MOOM: Um MOnitor de Objetos Móveis

Fernando Luís Dotti, Fernando Nygaard, Lúcio Mauro Duarte, Roberto Domingues Reznicek
fldotti@kriti.inf.pucrs.br, fnygaard@zaz.com.br, duarte@kriti.inf.pucrs.br, reznicek@kriti.inf.pucrs.br

Faculdade de Informática
Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681 - Prédio 16
CEP 90619-900 - Porto Alegre – RS – Brasil
Fone: +55 51 320-3611, Fax: +55 51 320-3621

Resumo:
Este artigo apresenta a arquitetura de um monitor de objetos móveis. Tal monitor foi denominado MOOM (MOnitor de Objetos Móveis ou *MObile Object Monitor*) e utiliza-se de suporte já existente, provido por plataformas de suporte à mobilidade disponíveis, como forma de obter as informações relevantes ao processo de monitoração. Estas informações são disponibilizadas ao usuário de maneira a dar suporte ao desenvolvimento de aplicações que utilizam código móvel e ao estudo do comportamento de códigos móveis, servindo de base, também, para o desenvolvimento de um esquema de contabilização de recursos consumidos por componentes móveis.

Abstract:
This paper describes an architecture for a mobile object monitor. The monitor was named MOOM (MObile Object Monitor) and it uses existent support, given by available plataforms which support code mobility, in order to obtain information that are used during monitor execution. Those information are made available to the user to give him support during development of applications based on code mobility and to help him in the study of mobile codes behaviour. It can also help in the development of a counting scheme for resources consumed by mobile components.

Palavras-chaves: Sistemas Distribuídos, Mobilidade de Código, Monitoração, Orientação a Objetos, Computação Móvel, Aplicações Móveis.

1 INTRODUÇÃO

A crescente disponibilidade de capacidade de processamento e a facilidade de interconexão entre máquinas aceleraram, nos últimos anos, a criação de técnicas de desenvolvimento e suporte de aplicações distribuídas. A maioria das aplicações criadas para execução via rede está baseada na arquitetura cliente-servidor, porém, recentemente, surgiu uma nova proposta de desenvolvimento de sistemas distribuídos envolvendo mobilidade de código. Este novo modelo baseia-se na idéia de um código, residente em um determinado nodo de uma rede, chamado neste texto de Ambiente Computacional (AC), poder, dinamicamente, mover-se através desta rede e ser executado em outro AC dentro da mesma.

Por ser a idéia da mobilidade de código algo muito recente, não encontram-se disponíveis ferramentas de suporte ao desenvolvimento de aplicações geradas a partir dessa abordagem, bem como suporte prático para o estudo do comportamento de aplicações móveis. A fim de suprir tais suportes, idealizou-se a criação de um sistema de monitoração que pudesse controlar a evolução de aplicações móveis através de uma rede. Uma proposta de sistema para realizar tal função é apresentada neste artigo.

O texto apresenta, primeiramente, uma visão geral sobre mobilidade de código através da ilustração de áreas de aplicação de códigos móveis e os novos paradigmas introduzidos pela idéia de mobilidade de código. No capítulo 3, descreve-se a motivação do trabalho aqui apresentado, justificando a realização

deste. Já no capítulo 4, é apresentado o sistema de monitoração proposto, juntamente com a sua idealização, objetivos, arquitetura, informações que serão disponibilizadas para o usuário e explicação de cada um dos módulos que formam a sua arquitetura e a relação entre eles. No quinto capítulo, é descrito o cenário de validação do monitor e, no capítulo seguinte as considerações finais do trabalho. Por fim, lista-se a bibliografia utilizada.

2 MOBILIDADE DE CÓDIGO

Os sistemas que provêm formas de mobilidade de código são geralmente chamados de Sistemas de Código Móvel (SCM) ou Mobile Code Systems (MCS). Os SCM apresentam inovações em relação aos sistemas convencionais que os caracterizam e se mostram como vantagens para a sua utilização nos modelos de redes atuais. Uma dessas vantagens é a sua concepção voltada para a operação em redes de larga escala compostas por hosts heterogêneos e por conexões por links de diferentes larguras de banda. Sua principal característica é que o programador tem controle sobre a mobilidade de seu código dentro da rede, mas, apesar disso, o processo de migração de código através da rede lhe é transparente [1].

