Arquitetura de *Grids* de Agentes Aplicada à Gerência de Redes de Computadores e Telecomunicações

Marcos Dias de Assunção[1], Carlos Becker Westphall[1], Fernando Luiz Koch[2]

[1]Laboratório de Redes e Gerência; Universidade Federal de Santa Catarina
Centro Tecnológico; C. P. 476, CEP: 88049-970, Florianópolis – SC – Brasil

[2]Intelligent Systems Group - Institute of Information and Computing Sciences
Utrecht University – The Netherlands

{assuncao, westphal}@lrg.ufsc.br, fkoch@acm.org

Abstract. *The centralized approach for computer and telecommunication network management can take us into situations where a large amount of data need to be handled and analyzed by only one administration station in the network. In this situation this approach is no longer cost effective and, moreover, it doesn't scale with the number of devices in the managed site. In this work we propose an alternative by creating a highly distribute computing environment through the use of grids of autonomous agent to analyze large amounts of data, which reduce the processing costs by optimizing the load distribution and resources utilization.*

Resumo. *O gerenciamento centralizado de redes de computadores e telecomunicações pode levar-nos a situações em que grandes volumes de dados precisam ser manipulados e analisados por uma única estação de gerenciamento na rede. Além do alto custo, tal abordagem tem se apresentado pouco extensível a medida que cresce o número de dispositivos no ambiente gerenciado. Neste trabalho propomos uma alternativa através da criação de um ambiente computacional altamente distribuído com a utilização de grids de agentes autônomos para a análise de grandes volumes de informação, os quais reduzem o custo de processamento através da otimização da distribuição da carga e da utilização de recursos.*

1. Introdução

A computação em *grid* tem surgido como uma iniciativa que possibilita a agregação de recursos conectados em rede, formando um sistema distribuído em larga escala e possibilitando a resolução de problemas científicos e comerciais complexos [Foster e Kesselman 1999]. Distribuindo a carga de trabalho de suas aplicações, um usuário pode dispor de uma capacidade computacional e de armazenamento que se tornariam financeiramente inviáveis de se atingir em um ambiente tradicional.

Este compartilhamento e agregação de recursos se diferenciam das tecnologias atualmente disponíveis na Internet, pela forma como esta integração ocorre. O objetivo é proporcionar um acesso mais barato, eficiente, fácil, abrangente e em larga escala. As diferenças fundamentais entre um *grid* e um sistema distribuído tradicional estão na

grande heterogeneidade de recursos, na dinamicidade do ambiente e na alta latência das redes que os interligam. Um *grid* é uma forma de agregação que pode acontecer de várias formas, inclusive através dos mecanismos de cooperação e negociação proporcionados pelos agentes.

No fluxo de trabalho tradicional do gerenciamento de redes temos 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]. Sistemas baseados em regras de produção e inferência podem ser utilizados para analisar estes dados, extrair as informações necessárias e identificar possíveis problemas. Porém, a consolidação destes dados em informações de gerência é uma tarefa intensiva e consome um grande poder de processamento. À medida que o ambiente cresce, a eficiência de um sistema centralizado diminui e aumenta o custo com hardware.

Este trabalho apresenta uma arquitetura alternativa, baseada em *grids* de agentes para distribuir a carga destas tarefas intensivas do gerenciamento. Apresentamos os resultados de nossos estudos com *grids* e sua possível aplicação na gerência de redes.

O trabalho está dividido em cinco seções: na seção 2 apresentamos a computação em *grid, grids* de agentes e sua utilização na gerência de redes de computadores; na seção 3 apresentamos nossa proposta para uma arquitetura de *grid* de agentes voltada à gerência de redes de computadores; já na seção 4 comparamos nossa arquitetura com outras abordagens; finalmente na seção 5 apresentamos nossas conclusões e os trabalhos futuros a serem realizados nesta linha de pesquisa.

2. Computação em *Grid* e *Grids* de Agentes

Na literatura computacional, um *grid* tem aparecido como uma infraestrutura de hardware ou software capaz de agrupar componentes, proporcionando novas funcionalidades a partir de componentes existentes [Rana e Moreau 2000].

Em [Jeffery 2000], é apresentada uma definição com três camadas, sendo elas:

1. *Grid* computacional: nível mais baixo, preocupado com a reunião em larga escala de recursos computacionais e de armazenamento. Esta reunião de recursos pode ser utilizada com o objetivo de criar um poder de processamento equivalente ou superior ao de um supercomputador e utilizá-lo no processamento de grandes volumes de dados. Os recursos neste caso podem não ser apenas poder de processamento, mas bases de dados e outros. Um *grid* computacional é definido com detalhes em [Foster e Kesselman 1999] e [Buyya 2002].
2. *Grid* de informação: camada intermediária, permitindo o acesso uniforme as diferentes fontes de informação e tornando possível que os serviços possam ser executados em recursos computacionais distribuídos. Esta camada procura localizar, integrar e gerenciar as diferentes fontes de informação.
3. *Grid* de conhecimento: é a camada mais alta do modelo e proporciona serviços especializados, os quais podem procurar por padrões nos repositórios de dados existentes e gerenciar os serviços de informação. O *grid* de conhecimento pode

auxiliar na tomada de decisões e na interpretação das informações manipuladas no grid de informação.

Problemas a serem investigados nesta área de pesquisa envolvem a forma como estes recursos e grandes volumes de dados são tratados e manipulados. O acesso uniforme é outro desafio. As tecnologias que podem ser consideradas relevantes nesta área, são oriundas de bancos de dados e sistemas operacionais distribuídos, programação paralela e distribuída e gerência de redes. Alguns exemplos de infraestruturas são o Jini [Edwards 1999], o Globus [Foster e Kesselman 1997] e o Legion [Legion].

A tecnologia de agentes móveis pode ser utilizada na camada de *grid* computacional para proporcionar a migração de serviços no *grid.* Os sistemas multi-agentes, através dos conceitos de cooperação e de negociação também podem auxiliar no balanceamento de carga. A alocação de recursos em um *grid* pode ser visto como um problema de cooperação entre vários nós, os quais podem aceitar a alocação dos recursos sob certas condições.

