Uso de Memória Compartilhada Distribuída na Coordenação de Multi-Agentes

Thiago Gonzaga[1], Cristiana Bentes[1], Maria Clicia Stelling de Castro[2], Ricardo Farias[3] e Ana Cristina Bicharra Garcia[4]

Depto. Engenharia de Sistemas- FEN - UERJ [1]
thrpg@ele.puc-rio.br
cris@eng.uerj.br
Brasil

Instituto de Matemática e Estatística - CTC - UERJ [2]
clicia@ime.uerj.br
Brasil

Prog. Engenharia Sistemas COPPE - UFRJ [3]
rfarias@cos.ufrj.br
Brasil

Depto. Ciência da Computação - IC - UFF[4]
bicharra@ic.uff.br
Brasil

Resumo

Em Sistemas multi-agentes, os agentes podem se comunicar de forma direta através de trocas de mensagens ou de forma indireta através de um blackboard. A comunicação por blackboard é mais simples, porém sua implementação em uma arquitetura distribuída não é eficiente. Este fato ocorre porque normalmente se utiliza um único elemento de processamento como mantenedor do blackboard. Neste trabalho, estamos propondo o uso de mecanismos de memória compartilhada distribuída para permitir a implementação de blackboards em sistemas distribuídos de forma eficiente. Nossa idéia é distribuir os dados do blackboard pelos elementos de processamento e usar um sub-sistema que realize a troca de mensagens entre eles de forma totalmente transparente aos agentes. Implementamos esta proposta num sistema multi-agentes baseado em leis sociais (Tri-coord), utilizando o sistema software DSM TreadMarks como sub-sistema de distribuição de dados e gerenciamento das mensagens. Criamos um simulador de conflitos para gerar situações onde a comunicação entre os agentes é necessária e mostramos que o sub-sistema de memória compartilhada distribuída é capaz de manter os dados do blackboard coerentes..

1. Introdução

São encontradas diferentes definições para agentes computacionais na literatura. Podemos dizer que os agentes assumem algum comportamento autônomo, de acordo com o modelo computacional desenvolvido para atingir os objetivos do usuário. Isto é, eles são dotados de tarefas bem definidas e ações limitadas para agir em nome do usuário. Além disso, são capazes de monitorar e intervir no ambiente a que pertencem de maneira racional, através da opção pela melhor ação em determinado momento.

Os agentes computacionais são capazes de interagir com outros agentes e de participarem de sociedades em algumas situações específicas. Uma sociedade é composta por dois ou mais agentes ou resolvedores de problemas. Eles interagem através de um ambiente comum para solucionar um determinado problema. Esta sociedade é denominada de Sistema Multi-Agentes (SMA), que pode ser definida como: "uma rede de resolvedores de problemas que trabalham juntos para resolver problemas que estão além das suas capacidades individuais" [Green97].

Para que os problemas sejam solucionados, de forma coerente, os agentes do SMA devem ser capazes de se comunicar, coordenar suas atividades e negociar os conflitos que possam surgir. A coordenação das atividades entre os agentes é fundamental num SMA. Sem coordenação o grupo de agentes poderia degenerar-se rapidamente, o que geraria uma coleção de indivíduos com um comportamento caótico. Podemos definir a coordenação das atividades dos agentes computacionais como o compartilhamento de conhecimento. Isso inclui tanto a compreensão mútua do conhecimento, como a difusão desse conhecimento.

A coordenação entre os agentes computacionais pode ser realizada através de um conjunto de técnicas de negociação ou métodos de resolução de conflitos. Os conflitos entre os agentes podem variar desde uma simples contenção de recursos até computações complexas, gerando diversos tipos de conflitos.

Os conflitos de sistemas multi-agentes podem ser resolvidos através de: i) monitoramento do ambiente [Vivacqua97]; ii) negociação direta entre os agentes [Kraus97]; iii) intervenção humana; ou iv) leis ambientais ou sociais [Placca98]. Dessa forma, para que uma sociedade de agentes atue e coopere para atingir um determinado objetivo, é necessário que uma arquitetura seja definida. Ela deve possibilitar a interação entre esses agentes. Isso porque são nas interações que ocorrem as trocas de conhecimento, dos objetivos, planos ou escolhas, dado que cada agente tem apenas uma visão parcial do sistema.

