Pontifícia Universidade Católica do Paraná
Programa de Pós-Graduação em Informática Aplicada

MOTHRA: Uma Proposta de Sistema para Busca de Informações Baseado em Agentes Móveis

Dissertação submetida à Pontifícia Universidade Católica do Paraná
como requisito parcial à obtenção do grau de

Mestre em Informática Aplicada

por

Paulo Vinícius Wolski Radtke

Curitiba, 10 de Outubro de 2000

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
Pró-Reitoria de Pós-Graduação, Pesquisa e Extensão

ATA DA SESSÃO PÚBLICA DE EXAME DE DISSERTAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA DA PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ.

Exame de dissertação nº 025

Aos 17 dias do mês de outubro de 2000, realizou-se a sessão pública de defesa de dissertação "**MOTHRA: Uma Proposta de Sistema para Busca de Informações baseado em Agentes Móveis**", apresentada por Paulo Vinícius Wolski Radtke, ano de ingresso 1997, para obtenção do título de Mestre em Ciências. A Banca Examinadora foi composta pelos seguintes professores:

MEMBROS DA BANCA	ASSINATURA
Presidente:. **Prof. Dr. Celso Kaestner (PUCPR)**	
Prof. Dr. Edson Scalabrin (PUCPR)	Edson
Prof. Dr. Hilton José da Silva Azevedo (CEFET - PR)	
Profa. Dra. Flávia de Almeida Barros (UFPE)	Flávia Barros

De acordo com as normas regimentais a Banca Examinadora deliberou sobre os conceitos a serem atribuídos e que foram os seguintes:

MEMBROS DA BANCA	CONCEITOS
Presidente: **Prof. Dr. Celso Kaestner (PUCPR)**	Aprov.
Prof. Dr. Edson Scalabrin (PUCPR)	Aprov.
Prof. Dr. Hilton José da Silva Azevedo (CEFET - PR)	Aprov.
Profa. Dra. Flávia de Almeida Barros (UFPE)	Aprov.
Conceito Final	Aprovado

Observações da Banca Examinadora

Correções na redação conforme sugestões da banca e acordo com os prazos regulamentares.

p/ Carlos Maziero

PUC - PR

Profº Júlio Cesar Nievola
Coordenador do Programa de Pós-Graduação em Informática Aplicada–PUCPR

Rua Imaculada Conceição, 1155 – Prado Velho – CEP 80215-901 – Telefone: +55 41 330-1669 / + 55 41 330-1654
Fax: + 55 41 330- 1669 – Curitiba – Paraná – Brasil - E-mail:ppgia@rla01.pucpr.br

À minha família, pelo apoio e compreensão na vida acadêmica.

Agradecimentos

Aos professores, orientadores e amigos Celso Antonio Alves Kaestner e Edson Emílio Scalabrin, que em muito colaboraram neste texto e na jornada acadêmica do meu mestrado.

Ao amigo e também professor Carlos Eduardo Hammerski Júnior, que em muitos momentos prestou grande auxílio na codificação do protótipo e realização dos testes.

Resumo

Este documento apresenta a proposta MOTHRA, um sistema para a busca e recuperação de documentos empregando agentes móveis. Inicialmente é dada uma introdução sobre busca e recuperação de textos e agentes móveis, para em seguida apresentar-se a proposta em si e detalhar a sua arquitetura. Também apresenta-se um protótipo do sistema, sua implementação e os resultados de testes realizados com o objetivo de validar a proposta sobre a base textual TIPSTER. Finalmente são apresentadas as considerações finais e perspectivas de trabalhos futuros.

Abstract

This document presents the MOTHRA proposal, a document retrieval system based on mobile-agents. First it introduces text retrieval and mobile-agents, followed by the proposal itself, detailing it's architecture. It also presents a prototype of the MOTHRA system, it's implementation and the results of tests undergone with this prototype over the TIPSTER text collection. The last section presents some considerations about the proposal and future works.

Sumário

Capítulo 1

Introdução

Uma característica do ser humano é a de registrar sua existência, suas atividades e realizações. Normalmente, este registro é feito em documentos escritos. Porém tal produção de documentos é contínua, acumulando-se ao passar dos anos em uma quantidade que não pode ser facilmente controlada.

Os primeiros esforços para se organizar as informações disponíveis em forma textual datam do final do século XIX e tratavam de métodos unicamente manuais. Tais métodos de indexação consumiam anos de trabalhos e uma posterior revisão era tão longa e trabalhosa quanto o processo original. Em 1911, o Professor Lane Cooper publicou um índice remissivo das poesias de William Wordsworth de maneira que um leitor ou estudante pudesse rapidamente localizar qualquer palavra em que tivesse interesse. O volume de 1.136 páginas lista todas as 211.000 palavras não triviais das obras do poeta, e o mais notável, levou apenas sete meses para ser confeccionado. A tarefa foi completada rapidamente porque foi realizada por um grupo altamente organizado de 67 pessoas, das quais 3 faleceram na época em que a concordância foi publicada, usando cartões de 3x5 polegadas, tesouras, cola e estampas.

Entre 1960 e 1970 o uso de computadores impulsionou a indexação de textos, reduzindo o trabalho realizado por seres humanos em anos para em horas. Assim foi possível se ter grandes quantidades de informações organizadas (uma *base textual*). Nestas bases, o acesso a palavras-chave é facilitado por um índice contendo as palavras e suas ocorrências nos documentos da base. Um exemplo clássico desta abordagem é dado por Witten [WMB94], que mostra passo a passo a construção de um sistema de *full text retrieval* (pesquisa em texto integral).

Tais sistemas de recuperação de informações e suas variações se encontram presentes em diversos elementos do dia-a-dia, tais como em enciclopédias eletrônicas, máquinas de busca da Internet, em manuais eletrônicos de programas e outros. Com o aumento

da disponibilidade de informações na Internet, as máquinas de busca como o Yahoo [YAH] e Altavista [Alt] se tornaram elementos chave para a busca de informações em uma rede na qual informações são produzidas e atualizadas diariamente. E é a dinâmica da produção de novos documentos e atualização dos já existentes que implica em uma restrição nesta abordagem: os documentos são considerados como parte de uma base estática que é indexada para posterior consulta. Desta maneira, se um documento é atualizado na rede ou até mesmo eliminado (cerca de 40% da *web* é atualizada mensalmente), a referência deste na base indexada permanece inalterada. E novos documentos devem ser submetidos a um processo de inclusão, que envolve a reindexação periódica.

Um segmento de pesquisas na área de Recuperação de Informações se concentra em investigar mecanismos não-convencionais para a busca e recuperação de documentos, utilizando em especial técnicas de Inteligência Artificial (IA). Diversos sistemas procuram auxiliar um usuário final na sua busca por informações [PMB96], [Paz97], [BSY95], [SAP], abandonando os mecanismos tradicionais em busca de novas soluções para este problema. Tais sistemas filtram mensagens em *newsgroups*, indicam páginas *web* que podem interessar ao usuário ou realizam outras funcionalidades não abordadas pelos mecanismos tradicionais de busca.

Seguindo esta linha, propõe-se neste trabalho uma arquitetura para um sistema de busca e recuperação de informações empregando agentes móveis, o *MOTHRA - MObile Text and Hyperdocument Retrieval Architecture*. Tal arquitetura realiza a busca de documentos no formato HTML em uma rede de grande porte, sem o uso de um índice de documentos, o que elimina a necessidade da manutenção de uma grande base de palavras chaves e suas ocorrências. Isto é possível no formato HTML devido às referências existentes entre documentos na forma de *hyperlinks*. A arquitetura proposta detalha um sistema multi-agente, com suas características e interações. Com o objetivo de testar a viabilidade do sistema, são definidas restrições sobre a proposta e é implementado um protótipo. Tais restrições são aplicadas devido a complexidade de funcionalidades propostas, cada qual suficiente para um trabalho longo e detalhado por si só.

Entre os problemas atacados pela proposta frente a outros sistemas e suas soluções estão o tráfego de dados pela rede e a interação entre os agentes (instâncias) durante o seu deslocamento na rede. A questão do tráfego consiste em reduzir a quantidade de dados brutos transitados pela rede pelo trânsito dos resultados da avaliação e do código que faz essa avaliação. Quanto à interação entre os agentes em trânsito, que deve ser o mais eficiente em termos de tempo e recursos empregados, é empregado um mecanismo de *pegadas*, que consiste na troca de informações através de objetos

depositados no ambiente de execução dos agentes, contribuição direta ao campo de pesquisa de agentes móveis.

Este texto é estruturado da seguinte forma:

- O capítulo 2 apresenta uma introdução aos mecanimos de busca e recuperação de informação.
- O capítulo 3 apresenta uma visão do que são os sistemas multi-agente, que consitem a base da proposta MOTHRA.
- O capítulo 4 apresenta a proposta MOTHRA, detalhando sua arquitetura e infraestrutura.
- O capítulo 5 apresenta o protótipo da proposta MOTHRA, e as restrições que lhe foram impostas. Introduz a plataforma de agentes móveis empregada e as técnicas utilizadas para implementar o protótipo.
- O capítulo 6 apresenta os resultados dos testes realizados para a validação do protótipo, constituídos por testes de recuperação e de performance efetuados sobre documentos da base textual TIPSTER [Uni].
- Finalmente, o capítulo 7 apresenta as considerações finais e a proposição de trabalhos futuros.

Capítulo 2

Busca e Recuperação de Textos

Este capítulo dedica-se a uma apresentação das metodologias atuais com o objetivo de estabelecer paralelos entre estes métodos e o que é proposto em busca e ercuperação de informações no MOTHRA. Em especial, a seção de busca na Internet introduz as características e problemas específicos desta realidade, para a qual o MOTHRA é projetado para executar.

Dentre os mecanismos disponíveis, o mais conhecido e empregado é o de *full text retrieval* [WMB94], que visa fornecer ao usuário uma resposta imediata ao seu questionamento. Nestes sistemas, os textos são totalmente indexados, permitindo que se localizem todas as ocorrências de cada palavra significativa do texto. Sistemas *full text retrieval* requerem um pré-tratamento completo da base de documentos, o que pode não ser possível (caso da WWW), devido à impossibilidade de acesso à base como um todo.

Outra linha de pesquisa concentra-se na filtragem de documentos [Paz97]. São abordagens que empregam normalmente técnicas estatísticas aliadas à Inteligência Artificial. Em muitos destes sistemas, o *feedback* do usuário é fundamental, permitindo a criação de um perfil (*profile*), para que o sistema se adapte a situações futuras e forneça melhores resultados. Em contraste aos sistemas *full text retrieval*, tais sistemas não fornecem resposta imediata ao usuário, mas procuram melhorar a qualidade da resposta, sendo aplicáveis ao caso de bases ilimitadas.

2.1 *Full Text Retrieval*

Witten [WMB94] descreve a criação de um sistema completo de *full text retrieval*, o *Managing Gigabytes*, suas diversas etapas e seus componentes. Basicamente, tal sistema compreende três etapas:

- Compressão: o texto é armazenado em uma forma compactada, visando economizar espaço em disco;
- Indexação: todas as palavras diferentes presentes na base têm suas frequências e ocorrências por documento determinadas;
- Questionamento (consulta): o sistema procura na base documentos que contenham ocorrências das palavras-chave indicadas pelo usuário.

Exemplos típicos deste tipo de sistema são as máquinas de busca na Internet, busca por palavras-chave em sistemas de ajuda de programas ou em enciclopédias eletrônicas, entre outros.

Há outras possibilidades de sistemas mais avançados, como indexação de informação gráfica, indexação de informações armazenadas em imagens e outros [WMB94]. Tais possibilidades, apesar de interessantes, estão além do escopo deste trabalho, que visa unicamente a manipulação de textos. São descritas a seguir as etapas de compressão, indexação e questionamento.

2.1.1 Compressão de Textos

Entende-se por compressão de textos a etapa na qual um arquivo contendo um texto é convertido para uma representação na qual ele mantém as informações do texto original, porém ocupando menor quantidade de memória. Um sistema *full text retrieval* deve possuir mecanismos para compressão e descompressão de documentos.

Considera-se aqui apenas a compressão de documentos armazenados na forma textual ASCII puro, sem imagens ou formatações especiais. Há alguns requisitos desejáveis ao se comprimir textos. Em relação ao uso de recursos de armazenamento do sistema, é prática comum o uso de técnicas de compressão. Mesmo comprimido, texto é fator dominante em termos de espaço em dispositivos de armazenamento em aplicações típicas de recuperação de documentos. Uma pequena melhora na compressão em gigabytes de dados é um retorno maior em termos de espaço do que uma pequena melhora na eficiência do algoritmo de indexação.

Também exige-se que o processo de descompressão seja rápido, e neste ponto ocorrem dois casos específicos:

- Os de usuários têm acesso simultâneo a uma base de textos, compartilhando um computador centralizador das consultas e pagando um certo preço pelo acesso às informações, às pesquisas realizadas e pelo uso de outros recursos. Neste caso, o

sistema deve processar consultas eficientemente para que a taxa de transação requerida seja mantida. *Hardware* mais eficiente pode atingir os mesmos resultados que elegância de projeto, mas o último terá custos menores;

- Os usuários têm sua própria cópia da coleção de documentos, ou seja, os usuários compram os dados e os programas de consulta, como um CD-ROM que contenha uma enciclopédia. Esses usuários não pagam por acesso à base, porém são notoriamente mais críticos na escolha dos programas, e se um produto em particular tem reputação de ser lento ele dificilmente será um produto comercial de sucesso.

Outra consideração é que documentos individuais devem ser decodificados com um custo de processamento mínimo (*overhead*). Para isto, o método de compressão deve permitir pontos de sincronização no meio da cadeia codificada. Sem estes, não há como acessar e apresentar documentos individuais sem utilizar um tempo razoável (percorrer a cadeia codificada até encontrar o documento requerido) e o sistema de recuperação não será viável para coleções de médio ou grande porte.

Como última consideração, deve ser dada uma atenção especial ao subsistema de compressão. Em ambos os casos de uso descritos acima, a decodificação deve ser rápida, porém esse requerimento não é válido para a codificação. Este é um fator importante quando projetando o sistema de compressão de texto, porque , se necessário, a codificação pode utilizar muitos recursos computacionais e tempo de processamento, visando fornecer uma estrutura de acesso a mais otimizada possível.

Para a compressão do texto, tem-se duas etapas, uma primeira passagem na qual as palavras são codificadas e a frequência destas é gerada, e uma segunda na qual o texto é definitivamente compactado.

2.1.2 Indexação de Textos

Indexação é o processo no qual é criada uma referência entre todas as palavras que ocorrem na coleção de documentos e suas ocorrências em cada um dos documentos. É um índice remissivo de palavras, onde para cada palavra é associada uma estrutura que permite o acesso à posição exata em que ela ocorre nos documentos da base textual.

Criado o índice, é possível compactá-lo também para diminuir o espaço utilizado em disco. Em especial, a indexação no MG [WMB94] é feita através de *d-gaps*, onde apenas a primeira ocorrência da palavra é absoluta, as demais são obtidas pela adição de deslocamentos relativos à cada índice sucessivo (*d-gaps*). Considere o seguinte exemplo:

> *Com o advento do computador, diversas tarefas mecânicas passaram a ser realizadas de maneira mais rápida e eficiente. Um computador pode realizar*

cálculos que utilizariam muitas horas-homem em poucos segundos. Com a popularização dos custos de processadores e outros componentes, a tendência é que o computador venha a se tornar cada vez mais popular.

Os índices absolutos para cada ocorrência da palava *computador*:

$$[computador \rightarrow 5.20, 47]$$

E os índices em *d-gaps*:

$$[computador \rightarrow 5, 15, 17]$$

Esta relação indica um conjunto de posições no texto nas quais a palavra *computador* ocorre. A motiviação por trás da escolha dos *d-gaps* reside no fato de que os deslocamentos entre índices são menores que os índices absolutos, ocupando menos espaço de armazenamento.

2.1.3 Questionamento

Nesta etapa o usuário realiza questionamentos ao sistema, acessando diretamente documentos já conhecidos na base ou fornecendo palavras-chave, esperando obter como resposta os documentos nos quais estas palavras ocorrem. Uma vez compactado o texto e gerado o índice, é possível realizar questionamentos ao sistema.

Em uma consulta baseada em palavras-chave (binária ou por palavras simples), acessa-se o índice de palavras e com o auxílio deste é feita a busca das ocorrências no texto. Outros tipos de busca, como busca por *ranking*, classificação do melhor para o pior documento de acordo com o questionamento, são obtidas com variações neste método.

2.2 Ontologias em *Full Text Retrieval*

Um dos problemas que ocorrem num processo de *full text retrieval* consiste no fato de que as ocorrências de palavras nos textos estão envolvidas em um contexto, o que pode inserir "ruído" no resultado fornecido, quando a busca é feita pela simples presença de palavras. Por exemplo, a palavra *estrela* possui diversos contextos. O contexto astronômico (corpos celestes) e o conceito artístico (artista de grande fama). Porém, uma simples busca com a palavra *estrela* não é o suficiente para indicar qual contexto se deseja. É necessário então um mecanismo que solucione o problema de

homônimos[1] através de sinônimos[2] destas palavras, que podem ser mais específicas em determinados domínio de conhecimento.

Diversas propostas procuram resolver este problema, como a proposta OMF – *Ontologies Manager Framework* – [BGS98], que consiste não de um sistema completo de busca e recuperação de informações, mas sim de um *plug-in* para sistemas deste tipo, fornecendo ferramentas para que o usuário enriqueça a sua busca pela adição de palavras-chave que possibilitem uma maior precisão em relação ao contexto desejado.

O sistema de ontologias possui diversas categorias, divididas cada uma em subcategorias específicas, até chegar-se a palavras-chave finais, que serão utilizadas na busca em si. A interface para a composição do questionamento é do tipo *drill-down* [BGS98]. Neste tipo de interface, cada nó corresponde a um domínio de conhecimento, e a cada expansão dos nós, encontram-se folhas ou novos nós que detalham ainda mais o domínio de conhecimento, possibilitando a especificação adequada das palavras a utilizar no questionamento.

Um ponto forte da OMF é que o seu resultado é um questionamento baseado em palavras-chave, que pode ser utilizado nas máquinas de busca atuais (ou mesmo no protótipo do MOTHRA), possuindo configurações no próprio sistema para realizar consultas nas máquinas de busca mais populares.

Voltada para a solução destes problemas na Internet, a proposta SHOE – *Simple HTML Ontology Extension* – [LSR96], [L+97] propõe a adição de um rótulo (*tag*) específico da ontologia relacionada ao conteúdo da página, que iria indicar à máquina de busca qual o conteúdo do documento. Os principais problemas com esta proposta residem no fato de que: (1) a base de pesquisa seria restrita aos documentos que possuem os *tags*; e (2) não há um consenso das ontologias que devam ser utilizadas globalmente.

2.3 Sistemas de Busca na Internet

Baeza-Yates e Ribeiro Neto [BYR99] indicam algumas características relevantes sobre os dados disponíveis na Internet:

- Dados distribuídos: pela natureza intrínsica da *Web*, os dados se expandem por diversos computadores e plataformas, sem seguir uma topologia definida ou garantia de velocidade de conexão (ou mesmo garantia de conexão) entre os diversos pontos;

[1] Significados diferentes para a mesma palavra, como o caso de estrela.
[2] Palavras diferentes de mesmo significado, como *hostil* e *agressivo*

- Dados voláteis: pela facilidade de adicionar-se novos computadores à *Web* ou de atualizar os conteúdos, estima-se que 40% destes conteúdos alterem-se mensalmente;

- Dados não-estruturados e redundantes: por não haver um modelo que defina precisamente a *Web*, o que garantiria a consistência dos dados e seus *hyperlinks*, a *Web* segue um modelo semi-estruturado, sem a presença de um modelo de dados. Muitas páginas são repetidas (através de espelhamento ou cópia de informação) ou muito similares;

- Dados de qualidade desconhecida: mesmo sendo tratada como um novo modelo de publicação, não há um processo editorial associado à *Web*. Logo, dados podem ser falsos, desatualizados, com escrita pobre e, principalmente, com muitos erros (gramáticos, de digitação, devido a processo de digitalização, etc).

- Dados heterogêneos: além de dados não-textuais, também há documentos em diferentes línguas, sendo que algumas utilizam alfabetos diferentes do latino (como os Kanjis japoneses).

Tais características não podem ser resolvidas de maneira simples por programas, e muitas delas não vão mudar, por serem características humanas, como, por exemplo, os alfabetos utilizados.

Para resolver problemas relacionados à qualidade dos documentos disponíveis na *Web*, muitas máquinas de busca empregam um grau de confiança à uma página indicado pelo número de páginas que apontam (possuem *hyperlinks*) para a página em questão. Desta maneira, uma página que seja indicada por muitas outras tende a ser de maior confiança do que páginas que não possuam muitas referências externas. Evidentemente este método não elimina a possibilidade de um boato ou de informação enganosa, mas oferece melhores resultados.

Os métodos tradicionais de busca na *Web*, como o Altavista [Alt] e Yahoo! [YAH] envolvem a indexação dos documentos de um *site*, que indica as palavras-chave que serão consideradas em uma busca, de maneira similar a um sistema *full text retrieval*. Aliados a técnicas como as decritas anteriormente, as máquinas de busca da Internet até o momento são o único mecanismo efetivo para a busca de informações na *Web*, mesmo que apresentem ainda várias questões fundamentais a serem resolvidas.

2.4 Filtragem de Documentos

Como abordagem alternativa à *full text retrieval*, algumas pesquisas têm se concentrado na área de filtragem de documentos. Ao contrário de busca e recuperação em uma base textual completa, os mecanismos de filtragem trabalham com apenas parte da informação e aplicam técnicas de Inteligência Artificial aliadas às técnicas convencionais de estatística.

Os documentos analisados variam desde documentos HTML, mensagens postadas em *newsgroups*, mensagens eletrônicas (*e-mail*), ou outros documentos que possam ser acessíveis a partir de um ponto conectado à Internet. Os serviços de filtragem de documento vão de filtros de e-mail, eliminando conteúdos indesejados, a assistentes de navegação, que através de um perfil (*profile*) do usuário indica documentos interessantes que podem ser acessados a partir do documento atual.

Pazzani [Paz97] apresenta diversos sistemas nestes moldes. É interessante observar estes sistemas, apesar de empregarem elementos semelhantes, não possuem a mesma funcionalidade, obtendo informações de maneiras não-convencionais, se comparadas com abordagens do tipo *full text retrieval*.

LIRA

[PS97] É um sistema de busca de documentos na WWW. Baseado em um questionamento do usuário, realiza uma pesquisa de documentos e seleciona as p melhores páginas, retornando o resultado para o usuário.

O diferencial está no fato de que o sistema apresenta uma interface com o usuário para visualizar o documento encontrado e fornecer uma avaliação do resultado fornecido (*feedback* de relevâncias, pelo método de Rocchio). Tais avaliações serão utilizadas nas próximas buscas como parâmetros iniciais, melhorando a qualidade do resultado em novas buscas.

Este sistema apresenta uma funcionalidade básica similar a do MOTHRA, porém sem empregar mobilidade de código e outras funcionalidades descritas no capítulo da proposta.

Letizia: *Assisting Web Browsing*

[Lie97] É um sistema assistente para navegação na Internet. A partir da página que o usuário está visitando, acessa os *hyperlinks* em busca de outros documentos que sejam do interesse do usuário. O usuário fornece *feedback* relativo às páginas sugeridas pelo

sistema, sendo que o critério adotado não concentra-se apenas nos resultados positivos, mas também nos resultados negativos. A busca considera os *sites* acessíveis a partir da página atual como espaço de busca, que é realizada em largura no primeiro nível abaixo do documento que o usuário está visitando.

Destaca-se neste sistema a facilidade de uso, um *plug-in* para navegador internet que realiza a tarefa em paralelo à navegação. Para determinar quais páginas são interessantes, o sistema acessa as páginas disponíveis nos *hyperlinks* da página sendo visitada e as avalia de acordo com o perfil do usuário. Isto é possível devido à característica em páginas *web* de uma página apontar páginas correlatas, característica utilizada no MOTHRA. A avaliação dada pelo usuário realiza o refinamento do perfil, permitindo que novas avaliações sejam mais consistentes com o esperado pelo usuário.