No caso dos agentes, o conceito de *grid* é aplicado com a visão chave de que suas características e os fatores a serem atingidos farão com que os sistemas multi-agentes sejam mais interoperáveis e possam rodar nos mais variados tipos de recursos. Estes conceitos tendem a tornar a aplicação mais escalar e adaptativa. Um *grid* de agentes pode ser visto também como um framework que combina as facilidades de um *grid* computacional e uma arquitetura baseada em agentes. No caso do coABS [COABS], temos um cenário onde o *grid* de agentes aproveita as facilidades proporcionadas por um *grid* computacional para proporcionar seus serviços aos níveis superiores.

Fica evidente que os conceitos envolvidos nas tecnologias de agentes e sistemas multi-agentes podem ser utilizados na construção de *grids*, e como uma forma de tornar a computação em *grid* possível. É possível proporcionar a integração de recursos e o compartilhamento em larga escala. Podemos enfrentar a dinamicidade do ambiente no *grid*, através da entrada e saída de agentes ou comunidades no sistema, e é possível ter agentes em execução nos mais variados tipos de recursos. Além destas facilidades, são toleráveis a latência que possa existir na comunicação em sistemas deste porte.

2.1. Estado da Arte

A computação em *grid* tem mostrado resultados em trabalhos científicos que requerem computação intensiva e um poder de computação que não pode ser encontrado em um ambiente tradicional. Estas aplicações, que em sua maioria, podem ser decompostas em atividades menores, não são sensíveis a alta latência que possa existir na comunicação entre os elementos do *grid*, e suas sub-tarefas não requerem a passagem de um grande número de mensagens de sincronização.

Como exemplos deste tipo de aplicações temos a análise de sinais de rádio na busca de vida extraterrestre inteligente Seti@Home [Korpela *et al.* 2001], o superprocessador virtual proporcionado pela *United Devices* [United Devices] e o *Distributed.Net* [Distributed.Net].

Existem trabalhos na tentativa de proporcionar *middlewares* que facilitem a computação em *grid*, proporcionando gerenciamento de recursos e APIs para

desenvolvimento de aplicações. Alguns trabalhos pioneiros são o *Legion* [Legion] e o *Globus* [Foster e Kesselman 1997]. O *Globus* é um *middleware* que proporciona uma pilha de 4 camadas para controle de *hardware*, comunicação, compartilhamento de recursos e coordenação de tarefas. Outra tecnologia importante é o *JINI* [Edwards 1999].

Quando encontramos o termo *grid* de agentes relacionado em algum trabalho, percebemos 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. Em [Rana e Walker 2000] a tecnologia de agentes é utilizada para generalizar e interligar ambientes de resolução de problemas (PSE).

Dentro do conceito de *grid* de informação, temos o projeto *ABIS* [Manola 1999], que descreve um conjunto de tecnologias necessárias à aquisição, tratamento, armazenamento, acesso e envio de informações necessárias as tropas militares e serviços de defesa dos Estados Unidos na execução de suas tarefas.

O *DARPA's Control of Agent-Based Systems* (CoABS) [Manola 1999] visa criar *grids* de agentes através da integração de sistemas heterogêneos baseados em agentes, sistemas de agentes móveis e sistemas baseados em objetos. Seu contexto é amplo e muita documentação tem sido gerada neste projeto.

Em [Wijngaards *et al.* 2002] é apresentado o *AgentScape* com o objetivo de facilitar a construção de sistemas baseados em agentes atendendo as características de larga escala. São apresentadas as dificuldades de se utilizar as tecnologias tradicionais de agentes em um cenário de larga escala. Apresenta uma plataforma para criação de um grande número de agentes, suporte a múltiplos tipos de código e sistemas operacionais e interoperabilidade entre diferentes plataformas. O *AgentScape* é um *middleware* que proporciona a interoperabilidade entre diferentes plataformas de agentes.

No nosso trabalho utilizamos conceitos dos trabalhos apresentados acima na elaboração de uma arquitetura própria para a construção de *grids* de agentes voltados à gerência de redes de computadores, baseados em pequenos agentes utilizando o *AgentLight* [AgentLight] e seguindo ao máximo as especificações da *FIPA* [FIPA SC00001L] em sua concepção.

2.2. *Grids* de Agentes na Gerência de Redes

À medida que as redes de computadores e telecomunicações crescem, cresce também a complexidade no seu gerenciamento e o fluxo de informações no sistema de gerência. Conforme descrito em [Koch e Westphall 2001], um sistema de gerenciamento de redes tradicional apresenta um fluxo de trabalho semelhante ao da figura 1. Neste modelo, primeiramente, os dados são coletados dos dispositivos gerenciados da rede utilizando um protocolo de gerenciamento. Depois as informações são analisadas e finalmente são condensadas, dando origem às informações de gerenciamento reais. Mais tarde, com base nestas informações, os gerentes da rede podem tomar decisões e corrigir pontos falhos no sistema.

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.

