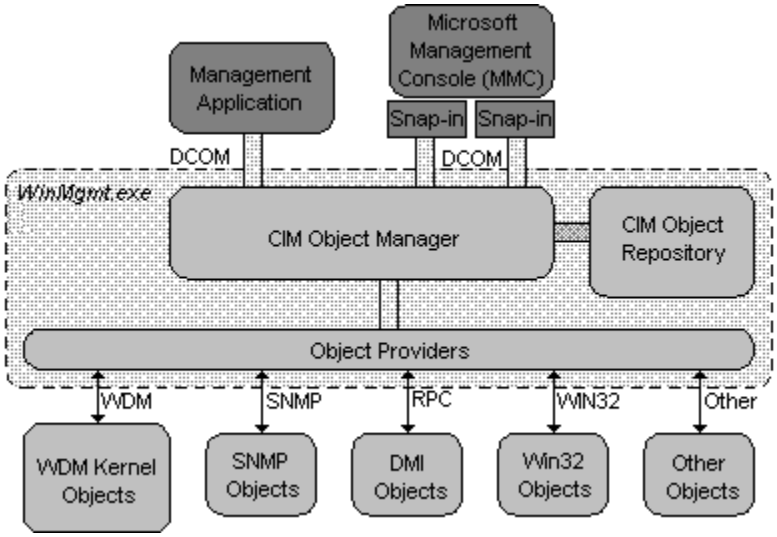


Figura 4 - Arquitetura do WMI [MICROSOFT].

O WBEM fornece uma aproximação para colecionar e distribuir dados de administração. No WMI da Microsoft, esta aproximação consiste em um mecanismo padronizado para armazenar definições de objeto (um CIM-COMPLIANT repositório de objeto), um protocolo padronizado para obter e disseminar dados de administração (COM/DCOM; outros protocolos também estão possíveis), e uma ou mais bibliotecas dinâmicas Win32 (DLLs) que funcionam como provedores de dados do WMI [MICROSOFT 2003].

O propósito da iniciativa do DMTF WBEM está em definir um conjunto não-proprietário de especificações de ambientes independentes para permitir que as informações de administração sejam logicamente organizadas e compartilhadas entre aplicações de administração, mesmo que operadas em ambientes de sistemas operacionais diferentes. Esta ajuda reduz o custo total de gastos da empresa, permitindo que os problemas do sistema sejam diagnosticados e resolvidos de uma maneira muito mais fácil de administrar.

3 – Agentes de Software e Sistemas Multi-Agentes

Agentes autônomos e sistemas multi-agentes representam uma nova maneira de análise, desenvolvimento e implementação de complexos sistemas de software [JENNINGS, SYCARA e WOOLDRIDGE 1998].

Existem diversas definições para o termo agente que segundo [Costa *apud* FERNADES 2003] é utilizado na literatura computacional para determinar diversos tipos de programas. Geralmente estas definições dependem da funcionalidade de cada agente. Para [JENNINGS, SYCARA e WOOLDRIDGE 1998] um agente é um sistema de computador, situado em algum ambiente, que é capaz de flexibilizar ações autônomas a fim de encontrar seus objetivos de desenvolvimento. De acordo com [Amandi *apud* SILVA, 2003], um agente é uma entidade computacional com um comportamento autônomo que lhe permite tomar suas decisões para agir, levando em consideração as mudanças acontecidas no ambiente em que atua e o seu objetivo.

Uma definição bastante utilizada é a de que “um agente é uma entidade que percebe o mundo através de seus sensores e atua sobre ele por meio de seus atuadores”. Desta forma, um agente pode ser qualquer entidade humana, de hardware ou software capaz de perceber o ambiente no qual está inserido e atuar sobre ele. Uma outra definição bastante abrangente e muito usada também é a de que “um agente é um componente de software e/ou hardware que é capaz de agir com o objetivo de realizar tarefas em nome de seu usuário” [Nwana *apud* ASSUNÇÃO 2004].

3.1 – Características dos Agentes de Software

A partir das definições, é possível extrair algumas características comuns aos agentes: devem estar inseridos em um ambiente, devem ter capacidade de sensoriar o seu entorno e atuar sobre ele, e devem agir segundo os objetivos para os quais foram implementados. Portanto, devem possuir uma série de atributos ou propriedades que os definem como agentes [Fernandez *apud* YEPES 2003].

As características e propriedades dos agentes não são iguais e necessárias a todos, porém, a capacidade de cada agente está ligada a presença destas funcionalidades. Algumas destas características são:

- Aprendizagem: segundo [FERNANDES 2003] uma real autonomia só pode estar presente quando um agente possui a habilidade de avaliar as variações do seu ambiente externo e escolher qual a ação mais correta. Entretanto, mesmo quando um agente não reconhece nenhuma ação a ser executada, é esperado que ele procure encontrar uma saída. A questão não é acertar sempre, mas aprender através da experiência.
- Autonomia: o sistema deve ser capaz de agir sem a intervenção direta dos humanos ou de outros agentes computacionais, e deve ter controle sobre suas próprias ações e estado interno. Deve basear suas ações no conhecimento que possui a respeito do ambiente. A orientação por objetivos e a capacidade de aprender são características importantes para que exista um comportamento autônomo.
- Cooperação: os agentes devem ser capazes de trabalharem em conjunto para finalizarem as tarefas interagindo com outros agentes computacionais ou com agentes humanos. A cooperação entre os agentes permite uma melhor e mais rápida solução de

problemas complexos, porém acarreta na necessidade de compartilhamento de conhecimentos, objetivos e preferências.

- Habilidade Social: habilidade dos agentes em interagir com outros agentes através de alguma linguagem comum.
- Mobilidade: esta característica refere-se à capacidade de um agente se mover dentro de uma rede de computadores. Esta característica, porém, vem acompanhada de problemas de segurança.
- Reatividade: capacidade de perceber mudanças em seu ambiente e reagir adequadamente de acordo com estas mudanças. Esta é uma característica essencial e deveria ser suportada por todos os agentes.

Essas são apenas algumas características desejáveis dos agentes de software. Vale lembrar que outros pesquisadores apresentam outras características ou mencionam que algumas não são tão importantes. Portanto, estas definições podem apresentar variações.

3.2 – Classificação dos Agentes de Software

A classificação dos agentes pode ser efetuada de diversas formas, sendo que a mais usual é aquela realizada de acordo com a linha de pesquisa e desenvolvimento, na qual é priorizada a função ou objetivo principal do agente. Assim, pode ser realizada a seguinte classificação: [Wooldridge *apud* YEPES 2003]:

Agentes de interface: também chamados assistentes pessoais ou agentes de usuário, possuem objetivo de simplificar as tarefas rotineiras realizadas por um usuário. Esse tipo de agente observa e monitora as ações tomadas pelo usuário na interface, aprende novos “atalhos” e sugere melhores formas de desenvolver a tarefa. O agente de interface age como um assistente

pessoal autônomo o qual coopera com o usuário na realização de determinadas tarefas.

Agentes colaborativos: enfatizam a autonomia e a cooperação com outros agentes de maneira a executar tarefas. Eles podem ter algumas características de aprendizado, mas este aspecto não é usualmente o foco central das suas operações. Para ter um grupo coordenado de agentes colaborativos, eles devem ser capazes de negociar entre si a fim de encontrar soluções adequadas.

Agentes móveis: o conceito de mobilidade implica na capacidade do agente de se deslocar entre diversas máquinas, evitando dessa maneira uma sobrecarga de comunicação ou permitindo-lhes utilizar recursos não existentes em sua máquina de origem. Um dos principais problemas relacionados a este tipo de agente é o relativo à segurança. Um agente dessa categoria seria, por exemplo, um excelente disseminador de vírus dentro da rede na qual estivesse inserido.

Agentes de informação: também conhecidos como agentes de Internet. Tem tido o seu desenvolvimento favorecido pela grande quantidade de informação disponível na Internet e a dificuldade gerada para encontrar e indexar essa informação de forma a fornecê-la ao usuário.

Agentes Reativos (ou Reflexivos): são geralmente agentes simples que possuem um mapeamento de situações e respostas associadas. Dessa forma, quando um estado ambiental é alterado, o agente executa a ação correspondente para satisfazer o novo estado. Este processo é conhecido por estímulo-resposta.

Agentes Híbridos: mostra-se como uma alternativa para dotar o agente com capacidades reativas apropriadas, visando solucionar a incapacidade de ação adequada por parte de um agente puramente deliberativo no momento em que o mesmo deve tomar uma decisão rápida e espontânea ao enfrentar uma situação imprevista. Em contra-partida, servem também para proporcionar a um agente puramente reativo, capacidade de raciocínio e planejamento quando o mesmo se deparar com uma situação na qual o ambiente diverge bastante dos seus objetivos iniciais.

Além deste tipo de classificação houve outras tentativas usando critérios diferentes. Como exemplo, pode-se citar:

- Classificação segundo as noções fraca e forte de agente;
- Classificação de acordo com a capacidade de se mover pela rede (móveis e estáticos);
- Classificação que leva em consideração alguns atributos primários que os agentes devem apresentar [NWANA e NDUMU 2004]. Na Figura 5 tem-se como atributos cooperação, aprendizado e autonomia e são identificados três tipos de agentes: colaborativos, de interface e inteligentes.

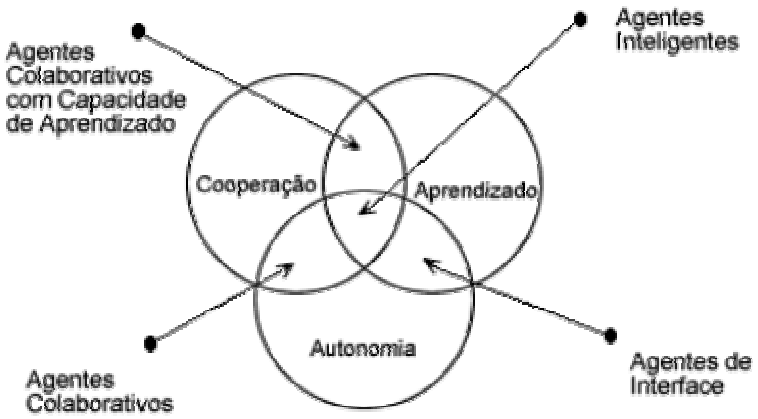


Figura 5 - Classificação baseada nos atributos primários [NWANA e NDUMU 2004].

3.3 – Inteligência Artificial Distribuída (IAD)

A constante busca por sistemas tomadores de decisões proporcionou o aparecimento e desenvolvimento de muitas áreas dentro da inteligência artificial (IA). Uma destas áreas, desenvolvida nos últimos anos, denomina-se IAD, Inteligência Artificial Distribuída. Contrapondo-se à inteligência artificial clássica -

cuja ênfase está na representação de conhecimentos e métodos de inferências - a IAD baseia seu modelo de inteligência no comportamento social sendo a ênfase transposta para as ações e interações entre entidades denominadas agentes [Weiss *apud* ROCHA 2003]. O desenvolvimento da IAD deve-se, principalmente, ao fato que um sistema IA precisa de muito conhecimento para solucionar problemas não triviais.

Nas abordagens clássicas de Inteligência Artificial, a ênfase da inteligência é baseada em um comportamento humano individual e o foco de atenção volta-se à representação de conhecimento e métodos de inferência. Já a Inteligência Artificial Distribuída (IAD), é baseada em comportamento social e sua ênfase é para cooperações, interações e para o fluxo de conhecimento entre unidades distintas [Oliveira *apud* YEPES 2003].

Pode-se dividir a IAD basicamente em duas áreas, sendo que ambas trabalham com o conceito de agentes: Resolução Distribuída de Problemas (RDP) e Sistemas Multi-Agentes (SMA). Apenas SMA será introduzido por ser relevante ao trabalho.

3.4 – Sistemas Multi-Agentes

Um sistema baseado em agentes pode conter um ou mais agentes para a solução dos problemas. Existem casos em que uma solução com apenas um agente é a mais apropriada. E existem outros onde há necessidade de multi-agentes onde os sistemas são desenvolvidos e implementados com muitos agentes. Um sistema multi-agentes pode ser definido como “uma rede de solucionadores de problemas fracamente acoplados que interagem uns com os outros para a solução de problemas que estão além das capacidades ou conhecimento individuais de cada um dos solucionadores” [Sycara *apud* ASSUNÇÃO 2004].

Sistemas multi-agentes são idealmentes projetados para representarem problemas que tenham muitos métodos de solução, múltiplas perspectivas e/ou múltiplas entidades de solução do problema [JENNINGS, SYCARA e WOOLDRIDGE 1998].

Segundo [Yepes *apud* FERNANDES 2003] os sistemas multi-agentes fazem uso do conceito de "comunidade de agentes inteligentes", cujo enfoque se baseia na existência de uma sociedade, composta por vários agentes que atuam no sistema por meio de cooperação e concorrência, sendo que é devido a esse "comportamento social" que emerge a inteligência do sistema. Assim, o objetivo maior dos pesquisadores em SMA está na coordenação de tal comportamento social inteligente, uma vez que dita coordenação envolve conhecimento, objetivos, habilidade e planejamento sobre os agentes. Pesquisas em sistemas multi-agentes concentram-se no comportamento de uma coleção de possibilidades pré-existentes em agentes autônomos objetivando a solução de problemas.

Nos sistemas multi-agentes, os agentes são a parte central que trocam conhecimentos para a resolução de problemas. Nesse tipo de sistema, pressupõe-se que os agentes que formam uma sociedade são dotados de certa autonomia e inteligência para que, independentemente da existência dos outros agentes, consigam percorrer o objetivo global que é a solução de um problema qualquer [Faraco *apud* FERNANDES, 2003].

3.5 – Características dos Sistemas Multi-Agentes

De acordo com [JENNINGS, SYCARA e WOOLDRIDGE 1998] e [Alves e Sichman *apud* YEPES 2003], as características de um sistema multi-agente são:

- Os agentes são concebidos independentemente de um problema particular a ser resolvido. O projeto de um agente deve resultar numa entidade capaz de realizar um determinado processamento.

- Cada agente possui informações incompletas, ou capacidades para resolver problemas, portanto cada agente possui um ponto de vista limitado.
- A concepção das interações também é realizada independentemente de uma aplicação-alvo particular. Busca-se desenvolver protocolos de interação genéricos, que possam ser reutilizados em várias aplicações similares.
- Não há controle global do sistema, este é implementado de forma totalmente descentralizada nos agentes.

Uma vez que os agentes são concebidos independentemente de um problema particular a ser resolvido, torna-se possível à reutilização de tais componentes quando se deseja projetar aplicações similares. Os agentes irão instanciar dinamicamente as organizações e interações quando um problema for apresentado ao sistema. A Figura 6 mostra a estrutura de um SMA segundo a concepção de Sichman [*apud* YEPES 2003].