A comunicação entre os agentes pode ser realizada de forma direta ou indireta. Na comunicação direta os agentes se conhecem a priori. Assim, trocam informações diretamente entre si. Na comunicação indireta, os agentes não têm nenhum conhecimento prévio. Deste modo, a comunicação ocorre através de uma estrutura de dados compartilhada.

Numa sociedade onde a comunicação é direta, os agentes devem conhecer as identificações dos outros agentes e enviar mensagens explícitas. Este procedimento requer a definição de um protocolo de comunicação, além de tornar a formulação do agente bem mais complexa. A comunicação indireta, normalmente, é realizada através de uma estrutura denominada *blackboard* [Carver94]. A comunicação é mais simples porque é implícita, através de leituras e escritas a uma estrutura que toda a sociedade tem acesso.

Embora a comunicação indireta seja considerada a forma de comunicação mais simples entre agentes, sua implementação é totalmente dependente da arquitetura do sistema computacional utilizado pela sociedade.

Nas arquiteturas com memória compartilhada fisicamente entre os elementos de processamento (nós), a implementação da comunicação indireta é simples e direta. O *blackboard* é implementado na memória compartilhada permitindo que todos os elementos de processamento tenham acesso a ele. Estas arquiteturas, são caras e possuem, em geral, um número reduzido de elementos de processamento.

No entanto, as arquiteturas com memória distribuída são menos custosas e não têm limitação em relação a quantidade de elementos de processamento. Elas podem conter um número elevado de processadores, permitindo, assim, a criação de sociedades maiores e mais complexas com um bom potencial de desempenho. Porém, a implementação de um *blackboard* em uma arquitetura distribuída não é óbvia. A solução mais adotada tem sido eleger um elemento de processamento como mantenedor do *blackboard*. As leituras e escritas são transformadas em mensagens explícitas para o mantenedor. Essa solução, em geral, não é satisfatória. Isso porque ela reduz de forma substancial a facilidade de comunicação fornecida pela utlização do *blackboard*. Além disso, o mantenedor pode criar uma perda de desempenho no sistema. Como toda a coordenação tem que passar pelo mantenedor do *blackboard*, ele pode se tornar um gargalo, pois pode ficar sobrecarregado, e como conseqüência diminuir a escalabilidade do sistema.

Neste trabalho, propomos o uso de mecanismos de memória compartilhada distribuída (*Distributed Shared Memory* - DSM), amplamente difundidos na área de programação paralela, para permitir o uso de *blackboards* em sistemas distribuídos de forma eficiente. Nossa idéia é distribuir os dados do *blackboard* de forma totalmente transparente aos agentes, do mesmo modo que um sistema software DSM faz. Os agentes acessam os dados como se eles fossem compartilhados. Há um sub-sistema responsável por distribuir os dados pelos nós de processamento do sistema distribuído e por transformar os acessos a esses dados em trocas de mensagens pela rede. Dessa forma, estamos propondo a implementação de um *blackboard* distribuído e escalável com um baixo custo.

Implementamos esta proposta num sistema multi-agentes baseado em leis sociais (Tri-coord), utilizando o sistema software DSM TreadMarks como sub-sistema de distribuição de dados e gerenciamento das mensagens entre os agentes. Neste estudo preliminar, estamos interessados em mostrar apenas como os mecanismos de memória compartilhada distribuída podem ser úteis para retirar do agente a responsabilidade de gerenciar as mensagens relativas às leituras e escritas no *blackboard*. Utilizamos um simulador de conflitos para gerar situações onde a comunicação entre os agentes é necessária e mostramos que o sub-sistema de memória compartilhada distribuída é capaz de realizar as trocas de mensagens de forma transparente ao agente.

Este trabalho está organizado da seguinte forma. Na próxima seção abordamos sistemas multi-agentes. Na Seção 3 descrevemos o modelo de resolução de conflitos Tri-coord. A Seção 4 resume as características de TreadMarks, o software DSM escolhido para nossa implementação, e a Seção 5 aborda a implementação do modelo Tri-coord baseado em TreadMarks. Nossas considerações finais estão apresentadas na Seção 6.

2. Sistemas Multi-Agentes

Um sistema multi-agente é considerado uma sociedade de agentes.

As sociedades de agentes podem ser classificadas de três maneiras diferentes de acordo com: i) o tipo de agente; ii) o número de agentes; e iii) suas regras de comportamente [Oliveira96].