Baseando-se na idéia de mobilidade de código, a construção de aplicações distribuídas passa a contar com outros paradigmas, além do cliente-servidor. Estes novos paradigmas são:

- **Execução Remota:** toda as operações necessárias para a completa execução de uma determinada tarefa requerida localmente são efetuadas de forma remota. Uma determinada tarefa é movida para um site remoto, executada utilizando-se dos recursos lá existentes e os seus resultados são entregues novamente para o site local;
- **Código por Demanda:** um componente local possui os recursos para efetuar uma determinada tarefa, mas não sabe como efetuá-la. Dessa forma, esse componente requer de um site remoto a forma (seqüência de operações) como ele deve se utilizar desses recursos para poder executar a tarefa a qual se propõe. Feito isso, o componente local realiza sua tarefa localmente;
- **Agente Móvel:** Utilizando-se desse paradigma, um componente móvel (agente), portador de todas as operações que deve efetuar, migra para outro site remoto e, fazendo uso dos recursos locais a esse último site, executa a(s) tarefa(s) que deve executar possuindo autonomia suficiente para migrar para outro site para completar a seqüência de tarefas, ou retornar ao site de origem, levando consigo os resultados da computação realizada [2].

As áreas de aplicação de código móvel são as mais variadas. Entre elas podemos citar a sua utilização em aplicações que realizam busca de informações distribuídas [3], podendo aumentar a eficiência desta tarefa através da migração do código até a base de informações, realizando uma consulta local em vez de uma consulta remota. A área de serviços avançados de telecomunicações [4] também pode utilizar aplicações com código móvel, a fim de fornecer mecanismos com capacidades facilitadas para reconfiguração dinâmica e customização de usuário quando oferecendo serviços tais como videoconferência, vídeo por demanda ou telemeeting. O setor de comércio eletrônico [5] apresenta-se como outra área onde o uso de código móvel pode colaborar para a obtenção mais rápida e precisa de dados altamente relevantes às negociações realizadas via rede.

A tecnologia do código móvel também pode ser usada para executar processos computacionais remotos enquanto o computador de origem está desconectado da rede, já que, ao moverem-se para um novo lugar, os códigos que implementam estes processos deixam de ter qualquer ligação com sua localização anterior. Pode, também, ajudar a superar os problemas de latência e largura de banda das redes por não realizar acessos remotas, uma vez que o código é executado localmente. Desse modo, em alguns casos, é possível diminuir-se o tráfego na rede, dado que o único trânsito na rede acontece quando o código movimenta-se do seu ponto original para o local onde será executado ou onde continuará uma execução anteriormente iniciada.

Existem hoje muitas tecnologias que dão suporte à mobilidade de código, entre as quais pode-se citar duas: o Voyager [6], [7] e o Aglets [8]. Estas duas plataformas foram utilizadas no desenvolvimento do presente projeto a fim de fornecer mobilidade às aplicações que servem de teste e como ponto de partida para a obtenção das informações que desejava-se.

3 MOTIVAÇÃO

Como já foi mencionado anteriormente, o campo da mobilidade de código é recente e, portanto, desprovido do suporte necessário para um melhor desenvolvimento de sistemas baseados em código móvel. Uma importante carência da área diz respeito ao supervisionamento da evolução da aplicação móvel através da rede em que esta está executando. É necessário que o desenvolvedor possua um ambiente de teste que apresente informações sobre o comportamento de seu código como forma de poder assegurar-se do perfeito funcionamento de sua aplicação.

Mais uma necessidade é a de suporte prático para o estudo do comportamento de aplicações móveis, permitindo a identificação de nichos de aplicações propícias ao uso de código móvel. Atualmente, o programador não tem subsídios para definir qual abordagem usar quando está desenvolvendo uma aplicação. Em alguns momentos, é necessário testar-se a validade de se usar um paradigma com mobilidade de código em vez de uma abordagem mais difundida, como a cliente-servidor. Para realizar essa comparação, seria necessário criar-se um protótipo da aplicação segundo cada uma das abordagens a fim de verificar o desempenho obtido em cada caso. Sem o devido suporte, o desenvolvedor não pode visualizar o andamento e consumo de recursos de sua aplicação baseada em mobilidade de código e, por conseguinte, não tem como certificar-se de qual opção é a mais adequada.

4 O MOOM

4.1 Concepção de um Mecanismo de Monitoração

Das carências citadas no capítulo anterior, vem a idéia da criação de um mecanismo para colaborar na gerência de ambientes de suporte a códigos móveis, monitorando seus comportamentos, auxiliando no desenvolvimento de aplicações com mobilidade de código. Através dessa ferramenta, o programador teria como testar as características móveis de seu software, acompanhando a execução de sua aplicação móvel e detectando eventuais erros, inclusive reconhecendo em que momento e lugar eles ocorrem.