Um dos problema no sistema é que ele depende muito do usuário, necessitando de sua avaliação constante e de que este efetivamente navegue pelas páginas em busca dos documentos. Não há um mecanismo que encontre automaticamente páginas de interesse do usuário e as indique como resultado. Outro consiste no fato de que um novo interesse do usuário, ou um interesse eventual, não serão avaliados corretamente pelo sistema pois não há informações no perfil, há a necessidade (caso de um novo interesse) de treinamento do sistema, o que ocorre através do *feedback* de avaliações.

Webdoggie (*Webhound*)

[ML97] Outro sistema de assistente à navegação na Internet, executando em navegadores como um *plug-in*, o *Webdoggie* é um sistema que trabalha sobre o perfil de vários usuários que fornecem páginas que estes consideram interessantes. Outros usuários do sistema recebem indicações de páginas interessantes a partir de indicações de usuários com interesses semelhantes.

Uma comparação intressante com o Letizia é na abordagem das avaliações. No Letizia, a avaliação é local e depende apenas do usuário, já no *Webdoggie* as avaliações são centralizadas em um ponto na rede, mas são realizadas por diversos usuários. A vantagem evidente é que um usuário recebe avaliações de páginas que são de seu interesse sem precisar navegar pela rede, colaborando com a comunidade quando eventualmente encontrar uma página que seja interessante. Uma segunda vantagem consiste no fato de que um novo interesse incluído no perfil do usuário irá ter melhores resultados iniciais, pois outros membros da comunidade com o mesmo interesse já podem ter fornecido avaliações de páginas.

Em contrapartida, a confiabilidade dos resultados do sistema depende da comunidade, que deve ser consciente para fornecer avaliações que sejam válidas. E mesmo

assim, divergências de opiniões nas avaliações podem acarretar em insatisfação com os resultados fornecidos, o que evidentemente não ocorre com o Letizia.

2.5 Considerações Finais

Os mecanismos de busca tradicionais têm sido cada vez mais utilizados para a Internet, tentando solucionar os problemas existentes e contornando as características da *Web* de maneiras próprias. Ao contrário de *full text retrieval*, que possui técnicas básicas consolidadas, o estudo de mecanismos de busca na *Web* ainda se apresenta como um campo de pesquisa em potencial, como os mecanismos de filtragem de documentos.

Capítulo 3

Sistemas Multi-Agente

Neste capítulo, é apresentada uma visão geral de sistemas multi-agente, da definição de um agente à interação entre estes. A proposição do MOTHRA como um sistema multi-agente é motivada por características dos agentes descritas nesta seção, como a cooperação para a resolução de uma tarefa. Assim, é possível dividir o problema em seus componentes e a interação entre as entidades responsáveis pela solução de cada um destes componentes fornece o resultado como um todo. Os agentes podem estar distribuídos em uma rede e até se deslocar nesta (um agente móvel), e mesmo alterações na implementação de um dos agentes não implicam em alterações na comunidade como um todo, já que a interação é feita através da troca de mensagens, não de chamadas a procedimentos remotos ou outros.

3.1 Definição de Agente

Na perspectiva de um usuário final, a mais abstrata possível, um agente é um programa que auxilia pessoas e age no benefício destas. Agentes servem para pessoas delegarem tarefas a estes. Apesar de não ser uma definição incorreta, ela é muito geral. Agentes podem ser de muitos tipos diferentes, e podem existir em diferentes cenários. Eles podem ser encontrados em sistemas operacionais, redes, bases de dados e assim por diante.

A visão de Inteligência Artificial de agente é mais genérica [Fer95], [RN95], e preocupa-se em definir as características de um agente, e não como ele é implementado. Em geral, um agente é analisado nas seguintes dimensões:

- É situado num ambiente de execução ;
- Possui as seguintes propriedades :

- é reativo : percebe mudanças no ambiente e age de acordo com essas mudanças para atingir seus objetivos;
- é autônomo : possui controle sobre si mesmo e tem capacidade de decidir suas ações;
- é pró-ativo: é dirigido pelo seu objetivo, tentando realizá-lo como prioridade máxima;
- é persistente - pode interromper sua execução a qualquer momento, ser armazenado e mais tarde restaurado com todos os conhecimentos.

Além disto, um agente pode possuir uma ou mais das seguintes características :

- ser comunicativo : pode se comunicar com outros agentes;
- ser móvel : pode se locomover de um ambiente de execução a outro;
- ser capaz de aprender : pode se adaptar de acordo com experiências anteriores.

Segue-se neste trabalho a visão de agentes na ótica da Inteligência Artificial, uma entidade autônoma com um objetivo definido. Por ser simples e fácil de observar, o exemplo clássico de sistema de agentes reativos em Inteligência Artificial é o de uma colônia de formigas [D+91], [Min86]. Analisando o comportamento destas, observa-se que cada tipo de formiga é especializada em realizar uma tarefa elementar, como buscar alimentos, defender o formigueiro, cuidar das larvas, etc. Elas são reativas, ou seja, atuam a partir de estímulos externos. Caso haja um ataque as formigas especializadas na defesa do formigueiro entrarão em ação, se for necessário mais alimento, as que tem a especialidade de buscar alimentos o farão, e assim por diante. Cada grupo especializado de formigas não sobrevive por si só, nem possui "inteligência" ou "capacidade de raciocínio"; sua capacidade em realizar tarefas é conhecida a priori.

Mas, se combinadas numa sociedade, na qual cada uma supre as necessidades de acordo com a sua especialização, elas sobrevivem e cumprem o objetivo básico de qualquer espécie, que é o de se perpetuar. Não há uma especialização de formiga que, sozinha, consiga manter o formigueiro, nem a possibilidade delas se adaptarem a novas especialidades. O fato de que a "especialização" e a "inteligência" surgirem do grupo e não do indivíduo são características fortes de um sistema multi-agente reativo [Min86].

Para Ferber, um agente deve ter um objetivo e representações do mundo, de si próprio e de outros agentes. Ele pode se comunicar com outros agentes, perceber o ambiente em que ele se encontra e interagir (praticar ações) com ele [Fer95]. Nesta visão, um agente pode ser definido de acordo com a figura 3.1. Um sistema multi-agente é composto pelos seguintes elementos :

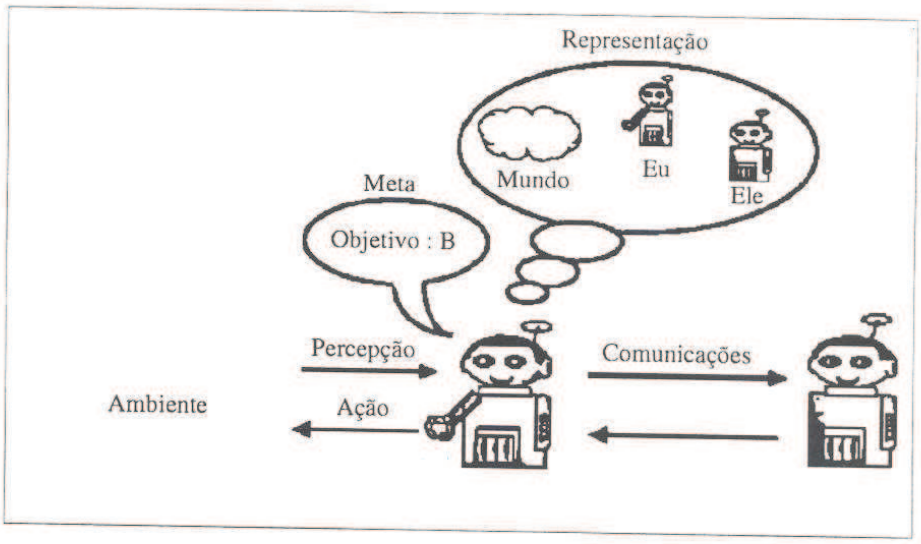

Figura 3.1: Modelo Conceitual de Ferber

- Um espaço ambiental E;
- Um conjunto de objetos O;
- Um subconjunto A de O que representa as entidades ativas do sistema (agentes);
- Um conjunto de relações R que unem os objetos;
- Um conjunto de operações F permitindo aos agentes de A perceber, produzir, consumir e manipular objetos de O.

Um agente ainda pode ser de dois tipos : reativo ou cognitivo. Um agente reativo reage unicamente a impulsos externos, como variações no ambiente, mensagens passadas por outros agentes, etc. A sua inteligência consiste no conhecimento do que fazer no caso de receber tais impulsos, ou seja, eles não possuem iniciativa própria. Já os agentes cognitivos tomam decisões, e possuem um objetivo definido, podendo assim tomar a iniciativa em certas situações. Esses agentes possuem a capacidade de tomar decisões e realizar ações independentemente de impulsos externos.

Revisando o exemplo das formigas sob a ótica de Ferber [Fer95], definem-se as formigas como agentes reativos, e um sistema baseado num formigueiro teria as seguintes características :

- A região aonde se encontra o formigueiro é o espaço E;
- As formigas, folhas, inimigos do formigueiro e outros elementos formam o conjunto de objetos O;
- As formigas formam o conjunto A de O;

- O conjunto de relações R e das operações F concerne a utilização do sistema.

Um projeto desenvolvido no sistema MANTA [Kus] simula as diversas especializações de formigas, mantendo assim um formigueiro virtual em um sistema multi-agente.

3.2 Mobilidade

Mobilidade é uma propriedade ortogonal de agentes, isto é, agentes não precisam ser necessariamente móveis. Um agente que não pode mover-se é chamado de agente estacionário [Lan98] :

> *"Um agente estacionário executa seu processamento apenas no sistema onde ele iniciou a execução. Se ele necessitar de informações que não estão presentes neste sistema, ou precisar interagir com um agente num sistema diferente, ele tipicamente usa um mecanismo de comunicação como RPC."*

Em contraste, um agente móvel não está ligado ao sistema onde ele iniciou sua execução, é livre para viajar através da rede. Criado num *host*, ele pode transportar seu estado e código para outro *host* na rede, onde ele continua a executar.

Por *estado* entende-se tipicamente os valores de atributos do agente que permitem a este determinar o que fazer quando ele continuar sua execução, no seu local de destino. Pelo termo *código* entende-se, num contexto orientado a objetos, o código de classe necessário para o agente executar. Tem-se então a seguinte definição para agente móvel [Lan98]:

> *"Um agente móvel não é ligado ao sistema onde começa sua execução. Ele tem a habilidade única de se transportar na rede de um sistema para outro. A habilidade de se transportar permite um agente móvel mover-se de um sistema que contém um objeto com o qual o agente quer interagir, e então ter vantagem de estar no mesmo sistema ou rede que o objeto."*

Franklin [FG96] nos dá uma definição mais abstrata de agentes móveis, voltada à visão do usuário, mais simples e direta :

> *"Um programa qualquer, realizando de maneira autônoma uma tarefa em benefício a um usuário, caminhando pela rede."*

Lange [Lan98] ainda nos mostra sete motivações para se utilizar agentes móveis :

1. Redução de tráfego de rede: o primeiro grande apelo para o uso de agentes móveis é a redução de tráfego de rede em sistemas distribuídos. Tais sistemas trabalham com protocolos de comunicação que envolvem múltiplas interações para cumprir uma tarefa, e o resultado é um aumento na carga da rede. Agentes móveis permitem o empacotamento de uma conversação, que é despachada ao *host* destino, onde a conversação ocorre. Pode-se desta maneira diminuir o tráfego de dados brutos na rede: ao invés de transferir uma quantidade grande de dados pela rede para processá-la localmente, um agente é enviado para o *host*, onde o mesmo processa os dados e envia pela rede apenas o resultado desejado. A idéia geral é a de enviar a computação aos dados ao invés de enviar os dados a computação.

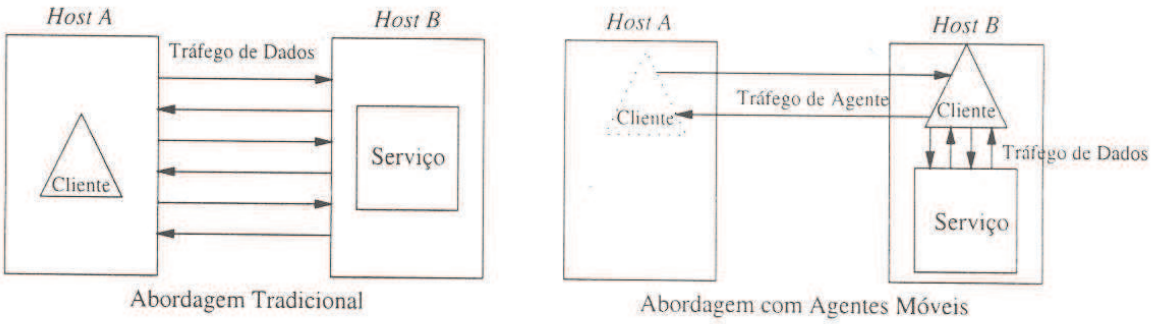

Figura 3.2: *Client-Server* em Duas Abordagens

2. Redução da latência em redes: sistemas de tempo real críticos, tais como robôs em processos de manufatura, precisam responder a mudanças no ambiente em tempo real. Realizar o controle através de uma rede de tamanho razoável envolve atrasos, que não são aceitáveis em sistemas de tempo real. Agentes móveis oferecem uma solução para este caso, já que podem ser enviados de um controle central para atuarem localmente uma programação completa.

3. Encapsulamento de protocolos: quando dados são trocados em um sistema distribuído, cada *host* possui código que implementa protocolos necessários para processar dados que entram e saem do sistema. Entretanto, protocolos evoluem para acomodar implementações mais eficientes ou atingir requisitos de segurança, o que torna difícil, se não impossível, a tarefa de atualizar o código de maneira apropriada. Isso torna protocolos um problema de legado. Agentes móveis são capazes, por outro lado, de mover-se para *hosts* remotos e estabelecer canais de comunicação baseados em protocolos proprietários.

4. Execução assíncrona e autônoma: muitas vezes temos conexões caras (em tempo de utilização) ou instáveis, e a implementação de tarefas que utilizem uma

conexão contínua durante o tempo da sua execução pode tornar-se inviável economicamente ou tecnicamente. Tais tarefas podem ser empacotadas num agente móvel, que é despachado e torna-se independente do seu local de origem, podendo operar assincronamente e autonomamente, e reconectar-se mais tarde para retornar os resultados desejados.

5. Adaptação dinâmica: Agentes móveis têm a habilidade de perceber seu ambiente de execução e reagir autonomamente às mudanças. Multi-agente móvel tem a habilidade única de se distribuir entre os hosts na rede, de maneira a obter a configuração ótima para resolver um problema particular.

6. Heterogeneidade natural: Computação em rede é heterogênea, tanto no aspecto de *software* como *hardware*. Como agentes móveis são geralmente independentes de plataforma e da camada de transporte, sendo dependentes apenas do ambiente de execução, eles fornecem condições para a integração transparente de sistemas.

7. Robustez e tolerância a falhas: A habilidade de agentes móveis para reagir dinamicamente a situações desfavoráveis e a eventos, os tornam mais simples para a construção de sistemas distribuídos robustos e tolerantes a falhas. Se um *host* estiver para ser desligado, todos os agentes executando na máquina serão avisados e terão tempo para se mover e continuar sua operação em outro *host* na rede.

Lange [Lan98] ainda indica algumas possibilidades de aplicações de agentes móveis:

- Comércio eletrônico: um agente móvel pode se deslocar pela rede em lojas na busca do melhor preço de determinado produto, efetuando a transação na loja que ofereça as melhores condições e custos;

- Assistentes pessoais: similares a agendas eletrônicas, porém com a possibilidade de agendamento de reuniões e trocas de informações com outras agendas inteligentes;

- Procurador seguro: dois agentes de transação podem marcar um *host* seguro para efetuar uma transação, sem riscos de segurança para nenhuma das partes. O enfoque dado não é a segurança na troca dos dados, mas sim em evitar que um dos agentes seja capturado em um *host* "armadilha" durante uma negociação;

- Recuperação de informação distribuída: aplicações nas quais a tarefa é procurar por alguma informação que esteja disponível de alguma maneira na rede. É neste contexto que se situa este trabalho.

- Serviços de redes de telecomunicações: agentes móveis podem ser empregados em tarefas de monitoração e configuração de tais redes, fato favorecido pela sua alta capacidade de adaptação ao ambiente;

- Aplicações de *workflow* e *groupware*: pela natureza distribuída de tais aplicações, agentes móveis servem naturalmente como uma base para o desenvolvimento de tais aplicações;

- Monitoração e notificação: quando faz-se necessário monitorar um ponto em um sistema, um agente de monitoração pode se deslocar pela rede, monitorar o ponto e retornar para o ponto de origem quando o monitoramento não for mais necessário;

- Disseminação de informação: ao invés de enviar notificações textuais sobre informações, um agente pode ser enviado interagir com o usuário de acordo com o seu perfil, podendo indicar produtos e informações correlatas;

- Processamento paralelo: a distribuição de carga obtida com o uso de agentes móveis é favorecida pela possibilidade de adaptação dinâmica do sistema, permitindo que um processo que esteja rodando em uma máquina sobrecarregada seja redirecionado para outra com menos carga.

Há diversas plataformas de agentes móveis disponíveis para avaliação e uso comercial acadêmico. Dentre elas podem ser indicadas as seguintes:

Nome	Origem	Linguagem
Aglets	IBM Corp. - Tokyo Research Labs	JAVA
Odyssey	General Magic Inc.	JAVA
Concordia	Mitsubishi	JAVA
Voyager	ObjectSpace	JAVA
MOLE	Universität Stuttgart Fakultat & Informatik	JAVA
Agent Tcl	Dartmouth College	Tcl
Ara	University of Kaiserlautern	Tcl

É notável nas plataformas citadas o uso da linguagem JAVA, em especial nas plataformas desenvolvidas mais recentemente. O advento desta linguagem colaborou para o impulso da tecnologia de agentes móveis, possibilitando o desenvolvimento de uma única aplicação que executa em diferentes plataformas (heterogeneidade, uma das características chave de agentes móveis).

Cada plataforma implementa de maneira diferente os métodos para transporte, comunicação e gerenciamento dos agentes. Esforços vem sendo feitos pela OMG (*Object*

Management Group[1]) para padronizar os ambientes de execução de agentes móveis, de forma a permitir que agentes de diferentes plataformas possam executar em servidores que não o da sua plataforma nativa. Esta proposta é denominada MASIF (*Mobile Agent System Interoperability Facility*).

3.3 Considerações Finais

Os agentes, incrementados pela mobilidade possibilitada pelas tecnologias de mobilidade de código, representam um modelo com potencial para a resolução de problemas de busca de informações de maneira não-convencional, com a aplicação de técnicas de Inteligência Artificial.

Tal potencial é explorado no MOTHRA, com agentes colaborando entre si na realização de tarefas menores, que não representam a solução total do problema, mas que possibilitam a obtenção da solução pela união dos resultados parciais obtidos.

[1] http://www.omg.org

Capítulo 4

A Proposta MOTHRA

Este capítulo dedica-se integralmente a apresentar a proposta MOTHRA, desde as motivações que levaram à sua proposição, arquitetura e descrição de funcionamento. Para testar a funcionalidade, um protótipo, que será descrito no próximo capítulo, foi implementado, no qual foram avaliadas questões relativas ao desempenho e qualidade dos resultados do sistema.

É importante notar que devido à complexidade de elementos da proposta MOTHRA e restrições de tempo para a execução deste trabalho, o protótipo é restrito a funcionalidades básicas descritas neste capítulo.

4.1 Motivação

As abordagens tradicionais de busca e recuperação de informação (do tipo *full text retrieval*) consomem recursos de sistema em grandes quantidades, sejam de memória (empregados no armazenamento dos documentos e seus índices) ou de processamento (para processar as informações da busca frente a um questionamento).

Um problema aberto em tais sistemas é quando a base textual é virtualmente infinita e/ou está em constante alteração. Novos documentos podem ser gerados diariamente nas bases textuais, ou então pode haver atualizações ou remoções de documentos que invalidam os índices gerados, o que implica na reindexação periódica de toda a base. Por último, tem-se o problema do tamanho da base: a produção de documentos pode ser elevada, ou os documentos podem estar distribuídos em uma rede de grande porte (como a Internet), ou podem ser em número tão elevado que seja praticamente impossível a sua indexação.

Propostas centradas no campo de pesquisa da Inteligência Artificial, conforme apresentado no capítulo 2, buscam resolver o problema por outros caminhos, ou então for-

necer funcionalidades ausentes nos sistemas de *full text retrieval* existentes. Porém, estas abordagens requerem uma conexão de rede constante para a sua realização, já que toda a execução está localizada em um único ponto, para onde são lidos os dados através da rede.

A tendência a médio prazo é que todos os sistemas estejam interligados em uma rede única, fato este que deve ser impulsionado pela computação móvel (*palmtops* [3CO], [Psi], ou em telefones celulares [WAP]). Frente a este cenário, propõe-se um sistema para busca e recuperação de informações com algumas características desejáveis para este paradigma iminente:

- A base de documentos que será tratada deve ser aberta, possuindo as seguintes características:
  - Base de documentos: os documentos que constituem a base podem ser atualizados, inseridos ou removidos de maneira transparente;
  - Disponibilidade: a base está distribuída em uma rede de computadores de topologia desconhecida;
  - Indexação: o conjunto de índices deve ser o menor possível, ou ainda, se possível, deve ser inexistente.
- Diferente dos sistemas atuais, o modelo a ser seguido no novo paradigma deve possuir estas características:
  - Execução Distribuída: a tarefa deve ser inicializada e executada sobre a rede;
  - Execução Assíncrona: uma tarefa de busca não deve depender do seu ponto de origem, a não ser para retorno de resultados;
  - Capacidade de Aprender: o sistema deve aprender com os resultados da busca e *feedback* dos usuários;
  - Execução Automática: a presença do usuário final deve ser requerida apenas ao término da tarefa, para visualização dos resultados e *feedback*.
- O usuário deve ter à sua disposição funcionalidades que facilitem a interação com o sistema e permitam a este especificar melhor a sua busca:
  - Perfil: a interface deve se configurar aos usos mais comuns do usuário;
  - Simplicidade: o sistema deve fornecer mecanismos simples para o que o usuário questione e visualize resultados;

  - *Feedback*: o usuário pode fornecer ao sistema avaliação de resultados fornecidos para melhorar outras buscas posteriores;
  - Ontologias: o sistema deve fornecer um mecanismo visando enriquecer a busca através de conhecimento de domínio.

- Do ponto de vista da busca e recuperação de documentos, o sistema deve possuir as seguintes características:
  - Aplicação de técnica composta: além de técnicas estatísticas já consolidadas em *full text retrieval*, deve combinar outras técnicas, como Inteligência Artificial;
  - Documentos: são considerados como organizados de forma similar a um grafo, com documentos indicando outros semelhantes através de links;
  - Mover a computação para os dados: o sistema não deve tranportar os documentos pela rede, mas sim os mecânismos de busca devem ir até o documento, obter uma avaliação deste e retornar apenas o resultado da avaliação.

Tal sistema pode ser imediatamente visto como um substituto das máquinas de busca convencionais WWW na Internet, ou em *intranets*. Um requisito anteriormente citado impulsiona o sistema na direção do formato de documentos HTML, que permite a realização de tal proposta: documentos que indicam outros semelhantes, organizados de forma similar a um grafo.

Esta característica é marcante na linguagem HTML, onde um documento pode indicar outros correlatos através de *hyperlinks*. Vale lembrar que a WWW não segue um modelo definido [BYR99], representando uma organização anárquica que segue apenas algumas diretivas básicas definida pela linguagem HTML. Ao se analisar os documentos armazenados na WWW, em geral um documento que trate de um assunto (como *estrela*, no contexto corpo celeste), irá indicar outros documentos que tratem de assuntos muito próximos ou do mesmo assunto em seus *hyperlinks*.

A principal vantagem do sistema reside no fato de que este não indexa estaticamente o conteúdo dos documentos para realizar a busca, mas realiza a busca dos documentos apenas no instante da solicitação, evitando a necessidade constante de adaptação do sistema devido a alterações no conteúdo e tamanho da base.