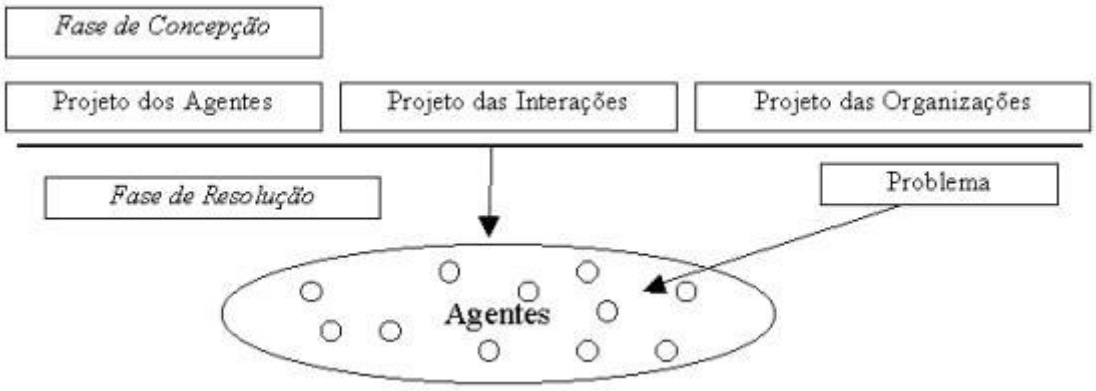


Figura 6 - Estrutura de um SMA de [Sichman *apud* YEPES 2003].

3.6 – Vantagens dos Sistemas Multi-Agentes

Segundo [Jennings *apud* SILVA, 2003] as características dos sistemas multi-agentes têm vantagens significativas sobre os modelos centralizados. Como exemplo destas vantagens podem ser citados:

- Maior rapidez na resolução de problemas através do aproveitamento do paralelismo.
- Diminuição da comunicação por transmitir somente soluções parciais em alto nível para outros agentes ao invés de dados brutos para um lugar central.
- Mais flexibilidade por ter agentes de diferentes habilidades dinamicamente agrupados para resolver problemas.
- Aumento da segurança por permitir que agentes assumam responsabilidades de agentes que falham.

Alguns dos motivos do aumento de interesse nas pesquisas em sistemas multi-agentes incluem: a habilidade de prover robustez e eficiência; a habilidade de permitir interoperações de heranças em sistemas; e a habilidade de resolver problemas nos quais os dados, perícia, ou controle são distribuídos [JENNINGS, SYCARA e WOOLDRIDGE 1998].

Os sistemas multi-agentes estão ocasionando uma verdadeira revolução na forma de resolver e de pensar os problemas. Estes estão passando de uma visão global, de atitudes baseadas em uma abordagem centralizada, a uma visão local, em nível de agente. A tecnologia de agentes está saindo rapidamente das universidades e laboratórios de pesquisa e começando a ser usado para solucionar problemas do mundo real nas extensões da indústria e aplicações comerciais.

4 – *Grids*

A grande melhoria de desempenho das redes de computadores originou a idéia do uso de computadores independentes, conectados em rede, como plataforma para execução de aplicações paralelas, originando a área de Computação em *Grid*. Computação em *grid*, ou *grids* computacionais, são uma forma de acesso universal, facilitado, confiável e seguro a agrupamentos dinâmicos de recursos computacionais heterogêneos dispersos geograficamente [FOSTER e KESSELMAN 1999]. Os principais atrativos desta idéia são a possibilidade de alocar uma grande quantidade de recursos a uma aplicação e fazê-lo a um custo muito menor do que alternativas tradicionais. *Grids* se tornaram possíveis nos últimos anos, devido à grande melhoria em desempenho e redução de custo, tanto de redes de computadores quanto de microprocessadores.

A diferença existente entre a computação distribuída e computação em *grid* é que a computação distribuída consiste na possibilidade de resolver um determinado problema computacional através da utilização de diferentes recursos distribuídos geograficamente. A computação distribuída passa a ser uma computação em g*rid* no momento em que existe uma infra-estrutura física e uma infra-estrutura lógica (software) que permita coordenar os trabalhos que vão ser processados e garantir a sua qualidade de serviço [PITANGA 2004].

O *grid* é um caso particular da computação distribuída, pois é orientado essencialmente para aplicações que precisam de uma grande capacidade de

cálculos, ou enormes quantidades de dados transmitidos de um lado para o outro, ou ambas.

O nome *grid* foi idealizado baseado nas malhas de interligação dos sistemas de energia elétrica (*power-grids*). A rede elétrica disponibiliza energia elétrica sob demanda e esconde do usuário detalhes como a origem da energia e a complexidade da malha de transmissão e distribuição. Ou seja, quando se tem um equipamento elétrico, simplesmente ele deve ser conectado na tomada para que receba energia. O *grid* computacional, portanto, seria uma rede na qual o individuo se conecta para obter poder computacional. O objetivo é criar a ilusão de um computador virtual simples, porém enorme e poderoso de sistemas heterogêneos conectados compartilhando vários recursos [CIRNE 2004].

Grids Computacionais nasceram da comunidade de Processamento de Alto Desempenho (PAD). O conceito foi apresentado pelos pesquisadores Ian Foster e Carl Kesselman, sendo composto por uma infra-estrutura de hardware e software que permite acesso a grandes capacidades computacionais geograficamente distribuídas, de forma confiável, consistente, econômica e persistente [PITANGA 2004]. Na verdade o conceito é antigo, mas com uma nova dinâmica, onde pode ser utilizada a capacidade de computação sem ter que se preocupar de onde vem ou como é mantida, fazendo uma metáfora às redes elétricas. Para se oferecer um conceito mais útil tecnicamente, é preciso pensar no g*rid* como uma plataforma para execução de aplicações paralelas [BERSTIS 2002]. Mas, certamente, há plataformas para execução de aplicações paralelas que não são *grids*. De maneira geral, pode-se dizer que *grids* são mais distribuídos, diversos e complexos que outras plataformas.

A tecnologia de computação em *grid* possibilita agregar sistemas heterogêneos e dispersos, formando a imagem de um único “supercomputador virtual”, oferecendo uma variedade de recursos virtuais, como ilustrado na Figura 7. Os usuários do *grid* podem ser organizados dinamicamente, cada qual com diferentes políticas de solicitações, mas podendo compartilhar seus recursos coletivamente.

O *grid* permite também o uso de técnicas de programação paralela por passagem de mensagens. Mas devido à alta latência provocada na comunicação entre processos, as aplicações devem ser construídas com uma granularidade bem projetada de tal forma que se comuniquem o mínimo possível. Então, as aplicações mais adequadas ao *grid* são as que possuem tarefas independentes, pois as tarefas não dependem da execução de outras e podem ser executadas em qualquer ordem. Além disto, são encontradas outras fontes de contenção entre tarefas na aplicação paralela em *grid* como: capacidades de comunicação na rede, sincronização de protocolos, largura de banda de entrada e saída para dispositivos e dispositivos de armazenamento, e as latências que interferem nas exigências de aplicações em tempo real [BERSTIS 2002].

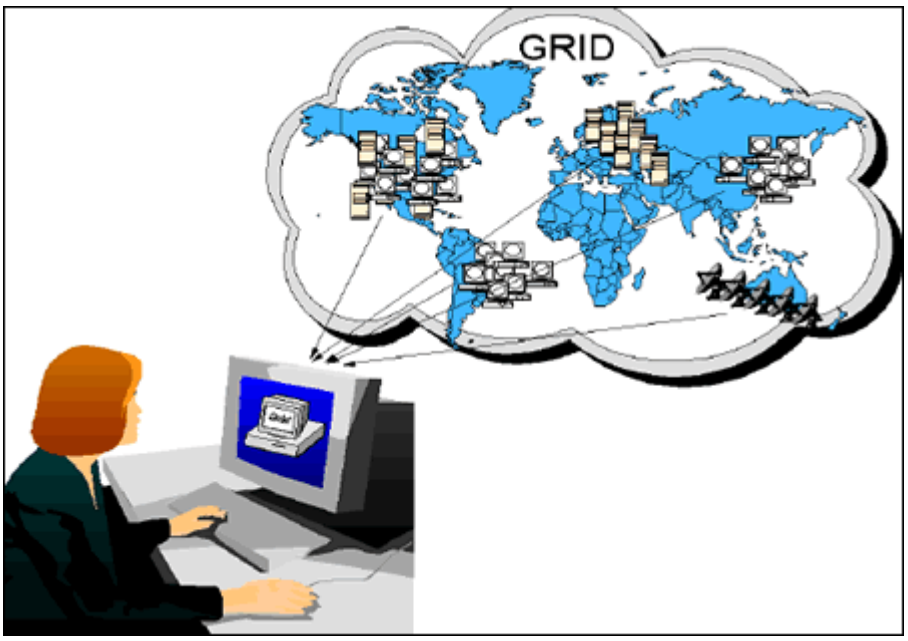


Figura 7 - O *grid* virtual com recursos heterogêneos e geograficamente dispersos representa uma simples visão aos usuários [BERSTIS 2002].

4.1 – *Grid* versus Internet

A Internet popularizou-se com o aparecimento dos serviços WWW. Seu objetivo era a interligação de diferentes ambientes computacionais e geograficamente dispersos. Nos *sites* desenvolvidos pela indústria o usuário dispõe de um menu de serviços fechados e em geral, permanecem no anonimato. O que ocorre em um ambiente de *grid* é o inverso, o usuário é que tem de submeter suas aplicações para serem resolvidas dentro do ambiente por ele montado.

4.2 – *Grid* versus *Cluster*

Um ambiente de *cluster* constitui em um sistema formado por hardware e software conectados em apenas um local, servindo a usuários que estão trabalhando em apenas um projeto. É usado exclusivamente para resolver os problemas computacionais de uma determinada organização. Além disso, os recursos são gerenciados por uma entidade central, e os computadores agem como se fossem um único dispositivo. Por outro lado, um *grid* presta serviços de uma forma geograficamente distribuída. Nas configurações em *grid*, cada "organização virtual" faz o gerenciamento de seus recursos não tendo a visão de uma imagem única do sistema. Ou seja, o usuário tem consciência dos diversos serviços disponíveis e que deverá requisitá-los para sua utilização. Portanto, os *grids* são mais heterogêneos, complexos e distribuídos [PITANGA 2004A].

Pode-se diferenciar *grid* e *cluster* também, utilizando a referência do Prof Buyya [*apud* PITANGA 2004A] - "Se acontece o compartilhamento de recursos gerenciado por um único sistema global sincronizado e centralizado, então é um *cluster*. Em um *cluster*, todos os nós trabalham cooperativamente em um objetivo comum e o objetivo é a alocação de recursos executada por um

gerente centralizado e global. Em *grid*, cada nó, possuí seu próprio gerente, recursos e política de alocação".

4.3 – Organizações Virtuais

É chamada Organização Virtual (VO) quando existem participantes que desejam compartilhar recursos para poder concluir uma tarefa. Este compartilhamento pode envolver troca de documentos, acesso direto a software remoto, computadores, dados, sensores e outros recursos [FOSTER e KESSELMAN 1997].

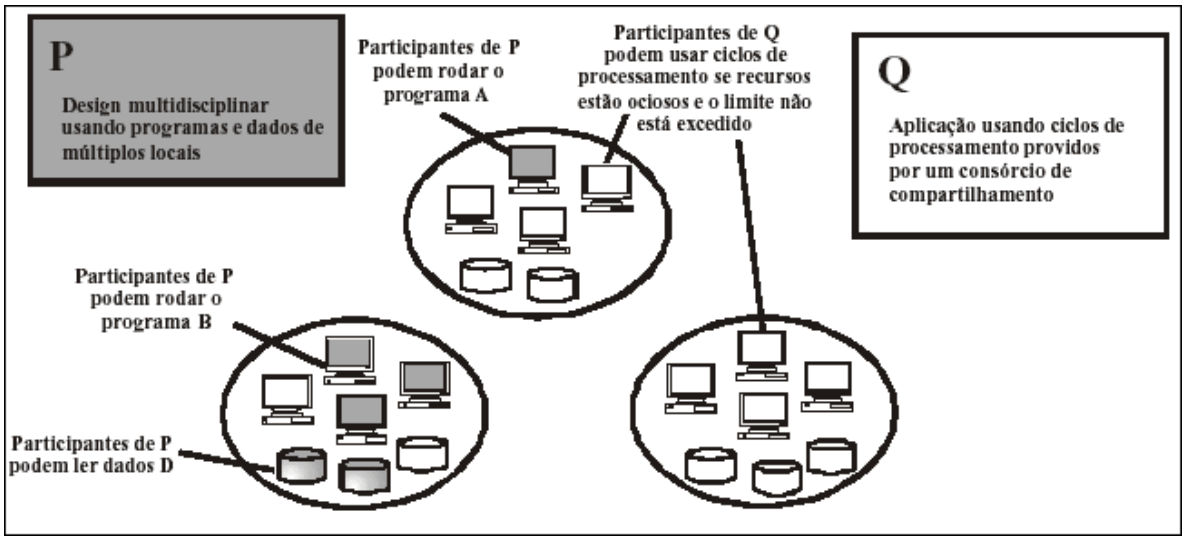


Figura 8 - Um exemplo de organização virtual [Foster et all. a*pud* ASSUNÇÃO 2004]

Um exemplo de organização virtual é apresentado na Figura 8. Uma organização, representada na figura pelas formas ovais, pode participar de uma ou mais organizações virtuais, compartilhando alguns ou vários de seus recursos. Neste exemplo existem três organizações e duas organizações virtuais. A OV **P** envolve participantes engajados no desenvolvimento de uma aeronave. A OV **Q** envolve participantes que estão compartilhando ciclos de processamento. A

organização à esquerda participa da OV **P**, a da direita participa de **Q**, e a do centro participa de ambas organizações virtuais [ASSUNÇÃO 2004].

4.4 – Características do *Grid*

Como benefícios do uso de um *grid,* diferenciando-o das tecnologias distribuídas e paralelas tradicionais, são citados:

Organizações podem agregar recursos: a computação em *grid* permite que as organizações possam agregar recursos não importando localização global. O compartilhamento não é limitado a arquivos, inclui também vários outros recursos, tais como: equipamentos, software, serviços, licenças, e outros. Estes recursos são "virtualizados" para dá-los mais uniformidade na interoperabilidade entre os participantes heterogêneos da grade. Isso elimina situações onde um *site* esteja sendo executado com sua capacidade máxima, enquanto outros tenham ciclos disponíveis.

Poderosa plataforma de suporte a Organizações Virtuais: organizações podem melhorar dramaticamente sua qualidade e velocidade de produtos e serviços disponibilizados, enquanto os custos com tecnologia são reduzidos por habilitar a colaboração transparente dos recursos compartilhados. Uma organização pode ter picos ocasionais inesperados de atividade exigindo mais recursos. Para aplicações que estão ligadas no *grid*, este pode oferecer um efeito balanceado do recurso através do escalonamento de tarefas em máquinas com baixa utilização, como ilustrado na Figura 9.