Esse mecanismo atuaria como um monitor que forneceria as informações necessárias sobre os códigos móveis presentes em uma rede a partir do acompanhamento de seus comportamentos. O controle exercido por esse monitor objetivaria dar o suporte ainda inexistente aos desenvolvedores que utilizam código móvel, oferecendo um ambiente controlado para teste de suas aplicações móveis, através da disponibilização de dados relevantes quanto a mobilidade de código como trajetória, tempo gasto em comunicação, local de finalização, etc. Desse modo, esses dados poderiam ser posteriormente analisados.

O projeto aqui descrito apresenta uma arquitetura para a implementação de um monitor para ambientes de código móvel. Trata-se de um mecanismo que visa monitorar o fluxo migratório de um determinado código móvel entre diferentes AC, dando condições para visualizar o trajeto percorrido por ele.

4.2 Objetivos do Monitor

Os objetivos principais deste projeto são o de dar suporte prático ao estudo do comportamento de aplicações móveis e dar suporte ao desenvolvimento de aplicações envolvendo mobilidade de código. Para isso, estabeleceu-se um sistema para coleta de informações sobre o comportamento das aplicações

móveis. A partir das informações coletadas pode-se disponibilizar estas informações através de interfaces (gráficas ou programáveis) para o desenvolvedor.

Baseando-se na coleta de informações e na visualização desse dados, torna-se possível exercer-se uma monitoração sobre a migração de códigos móveis, possibilitando o suporte aos desenvolvedores que os utilizam. Além disso, viabiliza-se a realização de ensaios monitorados com aplicações móveis e aplicações que utilizam outras abordagens, dando ao desenvolvedor subsídios para que ele possa optar pela aplicação mais eficiente e confiável para realizar a tarefa desejada por ele.

As informações de eventos das aplicações em que se baseiam os dados acima são enviadas ao monitor segundo facilidades da plataforma de mobilidade utilizada, tal como Voyager e Aglets. Cada uma dessas duas plataformas (bem como as demais existentes) oferece alguns meios de obterem-se informações sobre os códigos móveis que estão executando sobre elas. O sistema de monitoração em construção (MOOM) é projetado de forma independente de plataforma, bastando utilizar componentes de adaptação entre monitor e a plataforma de suporte. Como pode ser visualizado na figura 1, o monitor se encontra num nível acima do hardware presente, do Sistema Operacional utilizado e de uma plataforma que dá suporte à mobilidade. Acima do monitor, encontram-se apenas os objetos que estão rodando neste ambiente.

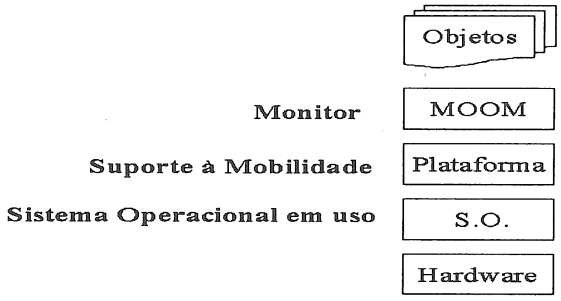

Figura 1. Níveis abaixo e acima do monitor que interagem e dão suporte ao mesmo.

4.3 Informações Disponibilizadas pelo Monitor

As informações disponibilizadas pelo monitor são:

- **Tempo total de transferência de um objeto:** Contabilização do tempo gasto no processo de migração de um objeto desde a sua saída do AC de origem até o instante de sua chegada no AC de destino;
- ***Trace* da mobilidade do objeto:** Itinerário do objeto desde a sua criação até seu término (ciclo de vida do objeto), obtido através da identificação dos AC de origem e destino de um objeto em cada migração (vide figura 2);

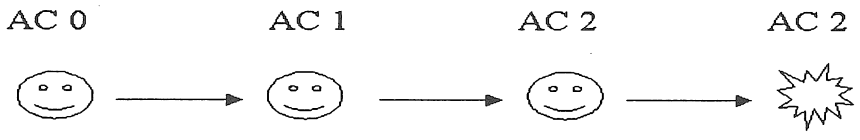

Figura 2. Ilustração de um exemplo de movimentação de objeto. No caso do exemplo acima, monitor teria condições de fornecer ao usuário o caminho percorrido pelo objeto. Ou seja, poderia dizer que o objeto foi criado em AC 0, moveu-se para AC 1 e moveu-se para AC 2, onde foi finalizado.