A classificação segundo o tipo de agente divide as sociedades em: homogêneas (todos os agentes são indênticos) e heterogêneas (possuem agentes diferentes). A classificação segundo o número de agentes divide as sociedades em: fechadas (há um número fixo e único de agentes) e abertas (o número de agentes pode variar, podendo ser incluídos ou excluídos novos agentes). A classificação segundo regras de comportamento divide as sociedades em: baseadas em leis (existem regras que determinam o comportamento dos agentes) e sem-lei (não há regras para reger os agentes).

2.1 Interação

Independente do tipo de sociedade, o grupo social formado pelos agentes realiza operações coletivas chamadas de interação.

A interação entre os agentes é necessária para que um agente possa ter o conhecimento do comportamente e atividades realizadas pelos outros agentes e para que todos possam conhecer o objetivo global, levando a ações de cooperação.

A cooperação pode ser classificada, em quatro tipos, de acordo com a relação de interdependência existente: horizontal, em árvore, recursiva e híbrida [Zhang92].

Na cooperação horizontal, os agentes não dependem de nenhum outro agente para a resolução dos seus problemas. No entanto, o uso de informação proveniente de um outro agente, pode aumentar o grau de confiança que o agente tem sobre suas soluções. Na cooperação em árvore, os agentes dependem de outros agentes para resolver seus próprios problemas. Na cooperação recursiva, vários agentes dependem uns dos outros para resolver os seus problemas. Já a cooperação híbrida envolve a cooperação horinzontal inserida em cooperação recursiva ou em árvore.

2.2 Resolução de Conflitos

Como os agentes de uma sociedade cooperativa possuem conhecimento e métodos de resolução distintos, podem existir diferentes situações de conflito [Oliveira93a].

Os conflitos podem ser positivos ou negativos. Conflitos positivos ocorrem quando existem vários agentes capazes de executar uma tarefa ou quando agentes possuem resultados diferentes, porém complementares. Conflitos negativos ocorrem quando não existe um agente capaz de executar uma tarefa ou quando agentes possuem resultados diferentes, mas contraditórios ou inconsistentes.

Todas essas situações de conflito devem ser tratadas por meio de negociações.

2.3 Negociação

A negociação entre dois ou mais agentes se destina ao estabelecimento de um acordo sobre a efetivação de uma dada forma de cooperação. Com a negociação, define-se quais tarefas serão desenvolvidas e atribuídas aos agentes. Este processo implica na comunicação entre os agentes, visando à coordenação de suas atividades. A coordenação entre os agentes é essencial para a resolução de problemas distribuídos e enfoca principalmente a utilização de recursos.

2.4 Comunicação

Para que uma sociedade de agentes possa interagir, resolver os seus conflitos e negociar é preciso que seja definida uma arquitetura que possibilite a comunicação entre os agentes. Durante uma comunicação ocorrem trocas de conhecimentos, objetivos, planos ou escolhas.

A comunicação pode ser realizada de forma direta ou indireta. Na comunicação direta os agentes trocam mensagens diretamente entre si. A comunicação indireta é implícita através de leituras e escritas a uma estrutura de dados compartilhada, denominada *blackboard*.

Arquitetura *Blackboard*

A estrutura de dados *blackboard* foi inicialmente proposta no sistema de reconhecimento de fala HEARSAY-II[Lee80]. Um *blackboard* é uma arquitetura que permite a integração de módulos ou programas individuais em uma aplicação única e integrada [Hayes-Roth95]. Ele é um repositório de dados central e compartilhado que mantém as informações disponíveis para leitura e escrita. As informações podem ser usadas cooperativamente por um grupo de agentes. Não existe comunicação direta entre os agentes. Somente o *blackboard* contém o estado corrente do problema a ser resolvido.

Troca de Mensagens

A troca de informação entre os agentes é realizada através do envio e recebimento de mensagens. Elas que possuem um formato comum e bem definido, necessariamente perceptível por todos os agentes

3. O Modelo Tri-Coord

Tri-coord é um modelo de auxílio à resolução de conflitos em ambientes fechados, com mútiplos agentes. Isto é, em ambientes onde leis ou regras de interação, comportamento e atuação estão bem definidas no ambiente.

