UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS DA EDUCAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

BRENO SYPERREK

AVALIAÇÃO EXPERIMENTAL DE ALGORÍTMOS DE NEGOCIAÇÃO APLICADOS AO BALANCEAMENTO DE CARGA NO GRID DE AGENTES PARA GERÊNCIA DE REDES

DISSERTAÇÃO DE MESTRADO

Florianópolis
2011

BRENO SYPERREK

AVALIAÇÃO EXPERIMENTAL DE ALGORÍTMOS DE NEGOCIAÇÃO APLICADOS AO BALANCEAMENTO DE CARGA NO GRID DE AGENTES PARA GERÊNCIA DE REDES

Dissertação de mestrado apresentada à Banca Examinadora do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Mestre em Ciência da Computação.

Orientador:
Prof. Carlos Becker Westphall, Dr.

Área de concentração: Ciências da Computação.

Linha de pesquisa: Rede de Computadores.

Florianópolis
2011

Catalogação na fonte pela Biblioteca Universitária
da
Universidade Federal de Santa Catarina

S994a Syperrek, Breno
Avaliação experimental de algorítmos de negociação aplicado ao balanceamento de carga no Grid de agentes para gerência de redes [dissertação] / Breno Syperrek ; orientador, Carlos Becker Westphall. - Florianópolis, SC, 2005.
84 p.: il.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da computação. 2. Algoritmos. 3. Sistemas de computação em grade. 4. Gerência de redes. I. Westphall, Carlos Becker. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU 681

BRENO SYPERREK

AVALIAÇÃO EXPERIMENTAL DE ALGORÍTMOS DE NEGOCIAÇÃO APLICADOS AO BALANCEAMENTO DE CARGA NO GRID DE AGENTES PARA GERÊNCIA DE REDES

Esta dissertação foi julgada adequada para obtenção do título de

MESTRE EM CIÊNCIA DA COMPUTAÇÃO

Especialidade **SISTEMAS DE COMPUTAÇÃO** e aprovada em sua forma final pelo **PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO.**

_____

Prof**.** Dr. Raul Sidnei Wazlawick,
Coordenador do Curso de Pós-Graduação
em Ciência da Computação

BANCA EXAMINADORA:

_____

Prof. Dr. Carlos Becker Westphall
Presidente

_____

Prof. Dr. Mário Antonio Ribeiro Dantas
Membro

_____

Profa. Dra. Carla Merkle Westphall
Membro

_____

Prof. Dr. Vitório Bruno Mazzola
Membro

AGRADECIMENTOS

Agradeço a minha família, especialmente a memória de meu pai, namorada, amigos e professores.

RESUMO

As atividades relacionadas ao gerenciamento de redes têm se tornado cada vez mais complexas devido ao crescimento das redes tanto em número de dispositivos como na variedade dos mesmos. Atualmente, o grande número de equipamentos existentes nas redes corporativas e a diversidade destes equipamentos tornam os modelos tradicionais de gerência realmente impraticáveis, pois a necessidade destes modelos por recursos de processamento, armazenamento e memória de alto desempenho torna o gerenciamento uma tarefa onerosa. Como alternativa ao modelo centralizado de gerência surgem os modelos distribuídos. Um destes modelos, descrito nesta dissertação, faz uso de agentes autônomos para compor um /grid/ de agentes para a gerência de redes. Neste /grid/ são necessários mecanismos para distribuição de tarefas, balanceamento da carga de trabalho e tolerância à falhas. O principal objetivo desta dissertação foi relizar uma análise do desempenho dos algoritmos english e dutch auction e contract-net na distribuição das tarefas e balanceamento de carga nesta arquitetura. Os resultados finais desta análise mostram que os algoritmos english e dutch auction realizam a distribuição de tarefas de processamento de dados de forma mais eficiente que o contract-net. Isso devido a característica de mudança rápida de disponibilidade de capacidade de processamento que o cenário apresenta. Já na distribuição para armazenamento de dados coletados os três algoritmos tiveram resultados parecidos, pois a capacidade de armazenamento é uma característica que não se altera em um curto espaço de tempo neste cenário.

ABSTRACT

The activities related to network management has become increasingly complex due to the growth of networks both in the number of devices like the variety of them. Currently, the large number of existing equipment in enterprise networks and the diversity of these devices make the traditional models of management really impractical, because the need of these models for the processing, storage and high-performance memory management becomes an onerous task. As an alternative to centralized model of management arise distributed models. One such model, described in this thesis makes use of autonomous agents to compose a grid of agents for network management. In this grid mechanisms are needed to distribute tasks, workload balancing and fault tolerance. The main objective of this thesis erform a performance analysis of Inglish Auction, Dutch Auction and contract-net algorithms in the distribution of tasks and load balancing in this architecture. The final results of this analysis show that the Inglish Auction and Dutch Auction algorithms conduct the distribution of data processing tasks more efficiently than the contract-net. This is because the characteristic of rapid change in availability of processing capacity. In the distribution for storing data, the three algorithms had similar results, because the storage capacity is a feature that does not change in a short period of time in this scenario.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O crescimento das redes e a diversidade dos equipamentos tem tornado o trabalho do administrador e gerente das redes uma tarefa que se torna complicada a cada dia que passa. Nos dias atuais já é inviável manter um sistema de gerência centralizado, pois para processar a grande quantidade de dados gerados por estas redes, vindos de equipamentos tão heterogêneos, a capacidade de processamento destes computadores é insuficiente. Várias propostas surgiram para descentralizar a gerência. Das mais promissoras podemos destacar as que envolvem o uso de agentes de software nos sistemas de gerenciamento[2][12][13].

Utilizando uma plataforma de agentes na gerência da rede, obtemos várias vantagens em relação ao gerenciamento centralizado[14], no sentido de descentralização do processamento das informações de gerência. Contudo, ainda assim, o problema da capacidade de processamento persiste. Numa rede de grande escala a quantidade de dados de gerência gerados pode saturar a capacidade de processamento de um sistema multi-agentes. Uma proposta para este problema foi apresentada por Assunção ,Koch e Westphall[3][4], onde um *grid* de agentes de gerência seria responsável por coletar, armazenar, analisar e fazer a interface entre o sistema e o usuário.

A diferença entre um *grid* de agentes e uma plataforma de agentes tradicional está na forma como os agentes de cada um dos sistemas interagem. Em um *grid* de agentes podemos ter agentes de plataformas diversas e com diversas morfologias, mas que precisam interagir entre si, pois estes agentes se conectaram ao *grid* para disponibilizar sua capacidade de processamento e serviços, ou para procurar outros agentes que o ajudem na tarefa que precisam desempenhar. Alguns padrões, no intuito de proporcionar uma integração mais fácil das várias plataformas de agentes, surgiram desta necessidade visando facilitar o trabalho dos desenvolvedores de plataformas multi-agentes.

Dentre os módulos da arquitetura de *grid* para gerência proposta por Assunção, Koch e Westphall[3][4], dois deles merecem uma atenção especial, que são: o módulo de armazenamento de dados e de análise das informações de gerenciamento. Para o sucesso da arquitetura estes módulos devem ser desenvolvidos de forma que possam proporcionar um melhor aproveitamento de recursos da rede nas atividades de gerenciamento. Por melhor aproveitamento se refere à possibilidade de agregar recursos, que não estejam sendo utilizados por seus usuários, nas atividades de análise de dados e armazenamento de informação de

gerenciamento. Estes recursos podem ser computadores da própria rede que está sendo gerenciada. Porém, para que isso seja possível, estratégias para proporcionar uma distribuição uniforme e eficiente das tarefas de análise de informação precisam ser investigadas.

Diante desta necessidade este trabalho se propõe a abordar, implementar e realizar experimentações de técnicas de negociação entre agentes para proporcionar esta distribuição. Dentre estas técnicas podemos destacar os protocolos de leilão e o contract-net. Verificar o desempenho destes protocolos na distribuição de carga nos cenários de coleta, armazenamento e análise das informações de gerência, bem como o desempenho do protocolo durante a fase de negociação.

1.1. JUSTIFICATIVA

A gerência de redes vem se tornando uma tarefa complicada a cada dia. A quantidade cada vez maior de dados gerados pelas redes atuais torna inviável o modelo de gerência de redes centralizado como vem sendo praticado. Para tratar esses dados e gerar as informações necessárias para a tomada de decisão dos gerentes de rede é necessário o uso de cada vez mais processamento e recursos tornando oneroso o trabalho de tratá-los de forma centralizada.

A IA (IAD - Inteligência Artificial Distribuída) veio em auxílio a este problema, trazendo os agentes autônomos que descentralizam o processamento e distribuem as tarefas entre vários recursos da rede[2]. Mas, mesmo com o auxílio dos sistemas multi-agentes a grande escala das redes e a heterogeneidade dos equipamentos torna esta tarefa trabalhosa. Surge então a idéia de se agrupar estes sistemas multi-agentes em *grids* de agentes[3][4] para executar a tarefa de análise de dados de gerência.

Todos os conceitos relacionados aos sistemas multi-agentes devem ser agora aplicados aos *grid* de agentes, o que por si só é um desafio[1]. Estes conceitos, quando padronizados e especificados garantem que agentes de plataformas diferentes e concebidos de maneiras diferentes possam interagir e se comunicar.

Um dos problemas principais em um sistema deste porte é a distribuição da carga de processamento. Como o objetivo principal de um sistema multi-agentes assim como de um *grid* é a utilização dos recursos ociosos da rede, mecanismos para efetuar esta distribuição de carga entre estes recursos ociosos é fundamental. Um destes mecanismos são as

técnicas de negociação e os protocolos de interação entre agentes. Com este trabalho, queremos avaliar o desempenho da distribuição de carga entre os agentes fazendo uso destas técnicas de negociação.

1.2. OBJETIVO GERAL

O objetivo geral deste trabalho é a aplicação de técnicas de negociação nos cenários de coleta, armazenamento e análise de dados no *grid* de gerência de redes proposto por Assunção, Koch e Westphall[3][4] para avaliar a distribuição de carga das tarefas pertinentes a cada cenário.

1.2.1. Objetivos Específicos

Os objetivos específicos que podem ser enumerados são os seguintes:

a) Implementar o protocolo *contract-net*[34][31] e algoritmos de leilão *dutch auction*[30] e *english auction*[29] na plataforma de *grid* de agentes para gerência de redes.
b) Examinar os benefícios gerados da distribuição de tarefas e balanceamento de carga pela aplicação das técnicas de negociação entre agentes.
c) Avaliar e comparar o desempenho dos três algoritmos de negociação propostos e sua tolerância a falhas
d) Avaliar e comparar o comportamento dos três algoritmos propostos em relação as características dos cenários de armazenamento e análise.

1.3. MOTIVAÇÃO E TRABALHOS RELACIONADOS

Como motivação para o trabalho com *grid* de agentes podemos citar alguns dos projetos que vêm sendo desenvolvidos nesta área. Podemos citar como um dos projetos de maior destaque o CoABS[5]. O CoABS é um projeto desenvolvido pelo DARPA (Departamento de Defesa Norte Americano) que visa aplicar os conceitos da tecnologia de agentes para auxiliar em operações militares, tanto no controle como na comunicação e acesso aos dados relevantes para serem utilizados no

campo de batalha. Além disso, como um objetivo secundário, o CoABS está sendo desenvolvido para integrar os vários sistemas dos pesquisadores do projeto.

O Agentcities[6] 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[8]. 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*[7]. O sonho de que a Internet será habitada por agentes de software tende a se concretizar. Neste cenário de larga escala, assim como em um *grid*, um conjunto de protocolos padrões de negociação e colaboração que garantam a interoperabilidade e a formação das organizações virtuais é extremamente importante.

Pode-se destacar trabalhos onde estão sendo propostas organizações virtuais [43] onde agentes com mesmos objetivos, ou voltados para as mesmas metas são agregados e interagem entre si utilizando métodos de negociação como os descritos neste trabalho.

Dinklon e Nimis [42] propõe métodos de conversação entre agentes para troca de informações sobre tarefas sendo executadas através da utilização do contract-net como padronizado pela FIPA.

Outro trabalho que merece destaque apresenta um estudo sobre a escalabilidade de algoritmos de negociação em ambientes de *grids* de agentes onde o número de indivíduos é muito grande [41].

1.4. ORGANIZAÇÃO DO TRABALHO

Visando o melhor entendimento dos resultados obtidos por este trabalho ele está organizado em 5 (cinco) capítulos da seguinte maneira:

1. O primeiro capítulo discorre sobre a introdução do problema, justificativas para este trabalho e objetivos geral e específico.
2. O capítulo dois apresenta conceitos relacionados a este trabalho como *grids* computacionais, uma breve definição sobre agentes

inteligentes e o desenvolvimento de *grids* de agentes e sua aplicação na gerência de redes.

3. O capítulo três apresenta conceitos de interação e colaboração entre agentes. Dentre os métodos de colaboração destaca-se com mais ênfase neste trabalho a negociação. Os algoritmos de leilão e o contract-net estão apresentados com detalhes, pois são a base sobre a qual este trabalho foi desenvolvido.
4. No quarto capítulo descrevemos a proposta deste trabalho, qual seja, aplicar as técnicas de negociação descritas no capitulo 3 no grid de agentes aplicado a gerência de redes, descrevendo os pontos específicos do grid onde esta negociação será necessária. Também neste capítulo serão descritos os testes realizados e os resultados obtidos nestes testes.
5. No capítulo cinco serão apresentadas as conclusões obtidas deste trabalho.

2 GRIDS

A palavra g*rid* está cada vez mais sendo utilizada na computação em geral. Ela é usada para caracterizar sistemas ou algum tipo de *frameworks* que agrupam *hardware* ou *software* com o objetivo de compartilhar recursos de forma fácil[21].

Em muitos casos onde o processamento de informações exige uma grande quantidade de recursos de *software* ou *hardware*, a computação em *grid* vem em auxílio, possibilitando a distribuição das tarefas entre os vários recursos conectados de forma mais barata e fácil.

Nos itens a seguir apresentaremos dois conceitos relacionados ao *grid* e algumas aplicações destas variantes.

2.1 COMPUTAÇÃO EM GRID

A computação em *grid* surge da necessidade de se compartilhar recursos de *software* e *hardware* de forma fácil, barata, segura, confiável e universal a dispositivos computacionais de alta capacidade[16]. Ou ainda como definido em[21], um mecanismo para interagir ou compartilhar componentes físicos e lógicos para que sejam utilizados como se fossem um único.

Os *grids* computacionais foram inspirados nas redes elétricas onde a energia que se necessita está disponível ao alcance de um *plug* de forma fácil e barata. Para que se utilize esta energia basta apenas conectar-se a este *plug*. Na computação em *grid* esta energia é substituída por recursos. Quando se necessita de recursos, o usuário pode conectar-se ao *grid* e estes recursos estarão disponíveis da mesma forma que a energia na rede elétrica.

Esta idéia surgiu principalmente motivada pelo surgimento de aplicações de larga escala que necessitam de grande quantidade de recursos para desempenhar suas tarefas. Em muitos casos, o processamento centralizado para aplicações deste tipo é inviável tanto financeira quanto computacionalmente. A quantidade e o preço dos recursos que deveriam estar disponíveis para a execução destas aplicações é dispendioso demais para muitas das empresas e instituições diversas. A saída mais viável foi o compartilhamento dos recursos já existentes entre estas empresas ou instituições. Quando os recursos de uma estão subutilizados, outra instituição pode utilizar estes recursos a seu favor. A

Figura **1** mostra um esquema que representa um *grid* computacional como uma fonte de recursos compartilhados.

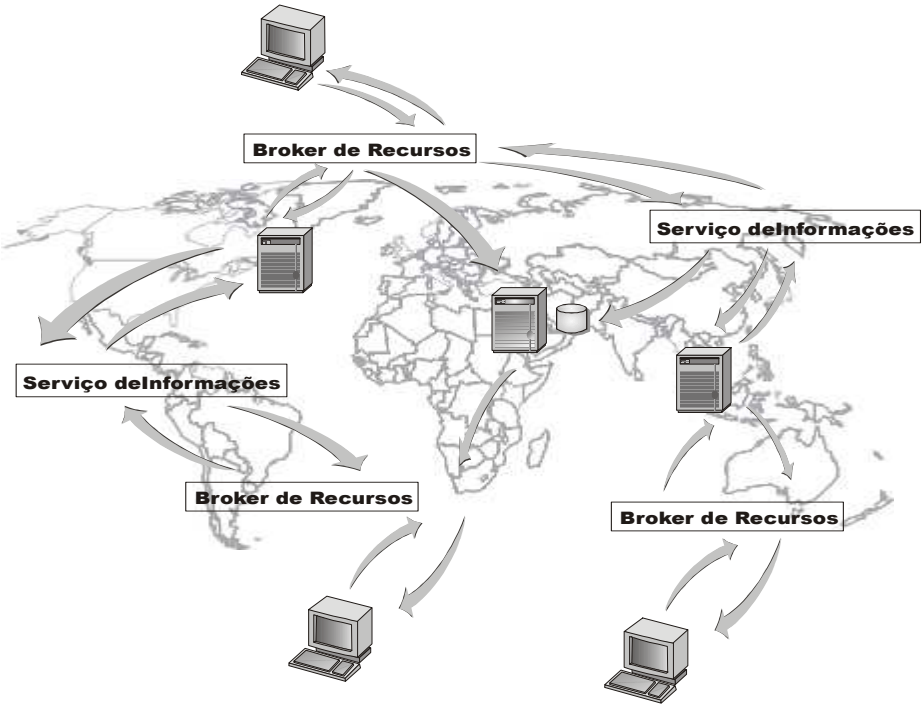

Figura 1: O Grid como fonte de recursos

O fator mais relevante para a união destes recursos é o alto desempenho e desenvolvimento das tecnologias de rede existentes atualmente. É neste cenário, com as organizações com objetivos afins e de compartilhar recursos, que surgiu o conceito de *grids* computacionais. Cada uma destas organizações é responsável pelo seu domínio de recursos, assim podemos observar que não existe um controle centralizado em um *grid* computacional. Segundo Manola[21], é neste cenário que lentamente está se desenvolvendo um ambiente computacional de larga escala com o compartilhamento de bancos de dados, computadores e dispositivos externos.