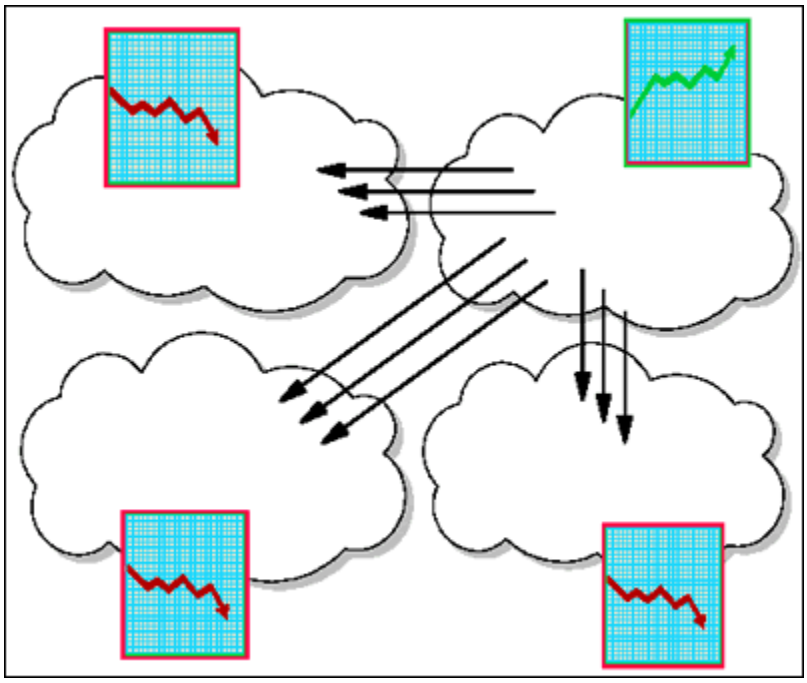


Figura 9 - Tarefas são migradas para partes menos ocupadas do *grid* para equilibrar recursos e absorver picos inesperados de atividade [BERSTIS 2002].

Acesso distribuído a diversos tipos de recursos: permite que empresas acessem e compartilhem bases de dados de forma remota. Isto é essencialmente benéfico para as ciências da saúde ou comunidades de pesquisa, onde volumes grandiosos de dados são gerados e analisados durante todo dia. Além disso, muitas vezes, as máquinas têm enormes capacidades de disco não usadas. Computação em *grid*, mais especificamente, um *data grid*, pode ser usado para agregar este armazenamento não utilizado dentro de um depósito virtual de dados muito grande, e configurado para alcançar aperfeiçoamentos na confiança e na execução em qualquer máquina.

Possibilidade de Falhas e Adaptabilidade: Falhas são fenômenos aleatórios, imprevisíveis e provocam perda de estado ou desconexão ao resto do sistema. Em ambientes heterogêneos, dinâmicos e de larga escala, como *grids*, falhas são a regra, não exceção [WEBER 2004]. Por isso, o software de gerência do *grid* pode automaticamente submeter tarefas a outras máquinas quando uma falha é detectada. Em situações críticas, como em tempo real, cópias múltiplas de tarefas importantes podem ser executadas em diferentes máquinas por todo o

grid, como ilustrado na Figura 10. Seus resultados podem ser checados para verificar qualquer tipo de inconsistência, tal como falhas do computador, dados corrompidos, ou correção de dados.

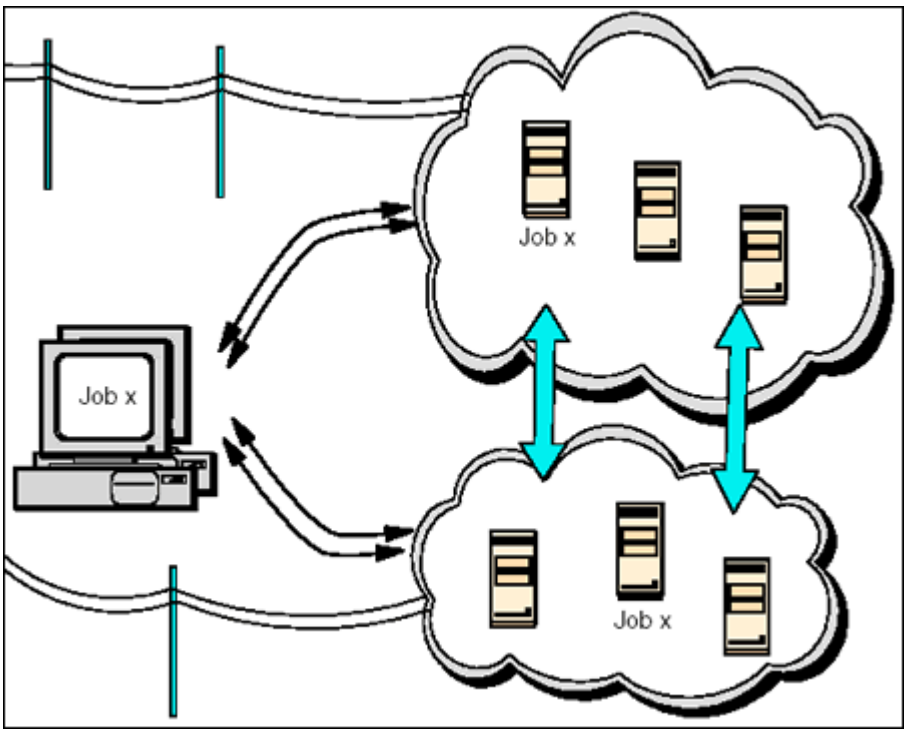


Figura 10 - Configuração do *grid* redundante e submissão de tarefa redundante para alcançar alta confiabilidade [BERSTIS 2002].

Colaboração entre centro de pesquisas: possibilita a larga dispersão das organizações para que facilmente possam colaborar em projetos pela criação da habilidade do compartilhamento de tudo, desde aplicações a dados, até projetos de engenharia, entre outros.

Melhor utilização de largura de banda: pode-se criar a mais robusta e resistente infra-estrutura de informações. Para o *grid* computacional, as aplicações leves são as ideais, pois estas requisitam relativamente menos das redes. Mesmo tendo máquinas com uma alta conexão, estas redes com baixo fluxo de dados constituem uma espécie de gargalo ao requisito fundamental para aplicações pesadas. A multiplicidade das velocidades das diversas redes implica em alguns

pontos de gargalo, e que compromete o desempenho do nosso supercomputador virtual [PITANGA 2004A].

Aproveitamento de recursos ociosos: pode-se aproveitar os ciclos de processamento não utilizados, disponíveis dos computadores pessoais que se encontram em várias localidades pelo planeta. Por exemplo, os computadores que se encontram ociosos durante a noite de uma empresa no Japão podem ser utilizados durante o dia para operações na América do Sul.

Escalabilidade: um *grid* pode crescer de uma quantidade pequena de recursos até a faixa de milhões.

O gerenciamento pode usar um *grid* para melhorar a visão do uso de padrões em grandes organizações, permitindo um melhor planejamento quanto à atualização de sistemas, aumento da capacidade, ou reservando recursos computacionais não mais necessários.

A computação em *grid* está gerando especulações não somente na área científica, mas também na empresarial (IBM, HP/Compaq, Oracle, Sun e Fujitsu), pois irá permitir a redução de custos e tempo, aumento de produtividade, compartilhamento de recursos e informações, dentre outras possibilidades. Com sua expansão, pode-se chegar, no final, em algo como a formação de um *Grid* Global, uma rede distribuída global de colaborações entre seus participantes, capaz de prover recursos talvez inatingíveis por um *cluster* isolado. E isto já está sendo feito em diversos países do mundo, inclusive no Brasil [PITANGA 2004].

O *grid* computacional é um conceito sobre o qual existe ainda uma grande expectativa e que poderá evoluir em diferentes direções, mas que hoje já é entendido como a próxima geração da *Web* para a comunidade científica.

4.5 – *Grids* de Agentes

Os agentes através de mecanismos como cooperação, pró-atividade, comunicação, entre outros, podem representar um papel importante na construção

de *grids* [Luck *apud* ASSUNÇÃO, WESTPHALL e KOCH 2003A], sendo que a integração de recursos pode ocorrer em um nível muito mais semântico.

Nos últimos anos houve um aumento no uso das tecnologias de agentes em sistemas de *grids,* quer seja na sua construção como no seu gerenciamento. Os agentes proporcionam características importantes em um ambiente de *grid*, possibilitando a conexão em larga escala de sistemas heterogêneos. Estes agentes podem ser flexíveis o suficiente para se adaptarem a sistemas operacionais específicos de computadores, ou a qualquer programa de controle de um equipamento.

Grids de agentes são, basicamente, *grids* que utilizam como infra-estrutura sistemas de agentes de software dispersos pela rede que possibilitam uma forma barata e eficiente de unir estes agentes e possibilitar o uso coordenado de recursos nas atividades de gerenciamento.

Um *grid* de agentes se difere de um sistema multi-agente pelas suas características de escalabilidade, distribuição de carga e cooperação. Ele pode ser visto como um sistema que provê a integração de sistemas multi-agentes e seus componentes em larga escala, geralmente transpondo os limites de uma organização [ASSUNÇÃO, WESTPHALL e KOCH 2003A]. Esta integração pode incluir um conjunto de mecanismos que envolvem serviços de diretórios, anúncio e descoberta de serviços, *gateways* entre diferentes plataformas, mecanismos de negociação de recursos, balanceamento de carga, além de serviços de segurança como autenticação e controle de acesso. Estes mecanismos devem garantir a interoperabilidade, o crescimento e aplicação de agentes em ambientes dinâmicos e multi-institucionais.

Em geral, um *grid* de agentes deve apresentar os seguintes serviços [SYPERREK, ASSUNÇÃO e WESTPHALL 2003]:

- Serviço de Informação: serviço de diretório responsável por registrar recursos e agentes existentes além de suas habilidades e serviços. O serviço de informação compreende a parte mais importante da infra-estrutura, possibilitando a procura e monitoramento dos agentes. Através deste mecanismo de

diretórios os agentes podem se registrar e disponibilizar seus próprios serviços e habilidades, além de poderem procurar outros agentes que sejam capazes de executar tarefas que eles próprios não podem, devido a sua limitação de recursos ou por questões estratégicas.

- Serviço de Nomeação: mecanismo de nomeação para que não ocorram dois agentes de um mesmo *grid* com o mesmo nome.
- Serviço de Visualização: ferramenta que informa os acontecimentos do *grid* e interações entre os agentes através de interfaces gráficas.
- Serviço de Segurança: devem existir mecanismos de autenticação e confidencialidade por integrar recursos de diferentes organizações. No *grid* de agentes, além dos critérios referentes a criptografia das mensagens trocadas, são necessários mecanismos de autenticação das mensagens trocadas.

Além das vantagens de distribuição da análise, a utilização de *grids* de agentes traz as seguintes possibilidades [ASSUNÇÃO, WESTPHALL e KOCH 2003A]:

- Exploração de recursos pouco utilizados, agregando-os na execução de tarefas de gerenciamento.
- Podem ser adicionados containeres de agentes para efetuar um tipo particular de análise, ou para prever valores futuros. Esta predição pode ser distribuída em vários equipamentos e sua utilização pode ser negociada no *grid*.
- Para reduzir o tráfego de informações na rede, o armazenamento de dados pode ser realizado de forma a mantê-los mais próximos do local onde serão processados.
- Balanceamento inteligente de carga e de recursos baseado na capacidade dos recursos, na sua ociosidade e na disponibilidade de conhecimento para o processamento. Os agentes possuem

inteligência para melhor distribuir a carga de processamento e encontrar os melhores recursos para processá-las.

- Integração e compartilhamento de software existentes, como de gerenciamento. Os softwares de gerenciamento existentes podem ser encapsulados e representados pelos agentes e proporcionar serviços usados pelo *grid*.

Segundo ASSUNÇÃO [2004], os *grids* de agentes podem ser vistos como uma coesão de uma série de tecnologias. São uma maneira de compor software mais facilmente e de possibilitar a conexão dinâmica de sistemas. Estas características fazem dos agentes entidades de extrema importância nos ambientes de *grid* e no estabelecimento das organizações virtuais dinâmicas. O *grid* de agentes pode ser também, uma infra-estrutura de software baseada em agentes construída sobre as funcionalidades providas por um *grid* computacional. Eles podem prover serviços e funcionalidades para as camadas superiores de informação e conhecimento.

4.6 – *Grids* de Agentes na Gerência de Redes

Nos últimos anos, houve um grande aumento na complexidade e no tamanho das redes de computadores e telecomunicações tornando a tarefa de gerenciamento destas redes morosa e complexa, pois, à medida que as redes de computadores e telecomunicações crescem, cresce proporcionalmente a complexidade de seu gerenciamento e fluxo de informações no sistema de gerência.

No fluxo de trabalho tradicional do gerenciamento de redes, como mostrado anteriormente (Figura 1), existe uma situação onde dados são extraídos dos equipamentos e precisam ser transformados em informações que possam orientar o gerente na tomada de decisões e na execução de ações de gerenciamento [KOCH e WESTPHALL 2001]. Estas metodologias de

gerenciamento centralizado tornam-se ineficientes quando consideradas a quantidade e diversidade dos equipamentos. O gerenciamento centralizado de rede pode levar a situações onde não existem recursos computacionais suficientes para realizar um gerenciamento eficiente [SYPERREK, ASSUNÇÃO e WESTPHALL 2003], assim como comprometer o sistema com a estação de gerência fora de funcionamento.

Para o gerenciamento são necessárias a coleta e análise de dados extraídos dos sistemas e de dispositivos da rede. Critérios que dizem respeito ao que coletar e ao que analisar podem ser orientados por um *Service Level Agreement* (SLA). Além do monitoramento de itens que sejam considerados vitais ao provimento de qualquer serviço, o monitoramento pode ser orientado pelo SLA [ASSUNÇÃO 2004].

Em um ambiente de rede composto por uma grande quantidade de equipamentos, o volume de dados coletados e que precisam ser analisados é bastante grande. Transformar estes dados em informações e relatórios que demonstrem as condições de operação da rede e seus serviços, ou que possam indicar eventuais problemas, pode se tornar uma tarefa intensiva, exigindo um grande poder de processamento das estações envolvidas no gerenciamento da rede.

Tomando a gerência de redes de computadores como uma aplicação em *grid*, onde existe um grande volume de dados que precisa ser transformado em informações de gerência, encontra-se um cenário onde *grids* de agentes podem ser aplicados. Pode-se ter uma grande quantidade de regras de análise que consegue-se manter e utilizar em um sistema como este. É possível efetuar uma distribuição e um balanceamento de carga da análise dos dados coletados, baseando-se na disponibilidade e capacidade dos recursos do *grid*. Se o sistema requer uma capacidade de processamento maior, basta adicioná-la ao *grid*. Dessa forma, pode-se obter ganhos significativos e uma redução de hardware considerável.