Para evitar o tráfego dos documentos pela rede, o que invalidaria a execução assíncrona do sistema, o sistema emprega agentes móveis para permitir que uma instância de agente se desloque na rede de um *host* a outro, visando a verificação do conteúdo de documentos, retornando, mais tarde, referências para estes documentos e uma ava-

liação da pertinência dos mesmos à consulta, que permitirá a escolha dos documentos que melhor atendem à mesma.

O sistema MOTHRA – *MObile Text and Hyperdocument Retrieval Architecture* – [RK98],[Rad98] é uma arquitetura multi-agente aberta visando a busca e recuperação de documentos seguindo estas características.

4.2 Arquitetura do Sistema MOTHRA

O sistema MOTHRA é composto por três modelos de agentes distintos, cujas instâncias cooperam entre si para realizar uma tarefa única. Essa divisão é similar à adotada no modelo da sociedade de agentes especializados citados por Deneubourg [D+91], [D+86], onde cada indivíduo não é capaz de realizar a tarefa como um todo, porém a interação entre eles forma um comportamento adequado à solução do problema.

Várias tem sido as propostas para a divisão das tarefas entre os agentes. Um modelo proposto por Sycara [S+96b] foi escolhido como base para o sistema MOTHRA, dividindo as funcionalidade em três modelos de agentes: agente interface; agente tarefa e agente de informação.

Na figura 4.1, uma visão do sistema MOTHRA, seus modelos de agentes e a troca de mensagens entre os mesmos.

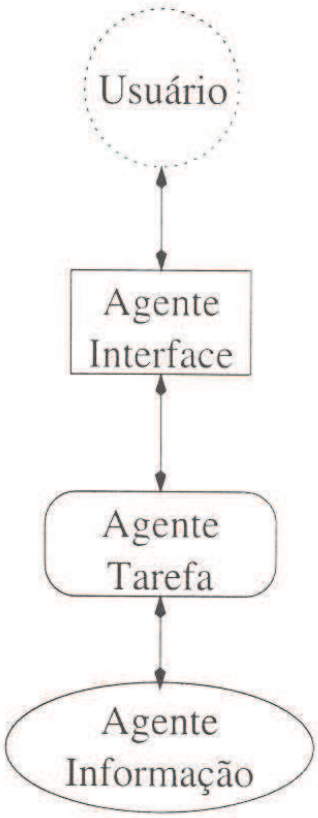

Figura 4.1: Arquitetura do Sistema MOTHRA

Tais modelos de agente possuem as seguintes funcionalidades

- Modelo Agente Interface: *front end* do sistema com o usuário. É responsável pela manutenção do perfil do usuário, por apresentar a este as opções disponíveis no sistema, formulação do questionamento e por recolher dados de *feedback* das tarefas executadas pelo sistema. A sua comunicação é com um agente tarefa, submetendo novas tarefas (buscas) e *feedback* disponível, bem como recebendo como resposta os resultados das buscas.

- Modelo Agente Tarefa: modelo responsável pela organização da tarefa. Coordena agentes de informação e compila os seus resultados para fornecer ao usuário através do agente interface (organizando um *ranking* dos documentos, por exemplo). Este agente aprende com o resultado das buscas, formando *clusters* de categorias de documentos que serão pontos iniciais para outras buscas. A comunicação deste se dá com um agente interface, recebendo pedidos de buscas e avaliações (*feedback*) do usuário, ou enviando resultados de buscas; e com um agente de informação, delegando as buscas propriamente ditas e recebendo os resultados brutos.

- Modelo Agente Informação: modelo que efetivamente realiza a busca, deslocando-se pela rede de *host* a *host* procurando por documentos que satisfaçam o critério de pesquisa do usuário. Também realiza uma avaliação da relevância do documento em relação à solicitação que será utilizada pelo agente tarefa, carregando consigo apenas a referência ao documento e esta avaliação. Sua comunicação ocorre com um agente tarefa, recebendo instruções para uma nova busca ou retornando os resultados.

Cada um dos modelos compartilha partes semelhantes em termos de estrutura, o que leva à proposição de uma arquitetura para um modelo de agente básico de busca e recuperação de textos, o BATHRA – *BAsic Agent for Text and Hyperdocument REtrieval Architecture* [RKS99], a partir do qual serão criados os modelos do sistema. Durante a execução podem ser criadas várias instâncias dos modelos de agente acima indicados. Outro problema a ser considerado nesta proposta é a interação entre os agentes durante o seu deslocamento, o que será detalhado na proposição de um mecanismo de interação entre agentes móveis em redes de grande porte, as pegadas [RKS00].

No restante desta seção será apresentado o funcionamento do sistema MOTHRA, seguido pela proposição do agente genérico para a implementação dos modelos de agente do MOTHRA e pela descrição do mecanismo para comunicação entre os agentes de busca.

4.3 Aprendizado

O agente tarefa deve armazenar conhecimento relativo ao resultados de outras buscas realizadas. Este conhecimento se traduz em documentos que serão utilizados para determinar endereços iniciais em novas buscas. Estes documentos são armazenados em *clusters* cujos centróides são as palavras-chave já empregadas.

O aprendizado é realizado de duas maneiras distintas:

- Automático: os n melhores resultados da busca são adicionados aos *clusters* no final da tarefa;
- Por *feedback*: de acordo com o *feedback* do usuário em relação aos resultados fornecidos, documentos podem ser adicionados ou removidos dos *clusters*.

Não há a necessidade de correlação entre as palavras-chave, já que o proposto é o uso de ontologias na formulação do questionamento.

4.4 Perfil

Cada instância do agente interface deve ser personalizada para melhor atender às necessidades do usuário. Tais informações personalizadas se concentram nos domínios de conhecimento de interesse do usuário, de maneira a existir conhecimento suficiente para uma formulação adequada do questionamento através do uso de ontologias.

4.5 Descrição de Funcionamentos

A execução de uma tarefa de busca e recuperação de documentos no sistema MOTHRA se dá em quatro etapas distintas: questionamento, preparação da busca, busca e exibição dos resultados. É importante notar que nestas etapas mais de uma sub-tarefa é realizada, envolvendo ao menos duas instâncias de agentes simultaneamente.

4.5.1 Questionamento

Nesta primeira etapa, o usuário fornece ao sistema os critérios que serão utilizados na busca. Tal mecanismo é similar ao empregado nas máquinas de busca atuais, porém pode ser incrementado em funcionalidade com o perfil e emprego de ontologias no modelo do agente interface.

O perfil serve de guia ao usuário dentro das ontologias disponíveis, facilitando a criação de uma busca precisa e que melhor represente o que o usuário deseja, utilizando informações configuráveis para determinar com maior precisão qual o domínio de conhecimento que se deseja dentre as ontologias disponíveis.

Além de formular o questionamento, o usuário deve nesta etapa estabelecer um tempo limite (*deadline*) para a execução da tarefa, o que evita que um agente de busca permaneça em uma busca indefinida de documentos pela rede. Opcionalmente, pode ser fornecido um ou mais documentos (endereços) iniciais para a busca, permitindo que o usuário possa sugerir os documentos iniciais para a pesquisa.

O agente interface também deve fornecer ao usuário, se for possível, um mecanismo para a escolha do mecanismo a ser adotado para a determinação de relevância do documento, caso exista mais de um modelo de agente informação no sistema.

Nesta etapa, instâncias de dois agentes estão envolvidas diretamente: agente interface e agente tarefa. O agente interface controla o perfil, gerencia as ontologias disponíveis e fornece ao usuário uma interface gráfica que permita a entrada e visualização destes dados.

O agente tarefa entra na etapa final do questionamento, após o usuário ter finalizado a submissão do questionamento ao agente interface. Ele recebe as informações relevantes ao questionamento do usuário para utilizar na próxima etapa.

As possíveis trocas de informações entre as instâncias agentes são:

- Agente interface: envia ao agente tarefa os dados do questionamento do usuário;
- Agente tarefa: responde ao agente interface se aceita ou não a busca, retornando uma identificação desta caso a resposta seja positiva.

Nesta etapa, as seguintes características do agente interface são empregadas:

- Ontologias: melhor formação do questionamento do usuário;
- Perfil do usuário: determinar o domínio de conhecimento do usuário na formulação do questionamento.

4.5.2 Preparação da Busca

Quando o agente tarefa aceita uma busca, esta entra em uma lista de tarefas em andamento. Como a busca está sendo iniciada, uma preparação, transparente ao usuário final, é necessária. Inicialmente, o agente tarefa deve determinar um ou mais documentos HTML a partir dos quais a busca deve iniciar. Tais documentos podem ser obtidos

de duas maneiras: indicação do usuário ou a partir de conhecimento obtido a partir do resultado de outras buscas (armazenados em *clusters* de documentos, de acordo com o conteúdo destes). Caso o sistema esteja sendo executado pela primeira vez, a indicação do usuário será a única maneira para se obter documentos iniciais para a busca.

Combinando documentos obtidos a partir destas duas abordagens, o sistema determina alguns endereços iniciais promissores para iniciar a busca. O agente tarefa prepara uma estrutura aonde irá armazenar os resultados desta busca específica e requer ao agente informação que realize a busca, de acordo com o método de busca escolhido pelo usuário, com os critérios determinados pelo usuário na etapa anterior.

Nesta etapa, cada agente informação recebe além do documento a ser pesquisado, o *deadline* e a identificação da tarefa, que serão utilizados mais tarde para o retorno dos dados ao agente tarefa, encerrar a busca e evitar que dois agentes de informação pesquisem o mesmo documento quando executando a mesma busca de documentos.

As possíveis trocas de informações entre as instâncias agentes são:

- Agente tarefa: envia para um agente de informação um documento inicial para a busca, o *deadline* e a identificação da busca. No caso de múltiplos documentos iniciais, devem ser requisitados novos agentes informações (clonagem, por exemplo) para receber os outros documentos;
- Agente informação: indica ao agente tarefa se aceitou ou não a busca requisitada.

Características do agente tarefa empregadas:

- Aprendizado: para determinar os documentos iniciais para a busca, o sistema baseia-se nos resultados obtidos em buscas anteriores.

4.5.3 Busca

Esta etapa é realizada apenas por instâncias do agente informação, sendo que a comunicação é feita entre agentes instanciados a partir deste modelo. Uma vez recebido o documento inicial da busca o agente deve então pesquisá-lo. É importante notar que, para cada documento inicial determinado no passo anterior um agente informação é designado, não ocorrendo casos de um agente informação receber dois documentos iniciais. O agente tarefa é envolvido em uma parcela secundária desta etapa, que consiste em receber os resultados coletados pelos agentes informação.

A etapa da pesquisa da base de documentos pode envolver ou não transferência do código e estado do agente (o uso de agentes móveis em todo o seu potencial). Se o agente estiver no *host* que armazena o documento HTML na rede, ele simplesmente irá

pesquisá-lo, caso contrário, ele primeiro irá deslocar-se até o *host* onde se encontra o documento para depois pesquisá-lo.

Ao pesquisar o documento, o agente informação determina a relevância do documento de acordo com os critérios fornecidos no questionamento do usuário (palavras-chave). Ao mesmo tempo, ao pesquisar um documento HTML o sistema determina os *links* para outros documentos e os coloca em uma fila de documentos a visitar.

Após terminar a pesquisa do documento, o agente armazena a referência deste (sua URL) com o resultado da avaliação de relevância do mesmo em relação à busca em uma estrutura interna, podendo agora tomar uma das quatro decisões possíveis, baseadas na lista de documentos a visitar que ele possui:

a) Caso o limite de tempo (*deadline*) para execução da tarefa tenha sido excedido, o agente deve ignorar documentos que estejam na lista de documentos a visitar e retornar os resultados obtidos ao agente tarefa;

b) Caso a fila de documentos a visitar esteja vazia, ele deve retornar ao seu *host* de origem, fornecendo ao agente tarefa os resultados encontrados. Neste ponto, o agente informação encerra sua execução;

c) Caso o próximo documento a ser pesquisado (primeiro da fila) esteja armazenado no mesmo *host*, o agente verifica se ele já foi pesquisado ou não por um agente da mesma tarefa. Se ele não foi ainda, o agente informação deve pesquisá-lo, caso contrário ele ignora o documento, voltando ao início deste processo de decisão;

d) No caso do próximo documento estar em outro *host* e haver mais documentos na fila (que podem possivelmente estar no *host* em que o agente se encontra no momento), o agente cria uma nova instância do agente informação que irá realizar a busca neste primeiro documento da fila. O agente original volta ciclicamente ao início deste procedimento de decisão.

   Caso a lista possuísse apenas um documento, o próprio agente original se deslocaria para o novo *host*. Em todos os casos verifica-se se o documento a analisar já foi pesquisado ou não por outro agente da mesma tarefa. No caso do documento ainda não ter sido pesquisado, o agente (clone ou original) realizam a pesquisa, caso contrário tal documento é ignorado. Se for um clone, não há resultados a serem retornados e o agente encerra sua execução. Já no caso de ser um agente original que se deslocou, este deve retornar os seus resultados.

Nesta etapa é introduzido um problema que será detalhado em uma seção específica: a interação entre os agentes. É necessário no caso da busca em documentos HTML

que os agentes troquem informações entre si para indicar que um documento já foi pesquisado.

Descarta-se a possibilidade do uso de mensagens do tipo *broadcast*, já que a rede em que tal sistema deve atuar não tem topologia e tamanho conhecido. Também descarta-se a possibilidade de comunicação pontual, devido a explosão combinatória do número de agentes em uma única tarefa, o que envolveria tráfego de dados pela rede em excesso e violaria uma das premissas da proposta.

A solução proposta para o problema é utilizar comunicação através do ambiente de execução dos agentes, empregando objetos portadores de informação, as pegadas [RKS99], [RKS00], que serão descritas detalhadamente adiante. No momento interessa o fato de que estas podem armazenar informações que indicam para agentes do sistema MOTHRA quais documentos foram pesquisados em um *host*.

Toda vez que um agente está prestes a pesquisar um documento, este verifica no ambiente a existência de pegadas. As pegadas, são depositadas no ambiente por outros agentes e contém informações úteis a outros agentes da mesma tarefa, possuindo um tempo de vida igual ao *deadline* da tarefa. No caso do sistema MOTHRA, a pegada contém a lista de documentos pesquisados no *host* por um dos agentes de informação de uma dada tarefa (nota-se aqui que dois agentes de tarefas de busca diferentes podem pesquisar o mesmo documento).

As possíveis trocas de informações entre as instâncias agentes são:

- Agente Informação:
  - questiona pegadas deixadas no ambiente se um determinado documento já foi lido anteriormente por outro agente da mesma tarefa.
  - fornece os resultados da busca ao agente tarefa.
- Pegadas: respondem a questionamentos de agentes, indicando se é verdadeiro ou falso que um documento já foi lido por outro agente qualquer.
- Agente tarefa: recebe resultados da busca, enviados pelos agentes informação.

Nesta etapa, as seguintes características do agente informação são empregadas:

- Pegadas: capacidade do agente de depositar informações no ambiente e de recuperar estas informações quando necessário;
- Interpretador HTML: necessário para pesquisar o documento e determinar sua relevância em relação à busca, bem como os *links* para os novos documentos a pesquisar.

4.5.4 Exibição dos Resultados

Esta última etapa envolve instâncias dos agentes interface e tarefa, fornecendo ao usuário final os resultados de uma busca. Convém ressaltar aqui que os resultados são apresentados ao usuário dentro do tempo limite especificado por este. São duas as situações possíveis:

- Quando o tempo limite é atingido e os agentes informação retornam os resultados encontrados;
- Quando os agentes pesquisam todos os links acessíveis a partir da página inicial da busca antes do tempo limite fixado pelo usuário.

Quando o agente tarefa possui todos os resultados, ele os organiza para formar um *ranking* dos documentos pesquisados, variando do melhor ao pior resultado. É possível também que o sistema apresente resultados parciais da busca à medida que os agentes informação retornam resultados. Neste momento, o agente tarefa pode escolher os n melhores resultados obtidos na busca e adicioná-los ao seu domínio de conhecimento. De acordo com os documentos pesquisados são formados *clusters* de documentos por palavra-chave, que serão utilizados como documentos iniciais em pesquisas futuras, caracterizando assim o aprendizado do agente.

Em seguida, o agente tarefa deve fornecer os resultados ao agente interface, que os exibirá ao usuário. A partir do agente interface o usuário poderá visitar os documentos encontrados na busca (pela execução de um *browser*), ainda tendo a possibilidade de fornecer um *feedback* ao sistema em relação ao documento, isto é, indicar explicitamente se o documento encontrado satisfaz os seus critérios de busca ou não.

As possíveis trocas de informações entre as instâncias agentes são:

- Agente Tarefa:
  - fornece ao agente interface os resultados ordenados da busca;
  - recebe do agente interface o *feedback* do usuário em relação a um dado documento.

  Não há troca de informações entre os agentes tarefa e agentes informação.
- Agente interface:
  - recebe do agente tarefa os resultados da busca;
  - fornece ao agente tarefa o *feedback* do usuário em relação a um dado documento.

Nesta etapa, as seguintes características do agente tarefa são empregadas:

- Aprendizado automático: de acordo com os resultados da busca, atualiza o seu conhecimento em relação aos possíveis domínios de busca com os n melhores documentos encontrados;

- Aprendizado assistido pelo usuário: atualiza o seu conhecimento em relação aos domínios de busca de acordo com o *feedback* do usuário em relação aos documentos.

4.6 O Agente Genérico BATHRA

A implementação do sistema MOTHRA envolve o uso de uma plataforma de agentes móveis como facilitador de desenvolvimento. Por características desejáveis de portabilidade e possibilidade de execução em sistemas abertos, a linguagem JAVA [Mic95], [MR00] foi escolhida para a implementação do MOTHRA. Porém são necessárias algumas considerações em relação ao que as plataformas de agentes móveis fornecem, e ao que é desejado como modelo para construir-se os modelos de agentes do sistema MOTHRA.

As plataformas de agentes móveis até então definem métodos e estruturas para mobilidade e comunicação simples, deixando de lado os mecanismos já clássicos e bem definidos de agentes de Inteligência Artificial (IA) [RN95], tais como representação de conhecimento, representação do agente e de seu objetivo, papel do agente na execução de uma tarefa, etc., cuja implementação é deixada a cargo do desenvolvedor. Sendo assim, são três os objetivos ao se definir um modelo genérico de agente para o sistema MOTHRA:

- Definir um conjunto de mecanismos que facilite a tarefa do desenvolvedor de agentes móveis utilizando técnicas de IA, devendo, estes mecanismos, serem transparentes ao desenvolvedor. Em outras palavras, o desenvolvedor não precisará conhecer os detalhes internos do funcionamento do agente. Este conjunto de mecanismos será fornecido ao desenvolvedor na forma de um agente genérico nomeado BATHRA – *Basic Agent for Text and Hyperdocument Retrieval Architecture*;

- Propor uma solução para o problema de interação entre agentes móveis numa rede de grande porte (como a Internet). A solução é proposta baseada na idéia de que os agentes deixam pegadas no ambiente de execução por onde eles passam.

Tais pegadas são elementos de informação sobre uma determinada tarefa que se desfazem com o passar do tempo;

- Fornecer uma camada de abstração entre a aplicação e a plataforma de agentes móveis, permitindo assim, futuramente, o desenvolvimento de aplicações com agentes móveis baseados na linguagem JAVA, independente da plataforma de agentes móveis na qual o agente genérico está implementado. Isto permitirá o desenvolvimento de um único agente para toda plataforma na qual o BATHRA esteja implementado.

Os modelos de agente do sistema MOTHRA (interface, tarefa e informação) são desenvolvidos a partir do agente genérico BATHRA. O comportamento do novo agente pode ser redefinido em partes, sendo que o agente genérico BATHRA assegura:

- A definição de mensagens aceitas ou competências;
- A percepção do ambiente;
- A gestão de comunicações;
- A gestão de mobilidade.

4.6.1 Visão Geral do BATHRA

Como discutido anteriormente, as plataformas de agentes móveis fornecem os mecanismos para transporte de código e de mensagens entre os agentes, que não são suficientes para a implementação facilitada de agentes mais sofisticados no modelo de Inteligência Artificial. Desta forma, há a necessidade de uma camada intermediária entre uma aplicação inteligente e a plataforma de agentes móveis, que defina a estrutura básica de um agente.

Tal modelo de agente deve definir os seguintes pontos [GBH88],[Fer95]:

- *Representação das competências e objetivos do agente*: o que o agente é capaz de fazer e qual é o seu objetivo;
- *Representação do conhecimento que o agente tem do ambiente*: o que o agente conhece do ambiente em que ele se encontra, em particular das informações depositadas no ambiente por outros agentes;
- *Representação do conhecimento que o agente tem dos outros agentes*: o que um agente sabe sobre outros agentes;

- *Forma de comunicação entre os agentes*: como são efetuadas as trocas de informações entre os agentes;

- *Papel do agente na comunidade*: a relação do agente para com outros agentes em termos dos objetivos gerais do sistema.

A figura 4.2 apresenta uma visão em camadas para uma aplicação típica.

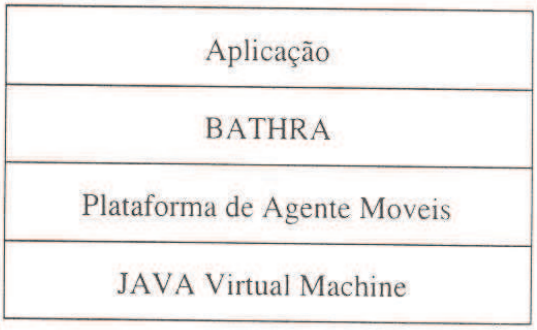

Figura 4.2: Camadas de uma Aplicação Típica

4.6.2 Arquitetura do BATHRA

As funcionalidades do BATHRA estão distribuídas em seis componentes, como mostra a figura 4.3: representação de conhecimento, roteamento de mensagens, gerenciamento de comunicações, gerenciamento de tarefas, mobilidade e controle. Esta arquitetura é baseada no modelo genérico proposto pela arquitetura OSACA [S+96a].

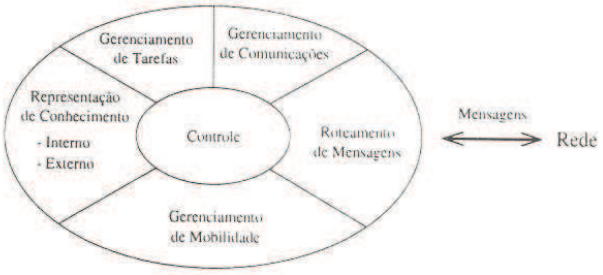

Figura 4.3: Arquitetura do BATHRA

- *Representação de Conhecimento*: uma representação de si mesmo (o que o agente sabe fazer) e uma representação dos conhecimentos que o agente possui do mundo exterior (ambiente e outros agentes);

- *Roteamento de Mensagens*: no BATHRA, todos os agentes podem ter acesso às mensagens enviadas por todos agentes presentes no mesmo *host*, bem como de agentes que estejam em outros ambientes mas que possuam representantes neste

(e vice-versa). Tais mensagens são filtradas e processadas em duas filas distintas: uma para mensagens endereçadas ao agente e uma para mensagens endereçadas para outros agentes. Esta última fila pode ser utilizada para possibilitar o agente a raciocinar sobre as tarefas em curso no sistema;

- *Gerenciamento de Comunicações*: a comunicação segue o protocolo KQML [F+93], empregando uma notação lógica para exprimir os conhecimentos a serem trocados [Gas88];

- *Gerenciamento de Tarefas*: componente responsável pelo gerenciamento das atividades realizadas pelo agente, que são realizadas em resposta às requisições internas ou externas;

- *Gerenciamento de Mobilidade*: fornece os mecanismos de mobilidade do agente e armazena um histórico da movimentação do agente;

- *Controle*: módulo responsável pela coordenação das atividades dos demais módulos dentro do agente. Contém os comportamentos básicos para cooperação com outros agentes, como primitivas de reação simples.