- **Tempo total de permanência de um objeto em um AC:** Contabilização do tempo em que um objeto permaneceu em um determinado AC, obtido a partir da identificação do instante de finalização de seu último processo migratório e do instante em que se inicia sua próxima migração;
- **Objetos pertencentes a uma aplicação:** Identificação dos objetos gerados a partir de uma aplicação;
- **Objetos existentes em um AC:** Identificação dos objetos que estão executando em cada um dos AC.

Outras informações poderão ser disponibilizadas caso sejam definidas como relevantes, tais como:

- Entrada de um objeto em um AC: Amostrar o instante em que um objeto inicia sua execução em um AC;
- Saída de um objeto de um AC: Amostrar o instante em que um objeto migra para outro AC;
- AC de origem de um objeto que migra;
- AC de destino de um objeto que migra;
- Instante de criação de um objeto;
- Instante de destruição de um objeto.

Contando com essas informações, o usuário tem um minucioso relatório de tudo o que é realizado por cada objeto móvel. Para demonstração da utilidade destas informações, tomemos o exemplo ilustrado na figura 3.

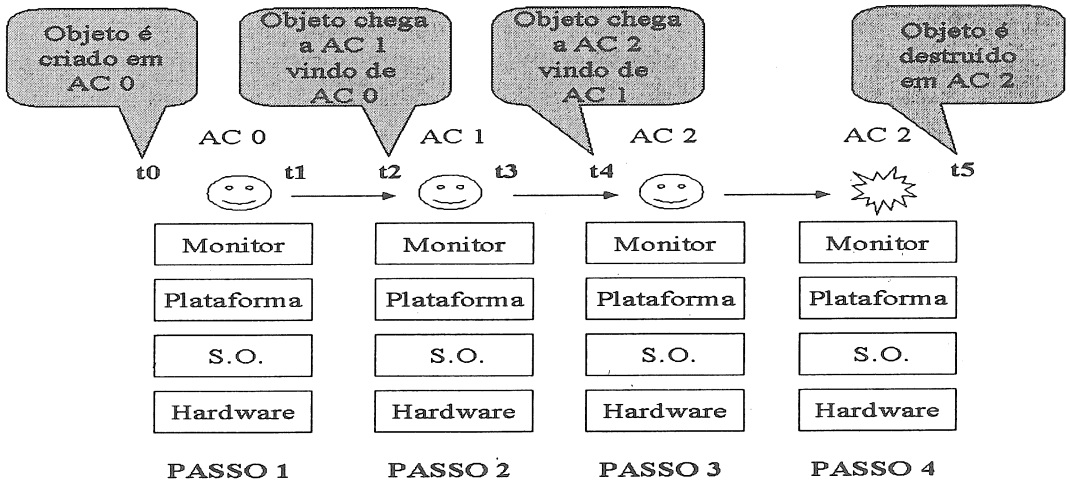

Figura 3. Exemplo de ciclo de vida de um objeto móvel.

Este exemplo demonstra o ciclo de vida de um objeto móvel. No passo 1, ocorre a criação do objeto em AC0. Neste momento, o monitor obtém o instante de criação, o identificador do objeto (gerado pelo próprio monitor quando da criação do objeto) e o local de criação (AC a que ele pertence). Podemos dizer, por exemplo, que o instante de criação foi t0, que o identificador do objeto é O1 e que seu local de criação foi AC0. À tabela de objetos pertencentes a AC0 são adicionadas as informações referentes a O1, bem como à tabela de objetos presentes em AC0.

No passo 2, o objeto O1 move-se para outro AC. Neste passo, o monitor identifica a migração e obtém o instante de início do processo migratório (t1), o local de saída do objeto (AC0), o instante de

finalização da migração (t2) e o local de chegada do objeto (AC1). Com base nesse dados, o monitor obtém o tempo total de transferência do objeto (t2 - t1), o tempo total de permanência do objeto em AC0 (t1 - t0) e atualiza sua lista de objetos presentes em AC0 (excluindo O1) e AC1 (adicionando O1).