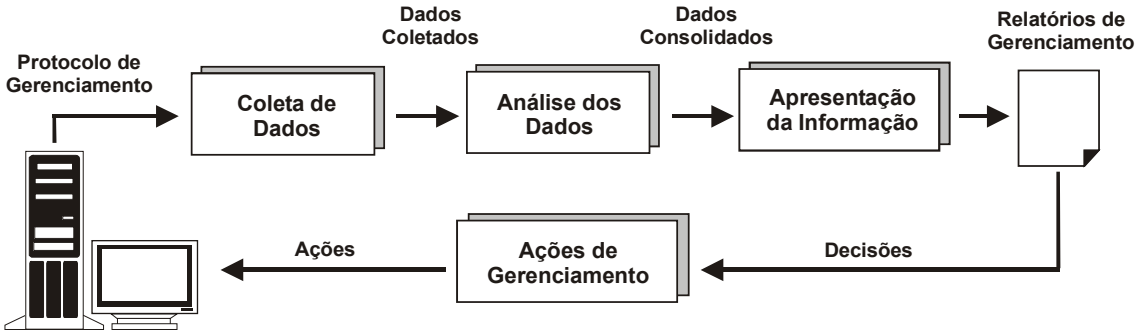

Figura 1. Fluxo do gerenciamento de redes

Muitas propostas baseadas em conceitos da Inteligência Artificial foram apresentadas com o intuito de aumentar a eficiência da gerência de redes. Alguns sistemas têm seu funcionamento baseado em regras de produção e inferência. Embora auxiliem no gerenciamento, estas aplicações também crescem no sentido de comportar o maior número possível de regras para identificar e tratar problemas, aumentando sua complexidade e diminuindo seu desempenho. Outro fator negativo neste cenário é a inexistência de paralelismo na análise. Esta situação pode ser melhorada se tivermos um grande número de agentes, possivelmente localizados em estações diferentes, analisando informações simultaneamente.

Com o crescimento da rede, a arquitetura centralizada apresenta sérios problemas de escalabilidade e um conseqüente aumento na demanda por poder de processamento, acarretando uma grande elevação no custo de *hardware* das estações de gerenciamento. Em tal modelo a estação de gerenciamento realiza o papel de coletora de informações. Além das tarefas de processamento e análise, temos um consumo de processamento e de memória utilizados no carregamento de bibliotecas para uso dos protocolos de gerência e com atividades de coleta de informações.

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, encontramos um cenário onde *grids* de agentes podem ser aplicados. Podemos ter uma grande quantidade de regras de análise que conseguimos 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*. Desta forma, podemos ter ganhos significativos e uma redução de *hardware* considerável.

Os agentes do *grid* podem aprender novas regras. Com isso, aumentamos o conhecimento acerca do gerenciamento. Vários agentes aprendendo novas regras podem proporcionar uma maior sinergia no sistema.

3. Arquitetura do *Grid* de Agentes Para a Gerência

Nesta seção apresentamos a nossa arquitetura de *grids* de agentes para a gerência de redes de computadores. Levando em consideração o fluxo de trabalho tradicional do

gerenciamento, apresentado na segunda seção, identificamos algumas atividades distintas: coleta de dados dos dispositivos da rede, a classificação destes dados, análise dos dados a fim de transformá-los em informações de gerência e apresentação das informações. Em uma primeira visão da arquitetura, apresentamos os *grids* de agentes que compõem o *grid* de gerência e em seguida apresentamos detalhes sobre cada ponto da arquitetura. A figura 2 mostra um esboço desta arquitetura.

3.1. *Grid* de Agentes Coletores (GC)

A responsabilidade dos agentes do *grid* coletor é coletar informações dos equipamentos da rede gerenciada usando um protocolo de gerenciamento ou outro meio. Chamaremos de "interface" esta habilidade dos agentes coletores de utilizar um protocolo ou outro. Um agente coletor pode ter uma interface *SNMP (Simple Network Management Protocol)* ou utilizar um utilitário de linha de comando, por exemplo.

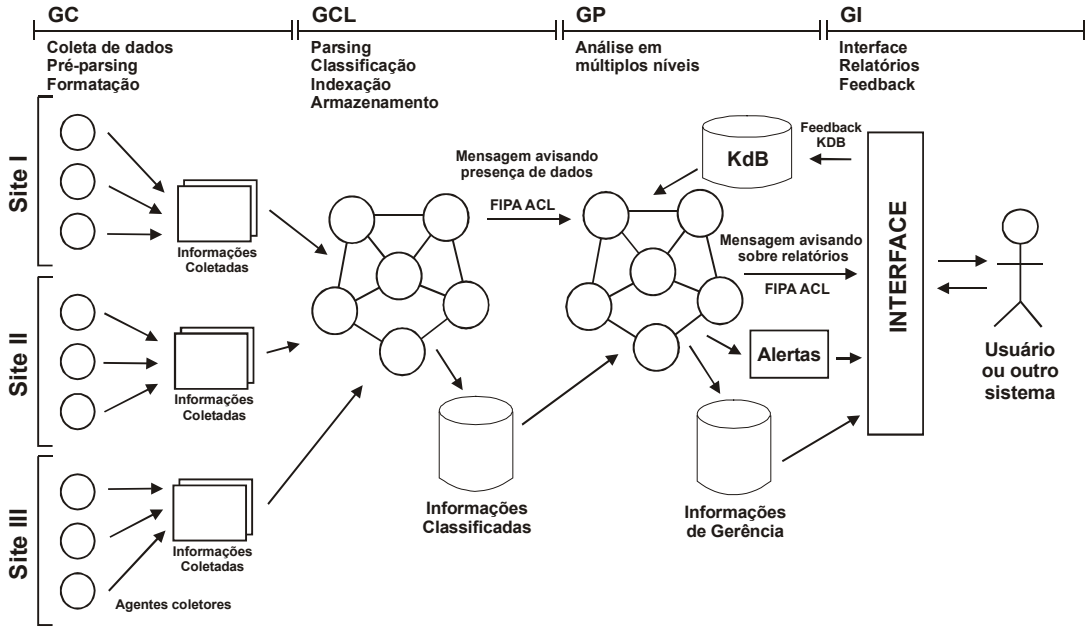

Figura 2. Arquitetura geral do sistema

As informações extraídas podem apresentar formatos bastante heterogêneos e por isso é necessário criar uma representação comum para estes dados. Esta representação pode ser feita usando XML [XML], em conjunto com ontologias [Gruber 1993], [FIPA SC00001L], [Thompson 1999]. Neste caso, garantimos que a representação das informações coletadas da rede poderá ser interpretada corretamente pelo *grid* de agentes que irá recebê-las.

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

As informações coletadas pelo *grid* coletor são enviadas para um *grid* classificador, já num formato padrão, através de um meio de comunicação qualquer – por exemplo, via SMTP ou HTTP.