No BATHRA, os módulos de *gerenciamento de mobilidade*, *roteamento de mensagens* e *controle* abstraem as características da plataforma de agentes móveis para os demais módulos (*gerenciamento de tarefas*, *gerenciamento de comunicações* e *representação de conhecimento*). Os módulos *gerenciamento de mobilidade*, *roteamento de mensagens* e *controle* são implementados em classes não extensíveis JAVA (classes *final*). Os demais módulos (*gerenciamento de tarefas*, *gerenciamento de comunicações* e *representação de conhecimento*) são independentes da plataforma de agentes móveis empregada, permitindo a portabilidade dos agentes desenvolvidos com o BATHRA em uma dada plataforma de agentes móveis para outra qualquer.

Para implementar um agente especializado utilizando o agente genérico BATHRA, será necessário especializar o gerenciamento de tarefas e a estrutura de conhecimento, ambos particulares à especialidade do agente. A especialização é obtida extendendo-se as classes fornecidas para agregar as necessidades específicas da aplicação.

4.6.3 Representações de Conhecimento

O conhecimento do agente é modelado em classes. O agente genérico BATHRA possui classes para representação de conhecimento sobre:

- Os agentes: representação de agentes que estão executando no mesmo ambiente;

- O ambiente: descrição do ambiente no qual o agente se encontra em um dado momento;
- As habilidades de um agente: descrição de quais tarefas um agente pode executar;
- As tarefas em curso: representa as tarefas em curso do agente.

Um exemplo de código JAVA utilizado na implementação do BATHRA, para representação de conhecimento (de um agente), é dado abaixo:

```
public class RepresentacaoAgente
     extends Representacao
     implements java.io.Serializable
{
     public String nome;
     public String documentacao;
     public String autor;
     public String ontologia;
     public Vector repHabilidades;
     public String papel;
     public Vector tarefas;
     public String localizacao;

     // Métodos da classe
     ...
}
```

Os métodos construtores fornecidos pelas classes de representação trabalham com dois tipos de argumentos, uma lista de *strings* que preenchem as informações de um agente, ou uma *string* que representa as informações de um agente em uma expressão lógica (obtida a partir de um questionamento a um agente), que são extraídas por um *parser*. Por exemplo, uma expressão válida representando conhecimentos sobre um agente é dada abaixo:

```
(nome 12345df32 : localizacao ''atp://10.1.1.71:4434'' : documentacao
''Operaria'' : ontologia ''BATHRA'')
```

Além de métodos construtores e *parsers* para extrair informações a partir de expressões lógicas, as classes de representação de conhecimento fornecem métodos para formatar informações em expressões lógicas para outros agentes.

4.6.4 Interação Entre Agentes

Nos sistemas multi-agente, a interação entre os agentes é o elemento fundamental. É a partir desta que a solução de um problema deve aparecer. O BATHRA disponibiliza dois mecanismos de suporte à interação: a troca de mensagens (formato KQML [F+93]) e a percepção de objetos característicos de informação sobre o ambiente de execução dos agentes, as Pegadas [RKS99].

Em uma rede de pequeno porte (um segmento de uma intranet), a interação entre os agentes se dá pelos mecanismos tradicionais de troca de mensagens estilo KQML. Nesta situação, o agente possui e utiliza uma representação dos outros agentes. Para a rede de grande porte, o agente emprega o mecanismo de pegadas. Maiores detalhes sobre pegadas são apresentadas adiante.

As pegadas são efetivamente o único meio pelo qual os agentes interagem durante o deslocamento em uma rede de computadores de grande porte (Internet, por exemplo). Esse tipo de comportamento caracteriza-se essencialmente por um mecanismo de interação reativo [FD92].

4.6.5 Considerações Sobre o BATHRA

Por independência de plataforma, a linguagem JAVA foi escolhida para a implementação do BATHRA e do MOTHRA. Também é nesta linguagem que roda a plataforma Aglets SDK [Tok98], que fornece serviços básicos para a construção de um objeto móvel.

A arquitetura apresentada simplifica o desenvolvimento de um agente móvel, deixando para o desenvolvedor da arquitetura a tarefa de implementar apenas as habilidades desejadas. A proposta BATHRA esconde todos os detalhes de implementação na parte de comunicação, tratamento de mensagens e outros, específicos para cada plataforma de agentes móveis, deixando para o desenvolvedor apenas a tarefa de resolver o problema específico.

4.7 Pegadas

Algumas plataformas de agentes móveis, empregam mecanismos centralizadores para permitir que os agentes possam se comunicar, desde que um agente possua co- nhecimento sobre um outro agente. Outras se valem do uso de *proxys* que representam os agentes, habilitando assim a comunicação. Estes mecanismos exigem o estabelecimento de conexões entre endereços diferentes.

O objetivo desta seção é apresentar um mecanismo para comunicação indireta entre instâncias de agentes, sem o uso da comunicação ponto-a-ponto. A solução proposta

é baseada na idéia de que os agentes apresentados nesta abordagem deixam *Pegadas* no ambiente por onde eles passam. No MOTHRA, toda vez que uma instância do agente informação deixa um ambiente de execução, é deixada uma pegada que indica os documentos já verificados por tal instância, o que permite que outro agente informação possa mais tarde tomar conhecimento dos documentos já pesquisados pelo agente que deixou a pegada, sem estabelecer comunicação ponto-a-ponto. Isto possibilita que os agentes informação otimizem a busca evitando que documentos sejam pesquisados redundantemente.

4.7.1 Interação entre Agentes

Assume-se neste trabalho que em um endereço fixo, ou uma rede de pequeno porte conhecida (ex: Intranet), a interação entre os agentes se dá pelos mecanismos tradicionais de troca de mensagens, ou seja, a sociedade é composta de agentes meramente comunicantes. Já em uma rede de grande porte (ex: Internet), os agentes empregam o mecanismo de pegadas para realizar a troca de informações. Este mecanismo possui dois níveis de detalhamento, que são diferenciados em termos de iplementação, não de funcionalidade:

- Pegadas simples, onde os agentes criam objetos contendo informações sobre as tarefas em que trabalham e os depositam no ambiente;
- Pegadas por clones, onde o agente cria um clone de si mesmo. O clone fica estacionário no ambiente para dar informações sobre a tarefa, e o seu criador segue para outro endereço.

A tarefa do agente deve ter um prazo limite (*deadline*) para execução, e é baseado neste limite que a pegada tem a sua duração determinada. Após ultrapassar o tempo limite a tarefa é dada como encerrada e as pegadas são eliminadas do ambiente.

Pegadas Simples

Uma pegada simples é implementada com o auxílio de objetos portadores de informação[1]. As informações contidas nestes objetos são, geralmente, uma descrição suscinta de uma dada tarefa e seu estado corrente.

[1]Este mecanismo é uma mímica do empregado por seres humanos ao deixar pequenas notas com avisos (mensagens presas por ímãs em geladeiras, lembretes telefônicos, etc.), com o intuito de notificar outro ser humano, que em um momento qualquer poderá obter a informação.

Estes objetos são estrategicamente deixados no ambiente pelos agentes, com o único intuito de comunicar informações aos demais agentes que por ali passarem e estiverem trabalhando sobre as mesmas tarefas. Tais objetos respondem reativamente a questionamentos simples sobre as informações neles armazenadas, ao contrário de uma instância agente da aplicação, que possui todos os mecanismos de comunicação e inferência de conhecimento,

Pegadas por Clones

Outra maneira dos agentes interagirem indiretamente é através de clones estacionários, ou seja, quando um agente vai se mover para outro endereço, ele cria um clone de si mesmo. O clone se fixa no local em que ele se encontra, e seu criador se movimenta para o novo endereço. O clone interage com os outros agentes da tarefa que porventura passem por aquele local, fornecendo informações da tarefa.

Estas informações podem ser relativas ao que já foi feito em uma tarefa e as decisões tomadas até a criação do clone, ou informações simples como as fornecidas pelas pegadas simples. A primeira diferença está na facilidade que algumas plataformas de agentes móveis, como no caso do Aglets, oferecem para se implementar pegadas por clones.

A segunda diferença é que a pegada por clone tem a possibilidade de não fornecer apenas informações de maneira reativa, mas também pode possuir mecanismos de raciocínio colaborativo com outros agentes tarefa. No MOTHRA tal possibilidade não é necessária, mas consiste em um teste interessante a ser realizado futuramente.

Comparação das Abordagens

As abordagens *pegadas simples* e *pegadas por clones*, embora diferentes em termos de implementação, apresentam características comuns. Vale ressaltar os seguintes pontos:

- *Recursos computacionais*: *Pegadas simples* consomem menos recursos computacionais, tanto em termos de execução como uso de memória, ao passo que *pegadas por clones*, consomem mais recursos computacionais;

- *Implementação*: *Pegadas simples* são implementadas de uma maneira mais simples, seguindo um padrão que pode ser facilmente reutilizado em outros agentes e aplicações, consistindo unicamente de um objeto comum que fornece apenas informações sobre uma determinada tarefa. Já as pegadas por clones possuem uma implementação diferenciada para cada agente da aplicação, cuja complexi-

dade aumentada quando deseja-se uma pegada que raciocine em conjunto com um agente interlocutor;

- *Mecanismos de interação*: em ambas abordagens, as trocas de mensagens empregam os mesmos mecanismos. A diferença fica por conta de como a pegada trata os questionamentos sobre a tarefa, se respondendo passivamente (*pegadas simples* ou *por clones*) ou com a possibilidade de raciocinar sobre os questionamentos feitos (*pegadas por clones* apenas).

Para um agente derivado do BATHRA, as pegadas são efetivamente o único meio pelo qual os agentes interagem durante o seu deslocamento (ex: na Internet).

4.7.2 Cenário de Aplicação

O objetivo do sistema MOTHRA é a busca de informações em documentos HTML em uma base textual distribuída em diversas máquinas em rede. Desta maneira, um agente percorre a rede a partir de um ou mais documentos iniciais, procurando em seus *links* documentos que satisfaçam um critério de busca indicado pelo usuário.

Um agente pode deslocar-se na rede, de um endereço para outro, ou enviar clones para realizar a tarefa em paralelo. A figura 4.4 mostra um cenário de aplicação da arquitetura, onde por serviços entende-se como o servidor HTTP disponível no ambiente de execução e outras funcionalidades disponibilizadas pela plataforma de agentes móveis.

O *Agente Interface* é responsável pela interface homem-máquina do sistema, possuindo informações do perfil de seu usuário.

O *Agente Tarefa* é responsável pelo andamento das buscas de informações em documentos, bem como por manter conhecimentos que possam facilitar futuras buscas, obtidos a partir dos resultados de buscas já realizadas.

Por fim, o *Agente Informação* é especializado em analisar documentos baseados em critérios fornecidos à busca. Este último agente é responsável pela busca dos *links* nos documentos HTML, que serão utilizados para continuar a busca em novos documentos.

Uma busca de informações consiste, para o usuário, de três etapas:

- *1*: o usuário questiona o sistema, através do *agente interface*, formulando um critério de busca. Nesta etapa o usuário também fornece um tempo limite para a execução da tarefa de recuperação de informações;

- *2*: o sistema automaticamente percorre a rede (em páginas HTML seguindo *hyperlinks*) e armazena informações de classificação sobre os documentos (e não

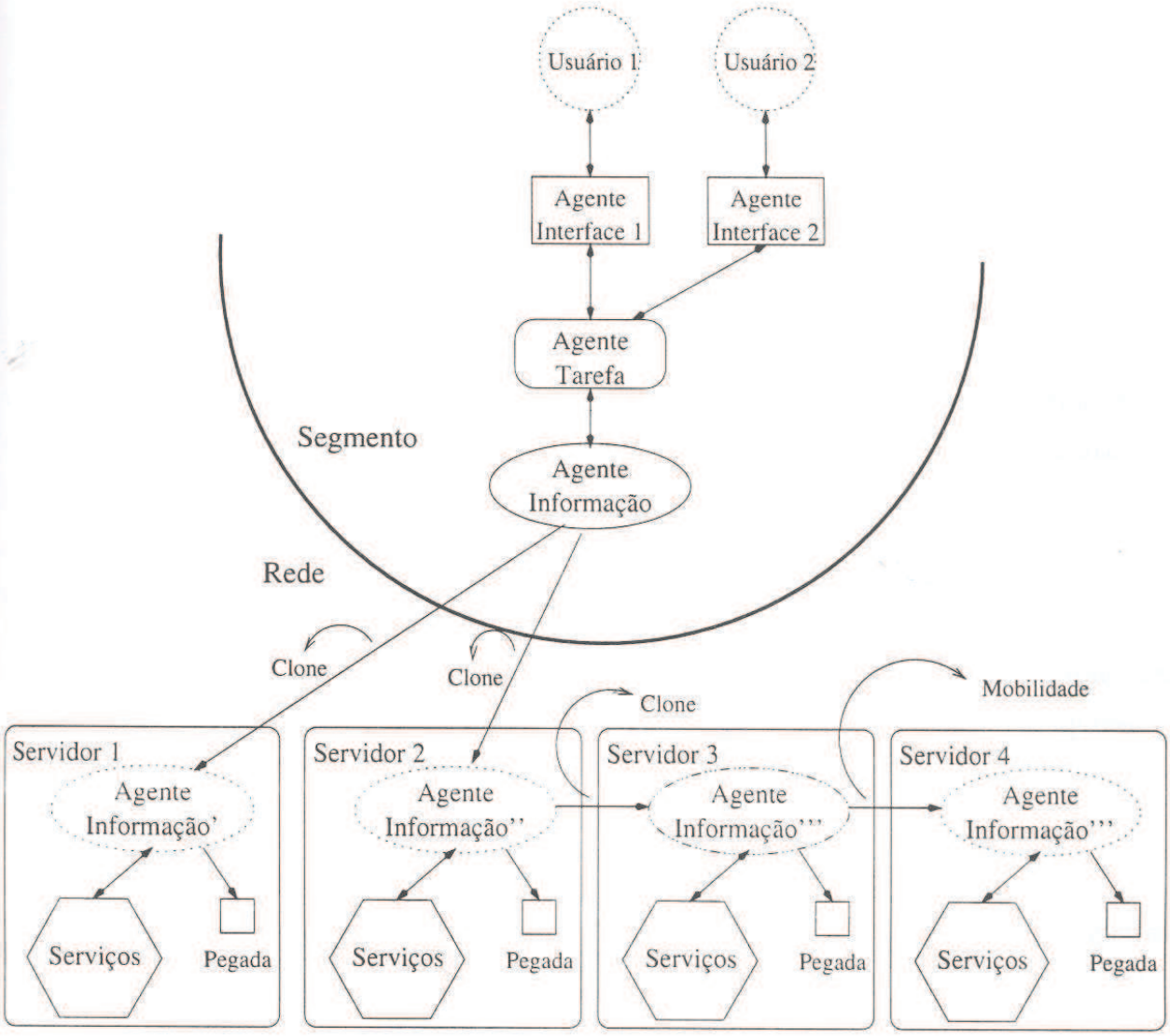

Figura 4.4: Arquitetura do Sistema MOTHRA.

necessariamente o documento como um todo), limitado pelo prazo fornecido pelo usuário na etapa anterior. Nesta segunda etapa, o usuário não tem sua presença requerida pelo sistema, podendo mesmo a estação que disparou a busca estar desligada;

- *3*: o sistema, tendo posse dos resultados retornados pelos agentes de informação, organiza as referências aos documentos e apresenta ao usuário o resultado.

Há uma quarta etapa que não faz parte da tarefa de recuperação em si, mas tem a ver com os mecanismos de interação dos agentes. No momento em que é ultrapassado o tempo limite de execução da tarefa, as pegadas são eliminadas do ambiente.

Nesta abordagem, como vários documentos HTML podem apontar para um mesmo documento, pode haver redundância na busca, mesmo que os caminhos percorridos inicialmente pelos agentes sejam diferentes, como na figura 4.5.

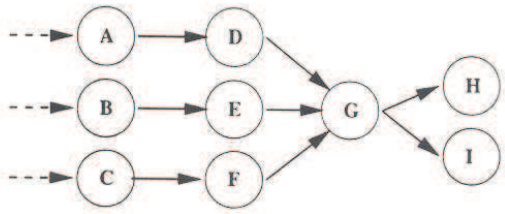

Figura 4.5: Hierarquia de Documentos

Se cada um dos três caminhos possíveis no grafo fosse percorrido por um agente, todos os três agentes visitariam o documento *G*. Evidentemente apenas um agente seria suficiente. Os documentos *H* e *I* são verificados posteriormente pelo agente que atingiu o documento *G* e por um clone seu.

4.7.3 Exemplo de Execução

Os agentes encontram-se em constante movimento pela rede. Cada vez que um agente chega a um novo endereço que possua um documento ainda não pesquisado, ocorrem as seguintes etapas (figura 4.6):

a) O agente *Ag* chega ao endereço;

b) O agente *Ag* envia uma mensagem de *broadcast* local (apenas para outros agentes ou pegadas que estejam no mesmo endereço) para verificar se o documento *Doc* já foi pesquisado por algum outro agente. Como nenhuma pegada é encontrada no ambiente, o agente assume que o documento ainda não foi pesquisado;

c) O agente *Ag* pesquisa o documento *Doc*;

d) Antes de sair do endereço, o agente *Ag* deposita uma pegada *P* no ambiente;

e) O agente *Ag* abandona o endereço, deixando para trás a sua pegada.

Caso um agente que tenha vindo por outro caminho também tente pesquisar o documento *Doc*, ocorre a seguinte situações (figura 4.7):

a) O agente *Ag'* chega ao endereço onde se encontra o documento *Doc*;

b) O agente *Ag'* envia uma mensagem de *broadcast* local (novamente, apenas para outros agentes ou pegadas que estejam no mesmo endereço) para verificar se o documento *Doc* já foi pesquisado por algum outro agente. Neste caso, uma pegada é encontrada no ambiente;

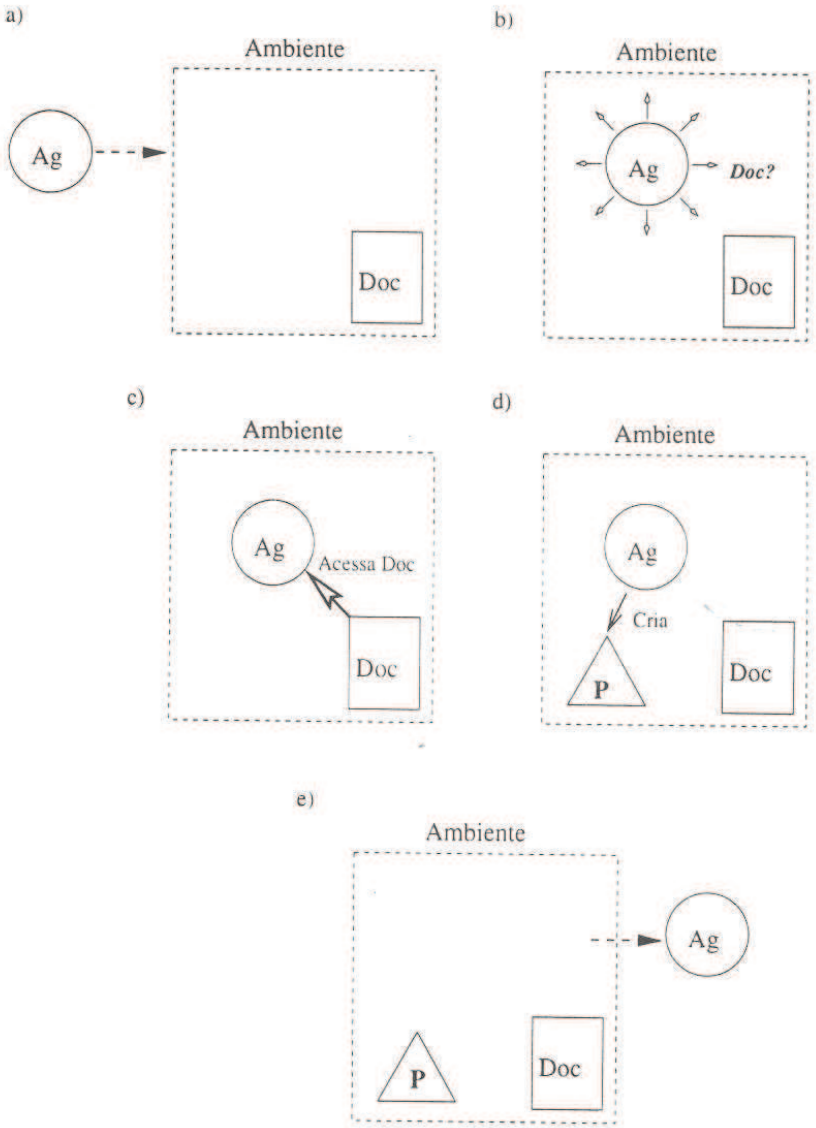

Figura 4.6: Agente pesquisando um novo documento

c) A pegada P indica ao agente Ag' que o documento Doc já foi pesquisado anteriormente;

d) O agente Ag' deixa o endereço, continuando a busca em outro endereço ou retornando os resultados encontrados.

É importante ressaltar que agentes envolvidos em duas buscas diferentes podem pesquisar o mesmo documento, já que os critérios analisados por ambos são diferentes. A diferenciação é dada pelo identificador da busca, que é de conhecimento tanto dos agentes da tarefa como das suas pegadas.

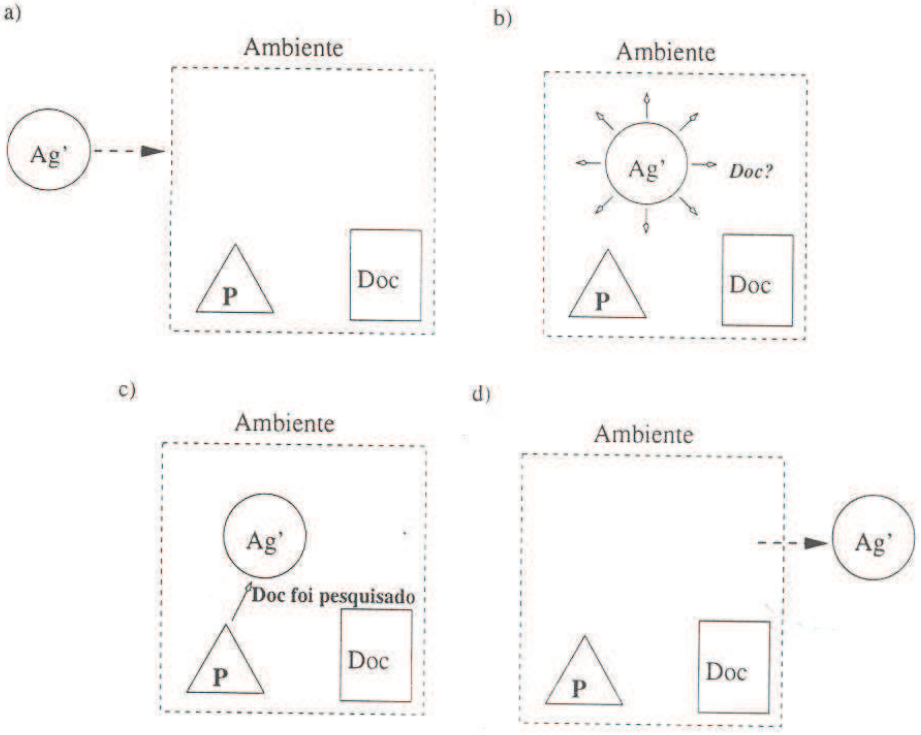

Figura 4.7: Agente tentando pesquisar um documento redundante

4.7.4 Considerações Sobre Pegadas

Os testes de agentes interagindo através de pegadas foram realizados no protótipo da proposta MOTHRA. Para os testes, foram implementadas pegadas por clones, de implementação facilitada por características da plataforma de agentes móveis Aglets SDK.

No sistema MOTHRA o ganho é claro, pois para um documento armazenado em um dado ambiente, uma pegada depositada neste ambiente é candidata a possuir informações relevantes a uma busca no documento em questão. Outros agentes que não passaram no ambiente não têm condições de fornecer informações, pois cada vez que um documento vai ser pesquisado o agente desloca-se para o ambiente em que ele se encontra.