No passo 3, ocorre o mesmo processo do passo 2. O objeto O1 move-se de AC1 para AC2. As informações obtidas pelo monitor são o instante de início do processo migratório (t3), o local de saída do objeto (AC1), o instante de finalização da migração (t4) e o local de chegada do objeto (AC2). A partir disso, o monitor obtém o tempo total de transferência do objeto (t4 – t3), o tempo total de permanência do objeto em AC1 (t3 – t2) e atualiza sua lista de objeto presentes em AC1 e AC2.

No quarto e último passo, o objeto O1 concluiu seu ciclo de vida (terminou suas tarefas). O monitor identifica a finalização de O1 e armazena sobre este o instante de sua destruição (t5) e o local de finalização (AC2). Com isso, a tabela de objetos presentes em AC2 é atualizada. Outra tabela que também é atualizada é a de objeto pertencentes a AC0, uma vez que, tendo sido finalizado o objeto, não é mais necessário mantê-lo nesta tabela.

Utilizando-se das informações obtidas em relação à criação, migrações e destruição de um objeto, o monitor tem a possibilidade de constituir o trace deste objeto. Utilizando o exemplo acima, podemos conhecer o itinerário cumprido por O1. Sabe-se que ele foi criado em AC0 em t0, saiu de AC0 em t1, chegou a AC1 em t2, saiu de AC1 em t3, chegou a AC2 em t4 e foi finalizado em AC2 em t5. Com isso, pode-se saber se este objeto realizou o trajeto correto e quanto tempo ele levou para tanto.

4.4 Arquitetura do Monitor de Códigos Móveis

A arquitetura do sistema de monitoração, a qual foi denominada de **MOOM** (**MO**nitor de **O**bjetos **M**óveis ou *MObile Object Monitor*), é composta por um monitor primário que centraliza as informações coletadas e as disponibiliza para acesso via interfaces gráficas ou programáveis e um monitor secundário rodando sobre cada um dos AC que se deseja monitorar, colhendo as informações relevantes ao processo de monitoração e enviando-as para o primário. O monitor primário possui uma lista dos AC monitorados por ele (AC onde se encontram os monitores secundários que trocam informações com esse primário), o que constituiu o que foi denominado de *domínio*. Isto é, os AC monitorados por um determinado monitor primário constituem o seu domínio de monitoração. Essa relação pode ser visualizada na figura 4.

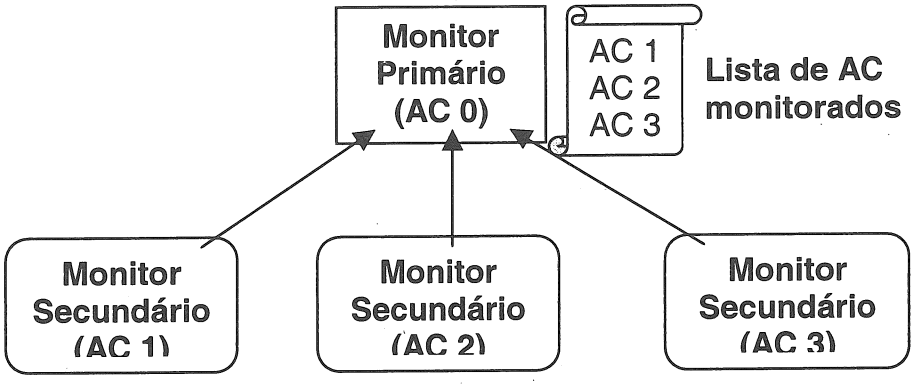

Figura 4. Ilustração de um domínio segundo a arquitetura descrita, onde os monitores secundários AC1, AC2 e AC3 pertencem ao domínio monitorado pelo monitor primário AC0.

Para controlar e administrar o monitor, dada a arquitetura brevemente descrita acima, é necessário apenas um usuário que atua sobre a máquina onde se encontra o monitor primário. Esse usuário

(administrador) escolhe as ações (vide seção seguinte sobre informações disponibilizadas) que se deseja monitorar no domínio onde o monitor se encontra.

4.5 Módulos do Monitor

Como já foi descrito anteriormente, o processo de monitoração envolve a comunicação entre monitores secundários locais a cada AC monitorado e um monitor primário, o qual fica localizado em um endereço pré-determinado e conhecido pelos secundários. Sendo assim, idealizou-se desenvolver o projeto através da implementação de módulos distribuídos em cada plataforma monitorável que trocam informações entre si. As relações entre esses módulos e suas localizações dentro do domínio monitorado podem ser visualizadas nas figuras 5 e 6. Estão representados nas figuras o suporte à mobilidade, que é a plataforma utilizada, o sistema operacional utilizado e o suporte à comunicação, que pode ser, por exemplo, por sockets ou RMI. Estes três elementos, como pode ser visualizado nas figuras, trocam informações com os monitores secundários e com o monitor primário. É a partir dessa interação que são obtidos os dados da monitoração e são trocadas informações entre monitores secundários e o primário.