3.2. *Grid* de Agentes Classificadores (GCL)

Em uma rede relativamente grande, podemos ter inúmeros coletores de dados. Por conseqüência teremos dados coletados de vários tipos de equipamentos chegando. O *grid* de classificação é responsável por classificar estas informações e armazená-las de uma forma estruturada e mais fácil de ser recuperada. Um arquivo de dados enviado por um *grid* coletor pode conter valores coletados de vários objetos gerenciados de equipamentos heterogêneos. É interessante organizar estas informações antes de realizar qualquer análise. Desta forma, o *grid* classificador desempenha as tarefas de *parsing*, classificação, indexação e armazenamento destes dados.

Na figura 3 temos um exemplo de um *grid* classificador recebendo dados coletados dos objetos gerenciados de três equipamentos diferentes. Os agentes do *grid* classificador possuem objetivos que consistem em classificar estas informações e armazená-las da forma correta, separando-as de uma forma que facilite a distribuição e análise (data-clustering). Depois de classificar estes dados, uma mensagem FIPA ACL [FIPA SC00061G] é enviada ao *grid* de processamento.

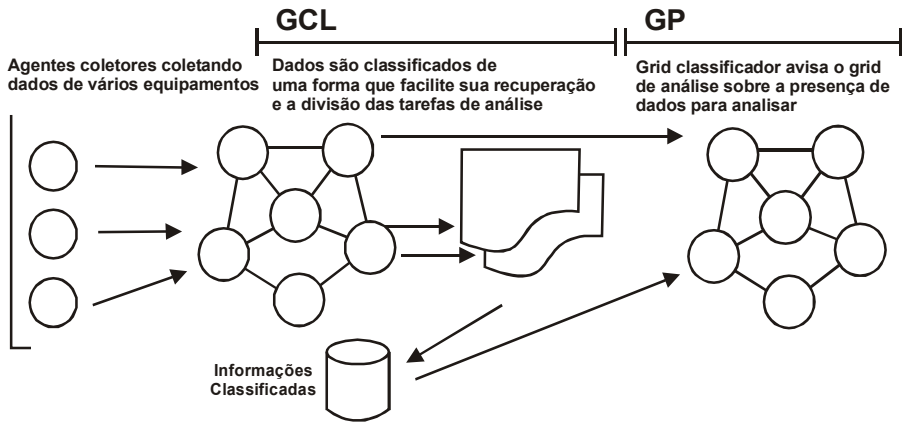

Figura 3. Classificação realizada pelo *grid* classificador

3.3. *Grid* de Agentes Processadores (GP)

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 darão origem aos relatórios de gerência e de alertas enviados ao *grid* de interface.

Este *grid* é composto por *containers* [Müller 1997] de agentes distribuídos em vários computadores. Temos desta forma um ambiente dinâmico em que novos recursos podem ser adicionados ou removidos do *grid*. Se houver a necessidade de aumentar o poder de processamento deste *grid*, podemos adicionar novos recursos ao mesmo.

O *grid* de processamento pode realizar uma série de análises em múltiplos níveis. Exemplos destas análises e como elas ocorrem são descritas a seguir:

1. Ao receber uma mensagem do *grid* classificador, indicando a presença de dados, o *grid* processador faz com que um número de agentes ou *containers*, baseados em suas regras, procurem por problemas nestes dados. Esta primeira análise não leva em consideração informações armazenadas anteriormente na base de dados e não procura por nenhuma relação entre os fatos.
2. Os agentes deste *grid* podem consolidar os dados, extraindo informações armazenadas anteriormente. Com isso podem procurar por problemas e agrupar dados, constituindo algumas informações que servirão para a construção de relatórios de gerência.
3. No terceiro nível de análise, os agentes do *grid* podem procurar por relações entre os fatos na base de dados. Tendo uma visão de mais alto nível das informações. Podem desta forma, identificar problemas que são possíveis de serem identificados através do cruzamento de informações de um conjunto de equipamentos e não apenas de dados isolados.

Dentro deste contexto, 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 e a raiz do *grid* de processamento coordena esta distribuição, funcionando como uma espécie de *broker*, no sistema.

A raiz do *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 *containers* pertencentes ao *grid.* Desta forma, é necessário que exista um registro dos *containers* que compõem este *grid* de processamento e suas habilidades, para que seja possível esta distribuição. Na figura 4 podemos ter uma visão abstrata da divisão das atividades de análise.

Como foi citado, podemos adicionar um novo *container* ao *grid* de processamento, com regras para identificar alguns problemas. Quando o *container* é adicionado, ele anuncia seus serviços ao *grid* e passa a analisar as informações que lhe são designadas.

3.4. *Grid* de Agentes de Interface (GI)

O *grid* de agentes de interface é o canal de comunicação entre o *grid* e o gerente da rede. É também uma forma de receber o *feedback* do usuário e fornecê-lo ao sistema. A interface é tanto um canal de saída quanto de entrada para o sistema. Através de uma interface o usuário pode receber as informações processadas pelo *grid* de processamento e as mensagens de alerta. É possível programar novos relatórios e interagir com a base de conhecimento, definindo novas regras e objetivos e alterando objetivos existentes.