4.6.1 – Arquitetura Proposta

Assunção [2003] propôs, utilizando *grids* de agentes, uma arquitetura para o gerenciamento de redes, que através da distribuição das tarefas de manipulação e análise dos dados proporciona maior extensibilidade e adaptabilidade do que os modelos centralizados tradicionalmente utilizados. Esta utilização de *grids* de agentes se mostra interessante na gerência, pois além de distribuir tarefas ligadas ao gerenciamento, possibilita uma distribuição da análise e do armazenamento das informações, conforme descrito em simulações [ASSUNÇÃO 2003] que demonstram os resultados esperados da arquitetura.

No *grid* de agentes de gerenciamento, conforme apresentado na Figura 11, pode-se identificar algumas tarefas distintas: a coleta de dados dos equipamentos da rede, a classificação e armazenamento dos dados e a análise e transformação dos dados coletados e classificados em informações de gerenciamento [ASSUNÇÃO, WESTPHALL e KOCH 2003A].

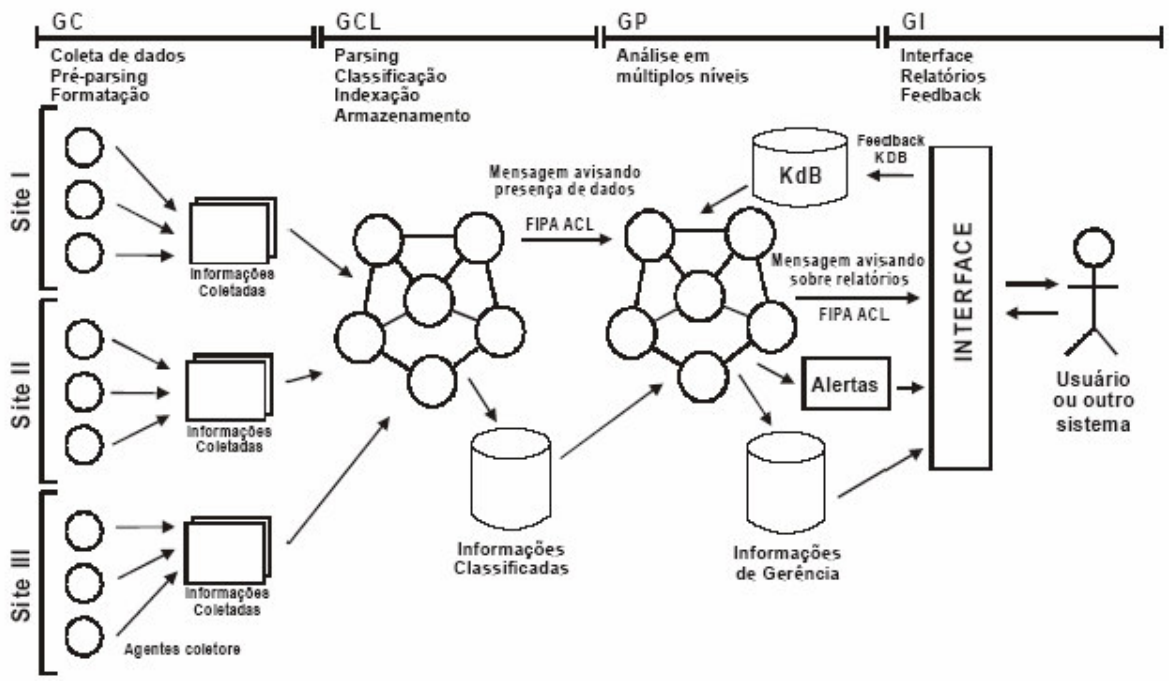


Figura 11 - Arquitetura geral do sistema [ASSUNÇÃO 2003].

Os *grids* de agentes coletores (GC) são responsáveis pela coleta das informações de gerenciamento dos equipamentos da rede gerenciada. Estes agentes podem fazer uso de um protocolo de gerenciamento ou de utilitários de linha de comando dos equipamentos onde estão instalados. Devem extrair apenas as informações relevantes para enviá-las ao *grid* classificador. Para o envio das informações, os agentes fazem uso de um protocolo como HTTP ou SMTP.

O *grid* classificador (GCL) realiza tarefas de *parsing*, classificação, indexação e armazenamento de dados. O *grid* de classificação é responsável por classificar as informações que recebe dos coletores, armazená-las de uma forma estruturada e mais fácil de ser recuperada, facilitando a distribuição e realização da análise.

O *grid* de processamento ou análise (GP) é a parte mais importante da arquitetura. É responsável pela consolidação dos dados coletados em relatórios de gerência. Os principais problemas enfrentados estão relacionados à forma como as tarefas de análise são distribuídas, controle de recursos, balanceamento de carga e tolerância à falhas. Os resultados das análises realizadas são relatórios e alertas enviados ao *grid* de interface.

O *grid* de interface (GI) é o canal de comunicação entre o *grid* de gerência e o gerente da rede. É também uma forma de receber o retorno do gerente humano e fornecê-lo ao sistema. Através de uma interface o usuário pode receber os relatórios resultantes do *grid* de processamento e os alertas. É possível programar novos relatórios e interagir com a base de conhecimento do sistema.

O foco deste trabalho é nos agentes de análise dos dados do *grid,* ou seja, no funcionamento do *grid* de agentes processadores, e por isto o assunto será detalhado em capítulo especial.

5 – Ambiente Experimental

Para o desenvolvimento da arquitetura de *grid* proposta foi escolhido o *framework* para a construção de agentes, *Agentlight,* proposta por Koch [AGENTLIGHT]. Alguns fatores que influenciaram na escolha desta ferramenta são:

- Disponibilidade do código fonte para alterações.
- Simplicidade e flexibilidade da plataforma.
- O *framework* é todo desenvolvido em Java.
- Um dos objetivos do *framework* é possibilitar a construção de agentes para pequenos dispositivos, o que facilita a sua utilização em diferentes plataformas, desde pequenos assistentes pessoais até servidores com recursos computacionais mais sofisticados.
- Como o *Agentlight* tem como princípio básico proporcionar uma plataforma para desenvolvimento de agentes para pequenos dispositivos, os agentes são criados com o intuito de serem bastante conservativos no que se refere ao tamanho de código compilado e utilização de memória. Estes cuidados são importantes no desenvolvimento da plataforma e são úteis também no *grid*, pois podem garantir a sua pervasividade e ubiqüidade.

Além destes fatores, o *Agentlight* está sendo desenvolvido e modificado para facilitar a realização das idéias sobre *grids* de agentes. Alguns módulos desta

plataforma estão sendo desenvolvidos para a experimentação das teorias propostas.

5.1 – Plataforma *Agentlight*

A plataforma *Agentlight* foi desenvolvida usando como princípio a necessidade de expandir a tecnologia de agentes para pequenos dispositivos, em geral, com limitados recursos de memória e processamento. Por isso, os agentes são criados com o intuito de serem conservativos em relação ao tamanho de código compilado e utilização de memória.

O *framework* é todo desenvolvido em Java, permitindo assim, que qualquer plataforma que possua uma máquina virtual Java possa executar as classes que formam o *Agentlight.* Isto possibilita a independência da plataforma de hardware utilizada e também, a reusabilidade de código. Outra característica importante desta linguagem é a segurança proporcionada pelas bibliotecas de criptografia, que permitem o uso de certificados digitais que podem ser usados para garantir a confidencialidade e autenticação das mensagens trocadas pelos agentes.

O *Agentlight* funciona com o conceito de containeres de agentes. Este conceito é utilizado para que se possa ter um bom número de agentes em execução com um uso mínimo de recursos do sistema. Segundo Assunção [2004], no *Agentlight,* um container é composto basicamente por:

- Um diretório onde são armazenadas informações referentes aos agentes do container*,* rotas para outros containeres, informações sobre as linguagens de conteúdo de mensagens que são aceitas pelos outros containeres, codificação de mensagem aceita e protocolos utilizados para transporte das mensagens.
- Uma base de conhecimento global que é compartilhada por todos os agentes do container. Assim, primeiramente a máquina de

inferência do agente infere sobre a sua base de conhecimento local e após isso sobre a base de conhecimento global.

- Um agente gerenciador, chamado de *manager*, que é o responsável por processar as mensagens destinadas ao container. Este agente possui em sua base de conhecimento um conjunto de regras necessárias para realizar as tarefas de gerenciamento do container.

Os agentes da plataforma utilizam um mecanismo de raciocínio baseado em um motor de inferência que infere sobre uma base de conhecimento e que possui fatos e regras armazenados usando uma estrutura baseada em lógica de primeira ordem. Os agentes no *Agentlight* são compostos basicamente por: uma base de conhecimento onde são armazenados os fatos e regras que constituem o conhecimento do agente; uma máquina de inferência usada para inferir sobre a base de conhecimento em busca de cláusulas e regras que casem com a cláusula dada como objetivo; e também os objetivos que o agente deve perseguir.

O núcleo básico do *grid* de agentes é construído sobre a infra-estrutura de agentes proporcionada pelo *Agentlight*. É chamado de núcleo, uma infra-estrutura básica necessária para que se possa iniciar um nó do *grid* de agentes, o que se pode chamar também de *kernel* do *grid* de agentes. A idéia é a de que para um usuário que queira construir ou doar os seus recursos para serem utilizados por um *grid* de agentes, tenha que baixar apenas alguns arquivos necessários à iniciação do aplicativo que ele irá executar, e a sua configuração.

5.2 – O Conceito de *Skills*

A definição de conhecimentos específicos que são necessários por algum agente é feita por meio de habilidades e ensinadas ao agente. Estas habilidades, por não serem representáveis de maneira lógica, são classes

implementadas em Java que agem como extensão das regras primitivas conhecidas e existentes no banco de regras do agente.

A ligação entre o conhecimento dos agentes, que é estruturado em forma de cláusulas de lógica de primeira ordem, e entre as coisas que não são lógicas, como o uso de classes e métodos de bibliotecas, deve ser feita através da implementação de uma interface chamada *SkillInterface*, apresentada na Figura 12.

```
public interface SkillInterface{
    public boolean skill(Term term, Stack stack);
}
```

Figura 12 - Interface Java *SkillInterface*

Nos arquivos de definição, quando uma regra está associada a uma *skill,* deve ser informada a *skill* correspondente para a qual os termos serão repassados. No momento de carregamento dos arquivos XML de definição, o nó realizará a instanciação da *skill* correspondente.

5.3 – Arquivos de Definição

Os arquivos de definição carregados por um nó *grid* no momento em que ele é iniciado são documentos XML que possuem uma estrutura simples e flexível. Estes documentos contêm a lógica da aplicação para a qual o nó será utilizado. Por exemplo, se um nó *grid* desempenha o papel de um coletor de dados, ele deve ser provido com conhecimento e habilidades necessárias para que possa realizar estas atividades. Se o nó for usado para realizar atividades de análise de informações, deve ter o conhecimento inicial necessário para realizar as atividades para as quais ele se destina. Estas informações iniciais ditam o comportamento que o nó tem no *grid* [ASSUNÇÃO 2004].

6 – *Grids* de Agentes Processadores

A camada de análise é a camada que realiza as inferências sobre os dados armazenados em busca de problemas que possam estar ocorrendo na rede gerenciada. Esta camada faz uso das informações armazenadas e classificadas pelo *grid* de armazenamento. Desta forma, além de técnicas de escalonamento, é necessário ter o maior número de regras possíveis, além de distribuí-las entre os vários agentes do *grid*. O desenvolvimento destas regras é extremamente importante para o desenvolvimento do sistema e tendo um número grande de regras pode-se variar a utilização do *grid* aplicando-o a cenários e redes diferentes.

O *grid* de processamento ou análise é a parte mais importante da arquitetura, e onde estão localizados os maiores desafios de desenvolvimento. É responsável pela consolidação dos dados coletados em relatórios de gerência. Os principais problemas que podem surgir dizem respeito à divisão das atividades de análise, controle de recursos, balanceamento de carga e tolerância à falhas. O resultado da análise é composto por informações que dão origem aos relatórios de gerência e de alertas enviados aos agentes de interface.

Este *grid* é composto por containeres [MULLER 1997] de agentes distribuídos em vários computadores. A idéia é possibilitar um ambiente dinâmico onde containeres de agentes podem ser adicionados ao *grid* ou removidos dele constantemente. Ao ser adicionado, ele anuncia seus serviços ao *grid* e passa a analisar as informações que lhe são designadas.

É apresentado o conceito de container de agentes porque a plataforma utilizada para os testes e validação da idéia usa este conceito. Porém, o *grid* processador poderia ser composto por vários agentes, e não necessariamente containeres, localizados em vários recursos. Um container nada mais é do que uma forma de possibilitar a execução de vários agentes em uma mesma aplicação. Para efeito externo, ele se comporta como se fosse um agente.

O *grid* de análise recebe uma mensagem do *grid* classificador, indicando que existem dados a serem analisados e que esta análise precisa ser distribuída entre os containeres pertencentes ao *grid*. Desta forma, é necessário que exista um registro dos containeres que compõem este *grid* de processamento e suas habilidades, para que seja possível esta distribuição. Na Figura 13 é apresentada uma visão abstrata da divisão das atividades de análise.

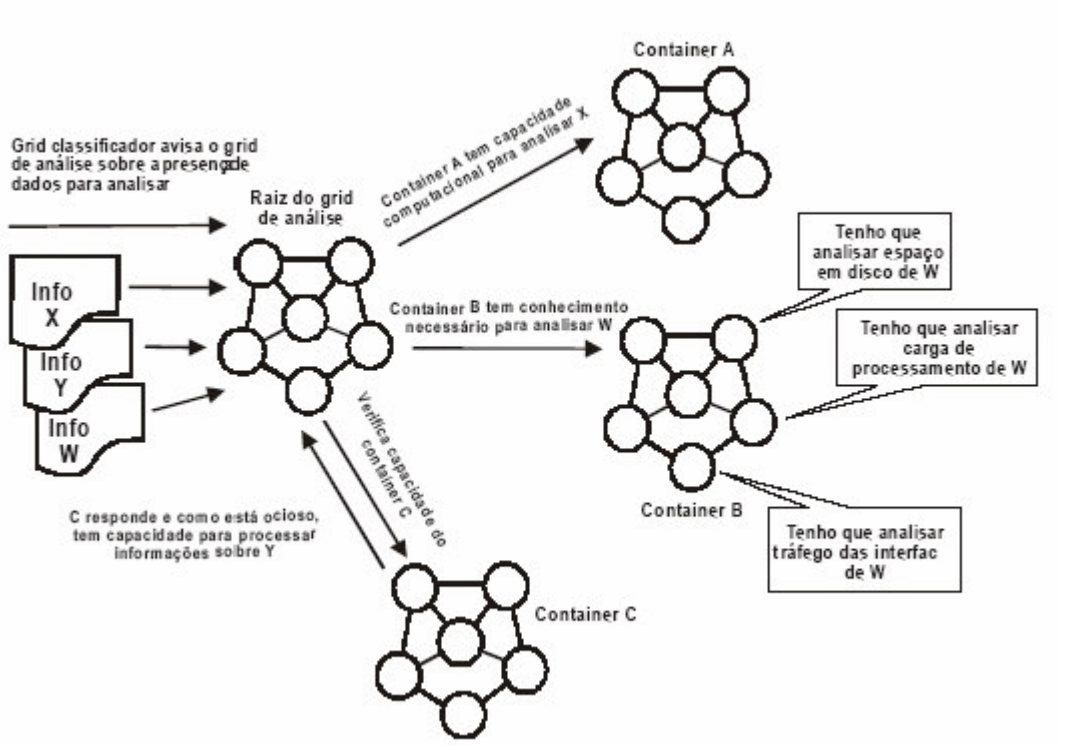


Figura 13 - Exemplo da divisão de tarefas no *grid* [ASSUNÇÃO 2003].