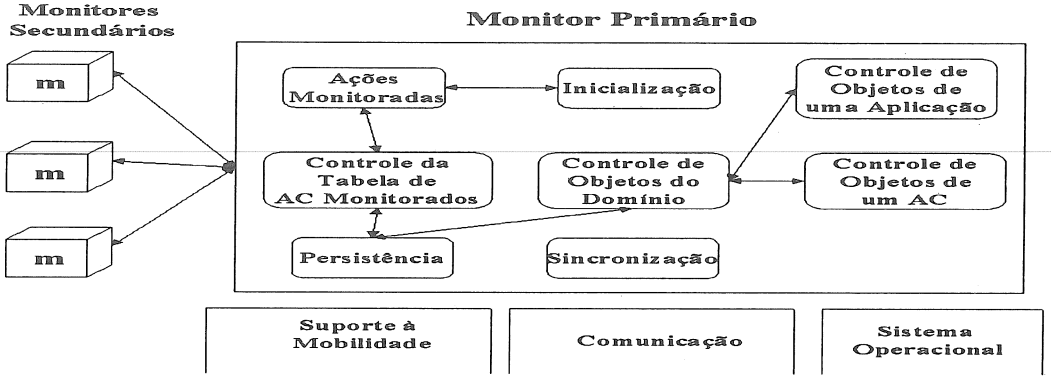

Figura 5. Apresentação da arquitetura em módulos do monitor primário.

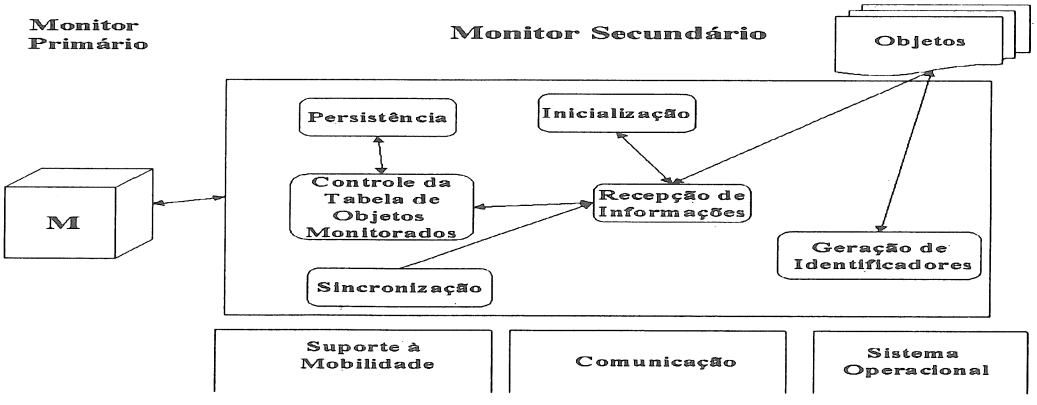

Figura 6. Apresentação da arquitetura em módulos do monitor secundário.

Os módulos que constituem a arquitetura do MOOM são a seguir descritos sucintamente:

- **Módulo de Persistência:** Este módulo é o responsável por garantir mecanismos, não só para o armazenamento das informações, mas também para a recuperação dessas;
- **Módulo de Geração de Identificadores para Objetos:** Gera identificadores para os objetos móveis;
- **Módulo de Recepção de Informações:** Responsável pelo recebimento das informações de geração de eventos relevantes à monitoração dos objetos do AC;
- **Módulos de Obtenção de Resultados:** A fim de proporcionar visualização de dados específicos sobre os objetos monitorados, alguns módulos auxiliares atuam sobre a base de dados. Tais módulos são apresentados abaixo:
  - ***Módulo de Controle de Objetos de uma Aplicação:*** Busca dos dados de monitoração relativos a todos os objetos de uma aplicação;
  - ***Módulo de Controle de Objetos de um AC:*** Este módulo é responsável pela filtragem das informações do Módulo de Controle de Objetos do Domínio relativas a todos os objetos de um AC;
  - ***Módulo de Controle de Objetos do Domínio:*** Módulo que fornece todos os resultados de monitoração dos objetos do domínio no qual se encontra;