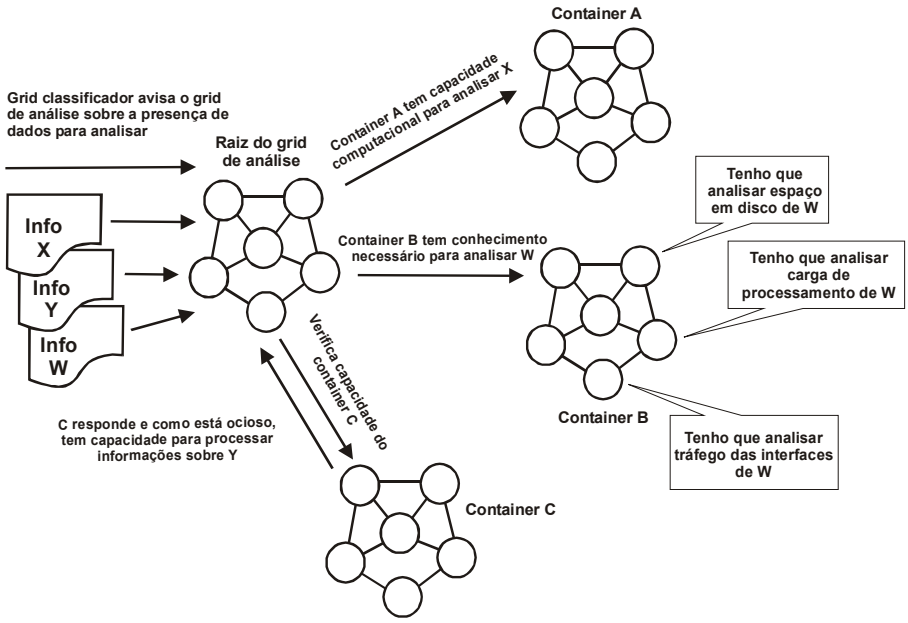

Figura 4. Exemplo da divisão de tarefas de análise no *grid*

Esta interface deve ser abrangente, flexível e multi-protocolo, permitindo várias formas de interação conforme a necessidade do usuário – por exemplo, páginas HTML, *e-mail*, *chat*, XML/HTTP – e o retorno do usuário também pode se dar sob estas formas. Esta interface, além de ser o canal com o usuário, pode servir de alimentação para um outro sistema, ou mesmo ser realimentada por este sistema, do qual pode aprender novas regras de gerência e transmiti-las ao *grid*. O *grid* de interface pode aprender novas regras através da análise das preferências e necessidades do usuário, customizando o sistema.

3.5. O Endereçamento no *Grid* de Agentes

Um problema que deve ser resolvido em um *grid* de agentes é o endereçamento dos componentes do *grid*. Adotamos uma forma de endereçamento semelhante ao sistema de resolução de nomes no qual o endereçamento no *grid* é feito usando-se o conceito de domínios e subdomínios. Por exemplo, podemos ter um *grid* de agentes cujo domínio ou identificador do *grid* se dá através de "*agentes.com*". Este é o identificador do nosso *grid* de agentes. Quando queremos nos referir a ele, usamos este domínio. Os *containers* criados dentro deste *grid* são identificados por um nome semelhante a um subdomínio de "*agentes.com*". Desta forma, um *container* pertencente a este *grid* teria que ter um nome semelhante a "*container1.agentes.com*". Na figura 5 temos um exemplo desta hierarquia.

Este nome utilizado pelos agentes não representa necessariamente o nome no servidor de nomes, do recurso onde o *container* está rodando. É apenas uma forma de endereçamento interno utilizado pelo *grid*. O envio de mensagens e arquivos para o *grid* ou para os *containers* ou os agentes que o compõem, é feito através de um endereço em nível de protocolo de aplicação, como SMTP ou HTTP, conforme descrito em [FIPA SC00001L].

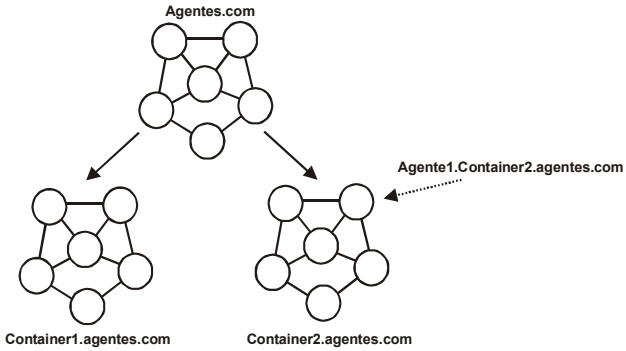

Figura 5. Hierarquia do *grid* de agentes

3.6. Balanceamento de Carga no *Grid*

Uma das vantagens da utilização de *grids* de agentes na análise dos dados coletados da rede é a distribuição inteligente da carga de trabalho para a análise no *grid*. Ao receber um volume de dados, o *grid* pode realizar a distribuição entre os *containers*, baseado nos seguintes princípios:

1. distribuí-los entre os *containers* com conhecimento para processá-los;
2. recursos que têm capacidade computacional para processá-los; e
3. utilizar recursos que estejam ociosos.

Para que isso seja possível, primeiramente o *grid* deve conhecer os recursos que o compõem em um determinado momento e suas capacidades. Estas informações são armazenadas na raiz do *grid* usando um serviço de diretório qualquer. Quando um novo *container* é adicionado ao *grid*, ele pode se registrar e informar a capacidade do recurso onde está rodando. A ontologia utilizada ao informar este perfil é definida pela FIPA em [Fipa PC00091B]. Esta interação pode ser vista na figura 6, onde o *container* 1 é adicionado ao *grid*, informa o perfil do recurso em que se encontra e os serviços que ele é capaz de proporcionar. A raiz registra estas informações no diretório D1. Quando é necessário atribuir alguma atividade, com o intuito de distribuir a carga de trabalho, a raiz usa estas informações para selecionar os recursos capazes de processá-las.

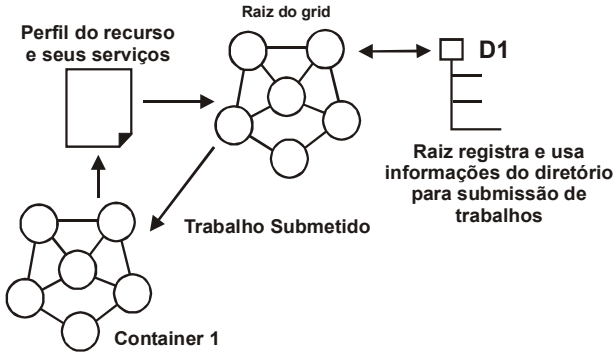

Figura 6. *Container* adicionado ao *grid* informa o perfil do recurso onde está

Quando o *grid* de processamento recebe um aviso de que existem dados para serem processados, além destas informações, pode solicitar que seja informado o perfil atual dos recursos ou negociar com os *containers* a possibilidade de enviar as

informações para serem processadas por eles. Para isso ele pode usar os protocolos de negociação estabelecidos pela FIPA [Fipa SC00029H], [Fipa SC00027H].