O modelo proposto é baseado na aplicação de Leis Sociais e foi inspirado na teoria do Contrato Social de Jean Jaques Rosseau, sendo denominado Modelo Tri-Coord. Isso porque, ele possui Tripla Coordenação, onde o comportamento dos agentes é implicitamente controlado pelo ambiente.

Este modelo é constituído por quatro tipos de agentes: i) o agente de tarefas, atua no ambiente em nome do usuário; ii) o agente executivo, que é responsável pelo gerenciamento imediato dos conflitos e a comunicação entre os agentes, fiscaliza as eventuais

punições dos agentes que infringirem as regras dentro de sua jurisdição; iii) o agente legislativo, que é responsável pela manutenção e renovação do conjunto de leis e regras utilizadas no ambiente, além disso, está constantemente observando o ambiente para incorporar regras que atendam as necessidades de um determinado momento; e iv) o agente judiciário, que é responsável por resolver os conflitos pendentes que não puderam ser resolvidos pelo agente executivo, ele utiliza um processo de audiência onde todos os agentes envolvidos são convocados para uma comunicação síncrona com a finalidade de resolver o conflito.

A distribuição das responsabilidades para cada agente especial tem o intuito de equilibrar as tarefas pertinentes à execução, julgamento e caracterização de conflitos. Essa distribuição otimiza a resposta do ambiente de intenção. Os componentes principais do Modelo Tri-Coord e suas relações estão ilustrados na Figura. 1.

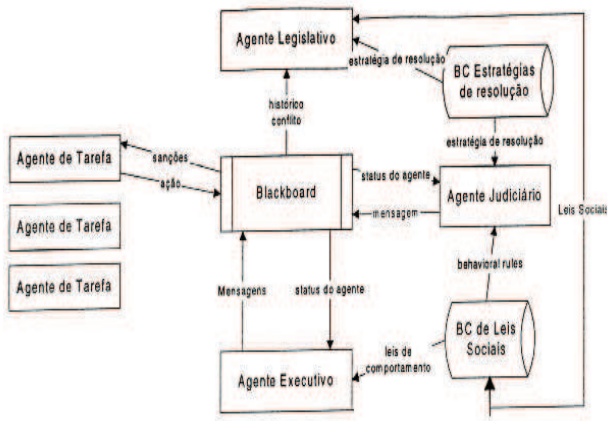

Figura 1. O Modelo Tri-Coord.

A base da comunicação em Tri-Coord é uma estrutura de *blackboard*. É através desta estrutura que informações são acessadas ou modificadas. No modelo Tri-coord, o *blackboard* garante um ambiente dinâmico de interação entre os agentes. A medida que as interações ocorrem, elas são automaticamente refletidas no *blackboard*. Esse por sua vez, é monitorado pelos agentes especiais que irão realizar as ações pertinentes.

Os agentes especiais têm função definida na resolução dos conflitos. Cada tarefa da aplicação tem sua especialidade, sendo tarefas: de sanção (Agente Executivo), julgamento de situações pendentes (Agente Judiciário) e atualização do conjunto de regras e Leis Sociais (Agente Legislativo).

O Agente Executivo lê o estado dos agentes de tarefas e, de acordo com o banco de leis sociais, dispara uma possível sanção. O Agente Judiciário lê o estado dos agentes e tarefas e intervém nas situações pré-estabelecidas. A intervenção é realizada baseada na decisão de um banco de leis sociais e um banco de estratégias de resolução de conflitos. Finalmente, o Agente Legislativo recebe o histórico dos conflitos ocorridos e atualiza o banco de estratégia de resolução de conflitos.

4. Sistemas de Memória Compartilhada Distribuída

Sistemas de memória compartilhada distribuída unem a facilidade do modelo de programação com memória compartilhada, em que todos os processos têm acesso a um espaço de endereçamento único, à escalabilidade de ambientes distribuídos [Li88]. Nesses sistemas, a memória compartilhada é implementada por mecanismos de hardware e/ou software que transformam, de modo transparente ao usuário, os acessos às memórias remotas em trocas de mensagens pela rede.

Sistemas de memória compartilhada distribuída implementados totalmente em software, também, chamados de sistemas software DSM, têm se mostrado uma alternativa de baixo custo para o suporte à memória compartilhada. O grande atrativo desses sistemas está na possibilidade de aproveitar os benefícios de uma arquitetura simples e largamente utilizadas como *clusters* de PCs. Um *cluster* de PCs pode ser facilmente formado conectando-se computadores pessoais. Essa arquitetura permite que o desempenho do sistema aumente à medida que mais máquinas forem agregadas ao *cluster*. Além disso, outra facilidade é poder ser facilmente atualizado seguindo as novas tendências em tecnologia.