- **Módulo de Inicialização:** Realiza todas as operações necessárias à inicialização e à finalização do monitor;
- **Módulo de Controle das Ações Monitoradas:** Administra quais as ações (criação, deleção, migração, ...) dos objetos móveis serão monitoradas;
- **Módulo de Sincronização de Relógios:** A partir desse módulo é possível a sincronização dos relógios entre os monitores primário e secundário, utilizando-se, por exemplo, o algoritmo de Cristian [9];
- **Módulo de Controle da Tabela de AC monitorados:** Executa o controle sobre todas as informações referentes aos monitores secundários que pertencem ao domínio;
- **Módulo de Controle da Tabela de Objetos Monitorados:** Controla os objetos pertencentes ao AC monitorado.

O sistema, como foi apresentado, pode ser adaptado a diferentes plataformas de mobilidade, uma vez que a única alteração se daria no Módulo de Recepção de Informações. Ou seja, a modificação se daria na forma em as informações são trocadas entre o monitor e a plataforma, sem a necessidade de qualquer alteração na arquitetura do sistema ou em outros módulos do mesmo.

4.6 Execução do Monitor

Durante a execução do MOOM, os monitores secundários ficam a espera de eventos gerados pelas aplicações presentes nos seus respectivos AC. A ser gerado um evento, o Módulo de Recepção de Informações (MRI) é avisado e providencia a geração de uma *thread* para o tratamento correto do evento e solicita ao Módulo de Sincronização a hora correta para adicionar a este evento. A hora correta é obtida pela sincronização entre secundários e primário realizada entre os respectivos Módulos de Sincronização, gerada a partir da aplicação de um algoritmo de sincronização, tal como o algoritmo de Cristian, sobre a hora obtida do Sistema Operacional. A geração desta thread é essencial para o bom funcionamento do monitor, uma vez que este processo libera o MRI para continuar a sua espera por eventos, deixando o tratamento dos eventos já recebidos por conta das threads, impedindo que o MRI fique ocupado e perca informações enquanto está tratando um evento.

Gerada a thread e adicionada a hora correta à informação, esta thread fica encarregada de encaminhar os dados relativos ao evento para o monitor primário utilizando o suporte à comunicação utilizado (sockets, RMI,...), onde estes serão disponibilizados ao usuário.

4.7 Inicialização do MOOM

Para executar o MOOM, deve-se inicializar primeiro o monitor primário. Durante sua inicialização, o Módulo de Inicialização (MI) realiza a ativação do Módulo de Controle das Ações Monitoradas (MCAM), o qual definirá que eventos serão válidos para fins de monitoração, e a abertura da comunicação com os secundários de acordo com o suporte à comunicação presente. Nesse ponto, o monitor primário fica pronto a receber cadastros de monitores secundários de seu domínio.

Inicializado o monitor primário, os AC que desejam ser monitorados devem providenciar a inicialização de seus monitores secundários. Essa inicialização ocorre com a ativação do MI do secundário, o qual abre a comunicação com o primário, segundo suporte à comunicação utilizado, enviando-lhe solicitação de inclusão no domínio. Tendo sido recebida a solicitação pelo primário e tendo este cadastrado o requisitante em seu domínio através do Módulo de Controle da Tabela de AC Monitorados, o primário envia, a partir de MCAM, os eventos válidos para monitoração para o secundário que solicitou sua inclusão no domínio. Ao receber os eventos que deve monitorar, o secundário tem seu MRI ativado, tendo como parâmetros estes eventos, e passa a monitorar o AC local.

5 CENÁRIO DE VALIDAÇÃO

Como cenário para testes do monitor, estão sendo utilizados computadores Pentium com 32 Mb de memória e Sistema Operacional Windows 95, todos ligados em rede, no Laboratório de Programação da PUCRS. Nesse ambiente, uma aplicação da área de comércio eletrônico que realiza uma pesquisa de preços em lojas virtuais será utilizada para testar o monitor. Esta aplicação é composta pelas lojas virtuais (localizadas em diversos AC) e por um agente pesquisador. Cada loja virtual possui uma tabela contendo nome e preço de cada produto oferecido por ela. O agente pesquisador é enviado de um determinado AC e circula entre as lojas virtuais em busca do menor preço para um determinado produto previamente especificado pelo desenvolvedor, retornando ao AC de origem ao fim do percurso.