Da mesma forma, a consolidação de dados e a busca por relações entre os fatos podem ser distribuídas entre os vários *containers* do *grid*. O *grid* de análise pode selecionar os recursos capazes de desempenharem estas tarefas e delegar as tarefas de processamento. Um exemplo desta interação pode ser visto na figura 7. A raiz, com base nas informações do diretório, seleciona os *containers* capazes.

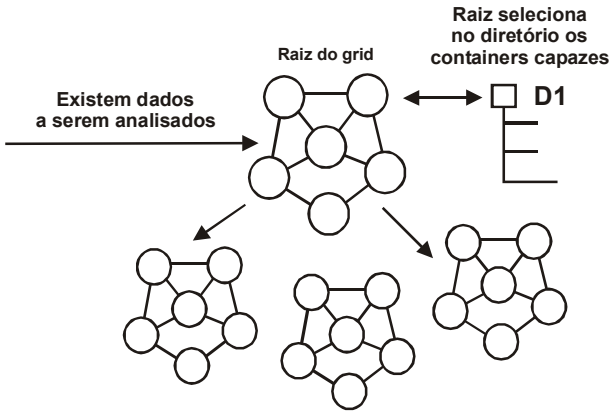

Figura 7. Raiz do *grid* de análise seleciona *containers* capazes

3.7. Especificação do *Grid* de Agentes

A especificação de um *grid* com centenas ou milhares de agentes de inferência pode se tornar uma tarefa extremamente complexa quando não possuímos uma metodologia bem definida e flexível para defini-los. Alguns dos fatores que precisam ser definidos são: que conhecimento os agentes possuirão inicialmente, como estarão distribuídos, qual é o conhecimento das bases de conhecimento globais e os objetivos a serem atingidos pelos agentes do *grid*. A ênfase é a adição de novos *containers* ao *grid* ou mesmo de conhecimento a agentes e *containers* já existentes.

É possível especificar o estado inicial dos *grids* de agentes usando documentos XML. Uma vez que já apresentamos como nosso *grid* de agentes está organizado, podemos descrever os elementos dos documentos XML que conterão as especificações do *grid*:

1. *agentgrid*: este elemento representa o *grid* de agentes. Tem um elemento filho que representa o domínio que identifica o *grid*;
2. *group*: este elemento representa os *containers* de agentes que integram o *grid*. Um *container* é constituído de um nome que chamamos de subdomínio, de uma definição da base de conhecimento global dos agentes do *container* e da definição dos agentes que constituem o *container*. Existe ainda um elemento que representa a linguagem em que as cláusulas da base de conhecimento global e local estão expressas;
3. *agent*: este elemento representa um tipo de agente que pertence ao *container*. Este agente é composto de um elemento que representa a quantidade de agentes a ser criada no *container* e de uma base de conhecimento local de cada agente. Este elemento possui ainda um elemento filho que contém o prefixo do nome

dos agentes. Neste caso, o nome que identificará o agente no sistema, será composto pelo prefixo e um número que é incrementado de 1 até a quantidade de agentes especificada pelo elemento *quantity*;

4. *localKDB*: representa a base de conhecimento do agente e é constituída de várias cláusulas;
5. *globalKDB*: representa a base de conhecimento global dos agentes de um *container*; e
6. *goal*: representa os objetivos do agente.

Um exemplo de um documento XML contendo a definição inicial de um *grid* de agentes pode ser visto na figura 8. Neste exemplo, definimos um *grid* de agentes composto por dois grupos (*containers*), com 20 agentes cada um. No momento de criação do *grid*, este documento XML é carregado por uma aplicação que é responsável por criar o *grid* a partir dele. Podemos criar documentos para definir *containers* ou apenas cláusulas que devem ser adicionadas na base de conhecimento de um *container* ou de um agente.

```
<?xml version="1.0" encoding="UTF-8"?>
<agentgrid>
  <domain>lrg.ufsc.br</domain>
  <group>
    <subdomain>grupo1.lrg.ufsc.br</subdomain>
    <language>prolog</language>
    <globalKDB>
      <clause>prolog-like.</clause>
    </globalKDB>
    <agent>
      <quantity>20</quantity>
      <prefix>agentesx</prefix>
      <localKDB>
        <clause>prolog-like.</clause>
      </localKDB>
      <goal>
        <clause>prolog-like.</clause>
      </goal>
    </agent>
  </group>
  <group>
    <subdomain>grupo2.lrg.ufsc.br</subdomain>
    <language>prolog</language>
    <globalKDB>
      <clause>prolog-like.</clause>
    </globalKDB>
    <agent>
      <quantity>20</quantity>
      <prefix>agentesy</prefix>
        <localKDB>
          <clause>prolog-like.</clause>
        </localKDB>
        <goal>
          <clause>prolog-like.</clause>
        </goal>
     </agent>
  </group>
```

Figura 8. Exemplo de documento XML especificando um *grid* de agentes

4. Vantagens da Arquitetura

Podemos destacar algumas vantagens da arquitetura utilizando *grids* de agentes em relação a arquitetura sem *grids*. Na gerência sem os *grids* de agentes temos um cenário semelhante ao apresentado na figura 9.

Neste modelo, usando agentes, temos coletores de dados extraindo dados de vários equipamentos, um classificador armazenando estes dados em um repositório e

um agente analisador analisando estas informações. Temos ainda uma interface com o usuário proporcionada pelos agentes de interface.

Cada rede possui uma estrutura semelhante e não existe nenhuma relação entre os diferentes *sites*. Não existe uma integração entre estas informações, nenhuma análise de alto nível ou cruzamento de dados pode ser realizada. Além disso, não existe também nenhuma forma de distribuição de carga de trabalho. A única evolução possível deste sistema seria a integração das bases de conhecimento.