O teste de validação das pegadas foi realizado sobre uma parte da base textual TIPSTER [NIS], [Uni], convertida para o formato HTML e distribuída em diferentes máquinas em uma rede, sendo que não houve redundância de documentos na busca, o que era esperado pela aplicação desta proposta, como detalhado no capítulo 6. Testes com a *web* ainda não são possíveis pela falta de servidores capazes de receber agentes móveis.

4.8 Considerações Finais

A proposta MOTHRA envolveu, paralelamente ao seu desenvolvimento, o estudo e integração de tecnologias ainda não usuais no desenvolvimento de *software*, como os agentes móveis e sistemas multi-agente. A aplicação de tais ferramentas resulta em um sistema robusto e que se adapta às próprias necessidades em redes em constante mudança, como a Internet e intranets.

As funcionalidades do MOTHRA propostas neste capítulo e que não foram implementadas no protótipo, serão objetos de estudo em trabalhos futuros:

- Aprendizado;
- Ontologias;
- Todas as funcionalidades do agente genérico BATHRA (no protótipo atual os agentes são implementados diretamente sobre a plataforma de agentes móveis Aglets).

Capítulo 5

Implementação do Protótipo MOTHRA

Esta seção detalha a implementação do protótipo da proposta MOTHRA, utilizado para a realização de testes de busca e obtenção dos resultados descritos na próxima seção. A primeira parte cobre as restrições impostas ao protótipo em relação à proposta original, para em seguida detalhar-se a plataforma de agentes móveis para a implementação do protótipo. Para finalizar a seção, é descrito o projeto do protótipo.

5.1 Restrições

Pelo escopo da proposta, decidiu-se proceder algumas restrições conforme indicado a seguir:

- Agente interface: no protótipo não possui o suporte a ontologias e perfil do usuário.
- Agente tarefa: no protótipo não possui o módulo de aprendizado, restringe-se apenas às buscas em andamento.
- Agente informação: não possui restrições no protótipo, estando em conformidade com a especificação.
- Pegadas: decidiu-se restringir as pegadas à utilização do tipo clone.
- Implementação: realizada diretamente sobre a plataforma de agentes móveis Aglets SDK.

Tais restrições não representam grandes limitações para a realização de testes de validação da proposta, que envolvem especialmente questões de performance e da viabilidade do sistema. Estes testes encontram-se detalhados na próxima seção deste texto.

Uma restrição, não relacionada a implementação do protótipo, mas sim ao mecanismo de busca utilizado e à base textual, é que os documentos a serem verificados pelo protótipo do sistema devem estar em inglês. A relação entre o mecanismo de busca empregado e a necessidade deste avaliar documentos em inglês é dada adiante.

5.2 Plataforma de Agentes Móveis Utilizada

A plataforma Aglets possui um modelo de desenvolvimento similar ao de uma *applet* JAVA [MR00], sendo o desenvolvimento de uma aplicação sobre esta plataforma uma extensão natural de mecanismos já conhecidos da linguagem JAVA. Curiosamente, o nome da plataforma é uma cópia da sigla *applet*: *applet* = *application* + *light-weight*, *aglet* = *agent* + *light-weight*. Em segundo lugar vem o fato desta encontrar-se disponível para avaliação e uso não-comercial sem custos.

Lange e Oshima apresentam os mecanimos do Aglets no livro "Programing and Deploying Java Mobile Agents with Aglets" [LO98]. Como citado anteriormente, um agente desenvolvido com o Aglets possui um mecanismo similar a uma *applet*, e como este ele também possui alguns métodos essenciais que fazem parte do seu ciclo de vida:

- *onCreation*: etapa de inicialização do agente, onde são criados recursos específicos do agente, como *threads* e outros. É invocado apenas uma vez no ciclo de vida do agente;
- *run*: método de execução do agente, que é invocado quando o agente é criado, despachado para outro *host*, ativado ou recuperado.
- *handleMessage*: método específico para tratamento de mensagens. Para cada mensagem recebida, este método é invocado, devendo tomar alguma ação;
- *onDisposing*: método invocado quando o agente está para ser destruído.

Além destes métodos, o Aglets disponibiliza *interfaces* Java para outras funcionalidades. Dentre elas, as mais importantes são as seguintes:

- *CloneListener*: adaptador para eventos relacionados à clonagem de um agente;
- *MobilityListener*: adaptador para eventos relacionados à mobilidade do agente pela rede;

- *PersistencyListener*: adaptador para eventos de persistência do agente. Não foi empregado no protótipo do MOTHRA.

Maiores informações sobre a plataforma Aglets, podem ser encontradas no livro de Lange e Oshima [LO98], bem como no site *Aglets Portal* [Pap99], um recurso gratuito na Internet com informações diversas, exemplos de aplicações feitas com o suporte do Aglets e artigos relacionados.

5.3 Implementação

Conforme detalhado anteriormente, o MOTHRA consiste em 3 modelos de agentes: um agente tarefa, um agente informação e um agente interface. Para o protótipo, os agentes tarefa e interface foram implementados em conjunto, visando a simplicidade de código. Esta modificação não é substancial, de forma que não se perde as características básicas da proposta.

Desta forma, o protótipo consiste em um agente informação, que realiza a busca na rede, e um agente tarefa que ao mesmo tempo possui o papel de agente interface. A implementação sobre o Aglets é simples e direta, utilizando as características fornecidas pela plataforma.

O agente tarefa possui uma implementação simples, já que praticamente todas as suas ações são baseadas no modelo de delegação de eventos Java, havendo apenas uma *thread* para o controle de eventos relacionados à passagem de tempo no sistema.

Já o agente informação é mais complexo, tendo seu processamento principal em uma *thread* que realiza a busca nos documentos pela rede. Tal *thread* é responsável pelas ações principais do agente: deslocamento na rede, criação de clones e avaliação da relevância de um documento de acordo com as palavras-chave da busca. O agente possui quatro estados possíveis de execução: ocioso (esperando o início de uma busca), procurando documentos, retornando resultados e esperando o fim da busca (quando o agente é uma pegada).

Ambos os agentes foram implementados em classes com funcionalidades específicas, como uma classe para ler o documento HMTL e serviços para realizar a avaliação do documento, outra para controlar *thread* da busca, outra como interface com o usuário no agente tarefa e assim por diante, visando possibilitar a troca futura destas classes, por outras que forneçam a mesma funcionalidade porém com mecanismos mais poderosos, possibilitando que a proposta seja facilmente aperfeiçoada.

5.4 Projeto do Protótipo

O protótipo foi implementado em dois agentes, cada um formado por várias classes. As figuras 5.1 e 5.2 mostram as classes que em conjunto formam os agentes tarefa (implementado em conjunto com o agente interface) e informação.

Figura 5.1: Agente Tarefa-Interface

As classes Stub e Busca são as classes principais dos agentes, herdando da classe Aglets. Os atributos destas classes servem para armazenar o conhecimento dos agentes, como as tarefas em andamento e seus resultados (agente Stub) e as avaliações de documento e estado de execução (agente Busca).

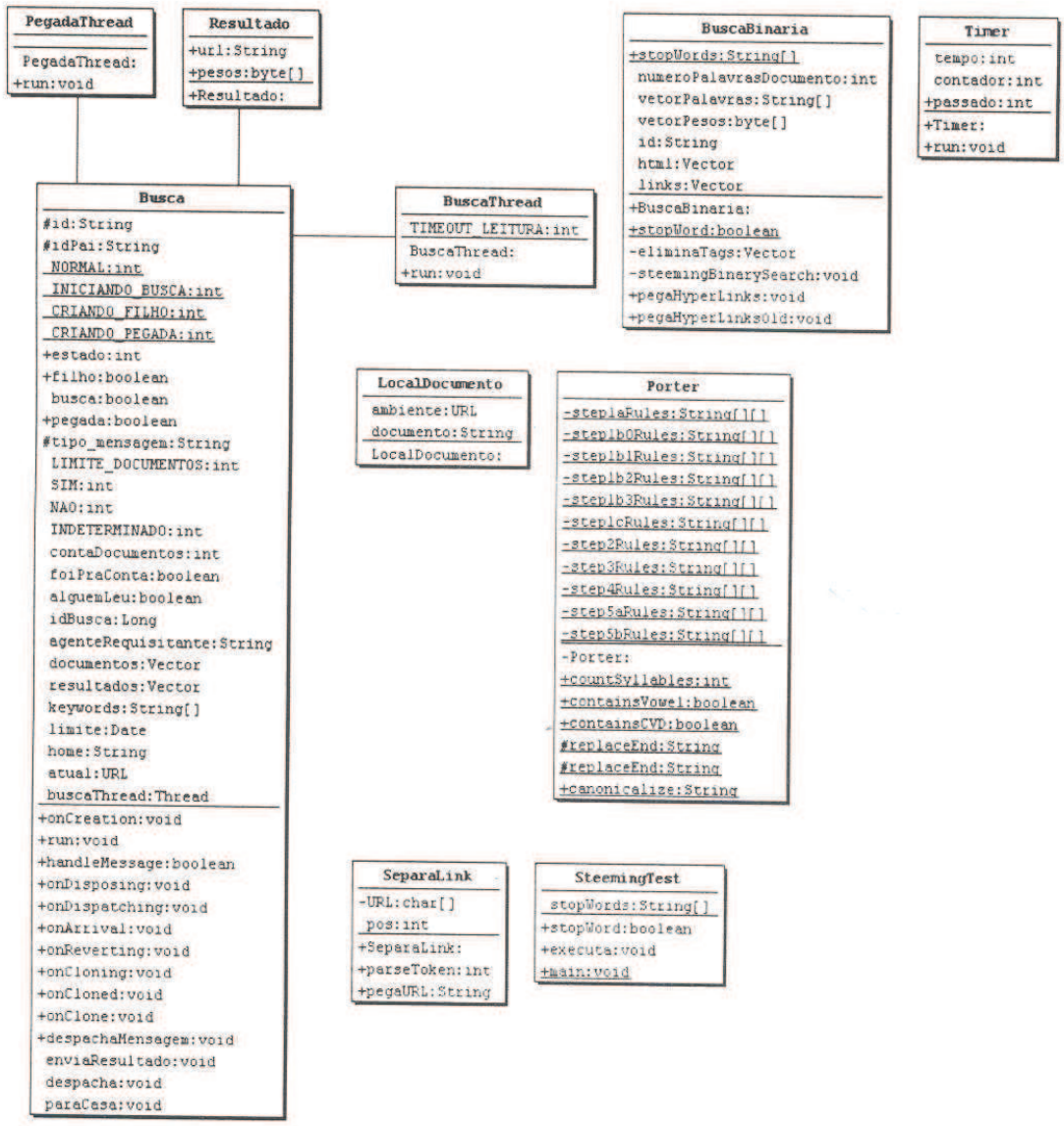

Figura 5.2: Agente Informação

5.5 Avaliação da Relevância dos Documentos

Para a avaliação dos documentos são eliminadas as *stop-words*, palavras de uso muito comum cuja relevância deve ser ignorada na busca, bem como também é aplicado o algoritmo para eliminação de sufixos das palavras do documento [Por93]. Vale notar que estes procedimentos também são aplicados às palavras-chave da busca, para permitir a compatibilidade entre estas e o documento em sua forma posterior ao tratamento.

A lista de *stop-words* e o algoritmo para eliminar os sufixos são para a língua inglesa, língua nativa da base textual empregada nos testes do MOTHRA. A conversão para outras línguas envolve a criação de uma lista de *stop-words* para esta língua, o que pode ser feito com relativa facilidade, e a conversão do algoritmo de eliminação de sufixos

ou a sua substituição por outro algoritmo equivalente.

Após o pré-processamento, são aplicadas duas medidas estatísticas sobre as palavras-chave no documento. Ambas têm funções específicas que são utilizadas para determinar um peso, que indica a relevância do documento em relação à busca.

A primeira medida é a média aritmética ($\overline{X}$) da contagem das n palavras-chave da busca no documento. Tal informação é utilizada como uma medida para indicar quantas vezes as palavras-chave foram citadas. Um documento com média 3 é melhor que um documento média 1, logo deve ter uma relevância maior.

A segunda medida é o desvio padrão (S), e indica o quanto a contagem entre as palavras-chave é correlacionada. Normalmente, as palavras-chave indicam um contexto e têm uma ocorrência em conjunto, logo, se deseja que numa busca os documentos possuam esta característica, porém somente a média não consegue indicar tal medida. O desvio padrão entra como uma medida auxiliar neste caso, evitando que documentos com contagem elevada em, por exemplo, apenas uma palavra, e baixa nas demais, tenha um peso elevado.

Assim, criou-se uma equação para determinar-se o peso da relevância de um documento, que é a seguinte:

$$\begin{aligned} Peso &= \frac{\frac{\sum_{i=1}^{n} P_i}{n}}{\sqrt{\frac{\sum_{i=1}^{n} {P_i}^2}{n} - \frac{\sum_{i=1}^{n} P_i}{n}} + 0.01} \\ &= \frac{\overline{X}}{S + 0.01} \end{aligned} \tag{5.1}$$

O valor 0.01 é inserido para permitir que o documento tenha um peso válido quando a contagem de todas palavras-chave forem iguais (desvio-padrão igual a 0), aumentando o seu valor em relação aos demais sem perder a consistência da avaliação, pois todos os documentos com desvio-padrão zero sofrerão o mesmo aumento.

Outros procedimentos podem ser considerados, como a frequência inter-documentos de um termo (*idf*), e a possibilidade de se avaliar a qualidade de uma página de acordo com os *hyperlinks* apontando para esta, porém o uso de tais medidas envolvem alguns problemas. Não há conhecimento prévio de quantas páginas apontam para um dado documento, sendo no máximo possível determinar no conjunto de páginas pesquisada na busca quantas apontaram para o documento, o que não reflete a medida desejada. A medida de frequência inter-documento de um termo possui um cálculo não-trivial, já que vários agentes realizam a busca em paralelo, sem um ponto central de processamento (à exceção do retorno dos resultados). Mesmo com a possibilidade do agente levar

consigo tais informações, a quantidade de dados trafegada pela rede seria muito grande, invalidando a proposta mais geral de se levar a computação aos dados para diminuir o tráfego na rede. Desta maneira, a avaliação do documento é feita isoladamente, sem levar em conta informações da relevância de outros documentos.

5.6 Exemplo de Avaliação de Documento

Considera-se o exemplo, retirado da base textual Tipster e as palavras-chave "*China*", "*reform*" e "*Taiwan*" como critério de busca:

> *Next year, the assembly is to revise Taiwan's constitution and introduce more democratic reforms. The issue at the heart of the campaign is the future of Taiwan and Communist China, divided since the 1949 revolution. On one side is the Chinese Nationalist Party, whose leaders have ruled Taiwan since fleeing to the island after their defeat by Communist forces on mainland China. The Nationalists, heir to a vast industrial empire and one of the richest political parties in the world, say Taiwan is part of China and must unite with the mainland one day. On the other side is the Democratic Progressive Party, Taiwan's scrappy 4-year-old opposition, which calls for abandoning any dream of uniting Taiwan and China. It wants a separate republic of Taiwan and a seat in the United Nations. The opposition platform is illegal under Taiwanese law. Communist China has threatened to invade Taiwan if the island announces its refusal to unite with the mainland. Because its goal is to establish a Taiwanese republic, the opposition wants direct elections for president. The Nationalists want to keep the current system, whereby the National Assembly elects the president. The Nationalist Party has opened state television to advertising by opposition candidates and is tolerating talk that a few years ago would have landed people in jail.; The ruling party also has forced hundreds of aging Nationalist legislators to step down.; Analysts say if the Nationalists win 75 percent of the National Assembly's 325 seats, they will be able to impose their version of Taiwan's future when the assembly revises the constitution next spring. But if the Democratic Progressive Party wins 30 percent of the vote or more, the Nationalists will be forced to bargain. The likely result is faster democratization and stronger opposition to the growing unofficial ties between China and Taiwan.; Since 1949, Taiwan has been transformed from an agricultural backwater into an industrial powerhouse. The country's $75 billion in bank reserves is among the world's highest.*

Ao eliminar-se as *stop-words* do mesmo, tem-se o seguinte texto:

year assembly revise taiwan constitution introduce democratic reforms issue heart campaign future taiwan communist china divided 1949 revolution side chinese nationalist party leaders ruled taiwan fleeing island defeat communist forces mainland china nationalists heir vast industrial empire richest political parties world taiwan part china unite mainland day side democratic progressive party taiwan scrappy 4 year opposition calls abandoning dream uniting taiwan china separate republic taiwan seat united nations opposition platform illegal taiwanese law communist china threatened invade taiwan island announces refusal unite mainland goal establish taiwanese republic opposition direct elections president nationalists current system national assembly elects president nationalist party opened state television advertising opposition candidates tolerating talk years ago landed people jail ruling party forced hundreds aging nationalist legislators step analysts nationalists win 75 percent national assembly 325 seats impose version taiwan future assembly revises constitution spring democratic progressive party wins 30 percent vote nationalists forced bargain result faster democratization stronger opposition growing unofficial ties china taiwan 1949 taiwan transformed agricultural backwater industrial powerhouse country 75 bank reserves world highest

Ao eliminar-se os sufixos das palavras do texto, tem-se o texto final que será utilizado na contagem das palavras-chave:

year assembli revis taiwan constitut introduc democrat reform issue heart campaign futur taiwan communist china divid 1949 revolut sid chines nationalist parti leader rul taiwan flee island defeat communist force mainland china nationalist heir vast industri empir richest polit parti world taiwan part china unit mainland dai sid democrat progress parti taiwan scrappi 4 year opposit call abandon dream unit taiwan china separ republ taiwan seat unit nation opposit platform illeg taiwanes law communist china threaten invad taiwan island announc refus unit mainland goal establish taiwanes republ opposit direct elect presid nationalist current system nation assembli elect presid nationalist parti open stat televis advert opposit candid toler talk year ago land people jail rul parti forc hundr ag nationalist legisl step analyst nationalist win 75 percent nation assembli 325 seat impos version taiwan futur assembli revis constitut spr democrat progress parti win 30 percent vot nationalist forc bargain result faster democrat stronger opposit grow unoffici ti china taiwan 1949 taiwan transform agricultur backwat industri powerhous countri 75 bank reserv world highest

Neste documento há 6 ocorrências da palavra-chave "*China*", 11 ocorrências da palavra-chave "*Taiwan* e 1 da palavra-chave "*reform*", totalizando 18 ocorrências das palavras-chave. Tem-se uma média de valor 6 e um desvio padrão de valor 1.25718,

que indica uma relevância de 4.734922 ao documento em relação às palavras-chave da busca.

5.7 Considerações Finais

O protótipo implementado visa demonstrar que os mecanismos fundamentais do sistema MOTHRA são funcionais, permitindo avaliações de performance e de relevância dos resultados fornecidos pelo mecanismo de avaliação de documentos escolhida para o protótipo.

Trabalhos futuros sobre o protótipo se concentrarão sobre o uso de outros mecanismos de avaliação dos documentos e implementação de mecanismos da proposta MOTHRA que foram deixados fora do escopo desta primeira análise.

Capítulo 6

Testes de Validação

Com o objetivo de testar a funcionalidade do protótipo do sistema MOTHRA, foram realizados testes em duas linhas de avaliação, que são exploradas neste capítulo.

A primeira analisa a performance do sistema, com o intuito de demonstrar as vantagens da aplicação do modelo multi-agente móveis. A segunda avaliação envolve testes de qualidade do mecanismo de busca adotado, realizadas sobre uma base textual. Em ambos os testes, foi empregada a base Tipster de documentos [Uni], convertida para o formato HTML.

Os gráficos relativos ao uso de memória e tempo de usuário no processador, obtidos nos testes de performance, são encontrados no apêndice B deste documento. A próxima seção apresenta as conclusões baseadas nos resultados obtidos neste capítulo e trabalhos futuros a serem realizados.

Um teste especial, realizado para avaliar a funcionalidade das pegadas evitando a redundância da busca, também é apresentado neste capítulo, porém não se trata de uma avaliação do sistema MOTHRA, mas sim do mecanismo de pegadas.

6.1 Testes de Performance

O objetivo principal destes testes é demonstrar a eficiência da abordagem multi-agente móveis proposta frente a um mecanismo centralizado. Para isto, este teste foi realizado em três situações com o objetivo de fornecer informações comparativas:

- Base centralizada, agentes centralizados;
- Base distribuída, agentes centralizados;
- Base distribuída, agentes distribuídos.

Os testes foram realizados em máquinas COMPAQ Pentium 200, 32Mb de RAM e 2Gb de disco rígido, sistema operacional Windows NT4.0, Aglets 1.03b e servidor HTTP Apache [Fou]. A configuração de hardware das máquinas é idêntica para permitir uma comparação simples.

Todos os testes de performance seguem o mesmo roteiro: um minuto de espera para estabilizar o sistema, a inicialização do servidor do Aglets, a instanciação dos agentes, o início da busca, a exibição dos resultados, a finalização dos agentes e o encerramento do servidor do Aglets, com mais um intervalo de um minuto para que o sistema atinja o estado estável após a busca. Os tempos discutidos a seguir englobam todos estes passos no teste, tendo cada passo um tempo mínimo alocado de um minuto para verificar a estabilidade do sistema nestas situações.

Para interpretação dos gráficos obtidos nos testes de performance e das etapas dos testes, considere o exemplo na figura 6.1

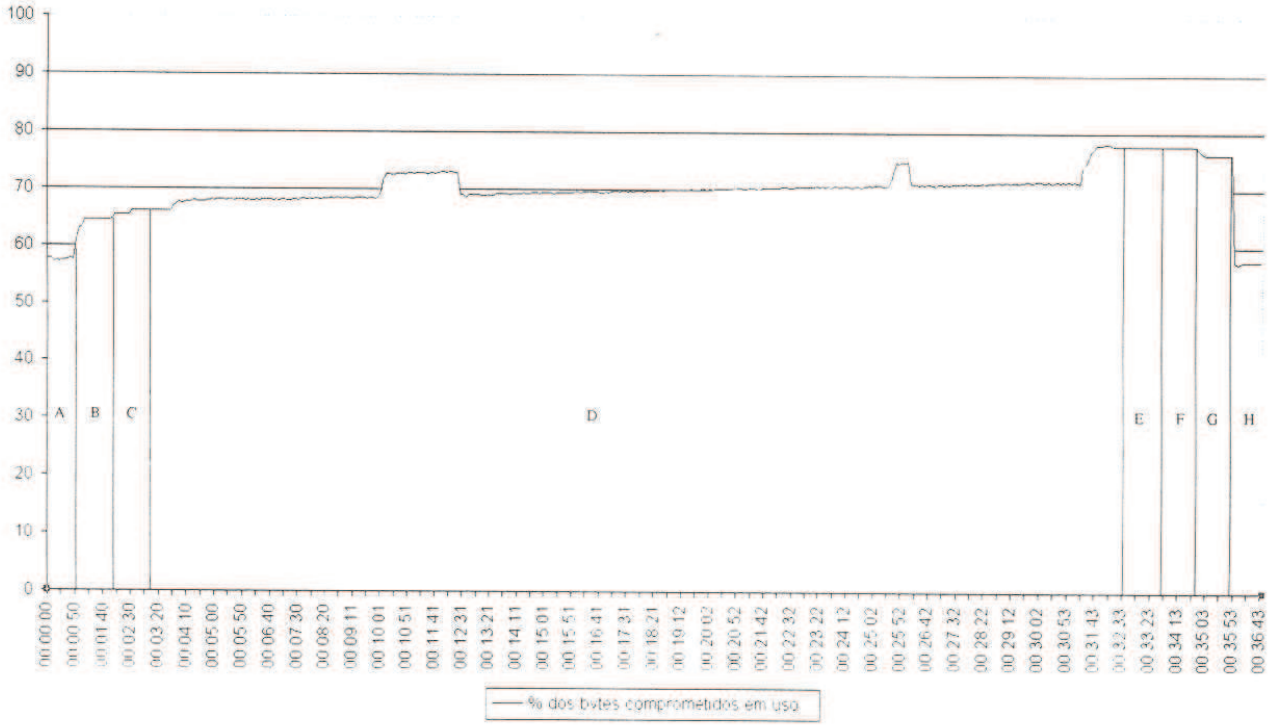

Figura 6.1: Etapas do Teste de Performance

Onde:

- A: espera inicial pela estabilizaçao do sistema;
- B: inicialização do servidor Aglets;
- C: instanciação dos agentes;
- D: início da busca

- E: exibição dos resultados;
- F: finalização dos agentes;
- G: encerramento do servidor Aglets;
- H: espea pela estabilização do sistema.