A principal dificuldade é proporcionar uma divisão das atividades de análise no *grid*. Distribuir esta análise sem que as informações percam a sua

essência é uma tarefa complicada. A divisão do conjunto de dados “problema” de aplicações distribuídas tem sido alvo de vários estudos e, na gerência de redes, esta divisão pode ser um pouco complicada, devendo ser tratada para não proporcionar uma perca de significado das informações. Daí a existência do *grid* classificador. O *grid* classificador prepara as informações para que as tarefas de análise possam ser distribuídas mais facilmente.

No fluxo de informações do *grid* de análise demonstrado na Figura 14, têm-se em I as informações sendo coletadas e quebradas em um conjunto de dados e armazenadas. O *grid* de análise é quem irá acessar estas informações. Em II, observa-se que o *grid* de análise distribui a carga, baseando-se em uma distribuição inteligente de acordo com a capacidade ou utilização do recurso ou ainda a disponibilidade de conhecimento para processá-la. Já em III, o conjunto de dados é processado localmente considerando a base de conhecimento global e a disponibilidade de recursos.

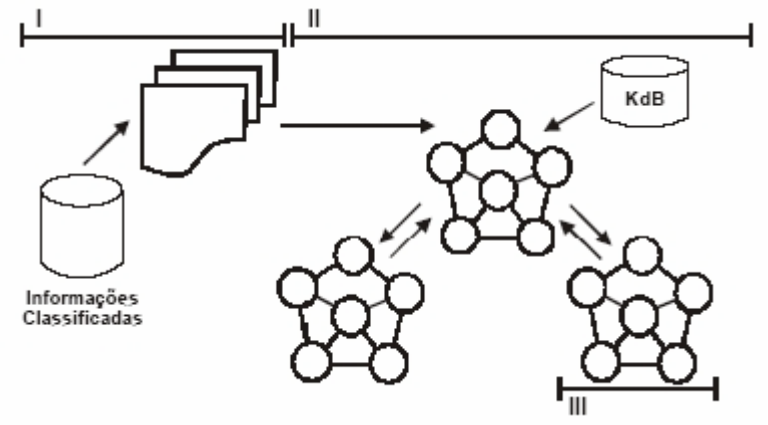


Figura 14 - Fluxo das informações no *grid* de análise [ASSUNÇÃO 2003].

Ao receber uma mensagem do *grid* classificador, indicando a presença de dados, o *grid* processador faz com que um número de agentes ou containeres, baseados em suas regras, procurem por problemas nestes dados. Esta análise não leva em consideração informações armazenadas anteriormente na base de dados e não procura por nenhuma relação entre os fatos. Neste caso, o objetivo é analisar dados e solicitar as informações de que precisam ao *grid* classificador, e

este se responsabiliza por encontrar no *grid* a melhor cópia destes dados e a mais rápida para ser recuperada. Porém, também se pode ter como análise, consolidações dos dados, extraindo informações armazenadas anteriormente. Com isso os agentes podem procurar por problemas e agrupar dados, constituindo algumas informações que servem para a construção de relatórios de gerência.

Como os agentes desta arquitetura são agentes de inferência orientados a objetivos, a análise dos dados no *grid* de análise se baseia em regras de inferência e sua distribuição no *grid* de análise é feita mediante a distribuição destas atividades. Este trabalho concentra-se no desenvolvimento de regras de inferência e análise utilizadas pelo *grid* analisador e da integração destas regras ao *grid* processador.

6.1 – Análise dos Dados

Sempre que dados relevantes são coletados, o *grid* de processamento é informado da existência destes dados para análise. O agente responsável pela análise busca no diretório, agentes com o conhecimento necessário para análise dos dados em questão. Quando este conjunto de agentes é conhecido, começa a fase de negociação. Dependendo do número de tarefas a serem executadas mais de um agente pode ser escolhido para este fim. Após a escolha dos agentes as tarefas são enviadas e executadas pelos agentes selecionados.

Porém, antes que possa existir uma distribuição de atividades de análise, é necessário que as regras para processamento dos dados sejam construídas. Como os agentes da plataforma utilizada para a implementação da arquitetura têm seu conhecimento representado em estruturas de lógica de primeira ordem, tais regras de análise devem ser construídas e modeladas de acordo com esta estrutura.

A construção de regras para analisar informações e identificar problemas que possam estar ocorrendo em estações da rede, como escassez de memória, uso do processador, entre outros, são exemplos de dados analisados no *grid* de análise. Para a construção das regras e identificação dos itens a serem analisados utilizaram-se as informações de atributos contidos nas especificações do modelo CIM e do pacote WMI da plataforma Windows da Microsoft.

A forma como a distribuição é realizada, ou quem realiza o escalonamento das atividades de análise, ou atribui as atividades aos agentes, é um outro fator a ser considerado. Assim, os nós que são inseridos no *grid* possuem tarefas de análise pré-definidas. Quando eles não são capazes de analisar estas informações usando os recursos que eles têm disponíveis, eles procuram no *grid* por outros agentes e delegam as atividades a eles. Eles então iniciam as atividades de análise, requerendo os dados de que precisam. Neste caso, cada agente responsável por um conjunto de análises é um escalonador em potencial, pois ele pode escalonar tarefas ou negociar com outros agentes a execução delas.

6.2 – Regras de Análise

Os relatórios de gerenciamento são construídos pelos agentes de análise de informações, que analisam um conjunto de dados resultando em relatórios ou notificações devolvendo-os aos armazenadores de dados para que, estes repassem aos agentes de interface no momento que o usuário solicitar. Em alguns casos devem ser especificados valores limites para compará-los com os coletados, a fim de identificar problemas como, por exemplo, superutilização.

Um exemplo de regra de análise pode ser o uso da memória. Neste caso, os dados coletados de cada sistema são comparados ao valor base e se ultrapassar este limite é criado um alerta que será mostrado ao usuário pela interface. Esta regra analisa informações das últimas coletas (como mostrada na

Figura 15), onde é usado como valor referência 80%. Ou seja, se o uso da memória está acima dos 80% é exibido um alerta. Têm-se também processamentos em relação à quantidade de vezes que o uso da memória ficou acima deste limite, como apresentado na Figura 16. Pode ser utilizado como valor referência, 50%. Assim, caso a memória fique mais de 50% do período analisado com o uso acima de 80% então é enviado outro alerta ao usuário.

```
[. . .]
analysis('memory_usage',System,InitialDate,FinalDate,Store) :-
     date(now,Date),
     date(string,Date,DateString),
     tree(create,TreeReport,"memory_usage"),
     tree(add,TreeReport,"system",System),
     agent(query-ref,Store,
                     listCollections(System,'baseline','MemoryAvailableMB','1',
                        InitialDate,FinalDate,ListAvailableMB)),
     eq('0',GThanThreshold),
     agent(query-ref,Store,
               getConfig(System,'memory','Capacity','1',Value)),
     __divide(MemoryAmount,Value,'1048576'),
     verifyMemory(System,ListAvailableMB,TreeReport,GThanThreshold,MemoryAmount),
     list(string,ListAvailableMB,String),
     list(size,ListAvailableMB,NumberCollections),
     math(percentage,GThanThreshold,NumberCollections,PercentageAbove),
     math(round,PercentageAbove,RoundedPercentage),
     memoryPercentTimesAboveThreshold(AcceptedPercentAbove),
     if(__gt(RoundedPercentage,AcceptedPercentAbove),
          agent(request,Store,
                      assertNotification(System,DateString,'Memory Overuse',
                         'The memory utilization is above the accepted threshold',
                         'high')),
          print(["\nMEMORY NOT ABOVE\n"])),
     tree(string,TreeReport,SerialReport),
     storeReport(System,DateString,"not-
          accumulative","memory_usage",SerialReport),
          print(["\nPASSED ANALYSIS - SYSTEM --> ",System,"\n"]),!.
[. . .]
```

Figura 15 – Parte da regra de análise do uso da memória.

```
[. . .]
   verifyMemory(System,ListAvailable,TreeReport,GTThreshold,MemoryAmount) :-
     list(head,ListAvailable,[Time,Value]),
     math(percentage,Value,MemoryAmount,Usage),
     math(round,Usage,RUsage),
     memoryThreshold(Threshold),
     if(__gt(RUsage,Threshold),
        __plus(GTThreshold,GTThreshold,'1'),
        eq(GTThreshold,GTThreshold)),
     list(tail,ListAvailable,RestOfList),
     tree(create,SubTree,"collection"),
     tree(add,SubTree,"time",Time),
     tree(add,SubTree,"value",Usage),
     tree(add,TreeReport,SubTree),
     verifyMemory(System,RestOfList,TreeReport,GTThreshold,MemoryAmount),!.
[. . .]
```

Figura 16 – Parte da regra referente ao percentual de vezes que o uso da memória ficou acima do limite estipulado.

A regra para uso do processador segue a mesma lógica, como é mostrado na Figura 17, e pode ter os mesmos valores de referência.

```
[. . .]
   analysis('processor_usage',System,InitialDate,FinalDate,Store) :-
     date(now,Date),
     date(string,Date,DateString),
     tree(create,TreeReport,"processor_usage"),
     tree(add,TreeReport,"system",System),
     agent(query-ref,Store,
                    listCollections(System,'baseline','ProcessorUsage','1',
                  InitialDate,FinalDate,ListUsage)),
     eq('0',GThanThreshold),
     list(create,TrashList,_),
     verifyProcessor(System,ListUsage,TreeReport,GThanThreshold,TrashList),
     list(string,ListUsage,String),
     list(size,ListUsage,NumberCollections),
     math(percentage,GThanThreshold,NumberCollections,PercentageAbove),
     math(round,PercentageAbove,RoundedPercentage),
     processorPercentTimesAboveThreshold(AcceptedPercentAbove),
     if(__gt(RoundedPercentage,AcceptedPercentAbove),
          agent(request,Store,
                   assertNotification(System,DateString,'Processor Overuse',
                     'The processor utilization is above the accepted threshold',
                     'high')),
          print(["\nPROCESSOR NOT ABOVE\n"])),
     tree(string,TreeReport,SerialReport),
     storeReport(System,DateString,"not-
           accumulative","processor_usage",SerialReport),
            print(["\nPASSED ANALYSIS - SYSTEM --> ",System,"\n"]),!.
[. . .]
```

Figura 17 – Parte da regra de análise do uso do processador.

Um exemplo de regra onde não se precisa de valores comparativos é a disponibilidade do sistema. Neste caso o relatório é criado apenas com a data e hora, inicial e final, em que o sistema não estava disponível. Além disso, também se calcula o tempo total de indisponibilidade e seu percentual. A Figura 18 apresenta parte do conjunto de regras necessárias ao agente de análise para

gerar o relatório de disponibilidade e na Figura 19 é apresentado o XML resultante.

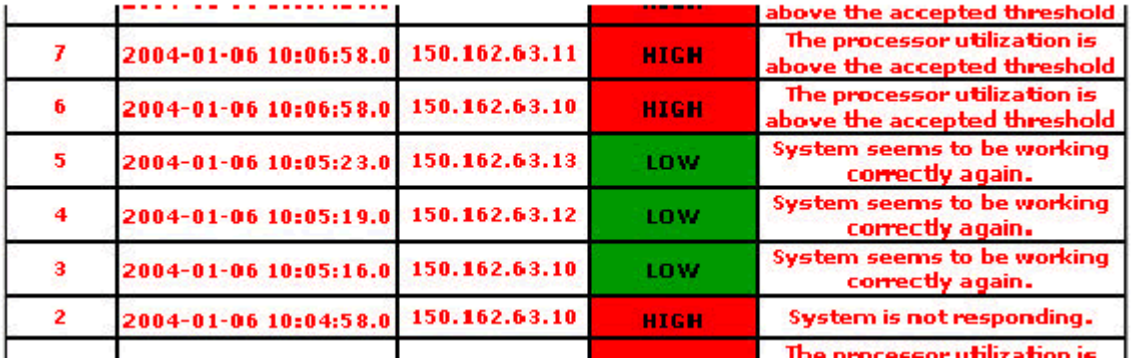
```
[. . .]
   analysis('system_availability',System,InitialDate,FinalDate,Store) :-
     date(diff,FinalDate,InitialDate,TotalTime),
     tree(create,TreeReport,"availability"),
     tree(add,TreeReport,"system",System),
     tree(add,TreeReport,"initialdate",InitialDate),
     tree(add,TreeReport,"finaldate",FinalDate),
     agent(query-ref,Store,
                    getOutages(System,InitialDate,FinalDate,ListOutages)),
     eq('0',TimeOutage),
     list(string,ListOutages,String),
     sumOutages(ListOutages,TimeOutage),
     calculateAvailability(TimeOutage,TotalTime,TreeReport),
     tree(string,TreeReport,SerialReport),
     storeReport(System,InitialDate,"accumulative",
                      'system_avail_daily',SerialReport),!.
[. . .]
```

Figura 18 – Parte da regra de análise da disponibilidade do sistema.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<availability>
     <system>150.162.63.13</system>
     <initialdate>2004-01-06 00:00:00.0</initialdate>
     <finaldate>2004-01-07 00:00:00.0</finaldate>
     <availability>100.0</availability>
</availability>
```

Figura 19 - XML resultante da análise de disponibilidade.

Os resultados das análises, como mencionado anteriormente, podem ser relatórios que são apresentados aos usuários, conforme apresentado na Figura 20, e/ou as notificações sobre os problemas que estiverem ocorrendo, Figura 21. As regras apresentadas são simples, porém demonstram o funcionamento do *grid* de processamento.



	[illegible]		[illegible]	above the accepted threshold
7	2004-01-06 10:06:58.0	150.162.63.11	HIGH	The processor utilization is above the accepted threshold
6	2004-01-06 10:06:58.0	150.162.63.10	HIGH	The processor utilization is above the accepted threshold
5	2004-01-06 10:05:23.0	150.162.63.13	LOW	System seems to be working correctly again.
4	2004-01-06 10:05:19.0	150.162.63.12	LOW	System seems to be working correctly again.
3	2004-01-06 10:05:16.0	150.162.63.10	LOW	System seems to be working correctly again.
2	2004-01-06 10:04:58.0	150.162.63.10	HIGH	System is not responding.
				The processor utilization is

Figura 20 – Tela com os alertas gerados pelo sistema.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" ref="../xsl/notifications.xsl"?>
<notifications>
[...]
    <notification>
        <system>150.162.63.11</system>
        <id>7</id>
        <text>The processor utilization is above the accepted threshold</text>
        <date>2004-01-06 10:06:58.0</date>
        <subject>Processor Overuse</subject>
        <severity>high</severity>
    </notification>
    <notification>
        <system>150.162.63.13</system>
        <id>17</id>
        <text>System is not responding.</text>
        <date>2004-01-06 12:20:57.0</date>
        <subject>System Outage</subject>
        <severity>high</severity>
    </notification>
    <notification>
        <system>150.162.63.13</system>
        <id>20</id>
        <text>System seems to be working correctly again.</text>
        <date>2004-01-06 14:25:54.0</date>
        <subject>System Responding</subject>
        <severity>low</severity>
    </notification>
[...]
</notifications>
```

Figura 21 – Relatório em XML contendo notificações.