Outra desvantagem é a falta de extensibilidade. A única forma de aumentar o poder de análise das informações é aumentar o poder do *hardware* que comporta o sistema. Temos um acréscimo caro e desnecessário e podemos somar a isto o fato destes recursos poderem estar alocados e não sendo utilizados mesmo quando não existirem informações para serem analisadas.

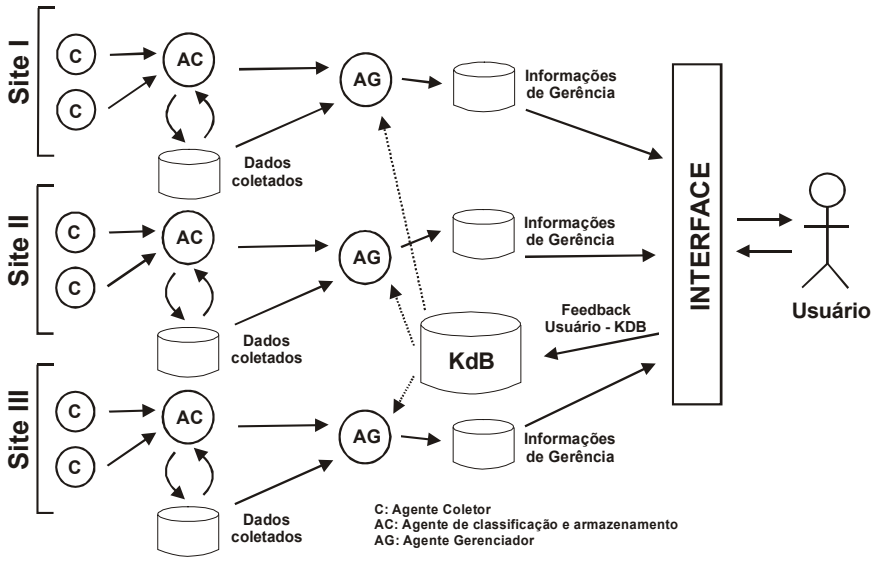

Figura 9. Arquitetura sem *grids* de agentes

Na arquitetura que estamos propondo teríamos uma estrutura mais flexível e extensível, conforme descrito na figura 2.

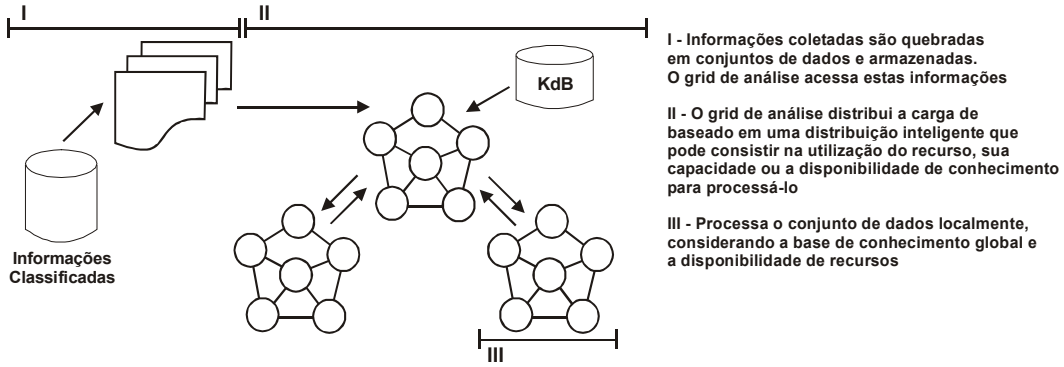

Figura 10. Fluxo das informações no *grid* de análise

Com *grids* de agentes, temos um fluxo único que os dados devem seguir. As principais vantagens desta arquitetura estão relacionadas com distribuição da carga de processamento e a utilização de recursos ociosos da rede. Conforme descrito na figura 10. Podemos ter ainda uma série de vantagens:

- Extensibilidade – A arquitetura pode ser expandida em qualquer um de seus níveis. Novos *containers* ou agentes podem ser adicionados, assim como novos objetivos podem ser atribuídos aos agentes já existentes. No *grid* de processamento, podem ser adicionados novos *containers* em equipamentos diferentes, de acordo com a necessidade de processamento e análise que se deseja atingir. Um *container* com vários agentes pode ser agregado ao *grid* para efetuar um tipo de análise mais específica ou na identificação de um tipo particular de problemas.
- Redução no tempo de análise – o tempo de análise dos dados pode ser reduzido, pois teremos vários *containers* analisando dados paralelamente. Desta forma um número maior de regras e de problemas pode ser verificado em um período de tempo menor. Podemos afirmar isso com base na distribuição das atividades e a possibilidade de agregação de recursos ao *grid*. Em uma aplicação tradicional de gerência, a única forma de agilizar o processamento destas informações seria aumentando a capacidade do *hardware* da estação de gerenciamento.
- Número maior de regras de análise – o sistema pode comportar um número maior de regras. Os agentes podem realizar análises em múltiplos níveis, realizando cruzamento de dados e com um número de regras que não seria passível de ser mantido e verificado sob os meios tradicionais.
- Balanceamento inteligente de carga – outra vantagem da utilização de agentes inteligentes é que eles possuem o conhecimento sobre como distribuir a carga de processamento, através do descobrimento de recursos disponíveis e dos princípios de colaboração e negociação.
- Poder de processamento maior e melhor distribuído – temos um poder de processamento maior, resultante da agregação de recursos ao *grid*. Temos também uma utilização mais barata e efetiva dos recursos, evitando que alguns estejam ociosos enquanto outros estejam sobrecarregados.
- Compartilhamento de conhecimento – em um sistema onde é realizado o gerenciamento de várias redes, existe a vantagem do compartilhamento de conhecimento.

Apesar das vantagens, *podemos perceber que a utilização de grids de agentes se mostra atraente quando a rede e o volume de informação são relativamente grandes*. Em ambientes menores, as abordagens tradicionais ou baseadas em sistemas multi-agentes ainda apresentam uma relação de custo mais efetiva.