Vale ressaltar aqui que para os testes de performance o resultado das buscas é irrelevante, já que o objetivo é comparar o tempo que o sistema utiliza para percorrer todos os documentos da base em um teste. A base de documentos, idêntica para todos os testes, foi composta de 4627 documentos, com a consulta "*market stock record*".

No final desta seção, será apresentado um comparativo resumido dos testes em uma tabela, visando uma visualização comparativa mais eficiente.

6.1.1 Base Centralizada, Agentes Centralizados

Neste teste, o sistema de busca é executado em apenas uma máquina, com a base centralizada. Este teste foi realizado de duas maneiras:

- Sem os mecanismos de interação. Desta forma não há *overhead* associado para determinar se um documento foi lido ou não, nem mesmo de transporte dos resultados obtidos pela rede.
- Com mecanismos de interação entre os agentes, com o intuito de comparar-se os resultado obtidos com o teste anterior e determinar o impacto na performance do sistema dos mecanismos de interação empregados no MOTHRA.

Os dois gráficos do teste *standalone* na máquina chamada Laplace, onde foi executado o teste, mostram a memória comprometida do sistema (figura B.1)e o tempo de usuário no processador durante o teste (figura B.2), para o caso sem interação. O tempo total para a execução deste teste foi de 33 minutos e 58 segundos.

Em seguida, o mesmo teste é repetido, porém com os mecanismos de interação dos agentes em ação. Os resultados de uso de memória e de processador são apresentados nas figuras B.3 e B.4 respectivamente. Neste segundo teste, o tempo total foi de 47 minutos e 34 segundos, um aumento de 13 minutos e 36 segundos em relação ao teste sem os mecanismos de interação, ou 39,71%. Este tempo representa o overhead dos mecanismos de interação entre os agentes ao determinar se um documento já foi pesquisado.

Comparando os gráficos do uso de memória observa-se que os valores médios são os mesmos, em torno de 70%. Porém o tempo de usuário médio no processador é menor

(55,22% contra 63,01% no teste anterior), já que o agente passa por momentos de ociosidade enquanto espera o *timeout* para determinar se o documento já foi pesquisado.

6.1.2 Base Distribuída, Agentes Centralizados

Neste teste, a mesma base foi distribuída na rede, em três servidores, porém o agente ainda continuou centralizado em apenas uma das máquinas, acessando os documentos através do protocolo HTTP. O objetivo é verificar como a carga de rede influenciaria uma busca no caso da impossibilidade de um agente deslocar-se para o servidor no qual o documento se encontra e qual a carga na memória e em tempo do processador do servidor HTTP empregado.

A base foi distribuída em três máquinas da rede, uma das quais chamada Laplace, onde os agentes foram executados e uma parte da base estava hospedada, e outras duas chamadas Descartes e Platão, que apenas continham documentos. Os resultados obtidos para o uso de memória e tempo de usuário do processador nas três máquinas podem ser visualizados nas figuras B.5, B.6, B.7, B.8, B.9 e B.10.

O tempo total do teste foi de 36 minutos e 43 segundos, apenas 7,1% a mais que os 33 minutos e 58 segundos no teste em apenas uma máquina sem os mecanismos de interação. O tempo de processador da aplicação do servidor *web* foi irrelevante, ficando na média de 0,24% na máquina Descartes. Na máquina Platão verificou-se um pico de atividade durante um intervalo de tempo, porém esta atividade é atribuída a algum *daemon* do sistema operacional, já que os valores no restante do tempo foram similares aos obtidos na máquina Descartes.

6.1.3 Base Distribuída, Agentes Distribuídos

Como teste final da performance do sistema, a mesma base distribuída foi submetida a testes de busca por agentes que se deslocam pela rede, podendo criar clones e utilizar todos os mecanismos de interação, conforme as funcionalidades do agente de busca do protótipo do MOTHRA descrito anteriormente.

Desta maneira, o agente começa a busca na máquina Laplace, cria clones e atinge as máquinas Descartes e Platão, realizando assim a busca em paralelo. Os gráficos obtidos neste teste para as três máquinas podem ser visualizados nas figuras B.11, B.12, B.13, B.14, B.15 e B.16.

O tempo da busca neste terceiro e último teste foi de 21 minutos e trinta e sete segundos, uma redução significativa frente ao primeiro teste (36,31% mais rápido). Considerando que há intervalos de tempos fixos associados em ambos os testes, antes e depois da busca, a queda no tempo da busca é ainda maior.

O tempo de usuário no processador ficou com valores médios entre 33,97% (máquina Platão) e 40,26% (máquina Laplace). A memória comprometida ficou entre valores de 55,92% (máquina Descartes) e 69,39% (máquina Laplace). Com estes dados em mãos é possível se concluir alguns pontos importantes, que são apresentados a seguir.

6.1.4 Comparativo dos Resultados de Performance

A tabela abaixo fornece um comparativo direto dos dados obtidos nos testes de performance:

Teste	Memória Laplace	Memória Descartes	Memória Platão	Processador Laplace	Processador Descartes	Processador Platão	Tempo
Agente Centralizado Base Centralizada	70,08%	-	-	63,01%	-	-	33:58
Agente Centralizado Interativo Base Centralizada	69,81%	-	-	55,22%	-	-	47:34
Agente Centralizado Base Distribuída	69,98%	47,52%	58,13%	59,48	0,24%	1,36%	36:43
Agente Distribuído Interativo Base Distribuída	69,39%	55,92%	65,79%	40,26%	38,02%	33,97%	21:37

Pode-se analisar o impacto do uso de memória e processador nas abordagens aplicadas a partir destes dados. Comparando-se as abordagens centralizadas, o uso de memória e processador são semelhantes, não apresentando vantagem em termos a base centralizada ou distribuída.

Já no teste com a base e agentes distribuídos os valores são expressivamente melhorados. Há uma diminuição no uso da CPU, bem como o tempo para completar a tarefa é reduzido drasticamente.

O uso de memória para armazenar as avaliações dos documentos não é fator relevante, pois para em torno de 5000 documentos avaliados o uso médio de memória foi próximo (diferença de apenas 0,59%) ao empregado para armazenar 15000 avaliações, o que pode ser observado comparando-se o valores de uso de memória na máquina Laplace na abordagem com base distribuída e agentes centralizados com o obtido com tanto agentes e base distribuídos. O impacto maior em uso de memória fica a cargo do servidor da plataforma de agentes móveis, em torno de 8% nas máquinas Platão e Descartes.

6.2 Testes de Qualidade dos Resultados

Com o intuito de verificar a qualidade dos resultados fornecidos pelo sistema MOTHRA, uma base de 15534 documentos HTML foi preparada a partir de documentos da base textual TIPSTER [Uni]. Esta base, por questões de velocidade do teste, foi instalada em apenas uma máquina, um Pentium Celeron 366, com 64Mb de memória RAM e 4,3Gb de memória de disco, rodando Windows NT4.0 e Aglets 1.03b.

Pela variedade dos assuntos abordados foi eleita a base do jornal San José Mercury, documentos SJMN91-06300045 a SJMN91-06364024 (identificação única na base para cada documento). Estão disponíveis na TREC [NIS] tópicos nas bases da TIPSTER e suas relevâncias, julgadas por seres humanos, sendo que nestes resultados é que os testes detalhados nesta seção foram avaliados.

Quatro tópicos foram escolhidos, para 5 testes (um deles é repetido para demonstrar uma característica do mecanismo de busca), para os quais são utilizadas as medidas de precisão, recobrimento e F-Measure [Rij79]. A escolha de tais medidas é feita pela avaliação realizada por estas da qualidade dos resultados fornecidos e objetividade das mesmas. Para a *web* há a necessidade de se definir medidas que avaliem o resultado obtido em uma base desconhecida, já que tanto precisão e recobrimento requerem o conhecimento da base integral (o que é válido no caso da Tipster).

Por *precisão* entende-se a porcentagem de documentos corretos retornados na busca em relação a todos retornados, ou seja, quantos documentos corretos foram retornados em relação ao total dos documentos retornados pela busca. Por *recobrimento* entende-se a porcentagem de documentos corretos retornados na busca em relação a quantidade de documentos corretos esperados (para esta medida, é necessário conhecer-se a base e desta quais documentos atendem ao critério de busca).

A última medida, a *F-Measure*, dada por Rijsbergen [Rij79], relaciona as medidas de precisão e recobrimento por meio de uma média harmônica. Analisando o significado de cada uma das medidas, tem-se que se ambas tendem a 1 (100%), o resultado tende a ser melhor, ao passo que se ambas ou uma diminui, o resultado degrada. Rijsbergen relacionou ambas as medidas na seguinte fórmula, que determina a F-Measure:

$$F - Measure = \frac{2 * Rel * Prec}{Rel + Prec}$$

Assim, para obter-se um bom resultado, é preciso obter os melhores valores possíveis para a relevância e precisão, conforme já discutido por Barros [BGS98], já que quando ambos se aproximam de 1 (um) maior é o valor da F-Measure. A figura 6.2 exemplifica

o relacionamento entre os valores de precisão e recobrimento no cálculo da F-Measure.

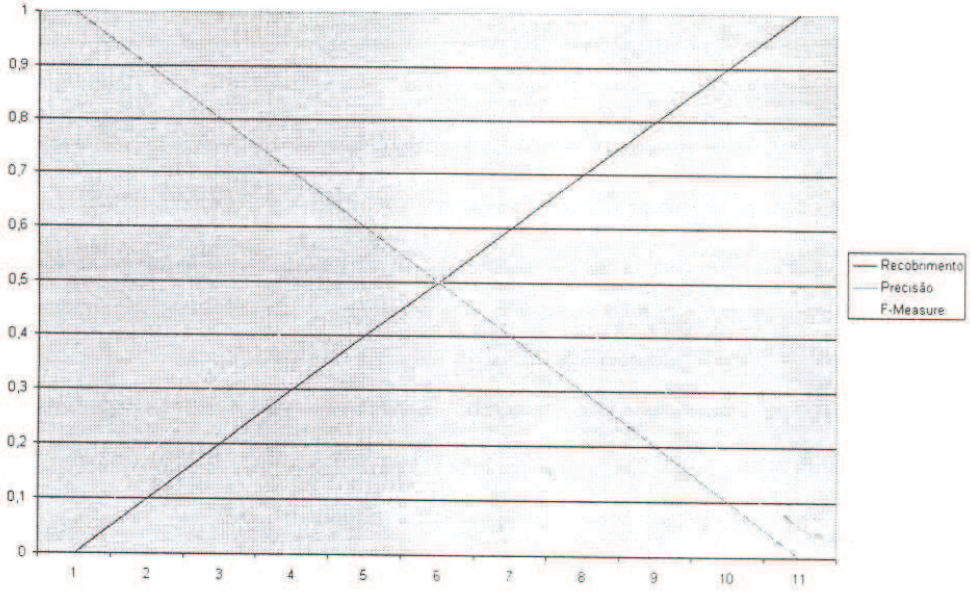

Figura 6.2: Relacionamento entre Precisão, Recobrimento e F-Measure

Os testes visam verificar a relevância das buscas feitas com o MOTHRA e demonstrar, para um efeito de comparação entre as buscas realizadas no teste e as avaliações feitas pela TREC (apresentadas a seguir), valores de relevância, precisão e F-Measure para futuramente estabelecer limites para os valores dos pesos dos documentos considerados como relevantes.

É importante ressaltar que os mecanimos de busca empregados no protótipo do MOTHRA são muito semelhantes à máquinas de busca como o Altavista [Alt], logo é esperado que documentos que não sejam realmente relevantes ao questionamento, mas contenham as palavras-chave esperadas, também sejam retornados com outros documentos que sejam relevantes à busca.

Uma maneira de se amenizar este problema é através da aplicação de um limite do peso dos documentos a serem visualizados. Isto faz-se necessário para efeitos de melhor visualização dos resultados podendo futuramente ser configurado no sistema pelo próprio usuário.

6.2.1 Tópicos e Relevâncias da TREC

A TREC (Text REtrieval Conference [NIS]) nos seus esforços para melhores mecanismos para busca e recuperação de informações, disponibiliza para pesquisadores tópicos (questionamentos) e suas relevâncias, avaliados por seres humanos.

Cada tópico contém uma descrição do domínio de conhecimento envolvido (ex: *international economics*), um título para o tópico (ex: *Airbus Subsidies*), um sumário,

uma descrição do tópico e uma narrativa que descreve o que deve conter um documento relevante à busca.

Como um complemento, cada tópico possui uma lista dos conceitos envolvidos; no exemplo de subsídios para a Airbus, encontram-se conceitos como tratados comerciais, subsídios federais, política *anti-dumping*, entre outros.

Como as buscas no protótipo do MOTHRA requerem apenas palavras-chave simples, foi escolhido o sumário do tópico, por fornecer uma pergunta suscinta e não envolver diretamente o uso de conceitos.

Para avaliar os resultados obtidos, foram comparados os documentos fornecidos pelo MOTHRA com os dados como corretos pela TREC em suas avaliações de relevância, sendo possível estabelecer uma avaliação correta de precisão, recobrimento e F-Measure nos documentos retornados em uma busca.

6.2.2 Tópico 52

Este tópico discute sanções internacionais aplicadas a África do Sul devido a política de discriminação racial do *apartheid*. Deste tópico, o sumário fornecido pela TREC é "*Document discusses international sanctions against South Africa*", a partir do qual foi criado o questionamento com palavras-chave "*international sanctions against South Africa*".

Na base de teste há 9 (nove) documentos que satisfazem o tópico, sendo estes 9 retornados corretamente pelo sistema MOTHRA. Como se esperava, documentos que contém as palavras-chave da busca, mas não se relacionam ao tópico, foram retornados, introduzindo ruído.

Abaixo, temos uma visão mais clara dos valores obtidos de precisão, recobrimento, F-Measure e número de documentos retornados frente ao número de documentos retornados corretamente, de acordo com o valor do peso limite aplicado (*threshold*). A figura 6.3 apresenta um gráfico relacionando os valores de precisão, recobrimento e F-Measure.

Peso	Precisão	Recobrimento	F-Measure	Total Retornados	Retornados Corretos
1	0.321428571	1	0.486486486	28	9
1,1	0.36	1	0.529411765	25	9
1,2	0.36	1	0.529411765	25	9
1,3	0.409090909	1	0.580645161	22	9
1,4	0,4	0,888888889	0.551724138	20	8
1,5	0,4375	0,777777778	0.56	16	7
1,6	0,454545455	0,555555556	0,5	11	5
1,7	0,333333333	0,333333333	0,333333333	9	3
1,8	1	0,222222222	0.363636364	2	2
1,9	1	0,222222222	0.363636364	2	2
2	1	0,222222222	0.363636364	2	2
0	0,003534957	1	0,00704501	2546	9

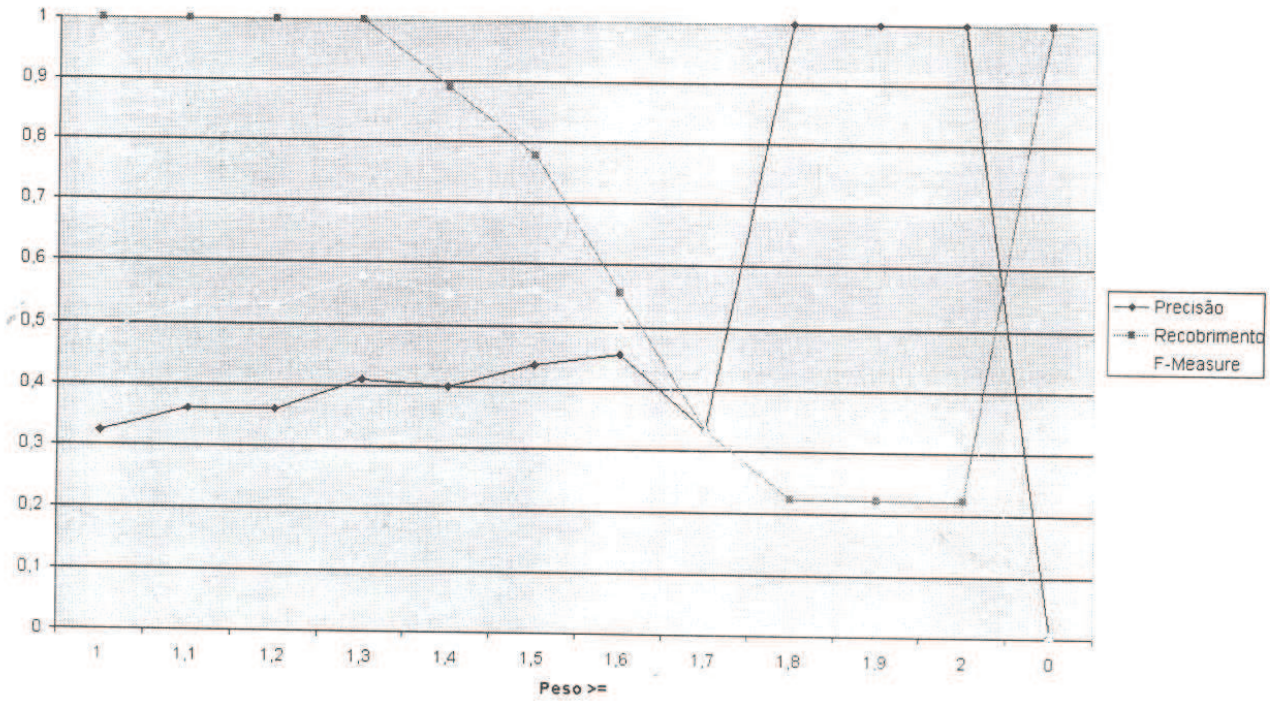

Figura 6.3: Resultados do Tópico 52

6.2.3 Tópico 51

Este segundo tópico escolhido discute disputas comerciais entre empresas de aeronáutica a partir de subsídios fornecidos pelo governo americano à Airbus Industrie. O sumário da TREC é "*Document will discuss government assistance to Airbus Industrie, or mention a trade dispute between Airbus and a U.S. aircraft producer over the issue of subsidies*". A partir deste resumo foi criado o questionamento com palavras-chave "*trade dispute between airbus industrie and aircraft producer over government subsidies*".

Em toda a base de teste há apenas um documento que satisfaz ao questionamento, o que torna interessante de observar o resultado obtido deste teste, já que a escolha das palavras-chave (como será demonstado claramente nos dois últimos testes) é fundamental para um bom resultado.

Este único documento foi retornado como quarto documento mais relevante da busca, e os vaslores das medidas obtidos no teste podem ser observados na tabela abaixo e na figura 6.4, como no teste anterior (novamente, o peso limite é o indicativo utilizado para determinar precisão, recobrimento e *F-Measure*).

Peso	Precisão	Recobrimento	F-Measure	Total Retornados	Retornados Corretos
0,5	0,001020408	1	0,002038736	980	1
0,6	0,003663004	1	0,00729927	273	1
0,7	0,005524862	1	0,010989011	181	1
0,8	0,027027027	1	0,052631579	37	1
0,9	0,05	1	0,095238095	20	1
1	0,2	1	0,333333333	5	1
1,03	0,25	1	0,4	4	1
0	0,00021097	1	0,000421852	4740	1

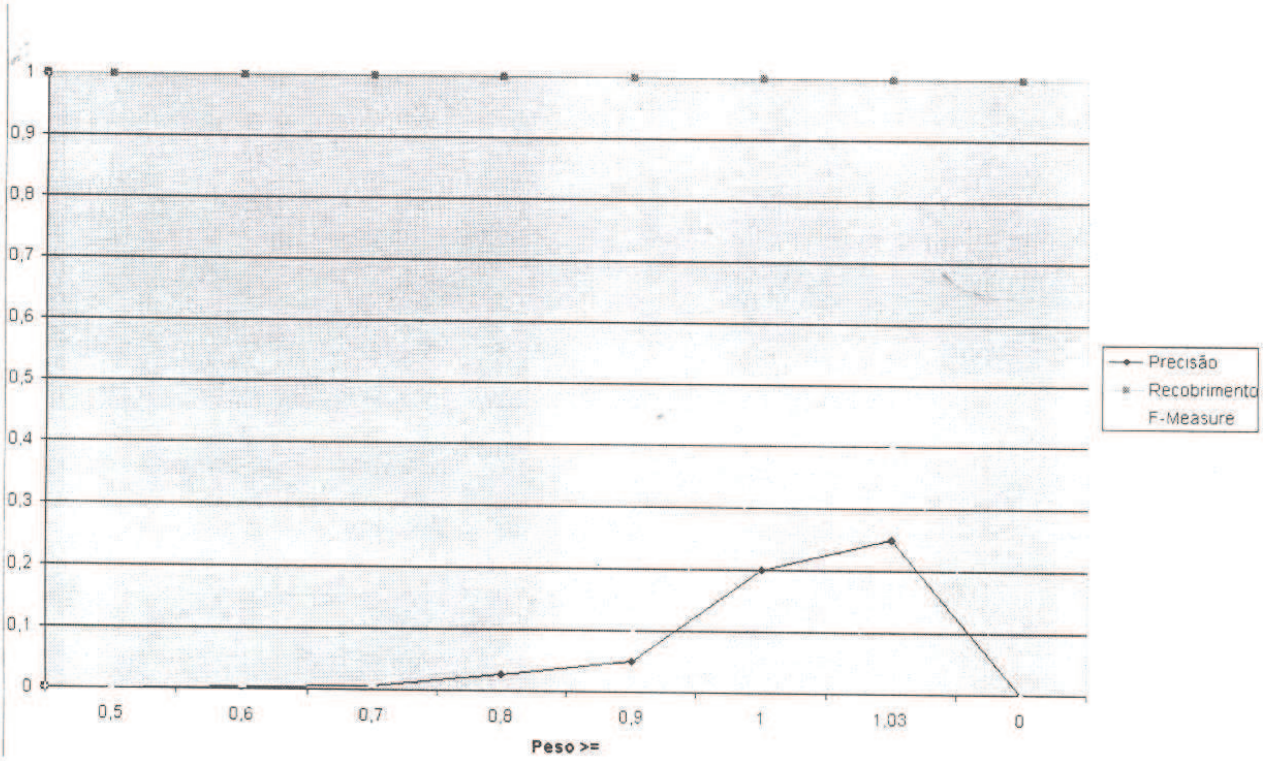

Figura 6.4: Resultados do Tópico 51

6.2.4 Tópico 59

Este tópico se relaciona com mortes causadas por fenômenos metereológicos, como furacões, enchentes, verões ou invernos intensos e outros. O sumário fornecido pela TREC é "*Document will report a type of weather event which has directly caused at least one fatality in some location*", dando origem ao questionamento empregado no MOTHRA "*weather event which has directly caused fatalities in some location*".

Tal busca apresenta um resultado regular, sendo que dos 42 documentos esperados apenas 33 foram encontrados. Isto deve-se ao fato de que as palavras-chave *weather event* não implicam nos tipos de fenômenos metereológicos esperados, como chuva, enchente, furacão e outros, ou seja, não há conhecimento de domínio envolvido.

Tal problema poderia ser resolvido com o uso de redes semânticas ou assistentes de ontologias, conforme proposto por Barros [BGS98], o que teoricamente deve melhorar

o resultado. Os próximos dois testes demonstram esse problema com mais clareza, modificando palavras-chave para melhorar o resultado obtido.

As medidas do teste com o tópico 59 podem ser visualizados na tabela abaixo e na figura 6.5.