Algumas características diferenciam um *grid* computacional de sistemas distribuídos e de computação paralela tradicionais:

- Controle distribuído: Como já foi citado no parágrafo anterior, o *grid* não possui um controle centralizado. Cada organização é responsável pelo seu domínio de recursos.
- Heterogeneidade: A diversidade dos dispositivos de rede, software e hardware fazem com que os recursos disponíveis sejam heterogêneos. Além disso, as políticas administrativas das organizações são diferenciadas o que caracteriza uma grande diferença de *grids* em relação aos sistemas paralelos e distribuídos tradicionais.
- Distribuídos geograficamente: Como também já foi citado, os recursos de software e hardware estão distribuídos em uma grande quantidade de locais diferentes conectados e acessíveis através das tecnologias de rede existentes.
- Dinamicidade: As organizações podem se conectar e desconectar do *grid* a qualquer momento. Assim, a capacidade de recursos do *grid* é dinâmica em relação ao tempo.

As características dos *grids* computacionais listadas acima, em principio diferenciam os *grids* de sistemas distribuídos e paralelos tradicionais. Mesmo assim a definição do que é um *grid* ainda esta sendo formada. A ausência de alguma destas características não descaracteriza um *grid* computacional.

2.2.1 Principais plataformas de Grid computacional

Várias plataformas de *grids* computacionais surgiram para o desenvolvimento de sistemas baseados em *grids*. Estas plataformas possibilitam, através de *middlewares*, protocolos e APIs a criação de tais sistemas. São citadas a seguir algumas destas plataformas e suas características:

Globus Tollkit

O Globus Tollkit faz parte do projeto de pesquisa Globus[17][19]. Este projeto abrange cinco áreas. São elas: Segurança, Serviço de informações, Ambiente de desenvolvimento de aplicações, gerência de dados e gerência de acesso a dados. Esta plataforma tem código aberto e pode ser utilizada livremente. Alguns sistemas em *grid* já vêm sendo

desenvolvidos com a utilização do Globus, como por exemplo, TeraGrid[22], o Information Power Grid da NASA[18].

O Globus Toolkit está dividido em módulos que podem ser utilizados separadamente pelos desenvolvedores de sistemas em *grid*. Tais módulos compreendem:

- Gerência de recursos: O gerenciador de dados é responsável pelo gerenciamento dos recursos solicitados pelas aplicações remotas. Entre outras coisas é o responsável por fazer o balanceamento de carga do *grid*, tarefa importantíssima nestas plataformas. Além disso, possui APIs para que possibilitam que as aplicações submetam, cancelem ou acompanhem os trabalhos submetidos ao *grid*.
- Serviço de Informações: o serviço de informações ou MDS é um serviço de diretórios onde recursos que se conectam ao *grid* são registrados. Ele está baseado no protocolo LDAP. Qualquer recurso que se conecte ao *grid* é registrado no MSD e está disponível para ser utilizado pelos membros que fazem parte do *grid* através do gerenciador de recursos.
- Gerência de Dados: o gerenciador de dados é baseado nos protocolos de transporte como HTTP e FTP e juntamente com o protocolo GSI provê acesso a arquivos distribuídos no *grid*. Além disso, possui ferramentas para manutenção de conjunto de dados.
- Segurança: como a *grid* se utiliza de uma rede pública para conectar recursos, o protocolo de segurança GSI possui mecanismos para prover autenticação e segurança no transporte das informações entre os membros o *grid*. É baseado principalmente em algoritmos de criptografia de chave pública e no protocolo SSL.

Legion

O Objetivo da plataforma Legion[20] é que os usuários que se utilizam dela como se estivessem utilizando um supercomputador em seu *desktop*. Ele foi desenvolvido pela universidade da Virgínia e seu objetivo é o suporte de autos graus de processamento distribuído e gerência do sistema físico para o usuário.

A arquitetura do Legion é baseada no modelo de objetos onde cada elemento ativo do *grid* é representado por um objeto. Os usuários podem desenvolver seus próprios objetos para estender a plataforma. Os objetos

representam recursos como sistemas de armazenamento, recursos computacionais, execução de códigos e identificadores de processos.

MyGrid

Muitas pesquisas vêm sendo feitas sobre a tecnologia de *grids* computacionais, apesar disso, as plataformas que implementam esta tecnologia são poucas[15]. O projeto MyGrid pretende mudar esta situação propondo uma plataforma em que os usuários se conectem ao *grid* e tenham recursos disponíveis para executar tarefas que podem ser divididas e executadas em qualquer ordem.

Esta plataforma pretende ser o mais *plug-and-play* possível sempre objetivando a segurança do ambiente do usuário.

Esta plataforma diferencia o tratamento dado a máquina do usuário, onde estão os dados e onde serão armazenados os resultados, das estações onde os dados serão processados.

2.2 AGENTES DE SOFTWARE

2.2.1 Uma noção “fraca” do conceito Agente

Segundo Wooldridge e Jennings[10], o termo agentes usado atualmente descreve uma entidade que possui as seguintes características:

- *Autonomia*: o agente pode existir e operar sem a intervenção de outras entidades relacionadas ao sistema sejam elas humanas ou outras. Além disso, o agente tem controle sobre suas ações e seu estado.
- *Capacidade social*: os agentes se utilizam de uma linguagem específica para interagir com outros agentes ou com humanos
- *Reatividade*: um agente pode sentir o ambiente onde vive e com isso reage a mudança deste ambiente. A Figura 2 representa um modelo geral de agentes onde podemos observar os mecanismos que possibilitam esta característica. Assim, um agente que está em um estado de inatividade pode mudar seu estado de acordo com as mudanças no ambiente, por exemplo, um *click* do mouse ou a chegada de um e-mail.
- Pró-atividade: além do ambiente, o agente pode mudar seu comportamento por conta própria, ou seja, tomar a iniciativa quando de alguma tarefa que deve ser executada.

- Continuidade temporal: os agentes estão continuamente em execução. São processos que não se encerram depois da execução da tarefa.
- Orientado por objetivos: o agente é uma entidade que executa tarefas complexas. A decisão de dividir a tarefa em problemas menores, quais destes problemas tem maior prioridade e se o agente vai executar estes problemas sozinho ou com a ajuda de outras entidades é dada ao agente.

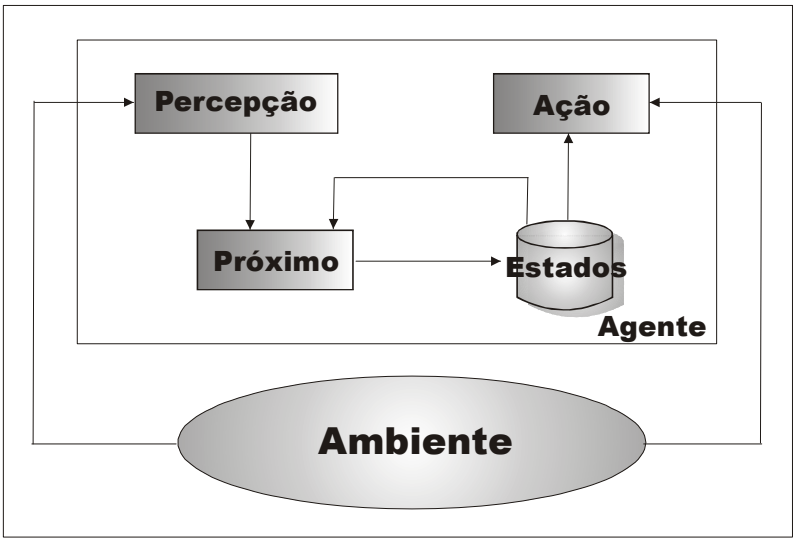

Figura 2: Modelo Geral de Agente

2.2.2 Uma noção "forte" do conceito agente

Para alguns pesquisadores da área de Inteligência Artificial o termo agente abrange um significado mais específico, além das características citadas na sessão anterior. Estas características adicionais são contextualizadas usando conceitos que geralmente são atribuídos aos humanos. Dentre estes conceitos podemos citar conhecimento, confiança, interação e obrigação. Alguns pesquisadores consideram até a possibilidade de agentes agindo com emoção em um futuro próximo.

Os agentes que normalmente se encaixam nesta definição forte do conceito possuem as seguintes características:

- Mobilidade: o agente pode se movimentar pela rede. No caso de um agente que esta executando em uma maquina cujos

recursos se tornaram escassos, o agente pode migrar para outro local da rede com mais recursos levando consigo seu estado e as informações necessárias para finalizar a tarefa.

- Aprendizagem: o agente é capaz de mudar seu estado com base em acontecimentos anteriores.
- Adaptabilidade: o agente é capaz de se adaptar as mudanças do ambiente, seja elas provocadas pelo usuário ou por outras entidades do sistema.
- Agilidade*:* o agente deve ser capaz de aproveitar as oportunidades que surgem no sistema. Por exemplo, um computador da rede possui mais recursos que o computador onde o agente está executando. O agente deve perceber o está acontecendo e tomar as medidas necessárias para aproveitar esta oportunidade.
- Colaboração: um agente não é um indivíduo isolado no sistema. Agentes são concebidos para colaborar com outros agentes de diversas formas. Com a colaboração surgem problemas como a execução de tarefas para outros agentes sem as informações completas para isto. O agente deve saber tratar estes problemas de alguma forma.

Todas as características citadas são desejáveis em um agente de software. A ausência de qualquer uma delas não descaracteriza o agente. Apesar de nenhum agente possuir todas estas características, vários agentes possuem um bom número delas. A ocorrências destas características nos agentes dependem da finalidade para que o agente está sendo utilizado. Por exemplo, em um sistema distribuído, a mobilidade é um fator importante. Já em sistemas robóticos o fator reatividade é mais importante.

2.2.3 Agência

A agência é o grau de autonomia e autoridade imputada ao agente. A agência de um agente pode ser medida qualitativamente pelo grau de interação do agente com os outros agentes e entidades do sistema.

Para isto o agente deve operar de forma assíncrona. A medida que o agente representa um usuário ou determinado recurso, seu grau de agência aumenta. Um agente pode interagir com entidades como dados, aplicações ou serviços e, além disso, agentes mais desenvolvidos podem negociar e colaborar com outros agentes.

2.2.4 Inteligência

Podemos definir inteligência de agentes em função de outras características. Para ilustrar esta idéia podemos utilizar as definições de Hayes-Roth e da IBM:

Um agente pode realizar continuamente as seguintes funções:

- Percepção das mudanças no ambiente;
- Mudança do estado em função das mudanças no ambiente;
- Raciocínio para interpretar as mudanças, resolver problemas, inferir e determinar ações a realizar.

Agentes inteligentes são entidades que realizam tarefas para um usuário ou aplicação, com um determinado grau de autonomia ou independência, utilizando-se de conhecimento ou representação das necessidades e objetivos do usuário

A verdade é que, quanto mais conhecimento conseguirmos modelar em um agente, mais próximos estamos da meta de ter algo similar a um agente inteligente.

2.2.5 Agentes móveis inteligentes

Um agente móvel, como o próprio nome sugere, tem a propriedade de se movimentar entre os nós de uma rede como mostra a Figura 3. Ele pode suspender a execução de uma tarefa e migrar para outro computador da rede. Neste novo computador ele continua a execução do ponto onde foi suspensa. O agente é que escolhe o momento e para onde deseja migrar.

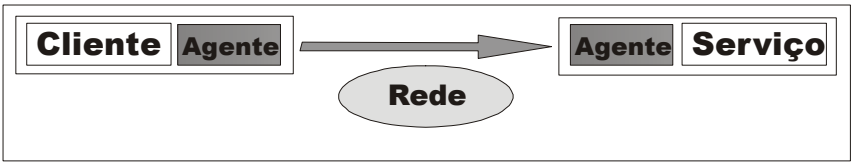

Figura 3: Paradigma de agentes móveis.

A tecnologia de agentes móveis emergiu dos avanços conseguidos na investigação em sistemas distribuídos. A idéia de transportar programas ao longo da rede não é nova, no entanto, a capacidade de um programa se transportar para novas localizações não é suficiente para descrever um agente móvel.

2.2.6 Uma definição mais completa

De acordo com a literatura existente[23] a seguinte definição caracteriza a essência de um agente móvel.

- **Modelo de agente**

Este modelo define a estrutura interna da parte inteligente do agente móvel.

Define a autonomia, aprendizagem e características de colaboração de um agente. A Figura **4** mostra como poderíamos classificar determinados agentes levando em conta as características de colaboração, aprendizado e autonomia presentes nos mesmos.

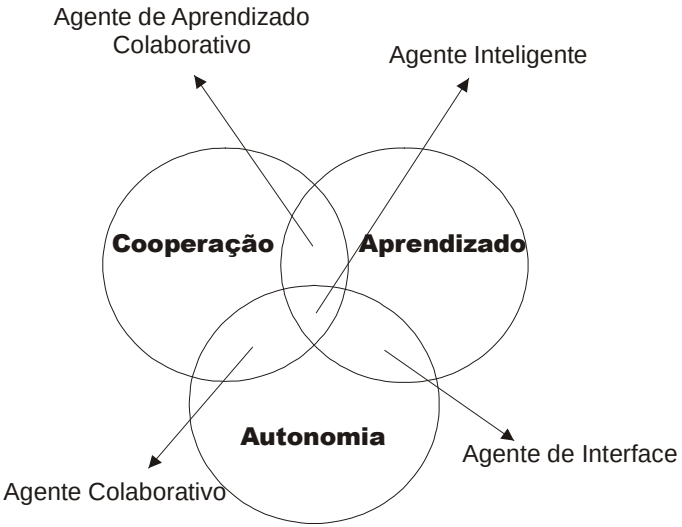

Figura 4:- Tipologia de agentes[NWAN 1996]

Além das características já citadas, este modelo especifica a natureza reativa ou pró-ativa dos agentes.

- **Modelo do ciclo de vida**

Este modelo define os diferentes estados da execução de um agente. Estes estados são influenciados por eventos que causam mudanças nos mesmos. Estes eventos podem ser internos ou externos ao agente. Assim este modelo esta intimamente ligado com o modelo computacional que define como a execução ocorre.

- **Modelo computacional**

O modelo computacional define como o agente executa suas funções quando esta no modo ativo. A este modelo estão relacionados recursos de hardware como o processador da estação onde o agente esta executando suas tarefas. Como recursos podemos definir a CPU de um computador ou, de forma mais abstrata, a máquina virtual Java.

- **Modelo de segurança**
  - **Proteção dos servidores de agentes maliciosos**

    Um sistema de agentes móveis, assim como um grid de agentes, como será definido os capítulos posteriores, é um sistema aberto e por isso está sujeito a uma variedade de ataques como, por exemplo, alteração de mensagens, mascaramento de identidade, vírus entre outros. Um agente está sujeito a qualquer destes tipos de ataques e os métodos tradicionais como criptografia, autenticação assinatura digital e hierarquia de confiança podem preveni-los.

    Além disso, o agente é executado por uma estação. Sendo assim, o agente tem acesso aos recursos desta estação. Neste nível de acesso, agentes maliciosos podem propagando vírus ou efetuar ataques que empeçam o funcionamento normal da estação. Tradicionalmente, para coibir este tipo de ataque, a abordagem é não deixar que código desconhecido seja executado pela estação. Em um *grid* de agentes, onde os agentes são heterogêneos, este método pode não ser eficaz comprometendo o funcionamento do *grid* de agentes.

  - **Proteção dos agentes de servidores maliciosos**

    Esta área lida com a proteção de agentes contra possíveis ataques provenientes da própria estação onde estão executando.

    Estes ataques podem alterar o código do agente, alterar o estado do agente, alterar o ciclo de vida do agente ou acessar informações contidas no agente.

    Neste caso, o agente está desprotegido em relação a estação, pois ele precisa expor seu código e dados ao ambiente da estação, a qual fornece os recursos necessários para sua execução.

- **Modelo de comunicação**

O modelo de comunicação é imprescindível em um sistema multi-agentes. È através deste modelo que os agentes conseguem se comunicar com as outras entidades do sistema em um ambiente computacional.

Estas entidades englobam outros agentes, usuários finais, recursos do sistema entre outros.

A comunicação é necessária quando se necessita de serviços fora do agente, quando se necessita cooperação ou coordenação com outros agentes ou outras entidades.

Um protocolo é uma implementação de um modelo de comunicação. Os protocolos variam desde o mais básico de transporte (e-mail, RPC) ao mais geral e complexo usados como linguagens de comunicação (KQML, FIPA ACL). Os protocolos também variam dependendo do tipo das entidades que se comunicam.

Devido ao elevado número de protocolos usados durante a comunicação é provável que os agentes móveis necessitem de mais do que um modelo de comunicação. Com os numerosos protocolos de comunicação surge a necessidade de tradução entre eles. Tal capacidade de tradução pode ser interna ao agente móvel ou pode ser contratada a um serviço externo de tradução.

- **Modelo de navegação**

Este modelo diz respeito a todos os aspectos da mobilidade dos agentes desde a descoberta e resolução de pontos de destino, até à forma como um agente móvel é transportado.

Apesar de cada um destes modelos ter sido descrito individualmente, eles estão intimamente integrados e interdependentes. A Figura 5 é uma representação da estrutura de um agente móvel.