Existe também um ponto de corte onde a utilização de *grids* de agentes deixa de apresentar vantagens. Como citado anteriormente, a divisão de dados e das atividades de análise pode acarretar uma perda de significado das informações.

5. Conclusões e Trabalhos Futuros

Neste trabalho apresentamos uma arquitetura de *grids* de agentes aplicada à gerência de redes de computadores. Também mostramos os conceitos envolvidos com a computação em *grid* e as vantagens de sua utilização no processamento e análise de dados de gerência.

Grids de agentes é um assunto relativamente novo e sua utilização na gerência de redes deverá ser cada vez mais explorada, sendo este um trabalho original e inédito na área. Nós destacamos as possibilidades de sua utilização e propomos uma arquitetura de *Grids* de Agentes para Gerência de Redes, validando a mesma teoricamente e via implementação de um protótipo que está em desenvolvimento. Aperfeiçoaremos as atividades até aqui realizadas, dando continuidade aos seguintes trabalhos:

1. Desenvolvimento de melhores protótipos, demonstrando as vantagens da utilização de *grids* de agentes através da realização de mais medições.
2. Evidenciar melhor o ponto em que a utilização de um *grid* de agentes passa a ser mais vantajosa e em que ponto deixa de ser.
3. Realizar estudos sobre o balanceamento de carga no *grid* de processamento. Também procurar meios eficientes de divisão das tarefas de análise. Executar mais medições da capacidade de processamento atingida com um *grid* de processamento e suas vantagens em relação as técnicas tradicionais.
4. Investigar melhor a utilização de agentes móveis na análise de dados e no balanceamento de carga. A mobilidade dos agentes proporciona uma migração das atividades de análise atribuídas a eles, melhorando a utilização dos recursos.
5. Melhorar a eficácia das formas de armazenamento, replicação, indexação e recuperação dos dados de gerência pelos agentes do *grid*.

Mostramos as vantagens de se aplicar *grids* de agentes na gerência de redes, reduzindo custos e proporcionando um processamento e análise mais eficiente através da arquitetura desenvolvida. Ainda existe muito a ser feito e acreditamos que a continuidade e o aperfeiçoamento deste trabalho merecem atenção especial de toda comunidade científica que atua na área, por se tratar de uma abordagem nova e promissora.

Referências Bibliográficas

AgentLight, Platform for Lightweight Agents, http://www.agentlight.org.

Buyya, R. (2002) “Economic-based Distributed Resource Management and Scheduling for Grid Computing”, PhD Thesis, School of Computer Science and Software Engineering Monash University, Melbourne, Australia.

CoABS DARPA Project, “Control of Agent Based Systems”, http://coabs.globalinfotek.com.

Distributed.Net Project, http://www.distributed.net/index.html.en.

Edwards, W. K, Core Jini, Addison Wesley, 1999.

FIPA SC00001L, “FIPA Abstract Architecture Specification”, http://www.fipa.org/specs/fipa00001/SC00001L.html, 2002.

FIPA SC00061G, “FIPA ACL Message Structure Specification”, http://www.fipa.org/specs/fipa00061/SC00061G.html, 2002.

FIPA PC00091B, “FIPA Device Ontology Specification”, http://www.fipa.org/specs/fipa00091/PC00091B.html, 2001.

FIPA SC00029H, “FIPA Contract Net Interaction Protocol Specification”, http://www.fipa.org/specs/fipa00029/SC00029H.html, 2002.

FIPA SC00027H, “FIPA Query Interaction Protocol Specification”, http://www.fipa.org/specs/fipa00027/SC00027H.html, 2002.

Foster, I. e Kesselman, C. (1997) “Globus: A Metacomputing Infrastructure Toolkit”, Intl J. Supercomputer Applications, 11(2):115-128.

Foster, I, e Kesselman, C. (1999) “The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann”, Morgan-Kaufmann, Orlando, FL.

Gruber, T. R., (1993) “Toward principles for design of ontologies used for knowledge sharing”, Technical Report KSL-93-94, Knowledge Systems Laboratory, Stanford University, August.

Jeffery, Keith G. (2000) “Knowledge, Information and Data”, A briefing to the Office of Science and Technology, UK. www.itd.clrc.ac.uk/Publications/1433/KnowledgeInformationData20000124.htm

Koch, F. L. e Westphall, C. B., (2001) "Decentralized Network Management using Distributed Artificial Intelligence" Journal of Network and Systems Management. Plenum Publishing Corporation. Meddletown, USA (2001, Vol. 9, No. 4). p. 291-313, December.

Korpela, E., Werthimer D., Anderson D., Cobb J. e Lebofsky M., (2001) "SETI@home-Massively Distributed Computing for SETI". Computing in Science & Engineering Magazine, Vol. 3, No. 1 pp. 78-83, January/February.

Legion - A WorldWide Virtual Computer, http://legion.virginia.edu.

Manola, F. (1999) “Characterizing Computer-Related Grid Concepts”, http://www.objs.com/agility/tech-reports/9903-grid-report-fm.html.

Müller, J. P., (1997) “The Design of Autonomous Agents: A Layered Approach”, volume 1177 of Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg.

Rana, O. F. e Moreau, L., (2000) “Issues in Building Agent-Based Computational Grids, UK Multi-Agent Systems” Workshop, Oxford.

Rana, O. F. e Walker D. W., (2000) “The Agent Grid: Agent based resource Integration in Problem Solving Environments”, 16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Lausanne, Switzerland.

Thompson, C. e Odell, J., (1999) "Agent Technology Glossary", http://www.objs.com/agility/tech-reports/9909-agent-glossary.html.

United Devices Inc, http://www.ud.com/home.htm.

XML Extensible Markup Language, http://www.w3.org/XML/.

Wijngaards, N. J. E., Overeinder, B. J., Steen, M. van e Brazier, F. M. T., (2002) “Supporting Internet-Scale Multi-Agent Systems”. Data and Knowledge Engineering. June 2002, Volume 41, number 2-3, pp. 229-245, June.