4.1 O Sistema TreadMarks

O sistemas escolhido para nossa implementação foi o sistema software DSM TreadMarks [Amza96]. Isso porque temos acesso ao seu código fonte e ele é amplamente difundido na comunidade acadêmica.

TreadMarks [Keleher94] é um sistema de memória compartilhada distribuída desenvolvido em software. Sua implementação é no nível do usuário e utiliza as bibliotecas padrão do Unix para a criação de processos remotos, para a comunicação entre processos e para o gerenciamento de memória. A sua unidade de coerência é a página, sendo a coerência dos dados mantida com uso do protocolo de invalidações. Este protocolo é implementado através da propagação de notificações de escrita em operações de sincronização.

A interface de programação oferecida por TreadMarks provê dois tipos de primitivas de sincronização: *lock/unlock* e barreira.

Outras características de TreadMarks são fornecer suporte a múltiplos escritores, através dos mecanismos de *twining* e *diffing, e* adotar o modelo de consistência *Lazy Release Consistency* (LRC). Com o modelo LRC envio e a aplicação dos *diffs* são atrasados até o primeiro acesso a uma página que foi invalidada.

Inicialmente, todas as páginas são protegidas contra escrita. Quando há uma tentativa de modificação numa página, é gerada uma falha de acesso. O TreadMarks intercepta esta falha, faz uma cópia da página (*twin*) e a libera para escrita. Quando for necessária a propagação das modificações feitas localmente, TreadMarks faz uma comparação entre o *twin* gerado e a versão modificada, e cria um *diff* contendo todas as modificações.

Quando ocorre uma falha num acesso à página é necessário buscar um conjunto de *diffs* para torná-la válida, havendo então a comunicação com um ou mais nós. Os *diffs* recebidos são aplicados á página para se obter uma versão coerente.

5. Estudo de Caso: O Ambiente Tri-Coord/DSM

Nossa proposta neste trabalho é utilizar o sistema Treadmarks, para implementar o *blackboard* do modelo Tri-coord de forma distribuída e transparente ao usuário. As mensagens utilizadas para o compartilhamento do *blackboard* deixam de ser controladas pelos agentes e passam a ser administradas pelo software DSM. Dessa forma, cada agente, então, acessa o *blackboard* como se fosse uma área local. Treadmarks se responsabiliza por distribuir o *blackboard* pelas memórias dos nós de processamento e coordenar a comunicação. Nossa idéia é abordar as seguintes questões:

- Desempenho: o gargalo no desempenho é reduzido. Isso porque as informações não estão mais limitadas a um elemento de processamento e a uma única área de memória;
- Modelo de Programação: é muito mais simples escrever um sistema de agentes em que a comunicação se baseia em leituras e escritas numa área compartilhada de dados.

Toda a responsabilidade pelas mensagens, integridade e replicação dos dados é realizada pelo software DSM. Treadmarks facilita a programação deste tipo de sistema, porque torna a atuação dos agentes mais simples.

Como o principal foco do nosso trabalho é o estudo da comunicação entre os agentes, para avaliar os benefícios de se manter o *blackboard* distribuído por TreadMarks, implementamos um simulador de geração de conflitos entre os agentes. O simulador gera situações de conflitos de modo que a coordenação baseada em leis sociais tenha os conflitos resolvidos pelos agentes especiais (Executivo e Judiciário). No simulador de conflitos implementamos um sistema simplificado baseado na arquitetura Tri-coord. Ele tem as suas funcionalidades e os agentes concluem tarefas simples e objetivas, sem capacidade de aprender. Porém, com plena capacidade de se coordenar de forma coerente.

A idéia básica do simulador de conflitos é de que os agentes computacionais têm como tarefa se moverem dentro de um plano bidimensional. Os agentes são objetos que se movem de um ponto inicial até um ponto final. Ao se inserir mais de um agente neste ambiente, existe a possibilidade dos agentes se encontrarem neste plano em uma determinada coordenada *(x,y)* num determinado instante de tempo como ilustra a Figura 2.