Esta aplicação foi escolhida para a validação da ação do monitor devido à possibilidade que ela apresenta de poder-se testar a obtenção de todas as informações previstas. Além disso, pode-se comparar a vantagem de tal aplicação em relação à equivalente sem mobilidade de código.

6 CONSIDERAÇÕES FINAIS

O projeto acima exposto encontra-se em fase de desenvolvimento. As plataformas Aglets e Voyager foram examinadas para a construção do MOOM, que está sendo implementado utilizando-se a linguagem Java [10], [11], através do Java Development Kit versão 1.1.6.

A maior dificuldade encontrada até então é o fato que boa parte das ferramentas estudadas encontra-se disponível tão somente em sua versão inicial ou beta, ou carente de determinados mecanismos que contribuiriam para o trabalho, o que torna seu estudo, aprendizado e utilização mais difíceis.

Existem algumas questões que devem ser consideradas quanto ao uso do monitor descrito aqui. Uma delas é o impacto causado pelo monitor no desempenho da aplicação. Deve-se verificar o gasto em tempo e recursos causado pela ação do monitor sobre as aplicações móveis monitoradas como forma de verificar a sua real validade e necessidade, pois, dependendo da aplicação em questão, é possível que não valha a pena realizar testes monitorados devido à simplicidade da aplicação ou outros fatores que tornem desnecessária a monitoração e transformem o monitor num desperdício de tempo de processamento que pode ser evitado. Além disso, existe também a questão do tráfego na rede devido à troca de informações entre os monitores secundários e primário, dado que, a cada evento gerado em um AC monitorado, o monitor secundário enviará o aviso desse evento ocorrido ao monitor primário. É previsível, portanto,

que, em um ambiente com muitas aplicações móveis se movimentando, o tráfego gerado seja bastante considerável. Por isso, deve-se atentar para o ambiente em que se pretende executar o monitor, de maneira a prever de que maneira o monitor poderá afetar o desempenho da rede.

O monitor, como apresentado no presente texto, restringe-se à monitoração de um domínio. Como apresentado em 4.3, entende-se por domínio o conjunto de AC monitorados por um primário; ou seja, todos os monitores secundários ligados a um monitor primário formam o domínio desse primário. Pretende-se estender o projeto de maneira a monitorar mais de um domínio, mas tal objetivo impõe algumas questões. A mais importante seria a definição da comunicação entre monitores primários, como forma destes trocarem informações sobre seus respectivos domínios. Assim, poder-se-ia controlar objetos que realizassem migrações inter-domínios. Tal mecanismo também auxiliaria na definição de um esquema de cobrança de recursos utilizados entre domínios. Desta forma, um objeto pertencente a um determinado domínio, ao mover-se para um domínio diferente, teria execução controlada de forma que ele utilizasse apenas recursos locais no limite permitido a ele.

7 BIBLIOGRAFIA

[1] FUGGETA, Alfonso, PICCO, Gian Pietro, VIGNA, Giovanni. Understanding Code Mobility. **Transactions On Software Engineering.** IEEE , v.24, 1998.

[2] CARZANINGA, Antonio, PICCO, Gian Pietro, VIGNA, Giovanni. **Designing Distributed Applications With Mobile Paradigms**. [s.l.], [199-].

[3] KNUDSEN, P.. **Comparing Two Distributed Computing Paradigms – A Performance Case Study**. M. S. thesis, University of TromsØ, 1995.

[4] MAGEDANZ, T., ROTHERMEL, K., KRAUSE, S.. **Intelligent Agents: An Emerging Technology for Next Generation Telecommunications?**. IEEE, Berlim, 1996.

[5] MERZ, M., LAMERSDORF, W.. Agents, Services and Eletronic Markets: How Do They Integrate?. In **Proc. of the Int'l Conf. On Distributed Platforms**. IFIP/IEEE, 1996.

[6] **http://www.objectspace.com/products/voyager**

[7] GLASS, Graham. **Voyager: The New Face of Distributed Computing**. [s.l.], [199-].

[8] **http://www.trl.ibm.co.jp/aglets/index.html**

[9] TANENBAUM, Andrew S.. **Modern Operating Systems**. Prentice-Hall International Editions, EUA, 1992. 463 à 475 p.

[10] CORNELL, Gary, HORSTMANN, Cay S.. **Core Java**. Makron Books, São Paulo, 1997. 807 p.

[11] FLYNN, Jim, CLARKE, Bill. **Visual J++: Programando em Java**. Makron Books, São Paulo, 1997. 639 p.