No relatório apresentado na Figura 21 existem exemplos de três notificações: um avisando que um sistema deixou de funcionar, outro que o sistema retornou, e um alertando que a utilização do processador em um determinado sistema ultrapassou os limites considerados ideais.

6.3 – Distribuição das Atividades de Análise

Assunção [2004] realizou testes da distribuição das atividades de análise de dados de gerenciamento. Os testes foram realizados no Laboratório de Redes e Gerência da Universidade Federal de Santa Catarina, que conta com uma rede local e poucos equipamentos. Foram utilizadas tarefas de análise, como avaliação da porcentagem do uso do processador dos sistemas e utilização de memória. Estas tarefas são designadas aos agentes de processamento do *grid*, de acordo com o critério usado para distribuição, para que estes realizem as análises correspondentes.

Como um primeiro experimento, todos os agentes foram registrados no *grid*, possuindo o conhecimento ou regras para realizarem as tarefas de análise. O agente responsável por delegar as tarefas efetuou uma distribuição cíclica das tarefas usando como parâmetros descritos na Figura 22.

Parâmetro	Valor
Tempo de chagada de uma tarefa de análise	10 segundos
Tempo de duração das medições	Aproximadamente 10 minutos
Intervalo de dados	2 hora

Figura 22 – Parâmetros de simulação.

Na segunda experimentação foi testada uma distribuição aleatória entre os agentes aptos a realizarem a tarefa.

E uma última abordagem foi a utilização de medidores para o fornecimento de informações de desempenho e capacidade dos recursos computacionais de forma a auxiliar a distribuição das tarefas. Estas informações são usadas para guiar o agente responsável pelo escalonamento, na decisão, a respeito de qual agente está mais apto a realizar uma determinada atividade de análise. Os medidores usados no teste foram as características e uso do

processador, porém outros fatores poderiam ser utilizados. A Figura 23 apresenta de forma abstrata como funciona esta abordagem de distribuição.

O resultado do tempo de resposta de cada experimentação pode ser visualizado na Figura 24. Embora o tempo de resposta da última abordagem seja menor, o tempo usado pelo mecanismo de decisão, e pergunta aos agentes do atual uso de recursos não foi contabilizado nestes experimentos. Esta questão pode causar sérios problemas como o aumento no tempo de resposta. Uma proposta apresentada na arquitetura de *grid* é que o serviço de diretórios mantido pela raiz armazene um conjunto de informações a respeito da capacidade computacional dos recursos existentes no *grid*, evitando que estas informações sejam buscadas a cada vez que uma tarefa precisa ser executada [ASSUNÇÃO 2004].

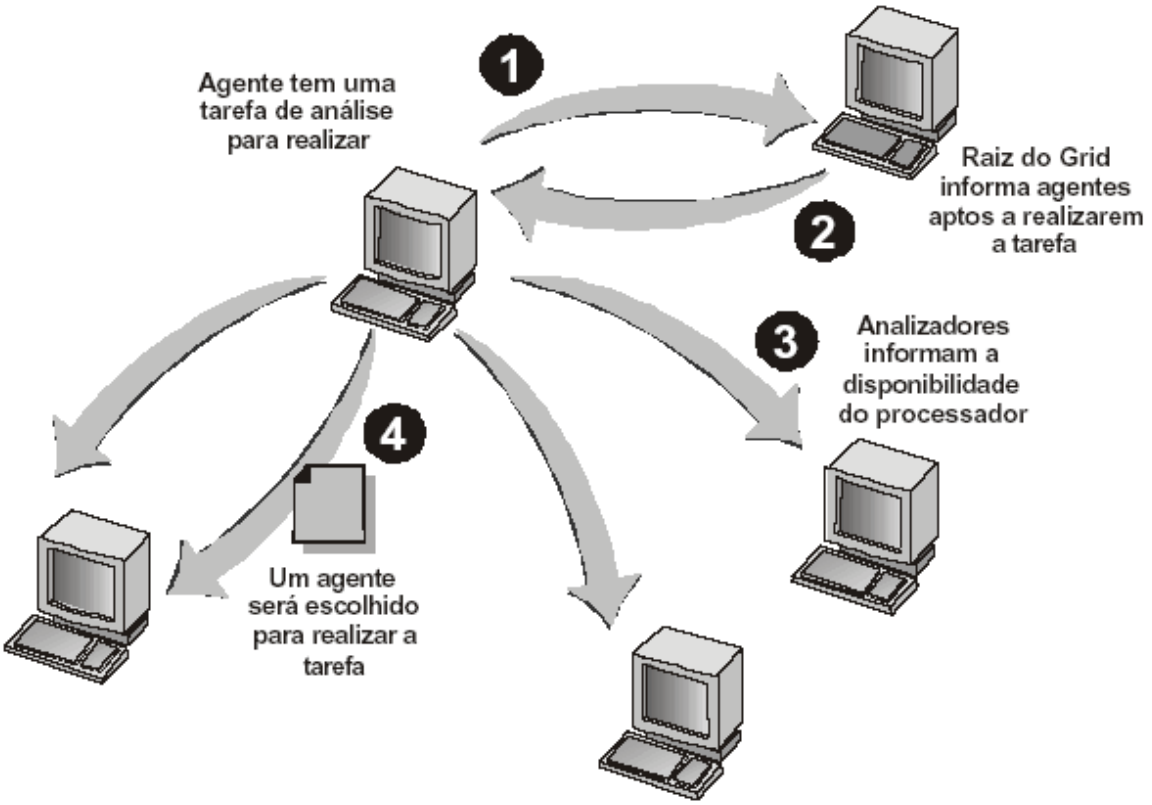


Figura 23 – Distribuição baseada no processador [ASSUNÇÃO 2004].

Tanto a abordagem de distribuição aleatória, quanto à distribuição cíclica apresentam uma equalização da utilização dos recursos nas atividades de análise de dados de gerência. Porém, em ambas, o tempo médio de respostas tende a ser um pouco maior, pois as tarefas são escalonadas também para máquinas com poucos recursos computacionais.

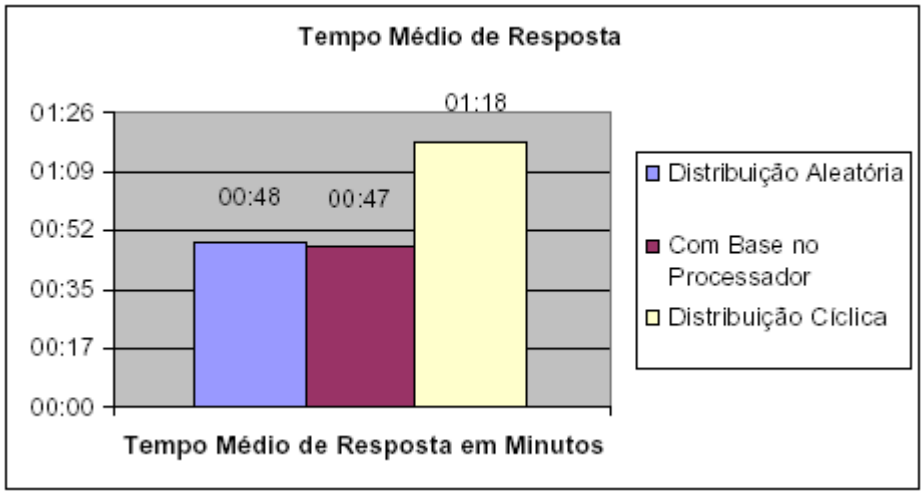


Figura 24 – Tempo médio de resposta na execução das tarefas [ASSUNÇÃO 2004].

7 – Conclusão e Trabalhos Futuros

Para um ambiente dinâmico como uma rede de computadores uma solução de gerência é a aplicação de um *grid* de agentes, que funciona de maneira distribuída. Desta maneira, o sistema não é sobrecarregado durante procedimentos gerenciais que utilizam muito processamento, como por exemplo, a tarefa de análise dos dados, pois esta pode ser escalonada a vários pontos conectados ao *grid* que tenham conhecimentos para tal.

O trabalho apresentou previamente uma introdução aos assuntos relacionados ao tema, como por exemplo, gerência de redes, o que são *grids* de agentes e como estes podem ser aplicados na gerência. Além dos benefícios desta abordagem e as vantagens, possibilitando uma melhor utilização dos recursos da rede.

Foi estabelecido um ambiente experimental no Laboratório de Gerência de Redes da Universidade Federal de Santa Catarina, composto por uma pequena rede local. Ali, a arquitetura completa proposta foi implementada e testada.

Com o foco voltado ao *grid* de agentes processadores, pode-se perceber que este é o centro da arquitetura. É nesta etapa que os dados coletados são transformados em informações gerenciais, consolidando-os em relatórios de gerência a serem exibidos aos usuários. Os relatórios podem ser apresentados aos usuários apenas com informações referentes a certo intervalo de tempo, ou podem ainda considerar análises anteriores, apresentando uma consolidação dos dados.

Devido a complexidade do paradigma de *grid* computacional, foram desenvolvidas apenas algumas regras simples de processamento para a gerência de redes. Porém, os resultados indicam que a abordagem atinge com sucesso os objetivos a que se propõe. Novas regras podem ser acrescentadas para atingir cenários maiores e com outras necessidades.

Neste momento é tratada a questão de continuidade do trabalho. Para tanto, são sugeridas algumas melhorias que, com certeza, contribuem para o enriquecimento do trabalho:

- Desenvolvimento de novas regras de análise, que atinjam um maior número de variáveis e cenários reais de gerenciamento.
- Aplicar as regras em redes maiores, envolvendo múltiplos domínios administrativos e com testes mais apurados.
- Aplicação em redes sem fio.

Referências Bibliográficas

[AGENTLIGHT] ***Platform for Lightweight Agents***. Disponível em: <http://www.agentlight.org>.

ASSUNÇÃO, M. D. **Implementação e Análise de uma Arquitetura de *Grids* de Agentes para a Gerência de Redes e Sistemas.** CPGCC – UFSC. Florianópolis – SC, 2004. Dissertação de Mestrado.

[ASSUNÇÃO 2003] ASSUNÇÃO, M. D.; WESTPHALL, C. B.; KOCH, F. L**. Arquitetura de *Grids* de Agentes Aplicada à Gerência de Redes de Computadores e Telecomunicações.** In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 21., 2003, Natal-RN. Proceedings... Natal: 21º Simpósio Brasileiro de Redes de Computadores, 2003, p. 789-804.

ASSUNÇÃO, M. D.; WESTPHALL, C. B.; KOCH, F. L. ***Grids of Agents for Computer and Telecommunication Network Management.*** *In: ACM/IFIP/USENIX INTERNATIONAL MIDDLEWARE CONFERENCE. INTERNATIONAL WORKSHOP ON MIDDLEWARE FOR GRID COMPUTING, 1.,* Rio de Janeiro, Proceedings... Rio de Janeiro: *1st Workshop on Middleware for Grid Computing*, p. 186-193, jun. 2003. (Artigo selecionado para ser publicado no *Journal of Concurrency and Computation*: *Practice and Experience*, John Wiley & Sons, Inc. Março/Abril 2004).

[ASSUNÇÃO, WESTPHALL e KOCH 2003A] ASSUNÇÃO, M. D.; WESTPHALL, C. B.; KOCH, F. L. **Vantagens do uso de *grids* de agentes no gerenciamento de redes de computadores.** Agent's Day – CBComp. 2003. p. 1358-1369

BERSTIS V. ***Fundamentals of Grid Computing.*** *ReadPaper IBM. International Technical Support Organization.* Outubro de 2002.

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

CASANOVA, H. ***Distributed Computing Research Issues in Grid Computing****. ACM SIGACT News Distributed Computing Column* 8. Julho 2002.

CIRNE, W. **Grids Computacionais: Arquiteturas, Tecnologias e Aplicações.** Universidade Federal de Campina Grande. Disponível em: <http://www.dsc.ufpb.br /~raissa/doutorado.htm>. Acesso em: 28/04/2004.

COABS – **CONTROL OF AGENT BASED SYSTEMS**. Disponível em: <http://coabs.globalinfotek.com>. Acesso em: 13/03/2004.

DISTRIBUTED.NET PROJECT, 2002. Disponível em: <http://www.distributed.net> Acesso em: 13/03/2004.

DMTF. **Web-Based Enterprise Management (WBEM) Initiative**. Disponível em: <http://www.dmtf.org/standards/wbem>. Acesso em: 28/08/2003.

FERNANDES, A. M. da R. **Inteligência Artificial – noções gerais.** Visual Books. Pg. 85-113, Mar. 2003.

FIPA - **The Foundation For Intelligent For Physical Agent**. Disponível em: <http://www.fipa.org >. Acesso em: 13/03/2004.

FOSTER, I ***What is the Grid? A Three Point Checklist****. GRID today. Volume 1,* no. 6 Julho, 2002. Disponível em: <http://www.gridtoday.com/02/0722/100136.html >. Acesso em: 28/08/2003

FOSTER, I, e KESSELMAN, C. ***The Grid: Blueprint for a New Computing Infrastructure****, Morgan Kaufmann, Morgan-Kaufmann*, Orlando, FL, 1999.

FOSTER, I.; KESSELMAN, ***C. Globus: A Metacomputing Infrastructure Toolkit,*** *International Journal of Supercomputer Applications*, v.11, n.2, p.115-128, 1997.

FOSTER, I; KESSELMAN, C.; TUECKE S. **The Anatomy of the Grid: Enabling Scalable Virtual Organizations**. International Journal of Supercomputer Applications, v.15 n.3, 2001.

[FOSTER 2002] FOSTER, I.; KESSELMAN, C.; NICK, J.; TUECKE, S. **The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration**. Open Grid Service Infrastructure WG, Global Grid Forum, jun. 2002. Disponível em: <http://www.globus.org/research/papers.html>. Acesso em: 28/04/2004.

GUBERT, L. C. **Utilizando o padrão de gerenciamento SNMP para gerenciar o tráfego *multicast*: a ferramenta *multicast* monitor.** CPGCC – UFSC. Florianópolis – SC, 2002. Dissertação de Mestrado.