Figura 5: Modelo Simples de Um Agente Móvel

O núcleo é baseado no modelo computacional, o que cousa bastante impacto nos outros modelos. Isto define como o agente se relaciona com outros agentes, estações e recursos, o que é importante para o modelo de segurança. O tipo de modelo de ciclo de vida adotado é dependente das características do modelo computacional. Por exemplo, todas as implementações de agentes móveis baseadas no Java foram forçadas a adotar o modelo baseado em tarefas. Tanto o modelo de segurança como o modelo de ciclo de vida estão estruturalmente muito próximos do núcleo. Os aspectos de segurança passam por todas as partes do agente móvel e, portanto, devem ser fornecidos nos níveis mais básicos.

As camadas exteriores contêm os modelos de comunicação, navegação e de agente. O modelo de agente define os aspectos "inteligentes" do agente móvel como a aprendizagem e funções de colaboração. O modelo de comunicação está fortemente dependente do modelo de segurança para que os agentes não sejam corrompidos por outros agentes ou estações. Finalmente, o modelo de navegação está também dependente do modelo de segurança, já que é o responsável pelo transporte do agente para outro nó. O processo de transporte muitas vezes move o agente para outro estado do ciclo de vida.

Isto define um agente móvel, no entanto, é necessário considerar o ambiente o qual o agente existe. Este é denominado por ambiente de agentes móveis. Assim, uma definição de ambiente de agentes móveis pode ser a seguinte[23]:

Um ambiente de agentes móveis é um sistema de software que está distribuído por uma rede de computadores heterogêneos. A sua principal tarefa é proporcionar um ambiente no qual os agentes móveis possam operar. O ambiente de agentes móveis implementa a maior parte dos modelos que apareceram na definição do agente móvel. Pode também fornecer: serviços de suporte que se relacionem com o próprio ambiente, serviços de suporte correspondentes aos ambientes em que os agentes móveis foram criados, serviços que suportem o acesso a outros sistemas de agentes móveis e, por último, suporte ao aceso a ambientes não baseados em agentes.

2.2.7 Vantagens dos agentes móveis inteligentes

Podemos citar como vantagens dos agentes móveis as seguintes:

- **Eficiência**

Os agentes móveis consomem menos recursos da rede uma vez que eles movem a computação para o local onde estão os dados;

- **Redução do tráfego da rede**

A maioria dos protocolos de comunicação envolvem diversas iterações, especialmente quando as medidas de segurança estão ativadas. Isto causa um elevado tráfego na rede. Com os agentes móveis é possível "empacotar a conversação" e enviá-la para o nó destino onde as iterações podem ter lugar, localmente;

- **Assincronismo**

O agente móvel pode operar assíncrona e independentemente da entidade que lhe enviou o pedido. Um exemplo disto poderia ser um dispositivo que após executar um agente para efetuar uma pesquisa autônoma numa determinada rede, desliga-se e volta a ligar-se mais tarde para recolher os resultados da pesquisa;

- **Interação com entidades de tempo real**

Entidades de tempo real como software que controla instalações nucleares requerem respostas imediatas a mudanças no seu ambiente. Controlar estas entidades numa rede potencialmente grande iria certamente significar enormes latências. Para situações críticas tais latências são inviáveis. Os agentes móveis oferecem uma alternativa: podem ser enviados de um sistema central para controlar entidades de tempo real em um nível local e processar ordens do sistema central;

- **Adaptação dinâmica**

Relacionado com o tópico anterior, os agentes móveis têm a capacidade de autonomamente reagirem a mudanças no seu ambiente. No entanto, tais mudanças têm de ser comunicadas aos agentes pelo ambiente de agentes móveis;

- **Lidar com grandes volumes de dados**

Quando vastos volumes de dados são armazenados em localidades remotas, é recomendado que o processamento destes dados seja feito localmente em vez de os transmitirem na totalidade pela rede;

- **Robustez e tolerância a falhas**

A capacidade dos agentes móveis reagirem dinamicamente a situações adversas torna mais fácil construir um comportamento tolerante a falhas, especialmente em sistemas largamente distribuídos;

- **Suporte a ambientes heterogêneos**

Tanto os computadores como as redes que fazem parte do sistema de agentes móveis são, por regra, de natureza heterogênea. Um sistema de agentes móveis é geralmente independente de computadores e redes e suporta operações transparentes entre eles;

- **Personalizar o comportamento do servidor**

Num domínio de redes inteligentes os agentes móveis são propostos como um meio para personalizar o comportamento de entidades da rede (por ex: roteadores) através do fornecimento dinâmico de novo comportamento;

- **Um conveniente paradigma de desenvolvimento**

A concepção e construção de sistemas distribuídos podem ser facilitadas com o uso de agentes móveis. Os agentes móveis são por natureza distribuídos e, logo, são aplicações dos sistemas distribuídas.

2.3 GRID DE AGENTES

Coordenar recursos múltiplos em um *grid* é uma tarefa com um nível de complexidade muito grande principalmente pela distância e heterogeneidade dos recursos. Além disso, a latência das redes que conectam estes recursos é uma barreira que pode aumentar a dificuldade de integrar estes recursos.

Muitas destas barreiras de integração de recursos podem ser vencidas com a utilização de sistemas multi-agentes. Segundo[24], a implementação de um serviço de localização de recursos em um grid pode ser uma tarefa difícil, porem pode ser facilmente resolvida com o uso de agentes. Além disso, a grande capacidade que os agentes têm de se adaptar aos ambientes onde existem faz com que possam ser usados para suportar e ampliar a capacidade de uma arquitetura computacional em *grid*.

Cada vez mais os sistemas multi-agentes estão sendo usados para controlar ambientes de computação em *grid*[21].Ainda assim, vários dos serviços que compõem um *grid* de agentes não estão bem definidos e ainda são objeto de estudo em vários laboratórios e universidades do mundo. Entre estes serviços podemos citar o serviço de localização, balanceamento de carga, registro de recursos em diretórios entre outros. Para melhor definir o comportamento, funcionamento e a utilização de

grids de agentes citaremos abaixo alguns dos projetos que vêm sendo desenvolvidos nesta área.

O CoABS[5], Controle de sistemas baseado em agentes, desenvolvido pelo DARPA, Agência de Projetos de Pesquisa Avançadas de Defesa dos Estados Unidos é um *middleware* que possibilita a interoperabilidade de sistemas, objetos dinâmicos e sistemas multi-agentes heterogêneos e distribuídos. A Figura 6 mostra um diagrama de funcionamento do CoABS.

Ele foi desenvolvido para um ambiente militar dinâmico e executa operações que precisam ser executadas rapidamente. O *software* e *hardware* que compõem o *grid* podem se conectar e desconectar a qualquer momento e estão sempre em movimento, pois o objetivo primário do projeto é interligar equipamentos e programas que estão sendo usados em campo de batalha.

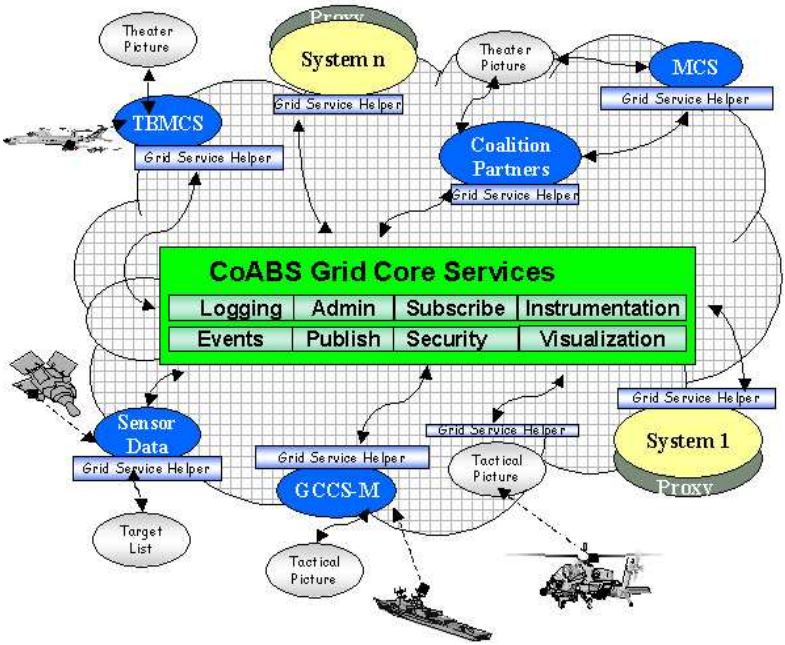

Figura 6: Diagrama de funcionamento do CoABS

Algumas das características necessárias para as operações propostas pelo projeto CoABS são:

- Serviço de diretórios: são registradas informações sobre os agentes como nome, localização, protocolos de comunicação ontologias entre outras informações. A busca por estas informações pode se dar pelo método de páginas brancas, onde a busca é feita pelo identificador do agente, ou através do método de páginas amarelas, onde a busca é feita utilizando-se, alem do identificador, outras informações que definem o agente registrado no diretório.
- Facilitadores: se dá pela utilização de *brokers* e *matchmakers*.
- Mobilidade: se dá pelo deslocamento dos agentes entre os nós que fazem parte do *grid.* A mobilidade possibilita uma melhor utilização dos recursos disponíveis e possibilita um método eficiente quanto a tolerância a falhas.
- Coordenação de times: possibilita a formação de times para a execução das tarefas e agrupamento de resultados.
- Gerência de Protocolos e Políticas: Possibilita a alteração dos protocolos de comunicação e políticas administrativas levando em consideração a mudança das circunstanciais do ambiente.
- Tradução: Por ser um sistema heterogêneo, é necessária a tradução entre interfaces e linguagens de comunicação e ontologias utilizadas pelos diversos agentes.
- Gerência de Ciclo de Vida: Define a criação, clonagem e morte dos agentes do sistema.
- Visualização: Serviço que possibilita uma compreensão do estado das aplicações.
- Segurança: devido ao fato de se tratar de um sistema aberto, o uso de métodos de segurança como autenticação e criptografia garante que os recursos não sejam utilizados de forma indevida, e agentes maliciosos não tenham acesso ao sistema.

Além do CoABS, citaremos o AgentScape[25][26]. O AgentScape é um *middleware* usado em sistemas de agentes de larga escala. Os três princípios básicos que servem de alicerce para o desenvolvimento deste *middleware* são:

- Prover uma plataforma para desenvolvimento de agentes em larga escala.
- Suportar a heterogeneidade entre sistemas operacionais e códigos de desenvolvimentos de sistemas.

- Interoperabilidade entre outras plataformas de agentes.

A plataforma tem como base duas entidades principais: agentes e objetos. Os agentes são considerados entidades ativas e interagem entre si através da troca de mensagens. Objetos são entidades passivas e são utilizados pelos agentes quando necessário. Ainda são definidos os locais onde os agentes e objetos existem e os serviços disponíveis na plataforma que possam ser utilizados pelas entidades.

O AgentScape é composto por um sistema operacional (AOS) que provê uma interface transparente para os níveis mais baixos da plataforma como por exemplo, recursos de *hardware*, acesso à rede e sistemas operacionais das estações que compõem a plataforma. Por ser uma arquitetura modular, o AOS possui um *kernel* com as principais funções que controlam todas as atividades e onde podem ser acoplados módulos para ampliação dos serviços. Estes módulos são:

- Comunicação: responsável pelo controle da comunicação entre os agentes.
- Localização: responsável por registrar o identificador do agente e sua localização, entre outras funções.
- Migração: módulo que é responsável pela mobilidade dos agentes entre as estações.
- Ciclo de Vida: responsável pela criação e morte dos agentes na plataforma.
- Segurança: responsável pela implementação dos métodos que garantam os parâmetros de segurança de um sistema aberto.

O acesso aos serviços oferecidos pelo *kernel* ou pelos módulos da plataforma é feito através de uma API desenvolvida para este fim.

2.7 O GRID DE AGENTES NA GERÊNCIA DE REDES

A proposta da utilização de *grids* de agentes na gerência de redes surge da necessidade de tratar uma grande quantidade de dados gerados por algumas redes atualmente. Assunção e outros [3][4] apresentam uma arquitetura de *grids* de agentes que pode ser vista na Figura **1**. Desta arquitetura merecem destaque dois cenários: coleta/armazenamento e análise de dados. Tais cenários serão descritos com mais detalhes neste trabalho pois é neles que a aplicação de técnicas

de negociação é necessária. A aplicação de técnicas de negociação nestes cenários tem por objetivo a distribuição das tarefas e o balanceamento da carga de trabalho entre os agentes que compõe entres *grids*.

O *grid* coletor é responsável por coletar os dados dos diversos equipamentos da redes. Esta coleta pode ser feita protocolos de gerência como SNMP, conjunto de ferramntas como WMI ou através de linhas de comando no sistema operacional. Os dados coletados são enviados então ao grid de classificação e armazenamento utilizando-se a lgum protocolo de transporte como http ou SNMP. Neste *grid* os dados são classificados para que possam ser armazenados e recuperados de forma eficiente. O *grid* de analise solicita a recuperação dos dados e gera as informações necessárias para que o *grid* de interface disponibilize tais informações para os gerentes das redes. A necessidade de distribuição das tarefas e balanceamento de carga é mais visível nos gris de armazenamento e de analise.

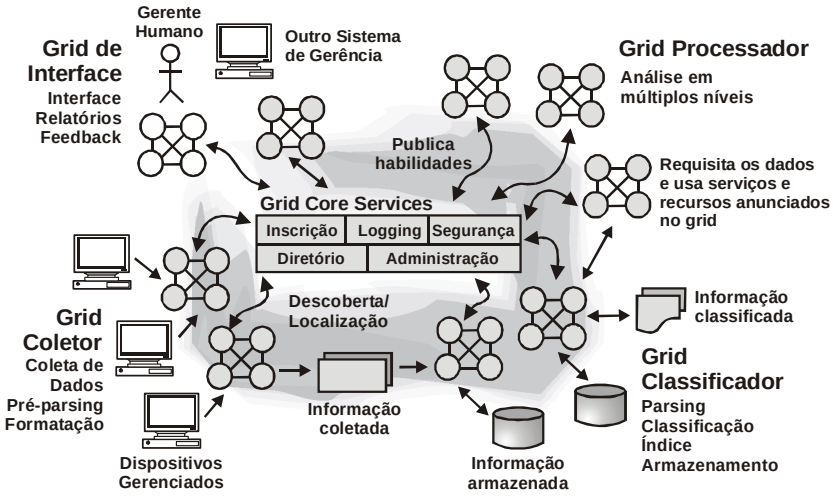

Figura 7: Grid agentes na gerência de redes [3][4]

Nos capítulos que seguem discutiremos as técnicas de negociação e descreveremos a utilização da negociação nos cenários onde ela se mostra mais relevante.

3 ACORDO, NEGOCIAÇÃO E COOPERAÇÃO ENTRE AGENTES

Um dos problemas relacionados à cooperação é a obtenção de acordos entre agentes com interesse próprios. No mundo dos agentes múltiplos requer-se, com freqüência, a interação entre os indivíduos para compartilhar objetivos comuns.

A habilidade de chegar a acordos é uma das capacidades fundamentais dos agentes autônomos inteligentes. Sem esta capacidade, é quase impossível estabelecer funções sociais entre os indivíduos. A capacidade de negociação e argumentação são pontos centrais na habilidade do agente de chegar a acordos com outras entidades.

Negociação, não ocorre por acaso, ela é governada por um mecanismo particular, ou protocolo. O protocolo define as regras para que uma negociação seja feita entre dois ou mais agentes[27]. Cada protocolo define as estratégias em particular adotadas pelo agente durante a negociação. Um agente usará a estratégia que melhor convier para que seu objetivo seja alcançado.

Neste capítulo são discutidos os processos de chegada a acordos através da negociação e argumentação. Também é apresentado o funcionamento de cada mecanismo ou protocolo e quais as propriedades que cada um deve ter para que a negociação e argumentação sejam realizadas.

3.1 DESENVOLVIMENTO DE MECANISMOS DE NEGOCIAÇÃO

O desenvolvimento de mecanismos de negociação compreende o desenvolvimento de protocolos que governam a interação entre agentes. É desejável que estes protocolos tenham algumas propriedades[28]. Estas propriedades são:

- Garantia de sucesso: O sucesso de um protocolo é garantido quando certamente um acordo é alcançado.
- Maximização do bem-estar social: um protocolo maximiza o bem-estar social assegurando-se de que o resultado da negociação satisfaça todos os participantes da interação.

- *Pareto Efitiency*: O resultado da negociação é dito *pareto efitiency* se nenhum dos membros que fazem parte da negociação leve algum tipo de vantagem em detrimento de outro.
- Racionalidade individual: um protocolo é dito individualmente racional se atingir os melhores interesses dos participantes. Esta propriedade é essencial, pois sem ela não existe incentivo para que os membros participem da negociação.
- Estabilidade: Um protocolo é estável se provê a todos os agentes um incentivo a se comportar de uma maneira particular.
- Simplicidade: Um protocolo simples é aquele que disponibiliza uma estratégia apropriada para os participantes da negociação, ou seja, um agente que faça uso de um protocolo simples pode facilmente determinar uma ótima estratégia para atingir suas metas.
- Distribuição: Um protocolo deve ser desenvolvido com o intuito de minimizar a comunicação entre os agentes.

3.2 ALGORÍTMOS DE LEILÃO

Leilões são exemplos muito simples de interação entre agentes[11]. Isto significa que é muito fácil automatizar leilões. Embora sejam simples, leilões apresentam uma série de problemas para os pesquisadores e é uma ferramenta poderosa para que agentes consigam alocar tarefas e recursos.