Peso	Precisão	Recobrimento	F-Measure	Total Retornados	Retornados Corretos
0,4	0,013452915	0,785714286	0,026452906	2453	33
0,5	0,058823529	0,452380952	0,104109589	323	19
0,6	0,062091503	0,452380952	0,109195402	306	19
0,7	0,060185185	0,30952381	0,100775194	216	13
0,8	0,162162162	0,142857143	0,151898734	37	6
0,9	0,291666667	0,166666667	0,212121212	24	7
0	0,012961508	0,785714286	0,025502318	2546	33

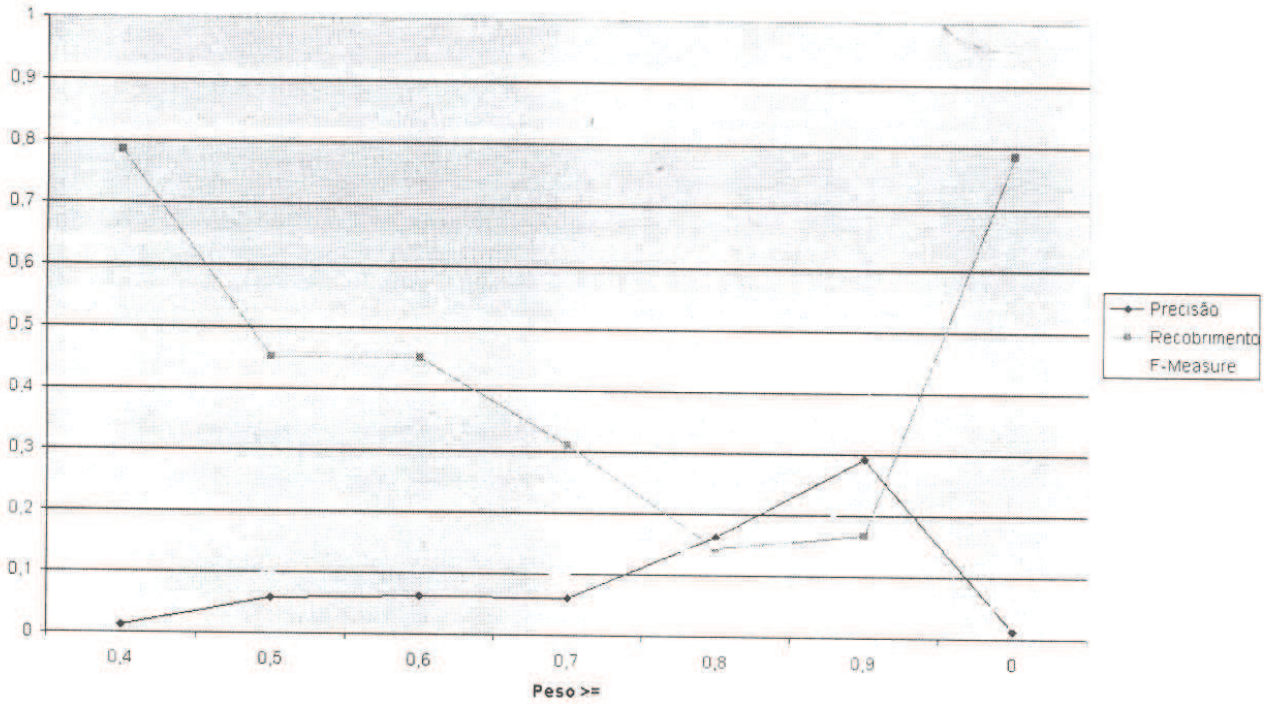

Figura 6.5: Resultados do Tópico 59

6.2.5 Tópico 54 - Primeiro Teste

Este tópico trata de reservas para o lançamento de um satélite comercial, sendo o sumário da TREC "*Document will cite the signing of a contract or preliminary agreement, or the making of a tentative reservation, to launch a commercial satellite*". Deste sumário, foi criado o questionamento "*contract or agreement or reservation to launch a commercial satellite*".

O resultado obtido neste primeiro teste com o tópico 54 não pode ser qualificado como bom. Os documento foram encontrados, classificados na terceira e vigésima quinta posições. Esta classificação requer do usuário do sistema muito tempo para acessar

diversos documentos em busca dos que realmente atendem os seus requerimentos (dos 25 melhores documentos, 23 são ruído).

Em seguida foi refeito o teste, porém com outras palavras-chave, com o objetivo de melhorar os resultados obtidos. As medidas obtidas neste teste podem ser observados na tabela abaixo e na figura 6.6.

Peso	Precisão	Recobrimento	F-Measure	Total Retornados	Retornados Corretos
0,6	0,005797101	1	0,011527378	345	2
0,7	0,010362694	1	0,020512821	193	2
0,78	0,08	1	0,148148148	25	2
0,8	0,055555556	0,5	0,1	18	1
0,9	0,1	0,5	0,166666667	10	1
1	0,333333333	0,5	0,4	3	1
0	0,000733945	1	0,001466813	2725	2

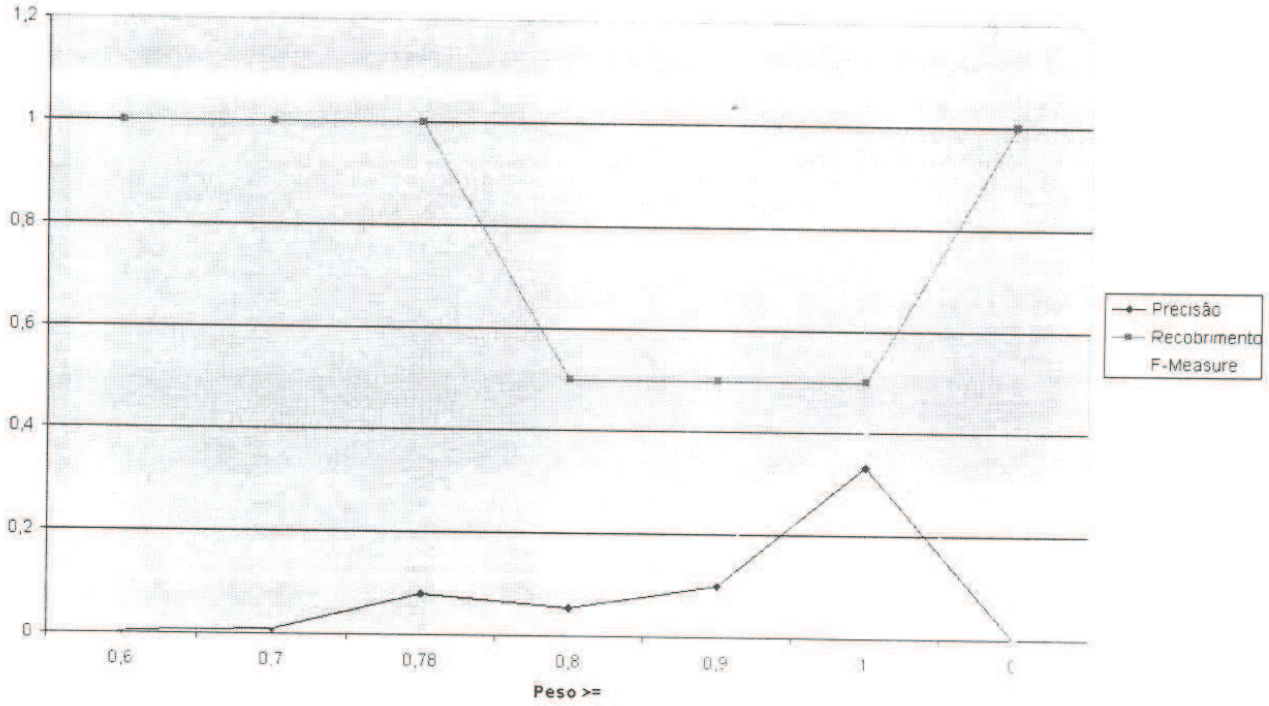

Figura 6.6: Resultados do Tópico 54 - Primeiro Teste

6.2.6 Tópico 54 - Segundo Teste

Para este teste, foram reformuladas as palavras chave de "*contract or agreement or reservation to launch a commercial satellite*" para "*reservation to launch a commercial satellite*". Tal mudança foi feita baseada em uma inspeção na base de documentos. Tal procedimento é possível nestes testes por existir uma avaliação de relevâncias dos documentos da base. Em casos como a *web*, tal tarefa não é possível. Nesta avaliação a inspeção foi feita a título de ilustrar a influência da escolha das palavras-chave com a qualidade do resultado fornecido pelo sistema.

Todos os documentos corretos nesta base parcial da TIPSTER tratam apenas da reserva para o lançamento de satélites comerciais, não havendo necessidade das palavras "*contract*" e "*agreement*" no questionamento, tornando-o mais restrito. Os documentos corretos foram classificados na primeira e quarta posições neste teste, representando uma melhora em relação ao primeiro teste com este tópico. As medidas obtidas com ests teste podem ser observados na tabela abaixo e na figura 6.7.

Peso	Precisão	Recobrimento	F-Measure	Total Retornados	Retornados Corretos
0,9	0,02247191	1	0,043956044	89	2
1	0,5	1	0,666666667	4	2
1,1	0,5	1	0,666666667	4	2
1,2	0,333333333	0,5	0,4	3	1
1,7	0,333333333	0,5	0,4	3	1
2	1	0,5	0,666666667	1	1
0	0,001212121	1	0,002421308	1650	2

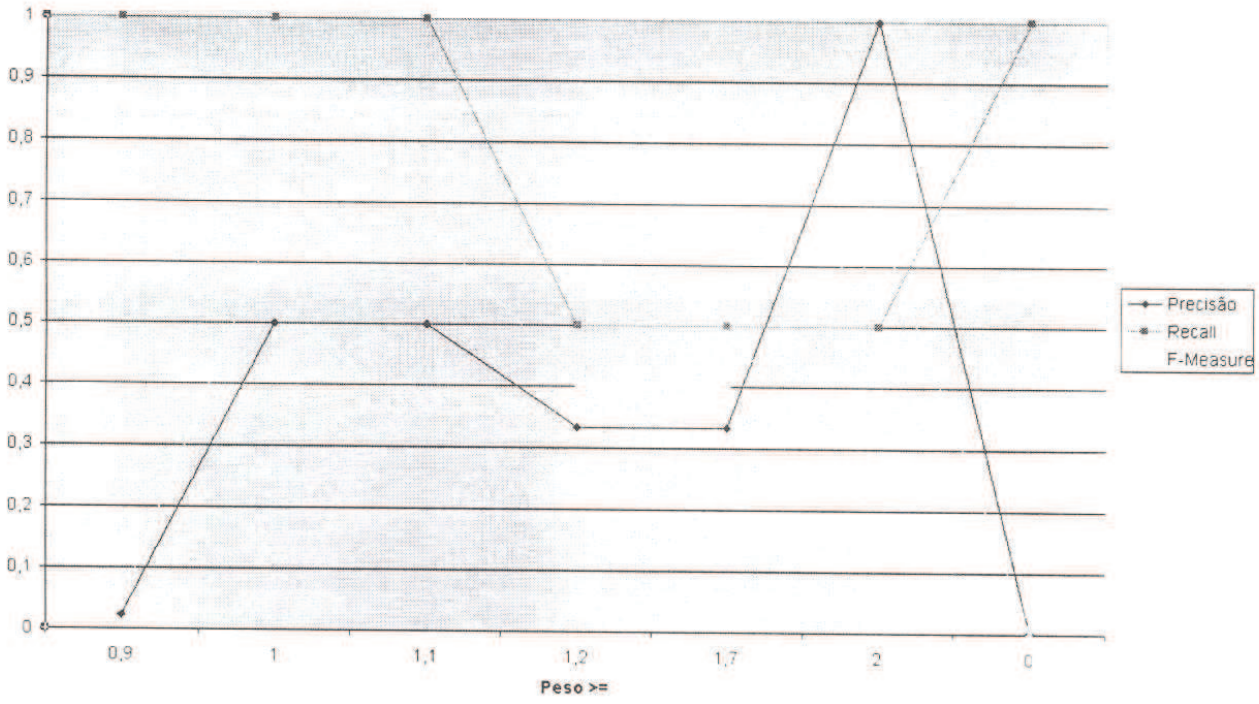

Figura 6.7: Resultados do Tópico 54 - Segundo Teste

6.3 Teste das Pegadas

Para avaliar a efetividade das pegadas para evitar a redundância na avaliação dos documentos, foi preparada uma base especial, distribuída em dois servidores na configuração apresentada na figura 6.8.

Todos os documentos possuem o mesmo conteúdo textual com o objetivo de que todos os documentos sejam retornados, possibilitando uma melhoor visualização da

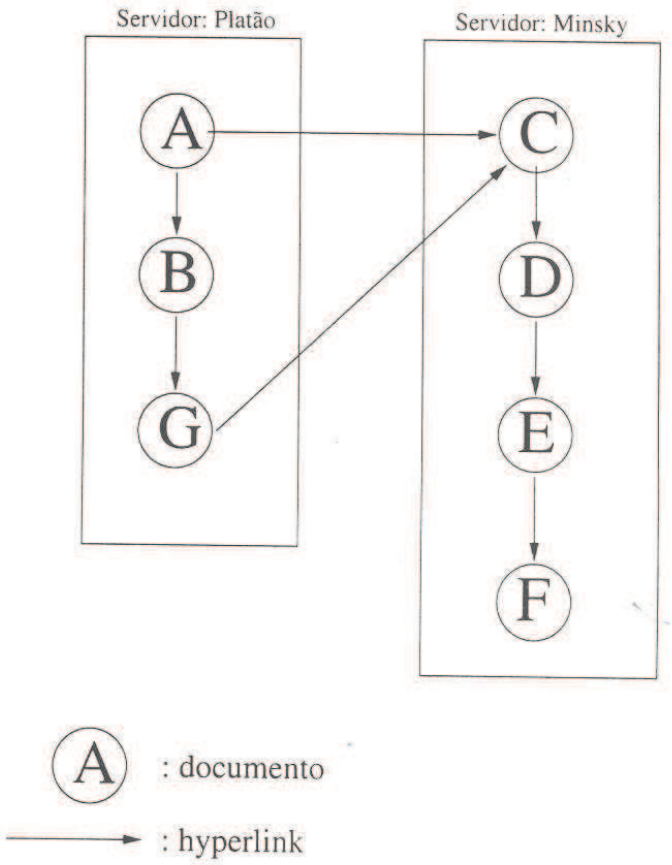

Figura 6.8: Distribuição dos Documentos

redundância. Ao executar-se o teste a partir do documento A com as pegadas, os documentos foram inspecionados uma única vez, ao passo que ao executar o mesmo teste sem as pegadas, os documentos C, D e E e F foram pesquisados duas vezes, uma vez pela instância de agente que iniciou a busca no servidor Platão e outra pela instancia que foi criada para verificar os documentos no servidor Minsky.

Observou-se nos testes de qualidade dos resultados que houve a eliminação de redundancias devido às pegadas. Conjectura-se que em um ambiente com muitas máquinas, como a Internet, o ganho em processamento não deperdiçado ao evitar-se a avaliação redundante de um documento seja considerável.

6.4 Conclusões

Em relação à performance do sistema, este comportou-se como se espera de um sistema multi-processado, ocorrendo uma diminuição do tempo para realizar a tarefa e a distribuição da carga de processamento nas máquinas envolvidas na tarefa.

Já em relação à qualidade dos resultados, é possível observar que mesmo com um mecanismo simples o sistema fornece resultados consistentes com os esperados (obtidos por avaliação humana), ficando aberto o caminho para o teste de novos mecanismos

mais sofisticados de busca que podem vir a fornecer resultados melhores, tirando vantagem da busca distribuída pela rede.

Capítulo 7

Conclusões e Trabalhos Futuros

Neste trabalho apresentou-se a proposta MOTHRA, um sistema de busca e recuperação de documentos baseado em agentes móveis. Uma característica, que ainda é pouco explorada na área de Recuperação de Informações, é o uso de agentes móveis, em especial com o emprego das pegadas para resolver a interação entre os agentes durante o seu deslocamento.

A proposta MOTHRA mostrou-se capaz de realizar a busca de documentos em uma rede, fornecendo resultados válidos de maneira similar a uma máquina de busca tradicional na Internet. A combinação de técnicas empregadas, Inteligência Artificial, *Text Retrieval* e agentes móveis possibilitou a proposição e avaliação no protótipo de um mecanismo não-convencional para o problema da busca e recuperação de informações em uma rede de grande porte. A principal característica desejável do sistema que foi a de reduzir o tráfego de dados na rede foi atingida, já que os agentes levam consigo informações reduzidas sobre a avaliação do documento e não realizam comunicação ponto-a-ponto, o que é possível pelo uso das pegadas.

As pegadas apresentam-se como uma alternativa para a comunicação em problemas de sistemas multi-agente em redes de grande porte, com uma característica importante, que é a comunicação assíncrona. A pegada existe durante um determinado intervalo e não possui restrições quanto ao número de acessos, podendo ser acessada por diferentes agentes em diferentes momentos.

Os resultados obtidos nas buscas foram consistentes com os indicados pela TREC, sendo o maior fator limitante à qualidade do resultado as palavras-chave empregadas para avaliar o documento, que devem abranger ao máximo os tópicos desejados, fato este demonstrado no teste do tópico 54, realizado duas vezes com dois questionamentos diferentes. Neste campo, uma pesquisa futura fica a cargo do uso de sistemas de ontologias para refinar o questionamento.

Além disso, futuramente poderão ser estendidos os trabalhos iniciados com o teste de novos mecanismos de avaliação distribuídos, procurando aliar técnicas de avaliação de documentos mais eficazes para a busca de documentos em uma rede desconhecida, como a possibilidade de a quantidade de *links* para um dado documento indicar a sua qualidade. Outra medida que deve ser pesquisada e deverá propiciar uma melhora significativa na qualidade do resultado da busca é o emprego de frequência dos termos inter-documento.

Outro campo de pesquisa possível é o aprendizado do sistema, em como montar a base de conhecimento prévio que irá determinar, de acordo com as palavras-chave fornecidas, os documentos iniciais da busca, que no protótipo consiste de apenas uma URL fornecida pelo usuário do sistema.

Apêndice A

Resultados Parciais do Tópico 52

Nesta seção é apresentado um exemplo do resultado fornecido pelo sistema MOTHRA na busca realizada com o tópico 52 da TIPSTER. Este resultado é parcial devido ao volume de documentos retornados, porém demonstra o resultado do peso do documento, o número de ocorrências de cada palavra-chave e uma indicação se o documento é relevante (1) ou não (0) ao questionamento. O campo da URL do documento foi suprimido por não ser necessário nesta demonstração, apenas no próprio sistema.

Documento	Peso	intern	sanction	south	africa	Relevante?
SJMN91-06340188	3,914925901	2	4	3	4	1
SJMN91-06340094	2,115322611	1	2	4	4	1
SJMN91-06357133	1,730718500	1	1	4	3	1
SJMN91-06320206	1,728060024	1	0	1	1	0
SJMN91-06323195	1,728060024	1	1	1	0	0
SJMN91-06330128	1,728060024	1	0	1	1	0
SJMN91-06336211	1,728060024	1	0	1	1	0
SJMN91-06343227	1,728060024	1	0	1	1	0
SJMN91-06349104	1,728060024	1	0	1	1	0
SJMN91-06341072	1,605419059	2	2	8	9	1
SJMN91-06309090	1,602710767	3	0	5	4	1
SJMN91-06337086	1,532285733	2	1	7	7	1
SJMN91-06301179	1,505740731	0	2	1	2	0
SJMN91-06329298	1,505740731	2	0	1	2	0
SJMN91-06349143	1,505740731	0	1	2	2	1
SJMN91-06352124	1,505740731	1	0	2	2	0
SJMN91-06315129	1,413547210	1	1	6	4	1
SJMN91-06343235	1,412216387	2	0	1	1	0

SJMN91-06350084	1,412216387	1	1	0	2	0
SJMN91-06351135	1,412216387	2	0	1	1	0
SJMN91-06338135	1,346632309	0	2	6	6	1
SJMN91-06302140	1,306024118	1	7	4	1	0
SJMN91-06344215	1,264111570	1	0	4	3	0
SJMN91-06362193	1,264111570	4	0	1	3	0
SJMN91-06327241	1,212090175	1	0	5	4	0
SJMN91-06312161	1,098544249	1	0	10	8	0
SJMN91-06354240	1,095045261	4	7	0	1	0
SJMN91-06301122	1,083211871	1	0	10	7	0
SJMN91-06356035	0,999777827	0	0	9	9	0
SJMN91-06357064	0,999714367	0	0	7	7	0
SJMN91-06309003	0,999500250	0	0	4	4	0
SJMN91-06309009	0,999500250	0	0	4	4	0
SJMN91-06327177	0,999500250	4	0	0	4	0
SJMN91-06347340	0,999500250	0	0	4	4	0
SJMN91-06320205	0,999333777	0	0	3	3	0
SJMN91-06334129	0,999333777	0	0	3	3	0
SJMN91-06343124	0,999333777	4	0	1	1	0
SJMN91-06304264	0,999000999	2	2	0	0	0
SJMN91-06316073	0,999000999	0	0	2	2	0
SJMN91-06316121	0,999000999	0	0	2	2	0
SJMN91-06318246	0,999000999	2	2	0	0	0
SJMN91-06322185	0,999000999	2	0	2	0	0
SJMN91-06323154	0,999000999	0	0	2	2	0
SJMN91-06329084	0,999000999	0	0	2	2	0
SJMN91-06330089	0,999000999	0	0	2	2	0
SJMN91-06336021	0,999000999	2	0	2	0	0
SJMN91-06344077	0,999000999	0	0	2	2	0
SJMN91-06355248	0,999000999	0	0	2	2	0
SJMN91-06362125	0,999000999	2	0	2	0	0
SJMN91-06301050	0,998003992	1	0	1	0	0
SJMN91-06301055	0,998003992	0	1	1	0	0
SJMN91-06302094	0,998003992	1	1	0	0	0
SJMN91-06302119	0,998003992	1	0	1	0	0
SJMN91-06303061	0,998003992	1	0	0	1	0
SJMN91-06303111	0,998003992	1	0	1	0	0
SJMN91-06303165	0,998003992	1	0	1	0	0
SJMN91-06304178	0,998003992	1	0	1	0	0

SJMN91-06304266	0,998003992	1	0	1	0	0
SJMN91-06305104	0,998003992	1	0	1	0	0
SJMN91-06305186	0,998003992	1	0	1	0	0
SJMN91-06305205	0,998003992	1	1	0	0	0
SJMN91-06306002	0,998003992	1	0	1	0	0
SJMN91-06306174	0,998003992	1	1	0	0	0
SJMN91-06308096	0,998003992	0	0	1	1	0
SJMN91-06308140	0,998003992	1	0	1	0	0
SJMN91-06309023	0,998003992	0	0	1	1	0
SJMN91-06309027	0,998003992	1	1	0	0	0
SJMN91-06309095	0,998003992	0	1	1	0	0
SJMN91-06309096	0,998003992	0	0	1	1	0
SJMN91-06310059	0,998003992	0	0	1	1	0
SJMN91-06310076	0,998003992	1	0	1	0	0
SJMN91-06310143	0,998003992	0	0	1	1	0
SJMN91-06311031	0,998003992	0	0	1	1	0
SJMN91-06311063	0,998003992	1	1	0	0	0
SJMN91-06311113	0,998003992	0	1	1	0	0
SJMN91-06311166	0,998003992	1	0	1	0	0
SJMN91-06311315	0,998003992	1	0	1	0	0
SJMN91-06312076	0,998003992	0	0	1	1	0
SJMN91-06312091	0,998003992	0	0	1	1	0
SJMN91-06312146	0,998003992	1	0	1	0	0
SJMN91-06314039	0,998003992	1	1	0	0	0
SJMN91-06314089	0,998003992	1	0	1	0	0
SJMN91-06315025	0,998003992	0	0	1	1	0
SJMN91-06315122	0,998003992	1	0	1	0	0
SJMN91-06315154	0,998003992	0	0	1	1	0
SJMN91-06316012	0,998003992	1	0	1	0	0
SJMN91-06316115	0,998003992	1	0	1	0	0
SJMN91-06317007	0,998003992	1	0	1	0	0
SJMN91-06317105	0,998003992	0	0	1	1	0
SJMN91-06317166	0,998003992	1	0	1	0	0
SJMN91-06317189	0,998003992	0	0	1	1	0
SJMN91-06318063	0,998003992	0	0	1	1	0
SJMN91-06318190	0,998003992	1	1	0	0	0
SJMN91-06319021	0,998003992	1	0	1	0	0
SJMN91-06319096	0,998003992	1	0	1	0	0
SJMN91-06319189	0,998003992	1	0	1	0	0

SJMN91-06319213	0,998003992	0	0	1	1	0
SJMN91-06320130	0,998003992	1	0	1	0	0
SJMN91-06320173	0,998003992	0	0	1	1	0
SJMN91-06320218	0,998003992	1	0	1	0	0
SJMN91-06322029	0,998003992	1	0	0	1	0
SJMN91-06322118	0,998003992	1	0	1	0	0
SJMN91-06323024	0,998003992	1	0	1	0	0
SJMN91-06323176	0,998003992	1	0	1	0	0
SJMN91-06324021	0,998003992	1	0	1	0	0
SJMN91-06324057	0,998003992	1	1	0	0	0
SJMN91-06324138	0,998003992	1	1	0	0	0
SJMN91-06324154	0,998003992	0	0	1	1	0
SJMN91-06325174	0,998003992	1	0	1	0	0
SJMN91-06326024	0,998003992	1	0	1	0	0
SJMN91-06326098	0,998003992	1	0	0	1	0
SJMN91-06326113	0,998003992	0	0	1	1	0
SJMN91-06326141	0,998003992	1	0	0	1	0
SJMN91-06326221	0,998003992	1	0	1	0	0
SJMN91-06327002	0,998003992	0	0	1	1	0
SJMN91-06329094	0,998003992	1	1	0	0	0
SJMN91-06329132	0,998003992	1	0	1	0	0
SJMN91-06329288	0,998003992	1	0	1	0	0
SJMN91-06330032	0,998003992	1	1	0	0	0
SJMN91-06330036	0,998003992	0	0	1	1	0
SJMN91-06330058	0,998003992	1	0	1	0	0
SJMN91-06330091	0,998003992	1	1	0	0	0
SJMN91-06330173	0,998003992	1	0	1	0	0
SJMN91-06332280	0,998003992	1	1	0	0	0
SJMN91-06332297	0,998003992	1	0	1	0	0
SJMN91-06333075	0,998003992	1	0	1	0	0
SJMN91-06333092	0,998003992	1	0	1	0	0
SJMN91-06333153	0,998003992	1	0	1	0	0
SJMN91-06333187	0,998003992	1	1	0	0	0
SJMN91-06334044	0,998003992	1	1	0	0	0
SJMN91-06334111	0,998003992	1	0	1	0	0
SJMN91-06336204	0,998003992	1	0	1	0	0
SJMN91-06336209	0,998003992	1	1	0	0	0
SJMN91-06337057	0,998003992	1	0	0	1	0
SJMN91-06339099	0,998003992	1	0	1	0	0

Apêndice B

Gráficos dos Testes de Performance

Esta seção apresenta os gráficos obtidos nos tests de performance do sistema MOTHRA. Tais gráficos visam avaliam duas medidas:

- Uso de memória: memória (global) alocada no sistema;
- Uso de processador: quanto do tempo disponível para processos do usuário no processador foi comprometido.