Um conflito é definido como o encontro de dois agentes em uma mesma coordenada no mesmo instante de tempo. Sempre que um conflito é gerado os dois agentes são postos em um estado de espera, até que este conflito seja resolvido pelos agentes Executivo e Judiciário. Estes agentes definem que um dos dois agentes continue na posição em que o conflito foi gerado, enquanto que o outro agente retorna para a posição imediatamente anterior. Após a resolução do conflito os agentes recebem novamente a condição de ativos e voltam a executar sua tarefa, que é se mover em direção ao seu ponto objetivo.

5.1 O *Blackboard* Distribuído

O *blackboard* do sistema Tri-coord/DSM contém informações a respeito do estado, posição e conflitos.

As estruturas de dados compartilhadas são: Tabela de Posições; Histórico de Posições; Lista de *Status*; Lista de Conflitos e Matriz de Conflitos. Estas estruturas armazenam as posições dos agentes nos planos, o número de conflitos nos quais um agente já se envolveu e com que outro agente ele se conflitou, o estado dos agentes (ativo, inativo, suspenso e aguardando julgamento) e o estado dos conflitos em termos de resolução e julgamento.

O sistema TreadMarks se encarrega de distribuir todas essas estruturas de dados pelos elementos de processamento do sistema. Todo o acesso às estruturas contidas no *blackboard* deve ser feito através das primitivas de sincronização fornecidas pela API do software DMS TreadMarks (Tmk_lock_acquire e Tmk_lock_release).

O uso destas primitivas de sincronização garante que os dados contidos nas estruturas sejam movidos de um elemento de processamento para outro de forma totalmente transparente para o agente em questão.

5.2 O Agente de Tarefas

O agente de tarefas tem como função somente se deslocar pelo plano. O agente verifica a lista de *status* e permanece aguardando sua liberação com estado ativo. Quando a liberação ocorre, o agente de tarefas calcula a nova posição e verifica se outro agente já ocupa aquela posição. Em caso afirmativo, o agente cria um conflito

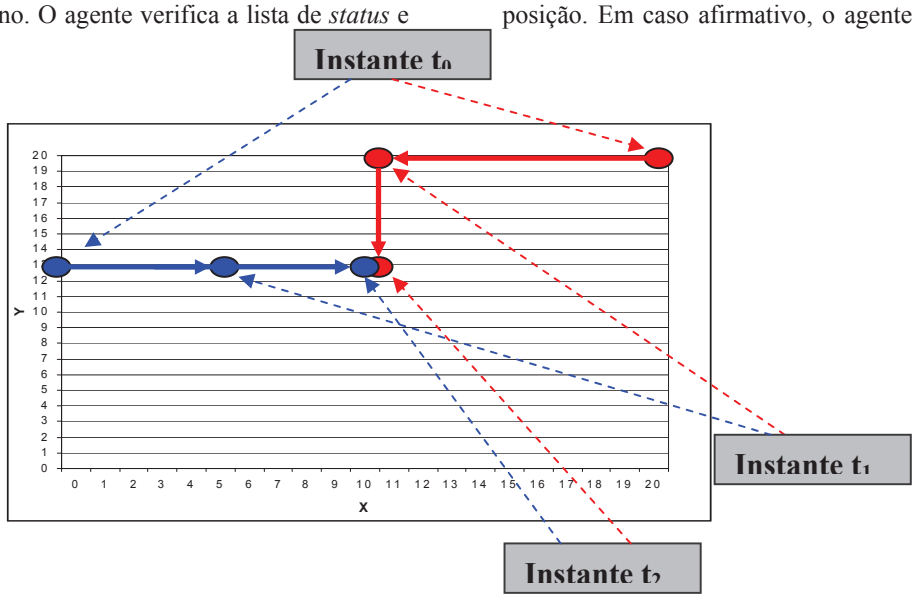

Figura 2. Exemplo de um conflito entre dois agentes no simulador

para a situação. Caso contrário, ele se move para a posição e reinicia o processo.

5.3 O Agente Judiciário

A função do agente judiciário é avaliar os conflitos e resolvê-los em última instância. Ele se mantém buscando conflitos que não tenham sido resolvidos e ainda não tenham sido julgados. Quando os encontra, o agente judiciário soluciona o conflito em favor de um dos agentes envolvidos.