Um leilão acontece entre um agente conhecido como leiloeiro e um grupo de agentes conhecidos como lançadores. O objetivo do leilão é fazer com que o leiloeiro venda ou aloque determinada coisa a um dos lançadores. Em muitos casos, sendo os mais usuais, o leiloeiro pretende alocar ou vender determinado recurso a um lançador pelo melhor preço possível enquanto que o lançador tenta minimizar este preço. O leiloeiro usa um mecanismo apropriado para satisfazer suas necessidades enquanto que o lançador usa outro mecanismo também para chegar a um resultado satisfatório. Existem vários fatores que podem influenciar ambos os mecanismos e estratégias que os agentes podem usar.

Um dos fatores mais importantes é o valor do recurso que está sendo negociada. Podemos relacionar estes valores como sendo valores privados, valores públicos e valores combinados. Para diferenciar estes valores podemos citar como exemplo uma mercadoria que tenha um

determinado valor real de 1. Para a maioria dos envolvidos no leilão este é um valor aceitável, sendo este um valor público. Já, para um determinado lançador esta mercadoria tem um valor sentimental o que faz com que seu preço aumente, sendo este, um valor privado. O valor combinado depende parcialmente do valor privado e do valor que outros lançadores dão à mercadoria.

Tipicamente, o lançador que faz a maior oferta é o vencedor do leilão. Aqui entra outro fator de definição do protocolo. Alguns leilões são conhecidos como *fist-price*. Neste caso o vencedor adquire a mercadoria pelo preço mais alto ofertado. Existem ainda leilões conhecidos como *second-price* onde o lançador que fez a oferta mais alta ganha o leilão, mas o valor pago pelo produto é o segundo valor mais alto.

Em relação ao preço ofertado pelos outros lançadores o leilão pode ser classificado como *open-cry*, onde todas as ofertas são abertas e os lançadores conhecem todos os preços ofertados. Se os lançadores não conhecem os outros preços ofertados o leilão é dito *sealed-bid*.

O número de rodadas que um leilão pode ter também é um fator determinante do protocolo. Se o leilão possui apenas uma rodada ele é conhecido como *one shot*. Leilões com mais de uma rodada podem ser classificados como *ascending* quando o preço inicial é baixo e as ofertas vão aumentando ou *descending* caso contrário.

A seguir são descritos os protocolos de leilão denominados *english auction* e *dutch auction*. Estes protocolos serão abordados com mais detalhes, pois serão utilizados no cenário de negociação que será descrito nos capítulos posteriores. Existem, além destes, outros protocolos de leilão que são utilizados na negociação e distribuição de tarefas em sistemas multi-agentes[11].

3.2.1 English Auction

O *english auction* é um dos tipos mais comuns e conhecidos de leilão. Pode ser classificado como *firs-price*, *open-cry* e *ascending*. O funcionamento do *english auction* se dá da seguinte forma:

- O leiloeiro oferta o produto determinando um preço inicial abaixo do valor real (que pode ser 0), se nenhum lançador fizer uma oferta melhor, o produto continua de posse do leiloeiro.
- Lances podem ser dados por qualquer agente que queira oferecer um preço maior do que o último preço oferecido. Todos os agentes podem ver as ofertas feitas.

- Quando um agente faz a última oferta, cujo valor esteja próximo do valor real do produto, o mesmo é alocado ao agente que paga o preço oferecido neste último lance.

Qual estratégia os agentes devem adotar para participar de um *english auction*? A estratégia dominante é fazer sucessivas ofertas com um valor um pouco maior que o valor corrente até que este valor atinja o valor real do produto ou um limite determinado. Atingido este limite o agente desiste de fazer suas ofertas.

As especificações da FIPA[29] descrevem um padrão para a implementação do *english auction*. Neste trabalho usaremos estes padrões de protocolos devido a grande aceitação destas especificações no desenvolvimento de sistemas multi-agentes. A forma como a FIPA padroniza este protocolo é ligeiramente diferente do protocolo tradicional. A Figura 8 descreve o funcionamento deste protocolo segundo estes padrões através de um diagrama UML.

Segundo estas especificações o leiloeiro envia uma mensagem informando que o processo do leilão será iniciado. Logo após o leiloeiro envia uma proposta inicial aos agentes que fazem parte da interação. Estes agentes podem recusar ou aceitar a proposta. Caso o preço aceito seja satisfatório é enviado uma informação aos agentes que aceitaram a proposta e logo após uma requisição para a execução da tarefa. Caso o preço esteja abaixo do preço que satisfaça o leiloeiro outra proposta é enviada aos agentes e assim por diante.

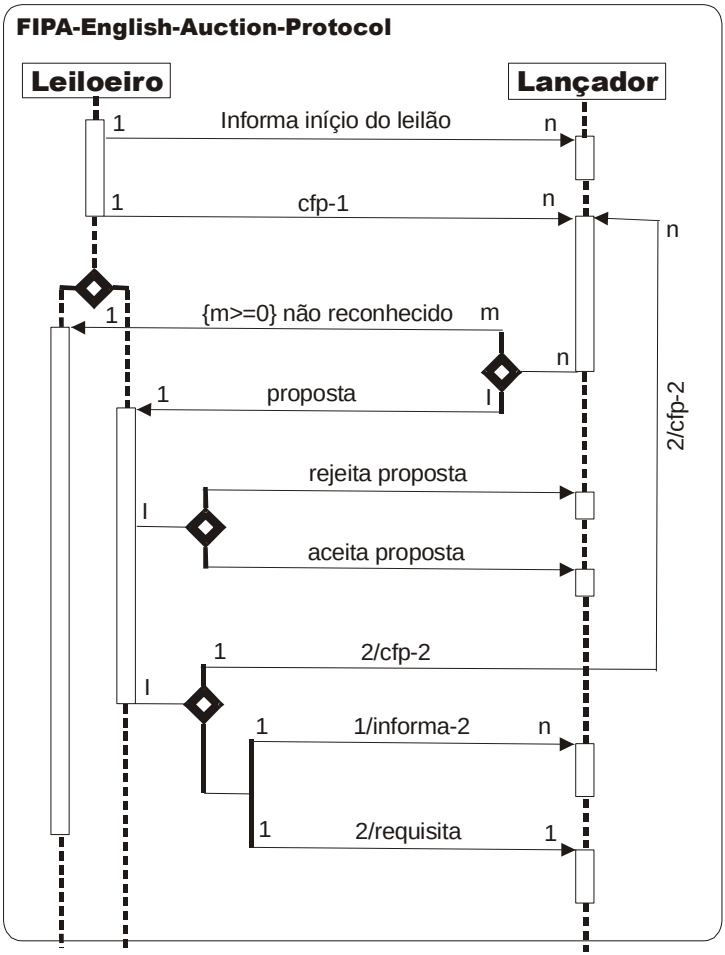

Figura 8: Protocolo de interação English Auction[29]

3.2.2 Dutch Auction

Dutch auction é um exemplo de leilão *open-cry* e *descending*. Seu funcionamento acontece da seguinte forma:

- O leiloeiro oferece a mercadoria por um preço bem mais alto que o valor real da mesma.

- O leiloeiro vai baixando gradualmente o preço da mercadoria até que algum lançador faça uma oferta que é igual ao valor corrente estipulado pelo leiloeiro.
- A mercadoria é alocada ao agente que aceita a melhor oferta.

Não existe uma estratégia dominante para os agentes participarem de um *dutch auction*.

De forma semelhante ao *english auction* a FIPA padroniza também o *dutch auction*[30]. A Figura 9 mostra um diagrama UML que ilustra o funcionamento deste protocolo de interação.

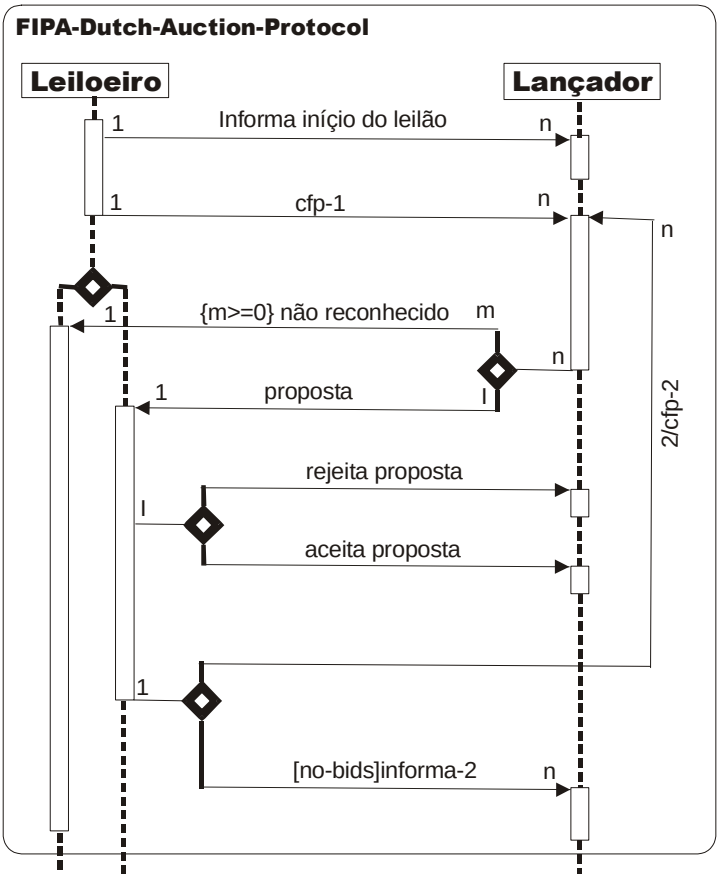

Figura 9: Protocolo de Interação Dutch Action[30]

Ao contrário do protocolo de interação *english auction*, o protocolo *dutch auction* definido nas especificações da FIPA atua de mesma forma que o protocolo originalmente concebido.

3.3 NEGOCIAÇÃO

Leilões são técnicas bem conhecidas de alocação de tarefas a agentes, porém, são muito simples para determinadas aplicações. Em muitos casos, onde agentes com interesse mútuo precisam chegar a acordos, técnicas mais elaboradas de interação são necessárias. Neste capítulo são abordadas técnicas que foram propostas por Rosenschein e Zlotkin[27] para negociação entre agentes artificiais.

Rosenschein e Zlotkin apresentam diferentes tipos de domínios de negociação: domínio orientado a tarefas e domínio orientado a valor. Antes de descrevermos estes domínios discutiremos as técnicas de negociação de uma forma geral.

Na maioria dos casos, qualquer negociação possui os seguintes componentes:

- Um conjunto de negociação, os quais representam o espaço de propostas possíveis que cada agente pode fazer.
- Um protocolo, que define as regras que o agente pode usar em suas propostas.
- Uma coleção de estratégias, uma para cada agente, que defina qual proposta o agente vai fazer. Freqüentemente, a estratégia que o agente utiliza é privada, não sendo visualizada por outros agentes, embora a maioria das negociações sejam *open cry* e a proposta feita pelo agente seja vista pelos outros participantes.
- Uma regra que determina quando se dá o acordo e o que é este acordo.

Normalmente o processo de negociação ocorre em várias rodadas, com cada agente fazendo suas propostas em cada uma delas. A proposta feita pelos agentes é definida por sua estratégia, que deve ser concebida no escopo da negociação e seguir as regras definidas por um protocolo. Se um acordo é alcançado, como definido pelas regras do acordo, a negociação termina.

Os quatro parâmetros listados acima podem fazer com que o ambiente de negociação torne-se complicado. O primeiro parâmetro pode

tornar a negociação complexa quando vários assuntos são negociados ao mesmo tempo pelos agentes. Isto ocorre, porque num cenário de negociação envolvendo múltiplos assuntos, os agentes não negociam sobre o valor de um atributo simples, mas sobre o valor de vários atributos que podem estar interligados.

Atributos múltiplos levam a um crescimento exponencial no espaço de possíveis acordos. Podemos citar, como um exemplo simples, agentes que estejam negociando sobre n variáveis que pode assumir os valores verdadeiro e falso. Neste caso, o espaço de possíveis acordos se restringe a 2^n resultados. Muitos domínios de negociação são mais complexos. Agentes que negociam sobre um número n de atributos que podem assumir m valores faz com que o espaço de possíveis acordos seja m^n. A negociação pode se tornar inda mais complexa quando valores como a crença dos agentes está envolvida no processo.

Outra fonte de complexidade na negociação é o número de agentes envolvidos no processo e o modo como estes agentes interagem. Podemos destacar as seguintes possibilidades:

- Negociação um-para-um: Um agente negocia apenas com outro agente. É um caso particular de negociação onde os agentes possuem preferências simétricas com respeito a um possível acordo. Por preferências simétricas entende-se que o objetivo dos dois agentes é o mesmo.
- Negociação muitos-para-um: Neste caso, um agente negocia com vários outros agentes. Leilões são um exemplo de negociação muitos-para-um. Este tipo de negociação pode ser tratado como um caso particular de várias negociações do tipo um-para-um.
- Negociação muitos-para-muitos: Vários agentes negociam com vários outros agentes ao mesmo tempo. No pior caso, podem existir n agentes no processo o que faz com que existam n(n-1)/2 *threads* de negociação sendo executadas simultaneamente.

Como o número de agentes está diretamente relacionado à complexidade da negociação, a automatização destes processos deve ser a mais simples possível.

3.3.1 Domínio orientado a tarefas

O primeiro domínio de negociação apresentado neste trabalho é o domínio orientado a tarefas como proposto por Rosenschein e Zlotkin[27]. Para formalizar um TOD será utilizada a notação desenvolvida pelos autores.

Um TOD é definida como uma tripla $\langle T, Ag, c\rangle$ onde T é um conjunto finito de possíveis tarefa, $Ag = \{1,...,n\}$ é um conjunto finito de agentes que fazem parte da negociação e $c : \wp(T) \rightarrow R^+$ é a função que define o custo da execução do subconjunto de tarefas e na maior parte dos casos é um número real.

A função que define o custo de execução do conjunto de tarefas deve satisfazer duas restrições. Primeira, devem ser monotônicas, isto é, adicionar tarefas ao conjunto nunca pode decrementar o custo. Formalmente podemos definir da seguinte maneira:

Se $T_1, T_2 \subseteq T$ são conjuntos de tarefas tal que $T_1 \subseteq T_2$, então

$$c(T_1) \leq c(T_2).$$

A segunda restrição afirma que o custo de não executar nenhuma tarefa é 0, ou seja, $c(\emptyset) = 0$.

Um encontro no domínio orientado a tarefas $\langle T, Ag, c\rangle$ acontece quando um agente AG tem tarefas associadas ao conjunto *T* para ser executado. Intuitivamente, quando um encontro acontece, existem potencialidades de que agentes entrem em um acordo através da realocação de tarefas entre eles. Formalmente, um encontro em TOD $\langle T, Ag, c\rangle$ é uma coleção de tarefas $\langle T_1,...T_n\rangle$, onde, para cada i, temos que $i \in Ag$ e $T_i \subseteq T$. Podemos observar que um TOD juntamente com um encontro neste TOD é um tipo de ambiente de tarefas onde os agentes operam.

As próximas definições se restringem a negociações do tipo um-para-um. Assumiremos que os dois agentes em questão são {1,2}. Agora, assumindo o encontro $\langle T_1, T_2\rangle$ podemos ver que um acordo é similar a um encontro: Ele será uma alocação de tarefas $T_1 \cup T_2$ para os agentes 1 e 2. formalmente um acordo puro é um par $\langle D_1, D_2\rangle$ onde $D_1 \cup D_2 = T_1 \cup T_2$. Podemos dizer que o agente 1 está comprometido

em executar as tarefas D_1 e o agente 2 está comprometido com a execução das tarefas D_2.

O custo de um agente para o acordo $\delta = \langle D_1, D_2 \rangle$ é definido como sendo $c(D_i)$, e deve ser denotado $\cos t_i(\delta)$. A utilidade de um acordo δ para um agente i é a diferença entre o custo do agente i executar a tarefa T_i, a qual foi originalmente associada no encontro, e o custo $\cos t_i(\delta)$ de uma tarefa que é associada em δ:

$$utility(\delta) = c(T_i) - \cos t_i(\delta).$$

Assim, a utilidade de um acordo representa o quanto o agente ganha com o acordo; se a utilidade é negativa, então para o agente o estado é poir do que se ele simplesmente desempenhasse as tarefas como estavam originalmente alocadas antes do acordo.

Em alguns casos podem ocorrer conflitos entre os agentes em uma negociação. Denotado como Θ, neste caso, o acordo $\langle T_1, T_2 \rangle$ consiste das tarefas originalmente alocadas.

A noção de dominação pode ser facilmente estendia para domínio. Um acordo δ_1 é dito dominante sobre o acordo δ_2 (escrito como $\delta_1 \succ \delta_2$) se e somente se:

- δ_1 é igual ou tão bom quanto δ_2 para qualquer agente
$$\forall i \in \{1,2\}, utility_i(\delta_1) \geq utility_i(\delta_2).$$
- δ_1 é melhor δ_2 para alguns agentes
$$\exists i \in \{1,2\}, utility_i(\delta_1) > utility_i(\delta_2).$$

Se um acordo δ_1 domina outro acordo δ_2 então, deve estar claro para todos os participantes que δ_1 é melhor que δ_2 e qualquer participante vai preferir δ_1 a δ_2. O acordo δ_1 é dito fracamente dominante em relação ao acordo δ_2 (escrito $\delta_1 \overset{\succ}{-} \delta_2$) se pelo menos a primeira condição de confirmar.

O acordo que não é dominado por nenhum outro acordo é chamado de pareto ótimo. Formalmente um acordo δ é chamado de pareto ótimo se não existe nenhum acordo δ' tal que $\delta' \succ \delta$. Se um acordo for pareto ótimo, então não existe nenhum acordo alternativo que melhore a

condição de um determinado agente, exceto com algum custo para outro agente. Se um acordo não é pareto ótimo, então os agentes podem propor outro acordo sem que não deixe outros agentes insatisfeitos

Um acordo é dito racional individual se ele domina fracamente o conflito ($\delta_1 \succeq \Theta$).