JENNINGS, N. R.; SYCARA, K.; WOOLDRIDGE, M. ***A Roadmap of Agent Research and Development.*** *Academic Publishers*, Boston. Pg. 275-306, 1998.

KOCH, F. L.; WESTPHALL, C. B. ***Decentralized Network Management using Distributed Artificial Intelligence****. Journal of Network and Systems Management. Plenum Publishing Corporation. Meddletown, USA* (2001, Vol. 9, No. 4). Pg. 375-388 , Dec. 2001.

LANGE, B. D.; OSHIMA, M. ***Seven Good Reasons for Mobile Agents****. Communications of ACM*, 42(3). pp. 88-89. Março 1999.

LEGION. **A Worldwide Virtual Computer**, 1997. *Site* do projeto Legion. Disponível em: <http://legion.virginia.edu>. Acesso em: 13/03/2004.

LIMA, F. S. **GERÊNCIA DE REDES DE COMPUTADORES**. FRB. Salvador – BA, 2003. Disponível em: <http://www.wernet.com.br/~fabricio/Gerencia_ de_ Redes. pdf>. Acesso em 29/08/2003

[LUCK 2003] LUCK, M., MCBURNEY, P., PREIST, C. **Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent Based Computing)**, AgentLink, 2003, ISBN 0854 327886.

MICROSOFT. Disponível em: <http://www.microsoft.com/technet/treeview/default. asp?url=/technet/prodtechnol/windows2000serv/evaluate/featfunc/wmiovw.asp>. Acesso em: 28/08/2003

MÜLLER, J. P. ***The Design of Autonomous Agents: A Layered Approach****. Volume 1177 of Lecture Notes in Artificial Intelligence*. Springer-Verlag, Heidelberg, 1997.

NWANA, H. S.; NDUMU, D. **An Introduction to Agent Technology**. Intelligent Systems Research, Applied Research and Technology, BT Labs. Disponível em: <http://more.btexact.com/projects/agents/publish/papers/intro_agents.htm>.Acesso em: 30/03/2004.

PITANGA, M. ***Grid Computing*****.** Artigo Clube do Hardware. Março de 2004. Disponível em: <http://www.clubedohardware.com.br/grid-2.html>. Acesso em: 15/05/2004

[PITANGA 2004A] PITANGA, M. ***Internet x Grid x Cluster*****.** Artigo Clube do Hardware. Março de 2004. Disponível em: <http://www.clubedohardware.com.br/ grid-3.html>. Acesso em: 15/05/2004

[ROCHA 2003] ROCHA, A. R.; SOUZA E. M.; ALVES J. J. C. **Introdução aos Sistemas Inteligentes e aos Sistemas Multi-Agentes.** Universidade Federal de Lavras. Abril de 2003. Disponível em: <http://www.comp.ufla.br/~undersun/ic/ia/ IntroduçãoAgentesInteligentesEAosSistemasMultiagentes.pdf>.
Acesso em 06/01/2004.

SETI@HOME. ***Search for Extraterrestrial Intelligence***, 1996. *Site* do projeto. Disponível em: <http://setiathome.ssl.berkeley.edu>. Acesso em: 13/03/2004.

SILVA, A. R. V. **Utilização de Agentes para Definição e Alteração de Bancos de Dados Heterogêneos.** CPGCC – UFSC. Florianópolis – SC, 2003. Dissertação de Mestrado.

SIMÕES, M. A. C. **Gerência Corporativa de Sistemas Baseada na Web utilizando Agentes Móveis Inteligentes.** Proposta de Dissertação de Mestrado, Recife. Fev, 28 2002

SPINDOLA, T. **Balanceamento de Carga em Grid de Agentes.** CPGCC – UFSC. Florianópolis – SC, 2003. Dissertação de Mestrado.

STALLINGS, W. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2.** Reading, Massachusetts: Addison Wesley Longman, Inc, 1999. ISBN: 0-201-48534-6.

SUBRAMANIAN, M. **Network management: An introduction to principals and practice**. [Reading, Massachusetts]: Addison-Wesley, 2000. 644p, ISBN: 0-201-35742-9.

SYPERREK, B.; ASSUNÇÃO, M. D.; WESTPHALL, C. B. **Grid de Agentes: Os Padrões FIPA na Construção dos Serviços do Grid para Garantia de Interoperabilidade.** 2003.

WEBER, T. S. *et all.* **Dependable Grid.** Projeto fomentado pela HP Brasil. Março de 2004. Disponível em: <http://www.inf.ufrgs.br/~taisy/projetos/GRIFIN.htm>. Acesso em: 15/05/2004

[WILLMOTT 2001] WILLMOTT, S. N., DALE, J., BURG, B, CHARLTON, C., O'BRIEN, P. **Agentcities: A Worldwide Open Agent Network**. Agentlink News. nov. 2001, p.13-15. Disponível em: <http://www.AgentLink.org/newsletter/8/AL-8.pdf>. Acesso em: 30/03/2004.

YEPES, Igor. **Sistemas Multi-Agentes**. Projeto ISIS. Disponível em: <http://www.geocities.com/igoryepes/agentes.htm>. Acesso em: 28/08/2003

APÊNDICE

Grids de Agentes Processadores para Gerência de Redes de Computadores e Telecomunicações

Maria Tereza Nagel

Resumo

Uma das etapas mais importantes no gerenciamento de redes e sistemas computacionais é a análise das informações coletadas. Os agentes de processamento são os responsáveis por realizar as inferências sobre os dados armazenados em busca de problemas que possam estar ocorrendo na rede gerenciada. O resultado da análise é composto por informações de gerência que darão origem aos relatórios e alertas para serem enviados aos agentes de interface.

Abstract

One of the most important steps in network and computational systems management is the information analysis. In order to that, it is utilized processors agents that are responsible to make inferences over stored data searching for problems that might be ocurring at managed network. The analysis' result is composed by management informations that will create reports and alerts to be sent to interfaces' agents.

1 - Introdução

Assunção, Westphall e Koch [ASS03] propuseram, utilizando *grids* de agentes, uma arquitetura para o gerenciamento de redes, que através da distribuição das tarefas de manipulação e análise dos dados proporciona maior extensibilidade e adaptabilidade que os modelos centralizados tradicionalmente utilizados. Levando em consideração o fluxo de trabalho tradicional do gerenciamento, algumas atividades foram identificadas: coleta de dados dos dispositivos da rede, classificação e armazenamento destes dados, análise dos dados a fim de transformá-los em informações de gerência e apresentação destas informações.

O *grid* de processamento ou análise é a parte mais importante da arquitetura e onde estão localizados os maiores desafios de desenvolvimento. É responsável pela consolidação dos dados coletados em relatórios de gerência. Os principais problemas que podem surgir dizem respeito à divisão das atividades de análise, controle de recursos, balanceamento de carga e tolerância à falhas. O resultado da análise é composto por informações de gerência que dão origem aos relatórios e alertas enviados ao *grid* de interface.

1.1 - Estado da Arte

As aplicações citadas por alguns como as primeiras tentativas de computação em *grid* são a análise de sinais de rádio na busca de vida extraterrestre inteligente (SETI@Home) [SETI], o superprocessador virtual proporcionado pela *United Devices* e o *Distributed.Net* [DISTR]. Embora atualmente estas aplicações sejam consideradas aplicações *peer-to-peer* (P2P) por alguns, elas são citadas por muitos como exemplos de *grid* e também como tecnologias precursoras. Uma segunda corrente de desenvolvimento do *grid* iniciou com a tentativa de criar *middlewares* que facilitem a computação em larga escala, ou em *grid*.. Alguns trabalhos pioneiros são o *Legion* [LEGION] e o *Globus* [FOS97].

Um dos trabalhos mais expressivos sobre *grids* de agentes é parte do projeto CoABS (*Control of Agent-Based Systems*) [COABS]. O CoABS é um programa do DARPA (*Defense Advanced Research Projects Agency*) e do *US Air Force Rome Labs.* O programa tem por objetivo investigar o uso da tecnologia de agentes para melhorar operações militares de comando, controle, comunicação, acúmulo e acesso a informações importantes no planejamento de ações militares. O *grid* CoABS também é responsável por facilitar a integração dinâmica dos vários sistemas desenvolvidos pelos pesquisadores do projeto CoABS no estabelecimento de coalizões construídas em campos de batalha.

2 - Gerência de Redes

A idéia básica na solução de um sistema de gerência consiste na utilização de um computador interagindo com os diversos componentes da rede a serem gerenciados, para ser possível extrair desses, as informações necessárias à gerência [GUB02]. Utilizando uma ferramenta de gerência de rede o administrador pode entre outras coisas, controlar melhor o ambiente verificando o uso dos recursos computacionais, planejar um estudo de capacidade, definir medidas pró-ativas na detecção de “gargalos” encontrados no ambiente, além de ser notificado imediatamente ao ocorrer uma falha em um dos elementos gerenciados, o que minimiza o tempo de parada do serviço prestado.

As etapas no processo de gerência de redes [STA99] são basicamente:

- Coleta de dados: processo responsável por extrair as informações de gerência dos equipamentos.
- Diagnóstico: analisa os dados coletados a fim de determinar causas dos problemas apresentados.
- Ação: após determinar a causa do problema, age-se para solucioná-lo na tentativa de evitar seu reaparecimento.

3 - Agentes

Existem diversas definições para o termo agente que segundo [Costa *apud* FER03] é utilizado na literatura computacional para determinar diversos tipos de programas. Geralmente estas definições dependem da funcionalidade de cada agente. De acordo com [Amandi *apud* SIL03], um agente é uma entidade computacional com um comportamento autônomo que lhe permite tomar suas decisões para agir, levando em consideração as mudanças acontecidas no ambiente em que atua e o seu objetivo. Uma definição bastante utilizada é a de que “um agente é uma entidade que percebe o mundo através de seus sensores e atua sobre ele por meio de seus atuadores”.

As características e propriedades dos agentes não são iguais e necessárias a todos, porém, a capacidade de cada agente está ligada a presença destas funcionalidades. Algumas destas características são:

- Aprendizagem: mesmo quando um agente não reconhece nenhuma ação a ser executada, é esperado que ele procure encontrar uma saída. A questão não é acertar sempre, mas aprender através da experiência.
- Autonomia: o sistema deve ser capaz de agir sem a intervenção direta dos humanos ou de outros agentes computacionais, e deve ter controle sobre suas próprias ações e estado interno.
- Cooperação: os agentes devem ser capazes de trabalharem em conjunto para finalizarem as tarefas interagindo com outros agentes computacionais ou com agentes humanos.
- Mobilidade: esta característica refere-se à capacidade de um agente se mover dentro de uma rede de computadores. Esta característica, porém, vem acompanhada de problemas de segurança.
- Reatividade: capacidade de perceber mudanças em seu ambiente e reagir adequadamente de acordo com estas mudanças.

4 - *Grids*

Computação em *grid*, ou *grids* computacionais, são uma forma de acesso universal, facilitado, confiável e seguro a agrupamentos dinâmicos de recursos computacionais heterogêneos dispersos geograficamente [FOS99]. Os principais atrativos desta idéia são a possibilidade de alocar uma grande quantidade de recursos a uma aplicação e fazê-lo a um custo muito menor do que alternativas tradicionais. A tecnologia de computação em *grid* possibilita agregar sistemas heterogêneos e dispersos, formando a imagem de um único "supercomputador virtual", oferecendo uma variedade de recursos virtuais. Os usuários do *grid* podem ser organizados dinamicamente, cada qual com diferentes políticas de solicitações, mas podendo compartilhar seus recursos coletivamente.

Como benefícios do uso de um *grid,* diferenciando-o das tecnologias distribuídas e paralelas tradicionais, são citados:

- Organizações podem agregar recursos: a computação em *grid* permite que as organizações possam agregar recursos não importando localização global. O compartilhamento não é limitado a arquivos, inclui também vários outros recursos, como: equipamentos, software, serviços, licenças, e outros.
- Acesso distribuído a diversos tipos de recursos: permite que empresas acessem e compartilhem bases de dados de forma remota. Além disso, muitas vezes, as máquinas têm enormes capacidades de disco não usadas.
- Possibilidade de Falhas e Adaptabilidade: Em situações críticas, como em tempo real, cópias múltiplas de tarefas importantes podem ser executadas em diferentes máquinas por todo o *grid*. Seus resultados podem ser checados para verificar qualquer tipo de inconsistência, tal como falhas do computador, dados corrompidos, ou correção de dados.
- Escalabilidade: um *grid* pode crescer de uma quantidade pequena de recursos até a faixa de milhões.

O gerenciamento pode usar um *grid* para melhorar a visão do uso de padrões em grandes organizações, permitindo um melhor planejamento quanto à atualização de sistemas, aumento da capacidade, ou reservando recursos computacionais não mais necessários.

4.1 - *Grids* de Agentes

Grids de agentes são, basicamente, *grids* que utilizam como infra-estrutura sistemas de agentes de software dispersos pela rede que possibilitam uma forma barata e eficiente de unir estes agentes e possibilitar o uso coordenado de recursos nas atividades de gerenciamento.

Além das vantagens de distribuição da análise, a utilização de *grids* de agentes traz as seguintes possibilidades [ASS03A]:

- Exploração de recursos pouco utilizados, agregando-os na execução de tarefas de gerenciamento.
- Podem ser adicionados containeres de agentes para efetuar um tipo particular de análise, ou para prever valores futuros.
- Para reduzir o tráfego de informações na rede, o armazenamento de dados pode ser realizado de forma a mantê-los mais próximos do local onde serão processados.
- Balanceamento inteligente de carga e de recursos baseado na capacidade dos recursos, na sua ociosidade e na disponibilidade de conhecimento para o processamento.
- Integração e compartilhamento de softwares existentes, como de gerenciamento.

4.2 - *Grids* de Agentes na Gerência de Redes

O gerenciamento centralizado de rede pode levar a situações onde não existem recursos computacionais suficientes para realizar um gerenciamento eficiente [SYP03], assim como comprometer o sistema com a estação de gerência fora de funcionamento. Tomando a gerência de redes de computadores como uma aplicação em *grid*, onde existe um grande volume de dados que precisa ser transformado em informações de gerência, encontra-se um cenário onde *grids* de agentes podem ser aplicados. É possível efetuar uma distribuição e um balanceamento de carga da análise dos dados coletados, baseando-se na disponibilidade e capacidade dos recursos do *grid*. Se o sistema requer uma capacidade de processamento maior, basta adicioná-la ao *grid*. Dessa forma, pode-se obter ganhos significativos e uma redução de hardware considerável.

5 - Arquitetura Proposta

Assunção [2003] propôs, utilizando *grids* de agentes, uma arquitetura para o gerenciamento de redes que possibilita uma distribuição da análise e do armazenamento das informações, conforme descrito em simulações [ASS03], que demonstram os resultados esperados da arquitetura.