5.4 O Agente Executivo

A função do agente executivo é de avaliar os conflitos em primeira instância e, quando ocorridos pela primeira vez, solucioná-los. Assim como o agente judiciário, o agente executivo, também, busca por conflitos que ainda não tenham sido resolvidos. Porém, os conflitos são resolvidos em primeira instância. Caso eles ocorram com uma certa freqüência são encaminhados para o agente judiciário.

6. Conclusões

Neste trabalho propomos o uso de mecanismos de memória compartilhada distribuída para a implementação de um *blackboard* distribuído e escalável. O uso destes mecanismos permite que sistemas multi-agentes possam ser implementados de forma simples em arquiteturas paralelas de baixo custo como *clusters* de computadores pessoais.

Nossa implementação foi baseada no sistema multi-agente Tri-coord cuja coordenação se baseia em leis sociais e a comunicação em um *blackboard*. Utilizamos o sistema software DSM TreadMarks para dar suporte ao *blackboard* distribuído. Além disso, criamos um simulador de conflitos para gerar artificialmente as necessidades de comunicação.

Os conflitos e suas soluções foram geradas no *blackboard* e a comunicação entre os elementos de processamento gerenciada por TreadMarks. Foram realizadas simulações com um agente executivo, um agente judiciário e dois agentes de tarefas. Mostramos que o sub-sistema de memória compartilhada distribuída é capaz de realizar as trocas de mensagens de forma transparente aos agentes.

Referências

[Amza96] Amza, Cristina. Treadmarks: Shared Memory Computing on Network of Workstations. IEEE Computer. Vol. 29 (2), pp: 18-28. 1996.

[Carver94] Carver, N. and Lesser, V. "A First Step Toward the Formal Analysis of Solution Quality in FA/C Distributed Interpretation Systems," Proceedings of the 13th International Workshop on Distributed Artificial Intelligence, July, 1994.

[Green97] Shaw Green, Leon Hurst, Brenda Nangle, Pádraig Cunningham, Fergal Somers, Richard Evans, Software Agents: A review. TCS-CS-1997-06, Dublin, 1997.

[Hayes-Roth95] Hayes-Roth, B. An anchitecture for adaptive intelligent systems. Artificial Intelligence Magazine. Vol. 72. 1995.

[Keleher94] Keleher, P., Dwarkadas, S., A. L. Cox, A.L. e Zwaenepoel, W. TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems, Proceedings. of the 1994 Winter Usenix Conference, January, 1994.

[Kraus97] Kraus, S. Negotiation and cooperation in multi-agent enviroments. Artificial Intelligence Magazine. Vol. 4 (1-2),1997.

[Lee80] Lee, E., Frederick, H-R., Lesser, V.R. e Raj, R.. The Hearsay-II Speech-understanding System: Integrating Knowledge to Resolve Uncertainty. Computing Surveys 12(2):213—253, 1980.

[Li88] Li, K. IVY: A Shared Virtual Memory System for Parallel Computing. Proceedings of the 1988 International Conference on Parallel Processing. Vol. 2, pp: 94-101. 1988.

[Oliveira93a] Oliveira, E.; Mouta, F.; Rocha, A. - Cooperation in a Multi-Agent Community, Proceedings 13th International Conference on Artificial Intelligence, Expert Systems and Natural Language, 1993.

[Oliveira96] Oliveira, Flávio Moreira de. Inteligência Artificial Distribuída. In: Escola Regional de Informática, 4, 1996, Canoas. Anais Sociedade Brasileira de Computação, 1996. p. 54-73.

[Placca98] Placca, José Avelino. Um modelo para interações de agentes em ambientes fechados baseado em leis sociais. Dissertação de mestrado-Universidade Federal Fluminense - UFF, 1998

[Singh98] Singh, Munindar P.; 1998; Agent Communication Languages: Rethinking the Principles"; IEEE Computer; Dec. 1998.

[Vivacqua97] Vivacqua, A. S., Garcia, A.C.B. Multiagent Active Design Documents in Group Design. Nos anais do XIV National Conference on Artificial Intelligence -Atificial Intelligence & Knowledge Management Workshop, Providence, Rhode Island,USA, 1997.

[Zhang92] C. Zhang. Cooperation Under Uncertainty in Distributed Expert Systems. Artificial Intelligence, Vol. 56, pp. 21-69, 1992.