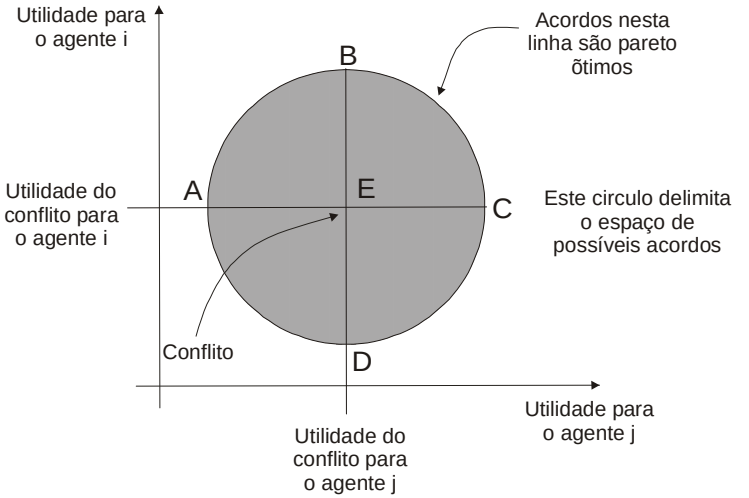

Figura 10: O conjunto de negociação[27]

Podemos agora definir o espaço de possíveis propostas que um agente pode fazer. O conjunto de negociação consiste de um conjunto de acordos que são (i) racionais individuais e (ii) pareto ótimo. Podemos dizer a partir da primeira restrição que não existe nenhum acordo que seja menos preferido pelos agentes do que o conflito e a segunda restrição nos diz que nenhuma proposta deve satisfazer um agente aumentado o custo de qualquer outro agente da negociação.

O conjunto de negociação esta ilustrado na figura Figura 10. O espaço de de todas as propostas aceitáveis pode ser plotado como pontos no gráfico, com a utilidade o agente i no eixo y e a utilidade do agente j no eixo x. O espaço de propostas compartilhado pelos agentes está restrito aos pontos A,B,C e D. O ponto de conflito entre os agentes é ilustrado pelo ponto E. Todos os acordos a esquerda da linha B-D não são racionais individuais para o agente j pois, neste caso, o conflito é melhor para o agente j. seguindo o mesmo raciocínio, todos os acordos abaixo de A-C não são racionais individuais para o agente i. Isso nos leva a concluir

que a área restrita B-C-E define o conjunto de negociação que contem os acordos que satisfazem os dois agentes, mas nem todos os acordos nesta área são pareto ótimo. Apenas os que estão contidos na linha B-C podem ser definidos como pareto ótimo. Observando o gráfico, espera-se que o agente i inicie a negociação propondo o acordo representado pelo ponto B e o agente j o acordo representado pelo ponto C.

3.4 TRABALHO COOPERATIVO

Neste capítulo vamos descrever como agentes podem ser concebidos para que possam trabalhar de forma cooperativa. A idéia de sistemas trabalhando cooperativamente inicialmente pode não parecer nova. O termo cooperação é largamente usado na literatura sobre sistemas concorrentes para descrever sistemas que interagem entre si para executar as tarefas que lhes convem. Existem duas distinções principais entre sistemas multi-agentes e sistemas distribuídos:

- Agentes, em sistemas multi-agentes, são desenvolvidos por diferentes pessoas com diferentes objetivos. Eles, a principio, não compartilham os mesmos objetivos, assim, um encontro entre agentes em um sistema multi-agentes são parecidos com um jogo, onde agentes agem estrategicamente com o objetivo de atingir um resultado em particular.
- Pelo fato de se assumir que agentes atuam de forma autônoma, eles devem ser capazes de coordenar suas atividades dinamicamente e cooperar uns com os outros. Em sistemas distribuídos e concorrentes tradicionais, coordenação e cooperação são definidos em tempo de implementação e não podem ser dinâmicos.

O trabalho cooperativo envolve uma série de atividades que serão descritas na seqüência, em particular o compartilhamento de tarefas e informações e a coordenação das atividades dos agentes.

3.4.1 Resolução de problemas distribuídos de forma cooperativa

Segundo Durfee[11], cooperação é necessária porquê nenhum nó isolado tem habilidade, recursos e informações suficientes para resolver um problema, e um conjunto de nós pode ter a habilidade de resolver diferentes partes do problema.

Historicamente, muitos dos trabalhos sobre resolução de problemas distribuídos de forma cooperativa são desenvolvidos levando em consideração a benevolência, isto é, agentes em um sistema compartilham o mesmo objetivo e não existem potenciais conflitos entre eles. Agentes não projetados para auxiliar na resolução de um problema, mesmo que isto implique em que um ou mais agentes sofram para fazer isto. Tudo o que importa é resolver o problema como um todo. A benevolência faz sentido quando todos os agentes de um sistema são desenvolvidos ou pertencem a uma única organização ou indivíduo. O fato de assumir a benevolência simplifica grandemente a tarefa dos desenvolvedores de sistemas multi-agentes.

Em contraste com o trabalho em resolução de problemas distribuídos de forma cooperativa, grande parte das pesquisas em sistemas multi-agentes está focada no desenvolvimento de sociedades de agentes com interesses próprios. Neste cenário, os interesses de um agente podem conflitar com os interesses de outros agentes. Mas, a apesar do potencial de conflitos ser grande, agentes em um sistema multi-agentes precisam cooperar para atingir seus objetivos.

É também importante distinguir CDPS da resolução em paralelo de problemas. Resolução em paralelo envolve simplesmente a exploração do paralelismo na resolução de problemas. Tipicamente, os componentes computacionais envolvidos são processadores. Um único nó é responsável por decompor o problema e alocar os sub-problemas aos processadores e reagrupar os resultados depois da execução. Freqüentemente, os nós que compões o ambiente paralelo são homogêneos no sentido de possuírem todos eles as mesmas habilidades.

Coerência e coordenação

Quando se está implementado uma sociedade de agentes artificiais com o objetivo de resolver algum problema, como o sucesso desta implementação pode ser alcançado? Que critérios devem ser levados em consideração?

A literatura de multi-agentes propõe dois assuntos que devem ser considerados:

- Coerência: O quanto um sistema multi-agentes ira comportar-se como um unidade segundo alguns parâmetros de avaliação. A

coerência pode ser medida em termos de qualidade das soluções alcançadas pelo sistema, eficiência dos recursos utilizados e o quanto o sistema degrada na presença de incertezas ou falhas. Wooldridge[32] faz um apanhado sobre o problema da avaliação da coerência nas ações de múltiplos agentes.

- Coordenação: em um sistema multi-agentes perfeitamente coordenado, agentes não interferem negativamente na execução e no resultado de outros agentes enquanto trabalham para atingir um objetivo comum. A presença de conflitos entre os agentes no sentido de interferência destrutiva entre eles é uma indicação de ma coordenação pobre.

Os principais questionamentos em relação a CDPS são:

- Como um problema pode ser dividido em tarefas menores para serem distribuídos entre os agentes?
- Como a solução de um problema pode ser sintetizada a partir dos resultados dos sub-problemas?
- Como a atividade dos agentes na resolução do problema como um todo pode ser otimizada para produzir uma solução que maximize as métricas de coerência?
- Que técnicas podem ser usadas para coordenar a atividade dos agentes para evitar interações destrutivas e infrutíferas e maximizar a eficiência?

Nos capítulos seguintes discutiremos algumas das técnicas que podem ser a resposta a essas perguntas.

3.4.2 Compartilhamento de tarefas e compartilhamento de resultados

Smith e Davis[33] sugerem que, para que um grupo de agentes possam trabalhar juntos com o intuito de resolver problemas, o CDPS pode ser visto como uma atividade de três estágios como mostra a

Figura **11**.

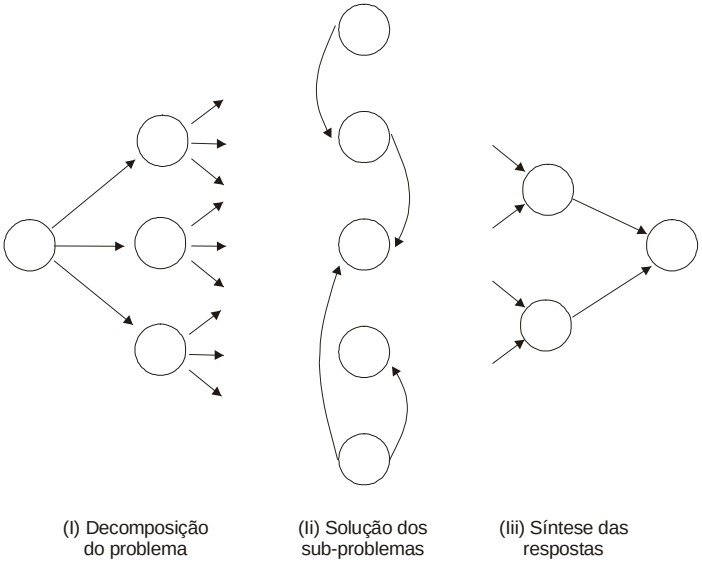

Figura 11: Os três estágios do CDPS[33]

I. Decomposição do problema: Neste estágio, o problema maior a ser resolvido é decomposto em vários sub-problemas menores. Esta decomposição pode ser hierárquica, ou seja, subproblemas podem ser decompostos em subproblemas ainda menores até que os sub-problemas tenha a granularidade adequada para serem resolvidos por um único agente. Os diferentes níveis de sub-problemas representam diferentes níveis de abstração do problema. Cada um destes diferentes níveis na hierarquia de resolução de problemas representa o problema maior em um nível diferente de abstração. Podemos observar que a granularidade do sub-problema é importante. Uma visão estrema do CDPS é que o problema pode ser decomposto até que se atinja sub-problemas que representam ações ou tarefas atômicas que não podem mais ser decompostas. Mas este cenário introduz uma serie de problemas, em particular, o overhead gerado pelo gerenciamento destes muitos subproblemas em detrimento do benefício da decomposição do subproblema como um todo.

Outro problema é como efetuar a decomposição. Uma possibilidade é que o problema seja decomposto por um único agente que deve ter a habilidade de executar esta tarefa. Para isto, ele deve conhecer a estrutura do problema. Se outros agentes possuem este conhecimento, eles podem ajudar na identificação da melhor decomposição. A decomposição pode ser melhor executada como uma atividade cooperativa.

II. Solução dos sub-problemas: Neste estágio, os sub-problemas identificados durante a decomposição são agora resolvidos. Este estágio envolve compartilhamento de informações entre os agentes. Um agente pode ajudar outro agente se ele possuir informações que sejam necessárias a outros agentes.

III. Síntese das soluções: Neste estágio, as soluções dos sub-problemas são integradas a solução do problema como um todo. Como na decomposição, este estágio deve ser hierárquico com as soluções parciais montadas em diferentes níveis de abstração.

A partir da visão geral acima, podemos enumerar duas atividades que devem estar presentes em uma CDPS e que são ilustradas na Figura 12:

- Compartilhamento de tarefas: Distribuição de tarefas tem lugar quando um problema é decomposto vários sub-problemas menores os quais são alocados a diferentes agentes. Possivelmente, o problema chave a ser resolvido na distribuição das tarefas é qual tarefa deve ser alocada a cada agente. Se todos os agentes são homogêneos em termos de suas capacidades, qualquer tarefa pode ser alocada a qualquer agente. Entretanto, em muitos dos casos mais triviais, agentes tem capacidades muito diferentes. No caso em que os agentes são realmente autônomos, alocação de tarefas envolve negociação entre agentes, possivelmente utilizando as técnicas descritas nos capítulos anteriores.
- Compartilhamento de resultados: compartilhamento de resultados envolve agentes compartilhando informações relevantes a seus sub-problemas. Estas informações podem ser compartilhadas de forma pro-ativa (agente envia a informação a outro agente sem que ela fosse solicitada apenas por acreditar que o outro agente pode se interessar pela informação), ou de forma reativa (agente

envia a informação a outro agente como resposta a uma requisição).

-

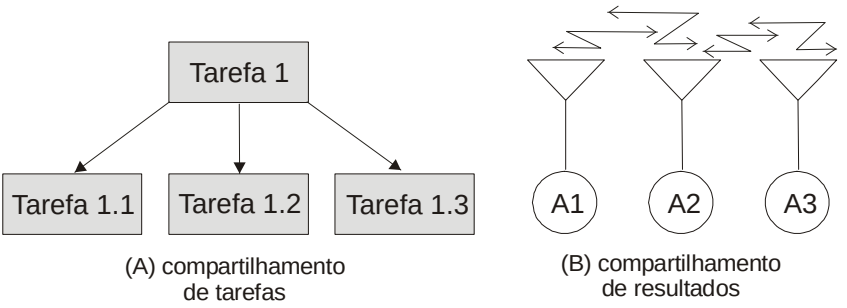

Figura 12:- (a) comtartilhamento de tarefas (b) compartilhamento de resultados[33]

3.4.3 Compartilhamento de tarefas usando contract-net

O contract-net (CNET) é um protocolo de alto nível desenvolvido por Smith[33][34][35][36], usado para alcançar uma cooperação eficiente através do compartilhamento de tarefas em redes de resolvedores de problemas que podem se comunicar. A Figura 13 mostra um diagrama simplificado do funcionamento do contract-net.

O nó que gerou as tarefas torna pública a informação de que existem tarefas a ser executadas através de um anuncio e se torna o gerenciador deste processo durante sua duração. Na ausência de informações sobre a capacidade dos outros nós da rede, o gerente é forçado a fazer um broadcast a todos os nós da rede. Se o gerente possui o conhecimento de quais nós da rede podem ser possíveis candidatos a execução da tarefa, ele pode enviar o anuncio apenas a esses possíveis candidatos. Finalmente, se o gerente conhece um agente específico que é o melhor candidato a executar a tarefa, ele pode fazer um anuncio ponto a ponto.

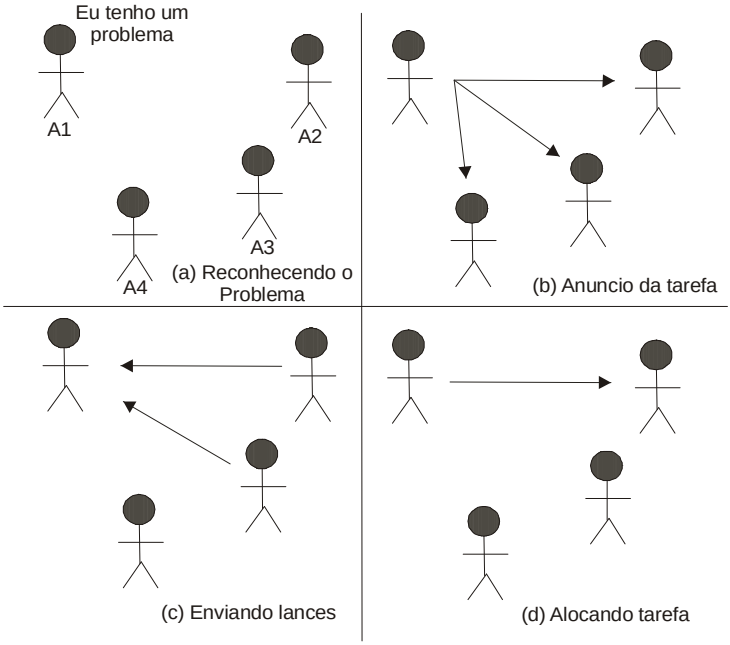

Figura 13: Protocolo contract-net (CNET)[31]

Os nós da rede escutam o anuncio feito pelo gerente e avaliam se os recursos que estão disponíveis são suficientes para a execução da tarefa. Se o nó chegar a conclusão que ele pode desempenhar esta tarefa ele faz um lance. O lance exprime a capacidade que o nó tem de executar a tarefa. O gerente poderá receber vários lances resultantes de um único anuncio. Baseado nos lances recebido, o gerente decide qual dos nós é o mais adequado para executar a tarefa. A seleção é comunicada ao agente escolhido que assume a responsabilidade pela execução da tarefa. Depois que a tarefa é executada, o executor envia um relatório ao gerente.

O processamento do conteúdo das mensagens trocadas durante as fazer que envolvem o contract-net podem ser as seguintes:

- Processamento do anuncio da tarefa: O receptor do anuncio da tarefa deve decidir se ele próprio é um candidato a execução da tarefa. Ele pode fazer isto comparando os requisitos necessários para a execução da tarefa, que estão contidos no anuncio, com a capacidade que ele tem disponível para executa-la. Se a capacidade for suficiente ele envia um lança ao nó gerente.

- Processando o lance: os lances que chegam ao gerente são armazenados até que um limite de tempo estipulado por ele tenha passado. Após este tempo, ele avalia os lances e decide qual dos nós que enviaram lances é mais capacitado. O nó escolhido é informado sobre sua escolha através de uma mensagem de decisão.
- Processando a decisão: Nós que enviaram os lances e não foram escolhidos simplesmente descartam as informações sobre a tarefa. O nó escolhido da continuidade a execução da tarefa.
- Processando requisição e envio de informação: estas mensagens são as mais simples de tratar. Uma mensagem de requisição apenas gera uma mensagem contendo a informação requisitada que deve ser enviada a quem requisitou.

A FIPA (*Foundation for Intelligent Physical Agents*) padronizou o protocolo de interação Contract-net[31] para ser usado por desenvolvedores de plataformas multi-agentes. O protocolo descrito neste documento é ligeiramente diferente do original. Esta diferença se dá pela inclusão dos atos comunicativos de rejeição e confirmação como pode ser observado no diagrama UML que descreve o funcionamento deste protocolo na Figura 14.