No *grid* de agentes de gerenciamento, conforme apresentado na Figura 1, pode-se identificar algumas tarefas distintas:

- Os *grids* de agentes coletores (GC) são responsáveis pela coleta das informações de gerenciamento dos equipamentos da rede gerenciada.
- O *grid* classificador (GCL) é responsável por classificar as informações que recebe dos coletores, armazená-las de uma forma estruturada e mais fácil de ser recuperada, facilitando a distribuição e realização da análise.
- O *grid* de processamento ou análise (GP) é a parte mais importante da arquitetura. É responsável pela consolidação dos dados coletados em relatórios de gerência. Os resultados das análises realizadas são relatórios e alertas enviados ao *grid* de interface.
- O *grid* de interface (GI) é o canal de comunicação entre o *grid* de gerência e o gerente da rede. É também uma forma de receber o retorno do gerente humano e fornecê-lo ao sistema.

O foco deste trabalho é nos agentes de análise dos dados do *grid,* ou seja, no funcionamento do *grid* de agentes processadores, e por isto o assunto será detalhado em capítulo especial.

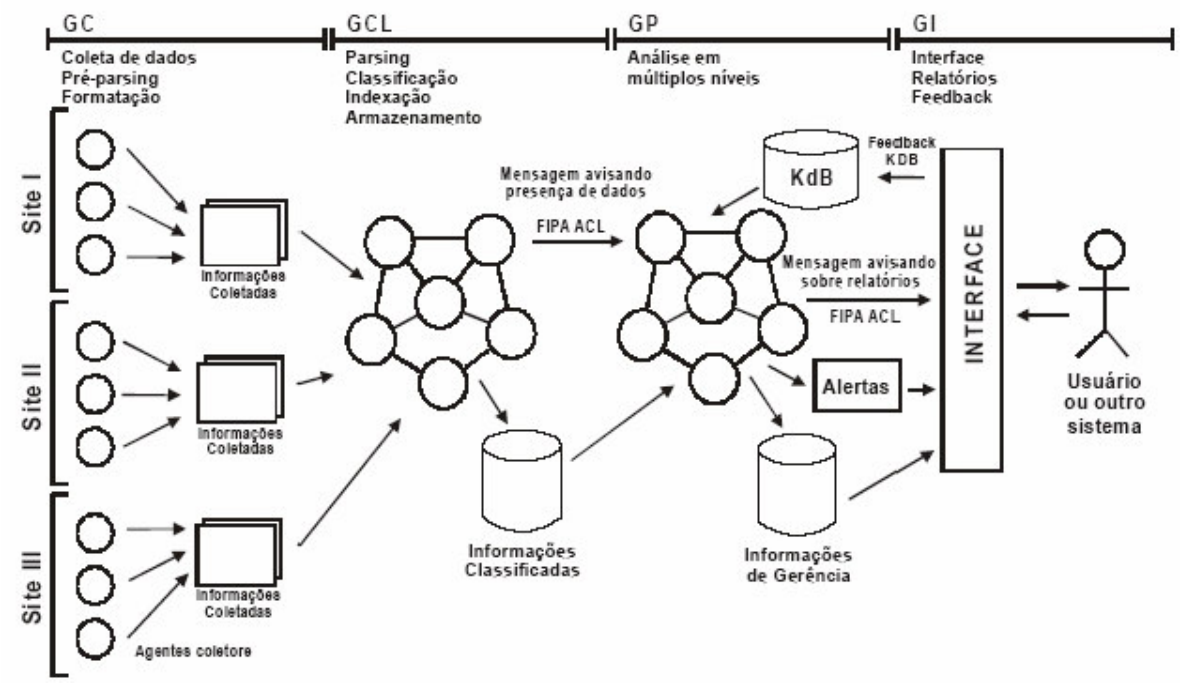


Figura 25 - Arquitetura geral do sistema [ASS03].

5.1 – Arquitetura de Agentes

Para o desenvolvimento da arquitetura de *grid* proposta, foi escolhido o *framework* para a construção de agentes, *Agentlight,* proposta por Koch [AGENT]. Alguns fatores que influenciaram na escolha desta ferramenta são:

- Disponibilidade do código fonte para alterações.
- Simplicidade e flexibilidade da plataforma.
- O *framework* é todo desenvolvido em Java.
- Um dos objetivos do *framework* é possibilitar a construção de agentes para pequenos dispositivos, o que facilita a sua utilização em diferentes plataformas.
- Como o *Agentlight* tem como princípio básico proporcionar uma plataforma para desenvolvimento de agentes para pequenos dispositivos, os agentes são criados com o intuito de serem bastante conservativos no que se refere ao tamanho de código compilado e utilização de memória.

O *Agentlight* funciona com o conceito de containeres de agentes. Este conceito é utilizado para que se possa ter um bom número de agentes em execução com um uso mínimo de recursos do sistema. Os agentes da plataforma utilizam um mecanismo de raciocínio baseado em um motor de inferência que infere sobre uma base de conhecimento e que possui fatos e regras armazenados usando uma estrutura baseada em lógica de primeira ordem.

6 - *Grids* de Agentes Processadores

A camada de análise é a camada que realiza as inferências sobre os dados armazenados em busca de problemas que possam estar ocorrendo na rede gerenciada. Esta camada faz uso das informações armazenadas e classificadas pelo *grid* de armazenamento. Desta forma, além de técnicas de escalonamento, é necessário ter o maior número de regras possíveis, além de distribuí-las entre os vários agentes do *grid*. O desenvolvimento destas regras é extremamente importante para o desenvolvimento do sistema e tendo um número grande de regras pode-se variar a utilização do *grid* aplicando-o a cenários e redes diferentes.

Este *grid* é composto por containeres [MUL97] de agentes distribuídos em vários computadores. A idéia é possibilitar um ambiente dinâmico onde containeres de agentes podem ser adicionados ao *grid* ou removidos dele constantemente. Ao ser adicionado, ele anuncia seus serviços ao *grid* e passa a analisar as informações que lhe são designadas.

Sempre que dados relevantes são coletados, o *grid* de processamento é informado da existência destes dados para análise. O agente responsável pela análise busca no diretório, agentes com o conhecimento necessário para análise dos dados em questão. Quando este conjunto de agentes é conhecido, começa a fase de negociação. Dependendo do número de tarefas a serem executadas mais de um agente pode ser escolhido para este fim. Após a escolha dos agentes as tarefas são enviadas e executadas pelos agentes selecionados.

Porém, antes que possa existir uma distribuição de atividades de análise, é necessário que as regras para processamento dos dados sejam construídas. Como os agentes da plataforma utilizada para a implementação da arquitetura têm seu conhecimento representado em estruturas de lógica de primeira ordem, tais regras de análise devem ser construídas e modeladas de acordo com esta estrutura.

6.1 - Regras de Análise

Os relatórios de gerenciamento são construídos pelos agentes de análise de informações, que analisam um conjunto de dados resultando em relatórios ou notificações devolvendo-os aos armazenadores de dados para que, estes repassem aos agentes de interface no momento que o usuário solicitar. Em alguns casos devem ser especificados valores limites para compará-los com os coletados, a fim de identificar problemas como, por exemplo, superutilização.

Um exemplo de regra de análise pode ser o uso da memória e do processador. Neste caso, os dados coletados de cada sistema são comparados ao valor base e se ultrapassar este limite é criado um alerta que será mostrado ao usuário pela interface. Têm-se também processamentos em relação à quantidade de vezes que o uso da memória ficou acima deste limite.

Um caso onde não se precisa de valores comparativos é a disponibilidade do sistema. Neste caso o relatório é criado apenas com a data e hora, inicial e final, em que o sistema não estava disponível. Além disso, também se calcula o tempo total de indisponibilidade e seu percentual.

Os resultados das análises, como mencionado anteriormente, podem ser relatórios que são apresentados aos usuários, e/ou as notificações sobre os problemas que estiverem ocorrendo. As regras apresentadas são simples, porém demonstram o funcionamento do *grid* de processamento.

6.2 - Distribuição das Atividades de Análise

Assunção [2004] realizou testes da distribuição das atividades de análise de dados de gerenciamento. Os testes foram realizados no Laboratório de Redes e Gerência da Universidade Federal de Santa Catarina, que conta com uma rede local e poucos equipamentos. Foram utilizadas tarefas de análise, como avaliação da porcentagem do uso do processador dos sistemas e utilização de memória. Estas tarefas são designadas aos agentes de processamento do *grid*, de acordo com o critério usado para distribuição, para que estes realizem as análises correspondentes.

Como um primeiro experimento, todos os agentes foram registrados no *grid*, possuindo o conhecimento ou regras para realizarem as tarefas de análise. O agente responsável por delegar as tarefas efetuou uma distribuição cíclica das tarefas usando como parâmetros:

- Tempo de chegada de uma tarefa de análise: 10 segundos.
- Tempo de duração das medições: aproximadamente 10 minutos.
- Intervalo de dados: 2 horas.

Na segunda experimentação foi testada uma distribuição aleatória entre os agentes aptos a realizarem a tarefa. E como última abordagem, utilizou-se medidores para o fornecimento de informações de desempenho e capacidade dos recursos computacionais de forma a auxiliar a distribuição das tarefas. Estas informações são usadas para guiar o agente responsável pelo escalonamento, na decisão, a respeito de qual agente está mais apto a realizar uma determinada atividade de análise. Os medidores usados no teste foram as características e uso do processador, porém outros fatores poderiam ser utilizados.

Embora o tempo de resposta da última abordagem seja menor, o tempo usado pelo mecanismo de decisão, e pergunta aos agentes do atual uso de recursos não foi contabilizado nestes experimentos. Tanto a abordagem de distribuição aleatória, quanto à distribuição cíclica apresentam uma equalização da utilização dos recursos nas atividades de análise de dados de gerência. Porém, em ambas, o tempo médio de respostas tende a ser um pouco maior, pois as tarefas são escalonadas também para máquinas com poucos recursos computacionais.

Conclusão

Foi estabelecido um ambiente experimental no Laboratório de Gerência de Redes da Universidade Federal de Santa Catarina, composto por uma pequena rede local. Ali, a arquitetura completa proposta foi implementada e testada. Com o foco voltado ao *grid* de agentes processadores, pode-se perceber que este é o centro da arquitetura. É nesta etapa que os dados coletados são transformados em informações gerenciais, consolidando-os em relatórios de gerência a serem exibidos aos usuários. Os relatórios podem ser apresentados aos usuários apenas com informações referentes a certo intervalo de tempo, ou podem ainda considerar análises anteriores, apresentando uma consolidação dos dados.

Devido a complexidade do paradigma de *grid* computacional, foram desenvolvidas apenas algumas regras simples de processamento para a gerência de redes. Porém, os resultados indicam que a abordagem atinge com sucesso os objetivos a que se propõe. Novas regras podem ser acrescentadas para atingir cenários maiores e com outras necessidades.

Neste momento é tratada a questão de continuidade do trabalho. Para tanto, são sugeridas algumas melhorias que, com certeza, contribuem para o enriquecimento do trabalho:

- Desenvolvimento de novas regras de análise, que atinjam um maior número de variáveis e cenários reais de gerenciamento.
- Aplicar as regras em redes maiores, envolvendo múltiplos domínios administrativos e com testes mais apurados.
- Aplicação em redes sem fio.

Referências Bibliográficas

[AGENT] *Platform for Lightweight Agents*. Disponível em: <http://www.agentlight.org>.

[ASS03] - ASSUNÇÃO, M. D.; WESTPHALL, C. B.; KOCH, F. L. Arquitetura de *Grids* de Agentes Aplicada à Gerência de Redes de Computadores e Telecomunicações. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 21., 2003, Natal-RN. Proceedings... Natal: 21º Simpósio Brasileiro de Redes de Computadores, 2003, p. 789-804.

[ASS03A] ASSUNÇÃO, M. D.; WESTPHALL, C. B.; KOCH, F. L. Vantagens do uso de *grids* de agentes no gerenciamento de redes de computadores. Agent's Day – CBComp. 2003. p. 1358-1369

[ASS04] - ASSUNÇÃO, M. D. Implementação e Análise de uma Arquitetura de *Grids* de Agentes para a Gerência de Redes e Sistemas. CPGCC – UFSC. Florianópolis – SC, 2004. Dissertação de Mestrado.

[COABS] - COABS – CONTROL OF AGENT BASED SYSTEMS. Disponível em: <http://coabs.globalinfotek.com>. Acesso em: 13/03/2004.

[DISTR] - DISTRIBUTED.NET PROJECT, 2002. Disponível em: <http://www.distributed.net> Acesso em: 13/03/2004.

[FER03] - FERNANDES, A. M. da R. Inteligência Artificial – noções gerais. Visual Books. Pg. 85-113, Mar. 2003.

[FOS97] - FOSTER, I.; KESSELMAN, *C. Globus: A Metacomputing Infrastructure Toolkit, International Journal of Supercomputer Applications*, v.11, n.2, p.115-128, 1997.

[FOS99] - FOSTER, I, e KESSELMAN, C. *The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, Morgan-Kaufmann*, Orlando, FL, 1999.

[GUB02] - GUBERT, L. C. Utilizando o padrão de gerenciamento SNMP para gerenciar o tráfego *multicast*: a ferramenta *multicast* monitor. CPGCC – UFSC. Florianópolis – SC, 2002. Dissertação de Mestrado.

[JEN98] - JENNINGS, N. R.; SYCARA, K.; WOOLDRIDGE, M. *A Roadmap of Agent Research and Development. Academic Publishers*, Boston. Pg. 275-306, 1998.

[LEGIO] - LEGION. A Worldwide Virtual Computer, 1997. *Site* do projeto Legion. Disponível em: <http://legion.virginia.edu>. Acesso em: 13/03/2004.

[MUL97] - MÜLLER, J. P. *The Design of Autonomous Agents: A Layered Approach. Volume 1177 of Lecture Notes in Artificial Intelligence*. Springer-Verlag, Heidelberg, 1997.

[SETI] - SETI@HOME*. Search for Extraterrestrial Intelligence*, 1996. *Site* do projeto. Disponível em: <http://setiathome.ssl.berkeley.edu>. Acesso em: 13/03/2004.

[SIL03] - SILVA, A. R. V. Utilização de Agentes para Definição e Alteração de Bancos de Dados Heterogêneos. CPGCC – UFSC. Florianópolis – SC, 2003. Dissertação de Mestrado.

[STA99] - STALLINGS, W. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2. Reading, Massachusetts: Addison Wesley Longman, Inc, 1999. ISBN: 0-201-48534-6.

[SYP03] - SYPERREK, B.; ASSUNÇÃO, M. D.; WESTPHALL, C. B. Grid de Agentes: Os Padrões FIPA na Construção dos Serviços do Grid para Garantia de Interoperabilidade. 2003.

[YEP03] - YEPES, Igor. Sistemas Multi-Agentes. Projeto ISIS. Disponível em: <http://www.geocities.com/igoryepes/agentes.htm>. Acesso em: 28/08/2003