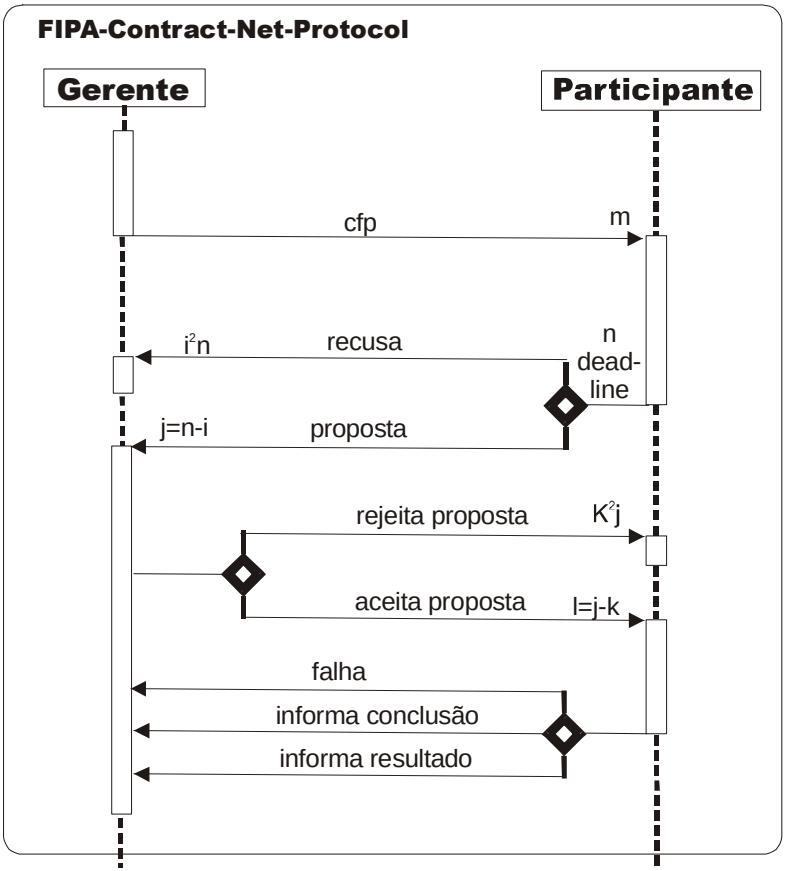

Figura 14: Protocolo de interação Contract-Net[31]

O protocolo se inicia quando um determinado agente, chamado aqui gerente, identifica uma tarefa e deseja que um ou mais agentes, chamados aqui de participantes, executem esta tarefa. O gerente anuncia aos participantes, ou potenciais contratados, a tarefa a ser executada. Cada participante avalia as condições necessárias para a execução da tarefa (que na maior parte dos casos esta expressa em termos de preço, mas também pode conter outros valores como tempo necessário para execução, capacidade de recursos,...). Dependendo da avaliação feita pelo participante ele pode retornar ao gerente uma proposta para a execução da tarefa ou rejeitar o anuncio. Neste ponto pode ser observada uma das diferenças entre as especificações da FIPA e o protocolo original. As

mensagens de recusa são descartadas pelo gerente e as propostas avaliadas. Os agentes que tiveram a proposta aceita pelo gerente são agora contratados e passam a ser responsáveis pela execução da tarefa e os agentes que tiveram suas propostas recusadas são informados da recusa. Durante a execução da tarefa o agente contratado pode mandar uma mensagem de falha caso alguma tenha ocorrido durante a neste período. Caso contrário, o agente, ao final da execução, envia uma mensagem informado e um relatório com os resultados ao gerente.

Por ser um protocolo relativamente simples, ele se tornou o protocolo mais implementado e estudado pelos desenvolvedores em *frameworks* para resolução de problemas.

3.4.4 Compartilhamento de resultados

No compartilhamento de resultados, a resolução de problemas prossegue através do envio, pelo agente, do resultado obtido da execução da tarefa. Este é o resultado de um pequeno problema que vai progressivamente sendo refinado para fazer parte da solução de um problema maior. Durfee[37], sugere que resolvedores de problemas podem melhorar sua performance no compartilhamento de resultados da seguinte maneira:

- Confidencialidade: Soluções derivadas independentes podem ser cruzadas, ressaltando erros e incrementado a confidencialidade da solução como um todo.
- Integridade: agentes podem compartilhar sua visão local para adquirir uma visão global.
- Precisão: agentes podem compartilhar suas soluções para que a precisão do problema como um todo seja melhor.
- Conveniência: Mesmo que um agente possa resolver ele mesmo o problema, pelo compartilhamento de resultados, a solução do problema pode ser adquirida mais rapidamente.

4 NEGOCIAÇÃO NOS GRIDS DE ARMAZENAMENTO E ANALISE DE DADOS

Dentre os módulos que fazem parte do *grid* de agentes para gerência de redes dois merecem destaque neste trabalho, pois são neles que serão aplicadas as técnicas de negociação propostas. Neste capítulo serão discutidos os aspectos de funcionamento destes módulos, que também são descritos por Assunção [3][4] como *grid* coletor e classificador e *grid* de processamento, bem como as formas de aplicação da negociação entre os agentes que compõe estes *grids* visando a distribuição das tarefas e o balanceamento de carga de trabalho resultante desta aplicação.

A plataforma utilizada para a implementação deste trabalho foi o JADE [39] . Uma descrição mais detalhada desta plataforma será feita no item 4.3 deste capítulo.

Com esta breve introdução passaremos agora para a descrição dos grids onde serão aplicadas as técnicas de negociação.

4.1 O GRID DE COLETA E ARMAZENAMENTO

A responsabilidade dos agentes que compõe o grid coletor é coletar dados de equipamentos através da utilização de protocolos de gerência, ferramentas ou até linhas de comando do sistema operacional. Tal habilidade de coletar dados depende do conhecimento que o agente possui.

Após a coleta de dados, estes são classificados e organizados de uma forma padronizada para serem enviados, através de algum protocolo de transporte, para o grid armazenador que vai efetuar o armazenamento dos dados. A tarefa de armazenamento é executada por agentes que possuem o conhecimento para que isso possa ser feito.

Este é um dos cenários onde pretendemos utilizar técnicas de negociação. A tarefa de armazenamento pode ser realizada por vários agentes o que possibilita o uso da técnica. A Figura 15 mostra um esquema onde um agente coletor, que tem dados relevantes para serem armazenados negocia com alguns agentes armazenadores.

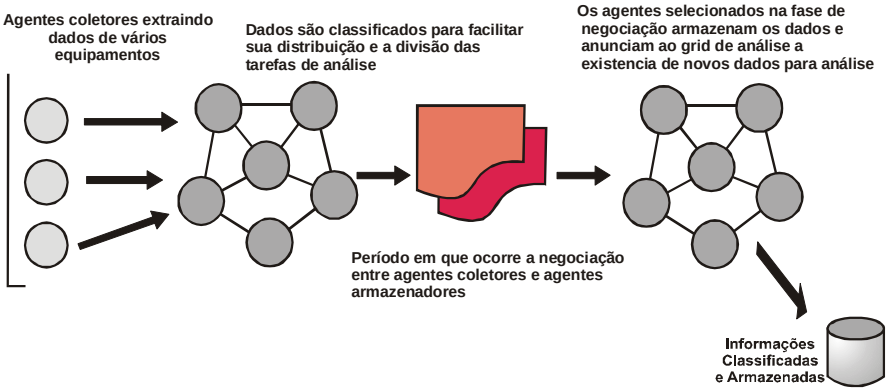

Figura 15: Negociação na fase de coleta/armazenamento.

Os agentes armazenadores disponíveis para a execução da tarefa estão registrados em um diretório. As informações registradas neste diretório contêm o nome do agente sua localização e pode conter outras informações como conhecimento do agente e recursos disponíveis para utilização. O agente coletor faz uma busca neste diretório a procura de agentes que satisfaçam as suas necessidades. Sabendo quais os agentes que podem executar a tarefa desejada começa o processo de negociação. Este processo e os algoritmos utilizados neste trabalho serão descritos posteriormente na seção 4.3.

Após a fase de negociação, o agente coletor possui a informação de qual o melhor agente armazenador que executara a tarefa de armazenamento, assim ele pode enviar os dados que devem ser armazenados a este agente e solicitar a execução da tarefa.

4.2 O GRID DE ANÁLISE

O *grid* de análise é responsável pelo processamento dos dados para a geração dos relatórios de gerência que são enviados para a interface. Este *grid* possui agentes que se organizam em forma de *containers* [38] localizados em vários pontos da rede. Tais agentes também possuem o seu registro no serviço de diretórios e podem ser localizados por outros agentes quando necessário.

Este módulo pode ser considerado o mais importante do *grid* de agentes para gerência de redes, pois é nele que, a partir dos dados

coletados, são extraídas informações relevantes sobre o desempenho da rede que está sendo monitorada. É também neste cenário que está a maior carga de trabalho do *grid* e a aplicação das técnicas de negociação se faz necessária.

A Figura 16 mostra um esquema do processo de negociação que ocorre no *grid* de processamento quando da necessidade de analise de dados.

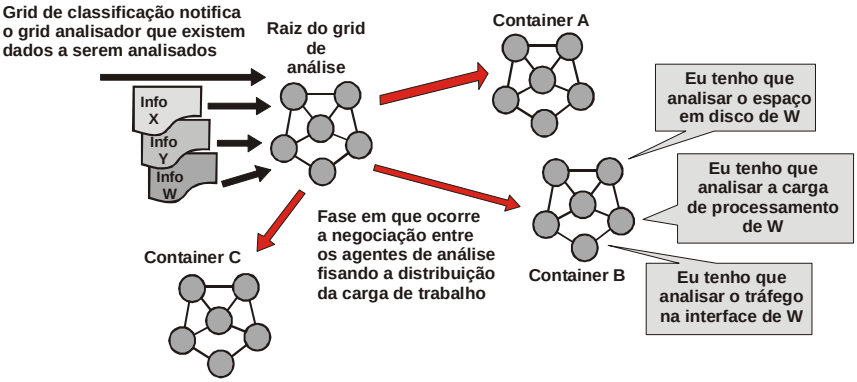

Figura 16: Negociação no grid de análise

Sempre que dados relevantes são coletados o *grid* de análise 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.

4.3 IMPLEMENTAÇÃO DO PROCESSO DE NEGOCIAÇÃO

A plataforma de agentes escolhida para a implementação dos algoritmos de negociação nos cenários descritos no capítulo anterior foi a plataforma JADE [39].

JADE é uma plataforma que facilita o desenvolvimento de sistemas multi-agentes. Esta plataforma inclui [40]:

- Um ambiente que é carregado em um determinado host quando um agente é criado e passa a existir neste host.
- Uma biblioteca de classes java que os programadores podem/devem usar para desenvolver seus agentes
- Um conjunto de ferramentas gráficas que permitem que administradores possam monitorar a atividade dos agentes que estão em execução.

Cada instância em execução em um ambiente JADE é chamada de container e pode conter vários agentes. Um conjunto ativo de containeres é chamado de plataforma. Cada plataforma deve conter um container principal e todos os outros containeres são registrados no principal assim que entram em atividade. A figura 17 mostra uma tela gráfica que representa uma plataforma e os containeres que a compõe.

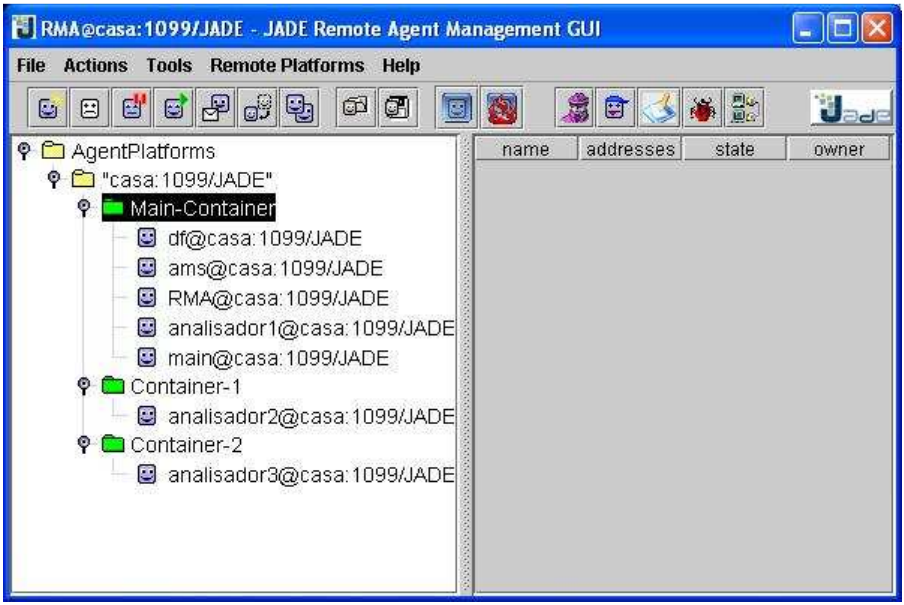

Figura 17: Interface de administração de agentes JADE.

Como podemos observar na figura 17, existem dois agentes que são iniciados no container principal chamados ams e df. O agente ams é responsável por prover um serviço de nomeação à plataforma e representa a autoridade da plataforma. Podemos iniciar/finalizar agentes e containeres na plataforma fazendo uma solicitação ao ams. O agente df provê o serviço de páginas amarelas, onde agentes podem encontrar outros agentes que executem tarefas que possam auxiliá-lo. A figura 18 mostra um exemplo de duas plataformas de agentes genéricos e sua organização.

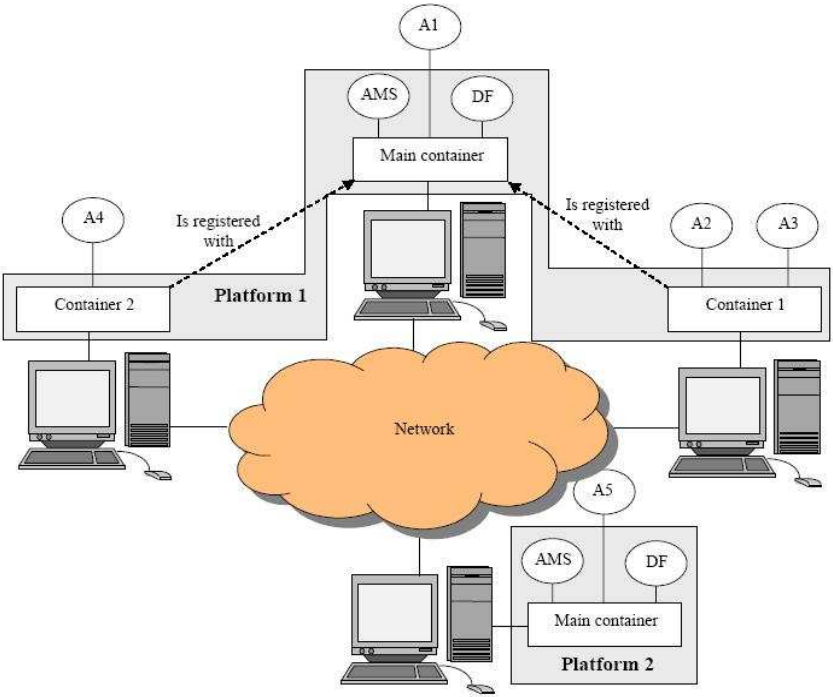

Figura 18: Plataformas e Containers de Agentes [39]

Esta plataforma foi utilizada para implementação dos agentes que atuaram nos cenários de armazenamento a análise de dados. Para o cenário de armazenamento, foram criadas duas classes chamadas de coletor, que representa agentes coletores de dados de gerência, e armazenadores, que são responsáveis por armazenar os dados coletados. No cenário de análise foram implementadas duas classes chamadas de

main, que é um agente responsável por distribuir a tarefa de análise de dados aos agentes analisadores, esses responsáveis pela execução da tarefa de análise. A interação entre estes agentes se dá através dos algoritmos de negociação *contract-net, english auction* e *dutch auction* descritos neste trabalho.

Os ambientes de simulação continham um agente que distribuía as tarefas e dez agentes responsáveis por executá-las.

Todos os agentes envolvidos no processo de negociação foram implementados para representar exatamente o estado em que se encontram no grid de agentes para gerência de redes no exato momento em que necessitam distribuir as tarefas de armazenamento (*grid* de armazenamento) e análise (**grid** de análise).

Para que se pudesse coletar dados que fossem avaliados com clareza, o agente que inicia a negociação já conhece o conjunto de agentes que podem executar as tarefas, dispensando assim a utilização das páginas amarelas para localizá-los. Desta maneira podemos obter dados dos mesmos agentes para todos os algoritmos de negociação, facilitando a análise do desempenho e do balanceamento de carga gerados destes processos de negociação, bem como o comportamento de cada algoritmo nos dois cenários.

No cenário de armazenamento o processo de negociação para os três algoritmos levava em conta a capacidade disponível em disco no host onde o agente estava executando. Já no cenário de análise a escolha do agente para executar a tarefa levava em consideração a capacidade de processamento no instante da negociação. Na próxima sessão serão apresentados os resultados obtidos dos experimentos feitos com estes agentes.

4.4 RESULTADOS OBTIDOS

Todos os algoritmos foram executados nas mesmas condições de ambiente para que se pudesse fazer uma comparação dos resultados de sua aplicação nos cenários de armazenamento e análise.

Todos os hosts estavam interconectados por uma rede local lan de 10Mb com configurações variadas de recursos de armazenamento e processamento. As configurações de cada host serão descritas quando apresentados os resultados das coletas de dados gerados da aplicação dos protocolos de negociação

4.4.1 Cenário de armazenamento

A quantidade inicial de espaço para armazenamento era o mesmo para cada agente no início do processo de simulação dos três métodos de negociação. A figura 19 mostra um gráfico que ilustra o espaço livre, em mega bytes, para armazenamento nos host em que cada agente estava executando. A tarefa distribuída consistia do armazenamento de um arquivo com aproximadamente 5MB de dados. Todos os agentes armazenadores no recebimento da cfp analisavam o espaço livre em disco e se este espaço fosse maior que a quantidade de dados a ser armazenada enviavam uma proposta para o agente coletor, caso contrário, enviavam uma recusa. O calculo do "preço" enviado junto com a proposta tinha como base a quantidade de espaço livre para armazenamento.

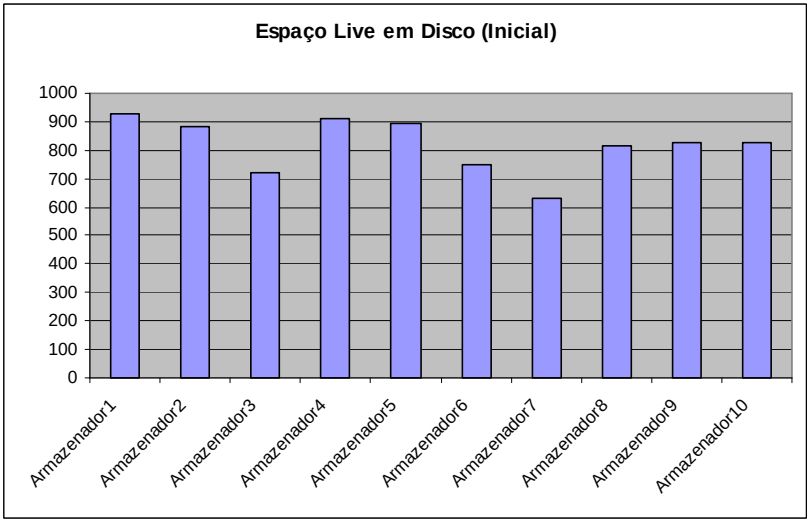

Figura 19: Espaço de armazenamento disponível no início da simulação

A aplicação do contract-net no cenário de armazenamento gerou uma distribuição de tarefas que pode ser observada no gráfico da figura 20.

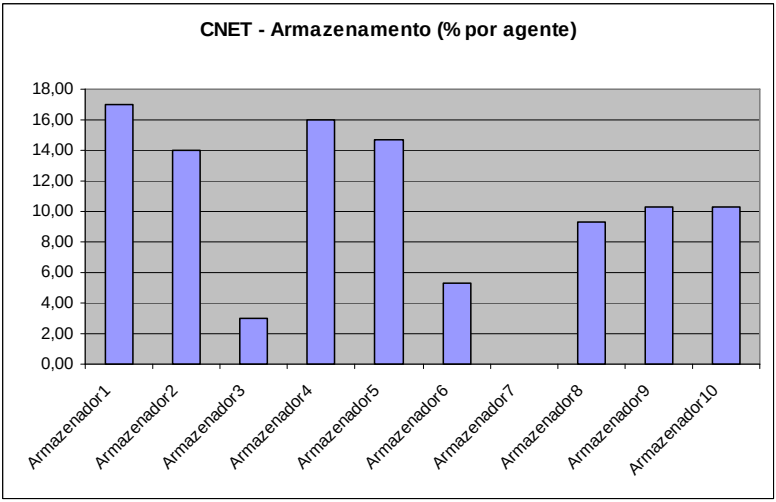

Figura 20: Resultado do balanceamento de carga gerado pelo *contract-net*

Podemos comparar a distribuição das tarefas com a quantidade de espaço disponível para cada agente no gráfico da figura 19 e observar que a carga de trabalho foi distribuída para os agentes que possuíam mais capacidade de armazenamento. Durante o decorrer do processo a capacidadede armazenamento de todos os agentes tende a ser a mesma. Isso faz com que a distribuição se torne uniforme a medida que o tempo passa. Esta situação acontece porque a tendência, com o passar do tempo, é que todos os agentes possuam a mesma capacidade de armazenamento como pode ser observado no gráfico da figura 21.

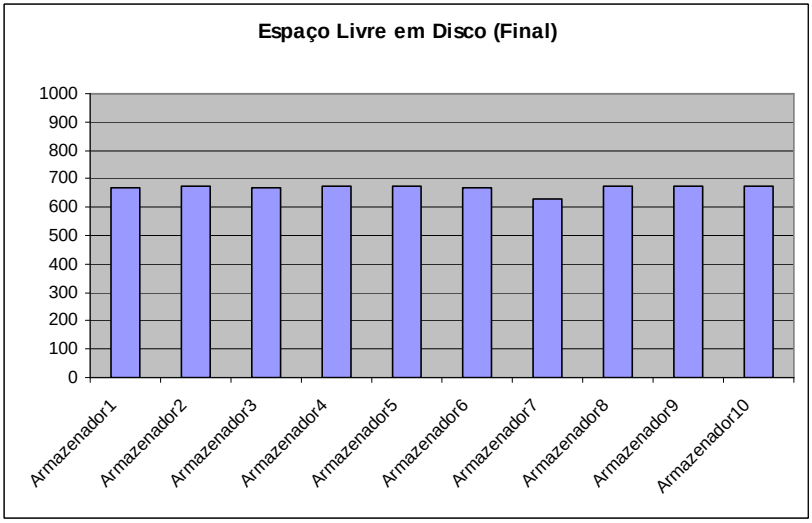

Figura 21: Espaço livre de armazenamento no final da simulação.

Os algoritmos *ducht auction* e *english auction* atingiram resultados muito parecidos com os resultados obtidos da aplicação do contract-net. A distribuição das tarefas, como mostram os gráficos das figuras 22 e 23, foi similar para os dois algoritmos bem como para o contract-net. Isso acontece principalmente porque o critério para a escolha do agente mais habilitado é o mesmo para todos os algoritmos, ou seja, a capacidade de armazenamento.

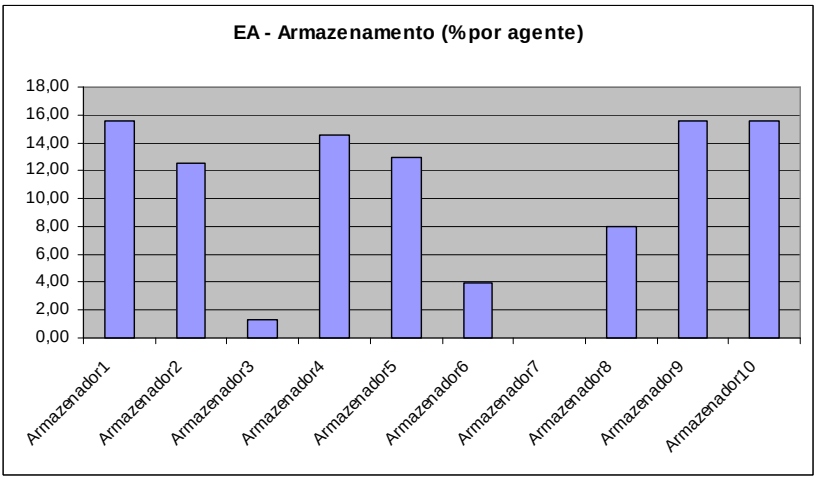

Figura 22: Resultado do balanceamento de carga gerado pelo *english auction*

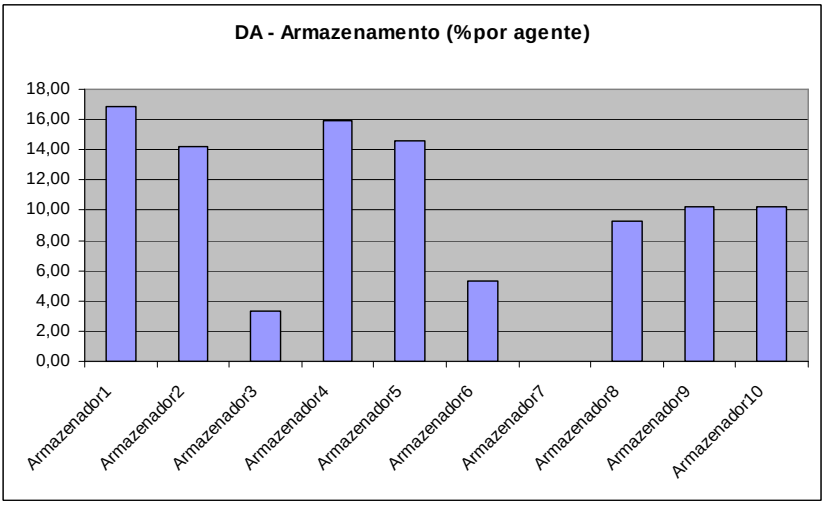

Figura 23: Resultado do balanceamento de carga gerado pelo *dutch auction*

A tendência de igualar a quantidade de espaço livre de armazenamento para todos os agentes também foi observada na aplicação do english auction e do dutch auction.

Durante a execução dos três algoritmos propostos nenhum dos agentes armazenadores recusou chamadas de proposta. Em todas as

interações dos três algoritmos, os agentes possuíam espaço livre para armazenamento. Nenhuma anormalidade foi observada em relação a agentes deixarem de exeistir ou executar durante o processo.

4.4.2 Cenário de análise

A configuração de processador e memória de cada host em que os agentes analisadores estavam executando pode ser vista na tabela abaixo.

Agente	Processador	Memória
Analisador1	Pentium 4 1.2GHz	256Mb
Analisador2	AMD Duron 1.0Ghz	128Mb
Analisador3	Pentium 4 1.2GHz	256Mb
Analisador4	AMD Duron 1.0Ghz	128Mb
Analisador5	Pentium 4 1.2GHz	256Mb
Analisador6	Pentium 4 1.2GHz	256Mb
Analisador7	Pentium 3 900MHz	128Mb
Analisador8	Pentium 4 1.2GHz	256Mb
Analisador9	Celeron 1.6GHz	512Mb
Analisador10	Celeron 1.6GHz	512Mb

Tabela 1: Ambiente de execução dos agentes analisadores

Estas informações eram usadas pelos agentes analisadores para calcular o preço para as propostas de execução das tarefas de análise. Os agentes que possuíam capacidade de processamento inferior a 30% da capacidade total de processamento enviavam ao agente principal uma recusa em resposta a chamada de proposta.

O gráfico da figura 24 mostra o resultado da distribuição das tarefas de análise utilizando o contract-net.

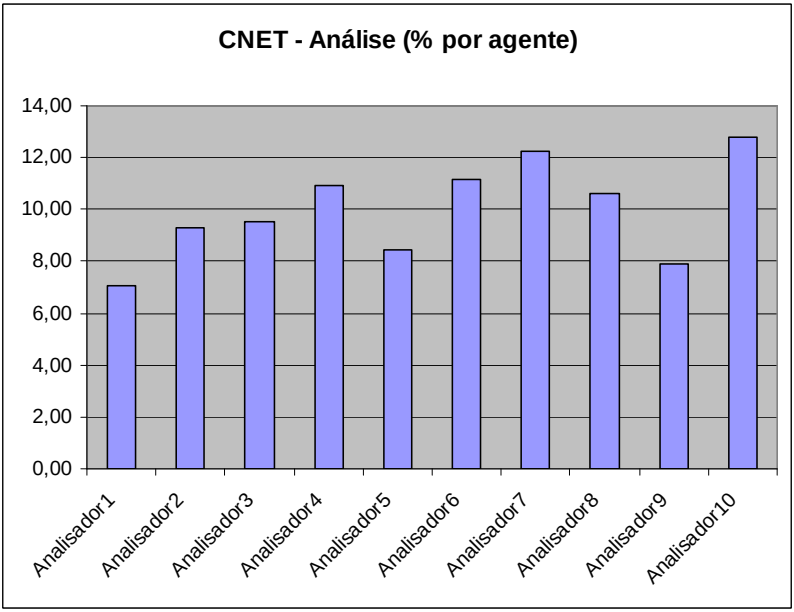

Figura 24: Resultado do balanceamento de carga gerado pelo *contract-net.*

A principal informação utilizada para avaliar qual o melhor agente para executar a tarefa foi a capacidade livre de processamento. Ao contrario da quantidade de armazenamento livre utilizada no cenário de armazenamento, a capacidade livre de processamento pode ter uma variação mais brusca num espaço menor de tempo. Isso faz com que a escolha dos agentes para a execução da tarefa de análise não seja tão obvia quanto a escolha para a tarefa de armazenamento. Em um determinado instante, o agente armazenador pode ter uma capacidade de processamento aceitável para poder participar do processo de negociação, mas num próximo momento esta capacidade não é suficiente para executar a tarefa. No caso de insuficiência de capacidade de processamento, o agente envia uma recusa. O gráfico da figura 25 mostra o número de propostas e recusas enviadas pelos agentes durante o processo de negociação utilizando o contract-net.

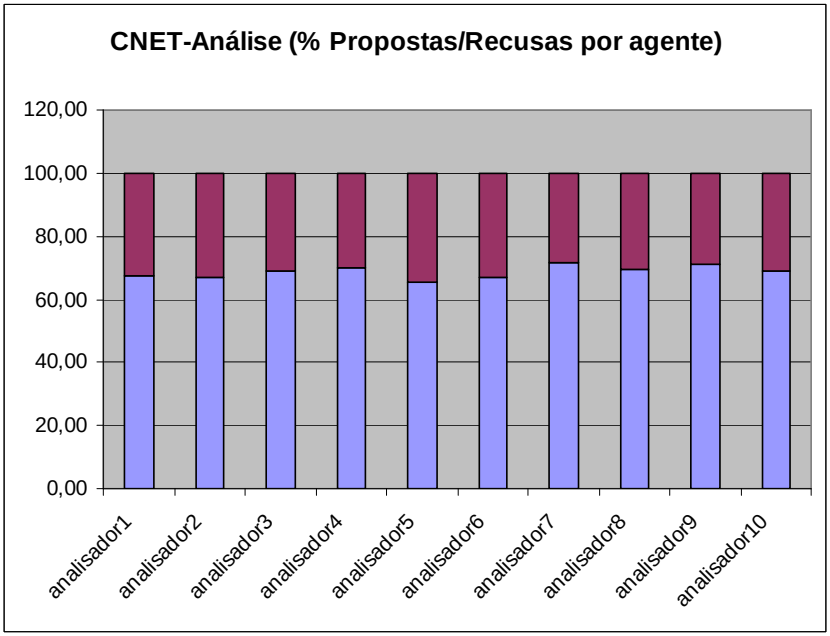

Figura 25: Propostas/Recusas enviada pelos agentes analisadores durante a execução do contract-net.

O critério que era usado por todos os agentes definia que se a capacidade de processamento fosse menos do que 30% do total uma recusa seria enviada em resposta a cfp. Podemos observar que o percentual de recusas é praticamente o mesmo para todos os agentes.

O algoritmo english auction apresentará o resultado da distribuição das tarefas mostrados nas figuras 26. Podemos observar que o balanceamento da carga de trabalho foi maior do que na utilização do contract-net. Isso ocorreu devido a característica diferenciada entre os dois métodos de negociação.

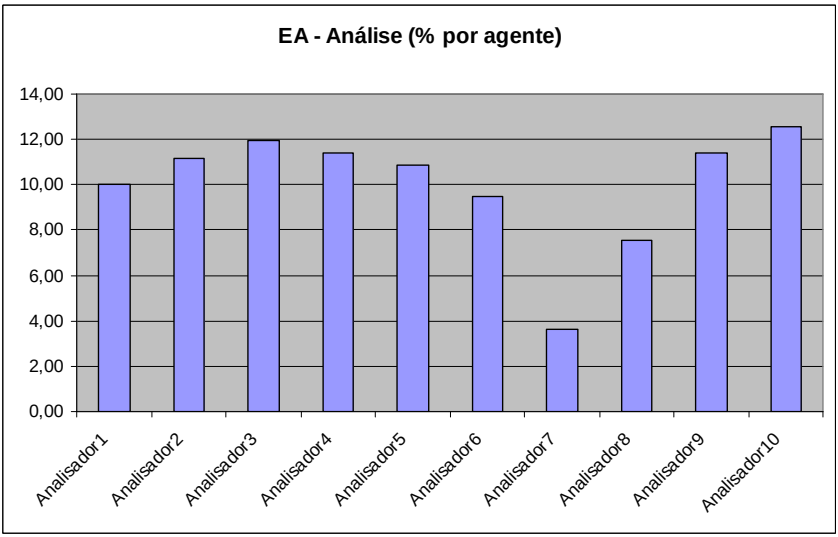

Figura 26: Balanceamento de carga gerado *pelo english auction.*

O contract-net é um processo de negociação conhecido como *one-shot*, ou seja, em apenas uma rodada o algoritmo seleciona o agente para executar a tarefa. Os algoritmos de leilão, em especial o english auction e o dutch auction tem uma particularidade. Se nenhuma das propostas enviadas pelos participantes da negociação satisfaz o agente que está distribuindo as tarefas, a negociação é encerrada sem que a tarefa seja delegada e outra rodada de negociação recomeça.

No gráfico da figura 27 apresentamos o número de propostas e recusas enviadas por cada agente analisador.

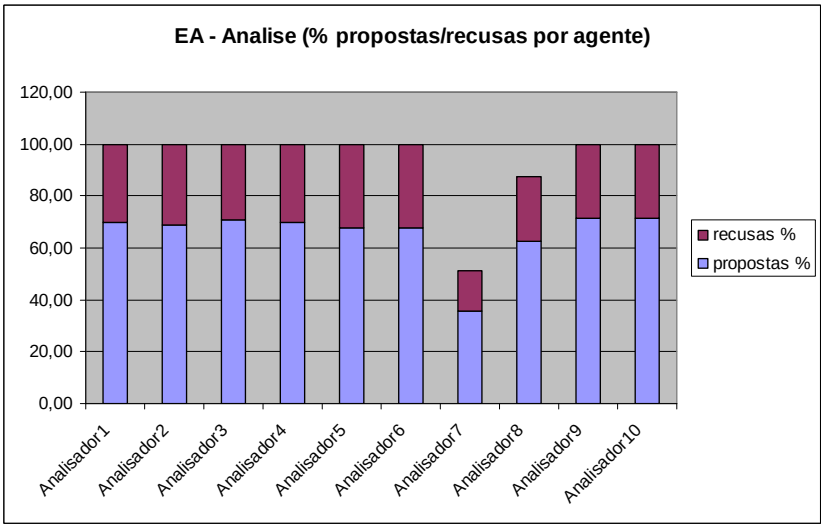

Figura 27: Propostas/Recusas enviadas pelos agentes analisadores durante a execução do *english auction*

Neste gráfico podemos observar uma particularidade. Os agentes analisadores 7 e 8 interagiram menos que os outros agentes. Isso se deve ao fato de que estes agentes foram encerrados de forma anormal antes do período de simulação terminar. Estes agentes foram encerrados no mesmo instante. O agente 7 não foi mais executado e o agente 8 foi reiniciado algum tempo depois e voltou a fazer parte do processo de negociação. O resultado desta falha pode ser observado também na distribuição das tarefas na figura 26. A falha destes agentes não comprometeu o funcionamento do sistema como um todo. Os outros agentes continuaram executando as tarefas que lhes eram delegadas.

O resultado semelhante foi observado na execução do dutch auction que é similar ao english auction em suas características. O gráfico da figura 28 mostra a distribuição das tarefas.

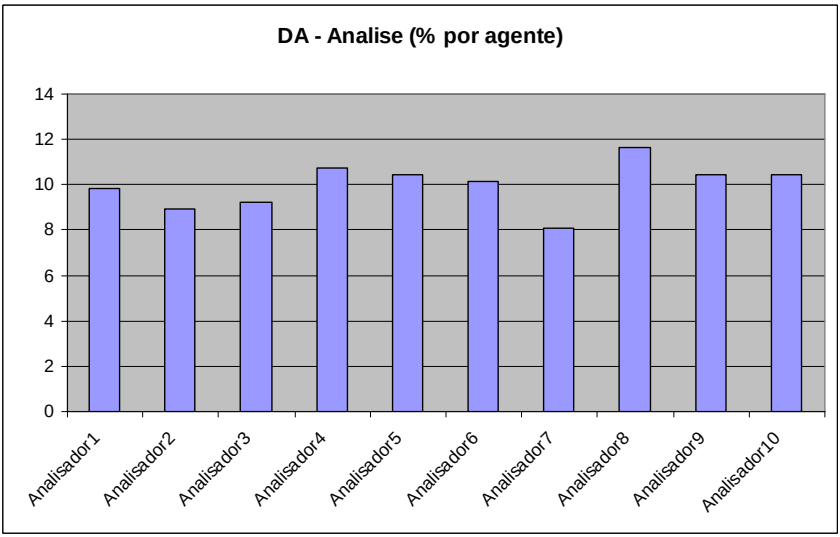

Figura 28: Balanceamento de carga gerado pelo *dutch auction*.

O gráfico da figura 29 nos mostra o número de propostas e recusas enviadas pelos agentes analisadores durante o processo, onde podemos observar que todos os agentes mandaram um número muito parecido de propostas e recusas.

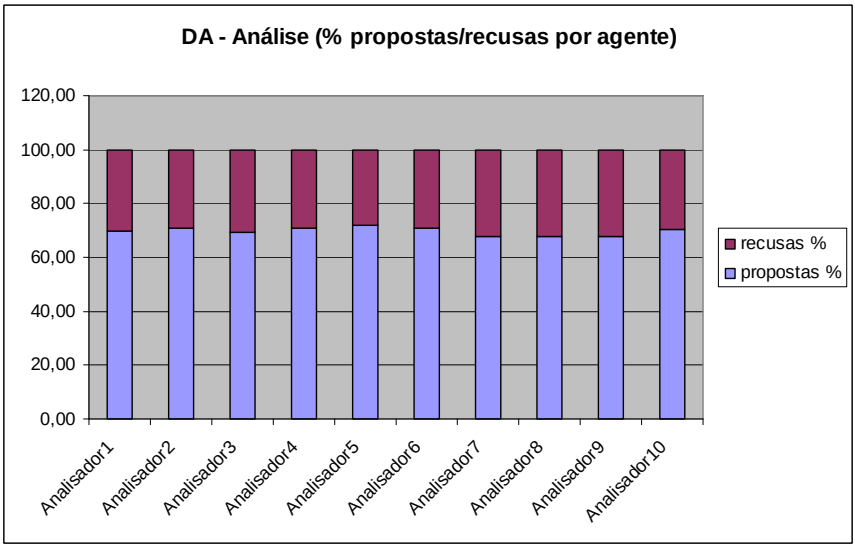

Figura 29: Propostas/Recusas enviadas pelos agentes analisadores durante a execução do *dutch auction*.

A principal observação feita no cenário de análise é que, apesar dos agentes terem enviados mais ou menos o mesmo número de propostas e recusas durante a utilização dos três algoritmos, os algoritmos de leilão tiveram uma melhor performance em relação a distribuição da carga de trabalho. Com isso podemos passar a próximo capítulo para as conclusões finais do trabalho.

5 CONCLUSÃO

Neste capítulo descreveremos as conclusões deste trabalho, quais os objetivos atingidos, dificuldades encontradas e trabalhos futuros em relação a interação entre agentes para distribuição de tarefas no grid de agentes para gerência de redes.

5.1 OBJETIVOS ALCANÇADOS

Todos os objetivos deste trabalho foram alcançados. A implementação do processo de negociação ocorreu de forma satisfatória sem que maiores problemas tivessem acontecido. Foi escolhida a implementação de um ambiente de teste controlado para que os três algoritmos propostos pudessem ser executados nas mesmas condições de ambiente e recursos disponíveis para que os resultados pudessem ser comparados e uma avaliação do desempenho e comportamento destes algoritmos pudesse ser feita quando aplicados aos cenários de armazenamento e análise.

Estas análises de desempenho e comportamento foram expressas nos dados apresentados no capitulo 4 onde podemos ver que o desempenho dos três algoritmos foi muito parecido quando aplicados ao cenário de armazenamento. Isto ocorre porque este cenário tem características que influenciam diretamente no comportamento dos algoritmos de negociação quando utilizados para distribuição de tarefas. Dentre estas características podemos destacar como a mais importante a capacidade de armazenamento que cada host possui. Podemos observar que a capacidade de armazenamento é uma característica que não tem uma mudança brusca num curto espaço de tempo. Assim, como esta característica era a mais relevante para a escolha dos agentes, os três algoritmos apresentaram resultados muito parecidos e a aplicação de qualquer um deles neste cenário pode ser satisfatória.

Já no cenário de análise os algoritmos tiveram resultados diferentes. Neste cenário os algoritmos de leilão obtiveram melhor desempenho na distribuição de tarefas. Isto ocorre porque a principal característica deste cenário, e que foi utilizada como critério para a distribuição das tarefas, é a capacidade de processamento. Este recurso pode ter uma mudança mais brusca em um curto espaço de tempo. Um host que tem 90% de capacidade de processamento livre, num curto

espaço de tempo pode passar a possuir 10% de capacidade livre. Esta variação é tratada melhor pelos algoritmos de leilão, pois se no instante em que a negociação está sendo feita nenhum dos *hosts* possui capacidade de processamento que seja suficiente para a execução da tarefa, o processo de negociação é encerrado e uma nova rodada de negociação é iniciada. Assim no instante que uma nova rodada é iniciada a característica de quantidade de processamento dos *hosts* envolvidos pode ter se alterado e um agente hospedado em um *host* com maior capacidade de processamento pode ser escolhido. Já o contract-net em uma rodada define qual melhor agente (*host*) para a execução da tarefa e a delega para este agente.

Podemos observar também através dos resultados que a inatividade de um ou mais agentes que executam tarefas não compromete o desempenho do processo como um todo. Agentes que por algum motivo deixem de existir e participar do processo podem ser criados novamente e voltar a executar suas atividades de forma normal.

5.2 PROBLEMAS ENCONTRADOS

O principal problema encontrado no decorrer do trabalho foi a mudança de plataforma de agentes que foi utilizada para a implementação da simulação de negociação. A plataforma Agentlight [9] foi abandonada e a plataforma JADE [39][40] foi adotada para o desenvolvimento deste trabalho. Isto ocorreu porque a plataforma Agentlight deixou de ser de domínio publico e não mais poderia ser utilizada para os fins deste trabalho. A plataforma JADE foi escolhida por ser uma plataforma completa que fornecia toda a infra-estrutura para o desenvolvimento deste trabalho.

5.3 TRABALHOS FUTUROS

Os resultados obtidos nos mostram que algoritmos de negociação têm um bom desempenho no balanceamento da carga de trabalho entre os agentes. Como trabalhos futuros podemos citar algumas atividades para a utilização efetiva destes algoritmos de negociação no grid de agentes para gerência de redes.

Os agentes que fazem parte do grid poderiam ter em sua base de conhecimento todos os algoritmos de negociação propostos neste trabalho. Com a implementação das facilidades de pesquisa em páginas amarelas que a plataforma JADE oferece, o agente que necessita iniciar um processo de negociação pode fazer uma pesquisa do diretório onde os agentes estão registrados, pesquisa esta que possui como critérios a atividade que os agentes estão aptos a executar e o protocolo de negociação que será utilizado. Tal pesquisa definiria o conjunto de agente que faria parte da negociação.

Neste trabalho as características dos ambientes dos cenários de armazenamento e análise que foram utilizados como critério para a escolha dos agentes aptos a executar as tarefas eram relativamente simples. Isto facilitou a análise e comparação do desempenho e comportamento dos protocolos de negociação, da distribuição das tarefas e balanceamento de carga de trabalho. Sabemos que a execução de tais tarefas dependem de um conjunto de recursos que interagem entre si, tais como, memória, capacidade em disco rígido, capacidade de processamento entre outros. Como trabalhos futuros podemos sugerir a melhoria dos critérios de escolha dos agentes para a execução das tarefas bem como um cálculo mais preciso do “preço” cobrado pelos agentes para a execução da mesma.

Podemos observar com este trabalho que muito ainda pode ser feito em relação a distribuição de tarefas e balanceamento de carga de trabalho através da utilização de protocolos de interação entre agentes. Outros protocolos de interação podem ser experimentados e, com o avanço cada vez maior e mais veloz das pesquisas nesta área, obter resultados ainda melhores.

BIBLIOGRAFIA

[1] NATIONAL COORDINATION OFFICE FOR INFORMATION TECHNOLOGY RESEARCH AND DEVELOPMENT. High Performance Computing And Communications: Foundation For America's Information Future. 1996. The Annual Supplement to the President's Budget. Disponível Online na Internet: http://www.ccic.gov/pubs/blue96.

[2] KOCH, F. L.; WESTPHALL, C. B. Decentralized Network Management using Distributed Artificial Intelligence. Journal of Network and Systems Management. Plenum Publishing Corporation. Meddletown, USA, v.9, n.4, p.291-313, dez. 2001.

[3] 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.

[4] 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).

[5] COABS – CONTROL OF AGENT BASED SYSTEMS. Sítio com informações sobre o projeto. Disponível em: < http://coabs.globalinfotek.com>. Acesso em: Ago. 2002.

[6] WILLMOTT, S. N., DALE, J., BURG, B, CHARLTON, C., O'BRIEN, P.: Agentcities: A Worldwide Open Agent Network, Agentlink News. 8 Nov. (2001) 13-15, http://www.AgentLink.org/newsletter/8/AL-8.pdf.

[7] LUCK, M., McBURNEY, P., PREIST, C.: Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent Based Computing), AgentLink, 2003.

[8] Foundation for Intelligent Physical Agents. Organização que tem por objetivo estabelecer especificações que garantam a interoperabilidade entre diferentes plataformas de agentes.

[9] AGENTLIGHT – PLATFORM FOR LIGHTWEIGHT AGENTS. Sítio com informações sobre o projeto. Disponível em: <http://www.agentlight.org>.

[10] WOOLDRIDGE, M. J., JENNINGS, N. R. (eds.): Intelligent Agents, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995.

[11] WOOLDRIDGE, M. J. An Introduction to Multiagent Systems. Baffins Lane,Chichester, West Sussex, England: John Wiley & Sons Ltd., 2001. 348p. ISBN0-471-49691-X.

[12] ROCHA, M. A, WESTPHALL, C. B. Pro-active Management of Computer Networks Using Artificial Intelligence Agents and Techniques. IFIP/IEEE International Symposium on Integrated Network Management, 5., 1997, San Diego, USA. Proceedings... Fifth IFIP/IEEE International Symposium on Integrated Network Management, p.610-621, 1997.

[13] FULLER, W. Network management using expert diagnostics. International Journal of Network Management, vol.9, num.4, 1999, p.199-208.

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

[15] CIRNE W. Grids Computacionais Arquiteturas Tecnologias e Aplicações. Anais do Terceiro Workshop em Sistemas Computacionais de Alto Desempenho. 2002.

[16] FOSTER, I.;KESSELMAN, K. The Grid: Blueprint for a new Computing Infrastructure. Morgan Kaufmann Publishers. San Francisco, CA. 1999.

[17] The Globus Project. Disponível em <http://www.globus.org>. Acesso em 12 out. 2003.

[18] JOHNSTON, W.E.; GANNON, D.; NITZBERG, B. Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. Proceedings 8th IEEE Symposium on High Performance Distributed Computing, IEEE Press. 1999.

[19] LASZEWSKI, G.; PIEPER, W. G.; WAGSTROM, P. Gestalt os the Grid. 2002.

[20] Legion A WorldWide Virtual Computer. Disponível em <http://legion.virginia.edu/>. Acesso em 12 out. 2003.

[21] MANOLA, F. Characterizing Computer-Related Grid Concepts, 1999. Disponível em: <http://www.objs.com/agility/tech-reports/9903-grid-report-fm.html>. Acesso em: 12 out. 2003.

[22] TeraGrid Project. Disponível em <http://www.teragrid.org>. Acesso em 12 out. 2003.

[23] NWANA, H. S. Software Agents: An Overview. Knowledge Engineering Review, v.11, n.3, p.1-40, set. 1996.

[24] MOREAU, L. Agents for the Grid: A Comparison for Web Services (Part 1: the transport layer). Editores: BAL, H. E.; LOHR, K-P; REINEFELD, A. IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2002), 2., 2002, Berlin, Germany. Proceedings... CCGRID Germany, p.220-228, May 2002.

[25] AgentScape. Disponível em: <http://agentscape.org/research/agentscape.html>. Acesso em 12 out. 2003.

[26] OVERINDER, B. J.; WIJNGAARDS, B. J. Et al. Multi-Agent Support for Internet-Sccale Grid Management. Proccedings of the AISB'02 symposioum on AI and Grid Computing. 2002.

[27] ROSENCSCHEIN, J. S.; ZLOTKIN, G. Rules of Encouter: Designing Conventions for Automated Negotiation Among Computers, Cambridge, MA. 1994.

[28] SANDHOLM, T. Distributed Racional Decision Making. In MultiagentSystem (ed. Wieb), pp. 201-258. MIT Press, Cambridge, MA.

[29] FIPA. English Auction Interaction Protocol Specification. Disponível em: < http://www.fipa.org/specs/fipa00031/>. Acesso em: 10 set. 2003.

[30] FIPA. Dutch Auction Interaction Protocol Specification. Disponível em: < http://www.fipa.org/specs/fipa00032/>. Acesso em: 10 set. 2003.

[31] FIPA. Contract Net Interaction Protocol Specification. Disponível em: < http://www.fipa.org/specs/fipa00029/>. Acesso em: 10 set. 2003.

[32] WOOLDRIDGE, M.; Coherent Social Action. In Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94). Amsterdam. 1994.

[33] SMITH, R. G.; DAVIS, R. Framework for Cooperation in Distributed Problem Solving. IEEE Transaction on Systems, Man and Cybernetics. 1980.

[34] SMITH, R. G., The CONTRACT NET: a formalism for the control of distributed problem solving. In Proceedings of the 5th International Joint Conference on Artificial Inteligence, Cambridge, MA. 1997.

[35] SMITH, R. G., The Contract Net Protocol. IEEE Transaction of Computers, C29(12), 1980.

[36] SMITH, R. G., A Framework for distributed Problem Solving. UMI Researsh Press, 1980.

[37] DURFEE, E. H., Distributed Problem Solving and Planning. In Multiagent Systems, MIT Press, Cambridge, MA. 1999.

[38] MULLER, J. P. The Design of Autonomous Agents: A Layered Approach. Lecture Notes in Artificial Intelligence, vol.1177, Springer-Verlag, 1996.

[39] BELLIFEMINE, F. JADE: what it is and what it is next, invited speech at ETAPS 2001, Workshop on Models and Methods of Analysis for Agent Based Systems (MMAABS), Genova, April 2001.

[40] BELLIFEMINE, F. , POGGI, A., RIMASSA G., Developing multi-agent systems with JADE, eventh International Workshop on Agent Theories, Architectures, and Languages (ATAL-2000), Boston, MA, 2000.

[41] JUHASZ, Z., PRASENJIT, P., Scalability Analysis of the Contract Net Protocol, (2002), CCGRID, 346-347.

[42] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, A. Gray e N. Fiddian (2004) “Agent-based formation of virtual organisations” Int. J. Knowledge Based Systems 17 (2-4) 103-111.

[43] M. Dinkloh e J. Nimis (2003) “A Tool for Integrated Design and Implementation of Conversations in Multiagent Systems” , 1st International Workshop on Programming Multiagent Systems languages, frameworks, techniques and tools, 2nd International Conference on Autonomous Agents & Multiagent Systems, Melbourne, Austrália.