A descrição dos testes e de como devem ser interpretar os resultados em relação ao eixo do tempo é ecnontrada no capítulo 6.

B.0.1 Base Centralizada, Agentes Centralizados

B.0.2 Base Distribuída, Agentes Centralizados

B.0.3 Base Distribuída, Agentes Distribuídos

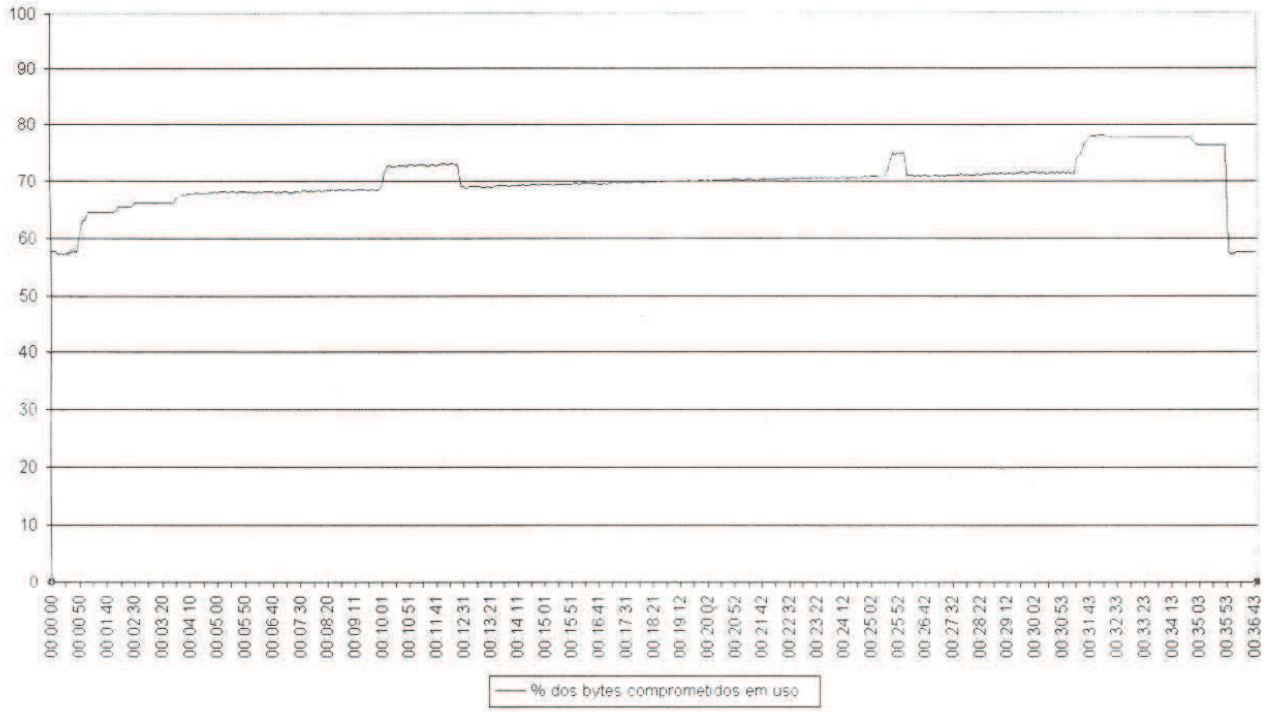

Figura B.1: Máquina Laplace – % dos bytes comprometidos em uso

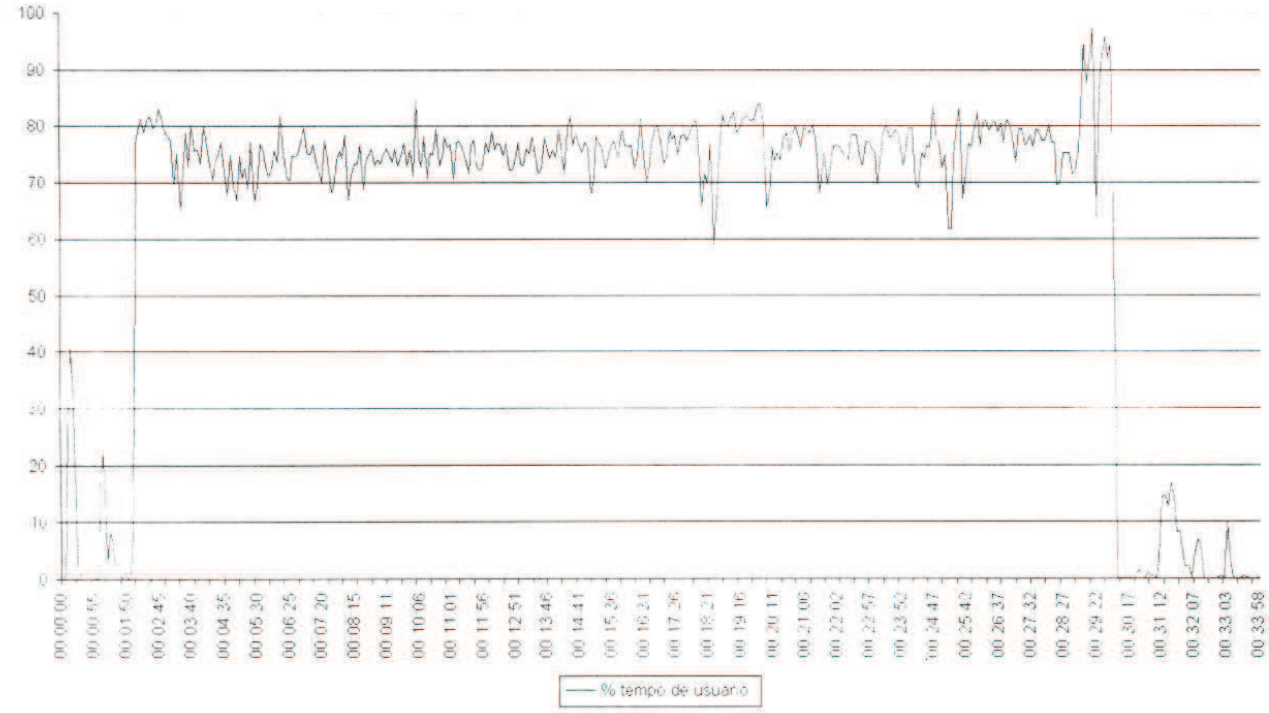

Figura B.2: Máquina Laplace – % do tempo de usuário no processador

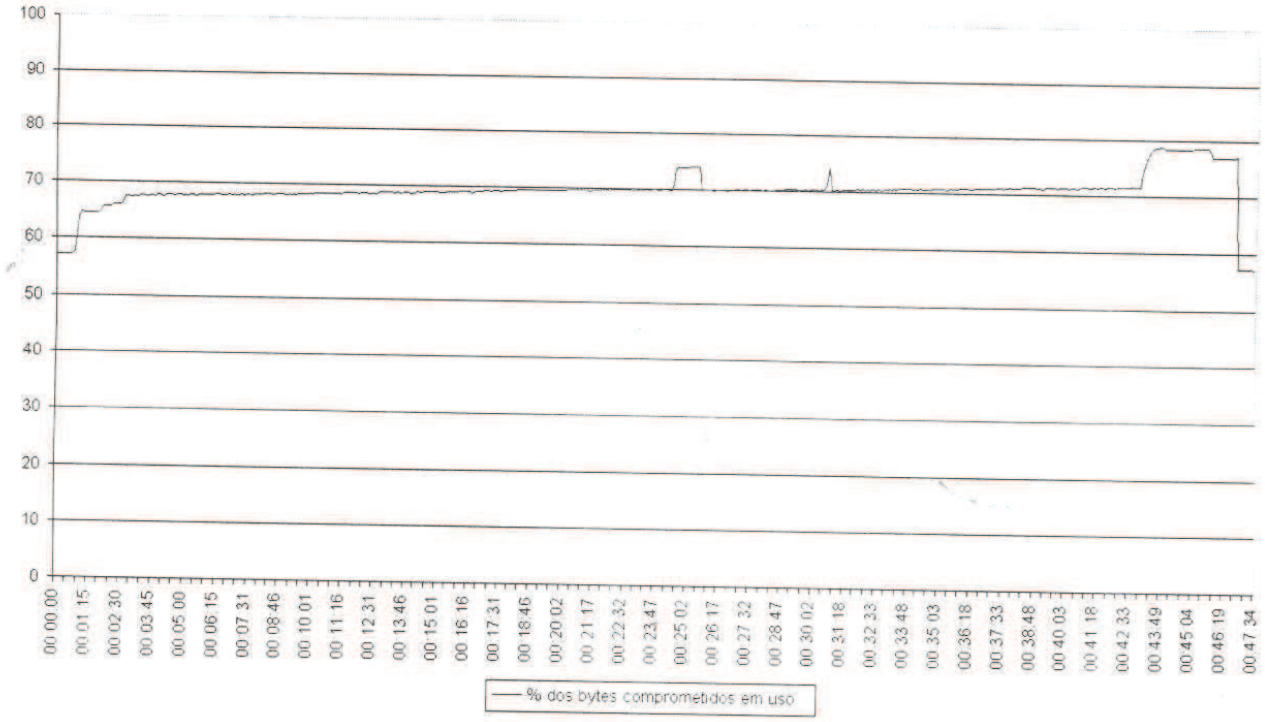

Figura B.3: Máquina Laplace – % dos bytes comprometidos em uso

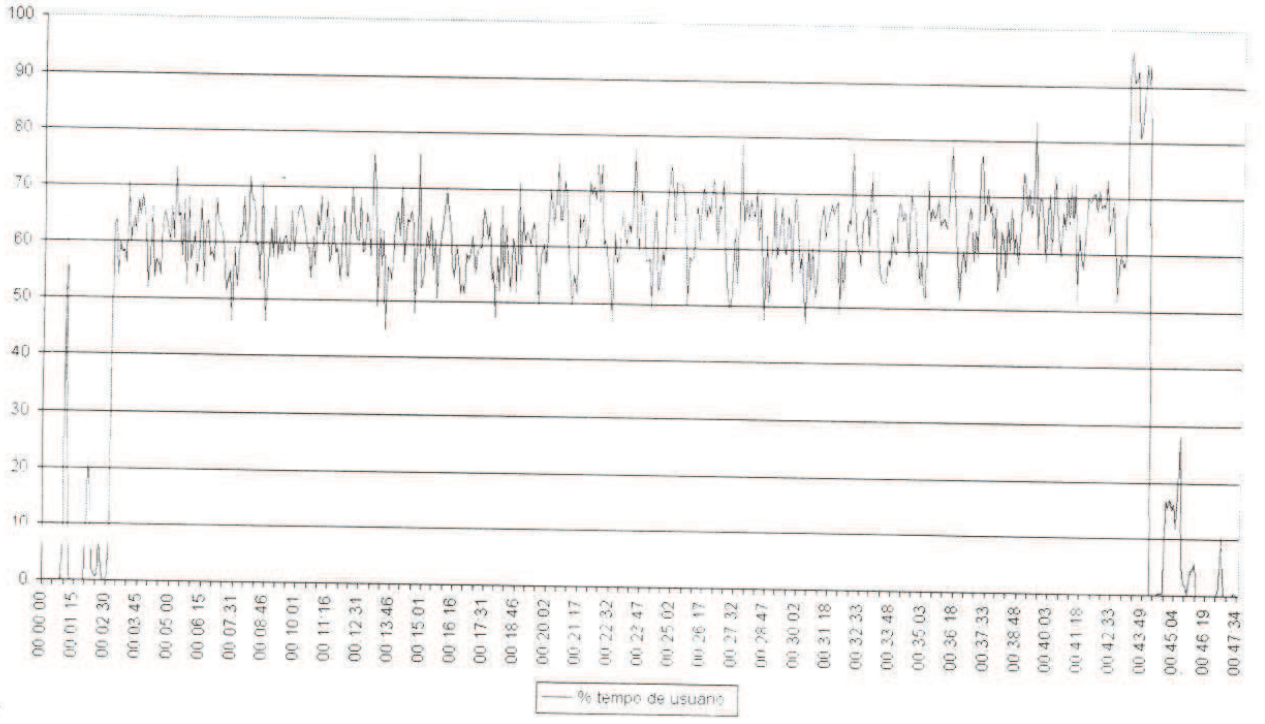

Figura B.4: Máquina Laplace – % do tempo de usuário no processador

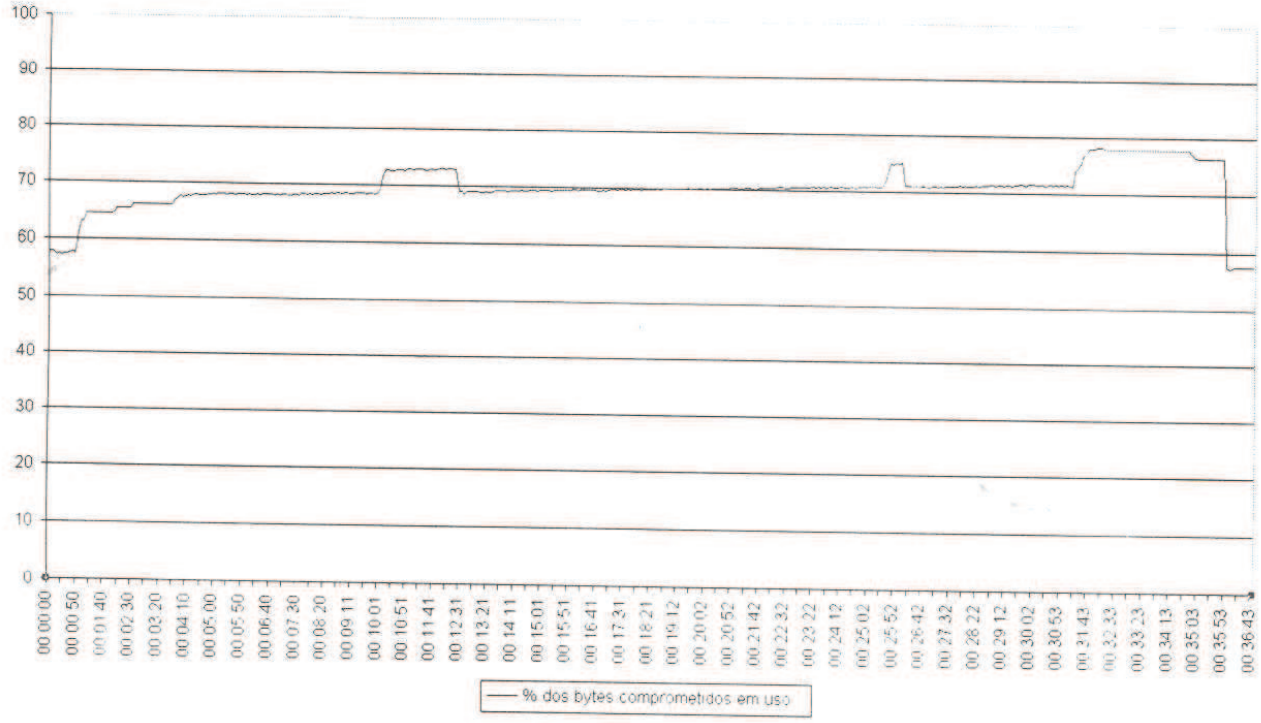

Figura B.5: Máquina Laplace – % dos bytes comprometidos em uso

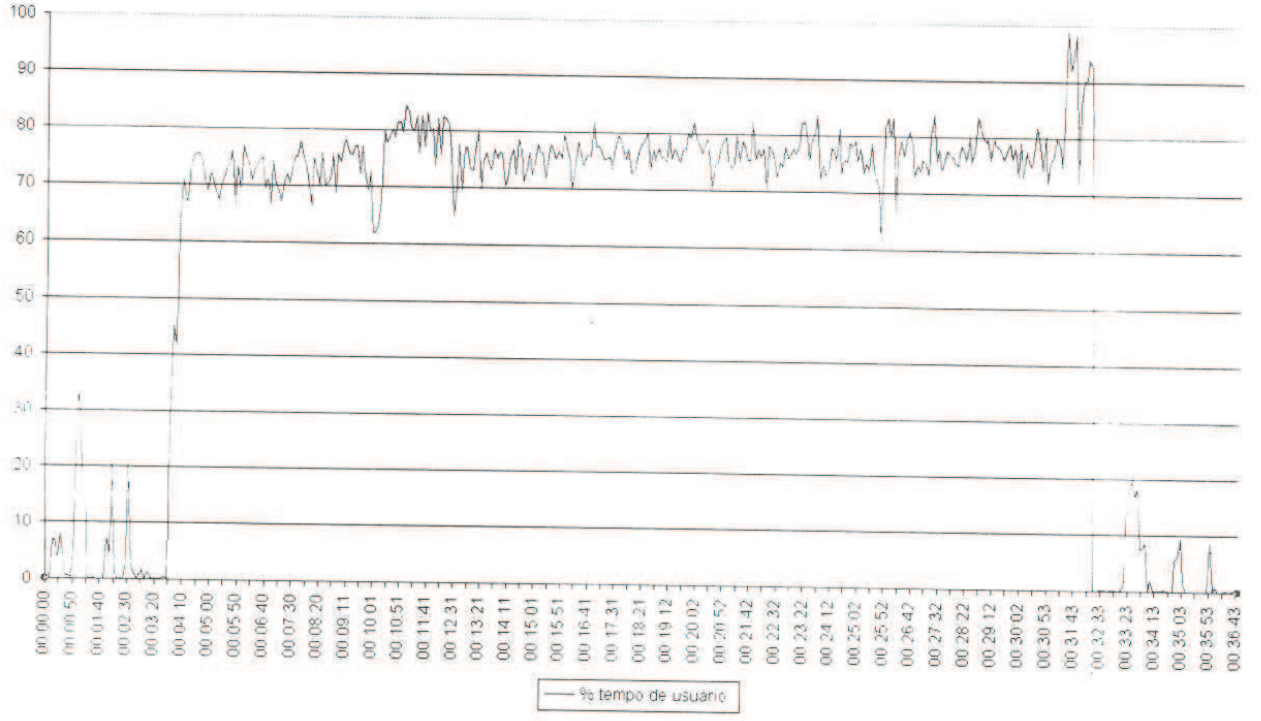

Figura B.6: Máquina Laplace – % do tempo de usuário no processador

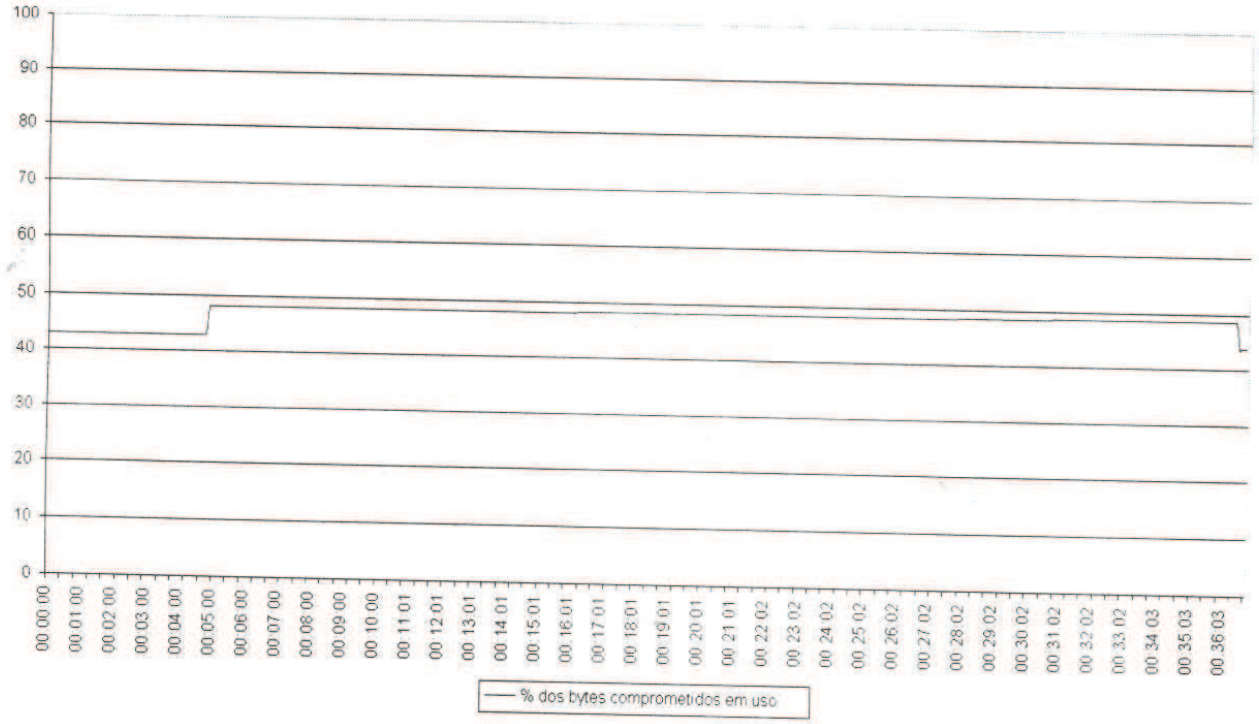

Figura B.7: Máquina Descartes – % dos bytes comprometidos em uso

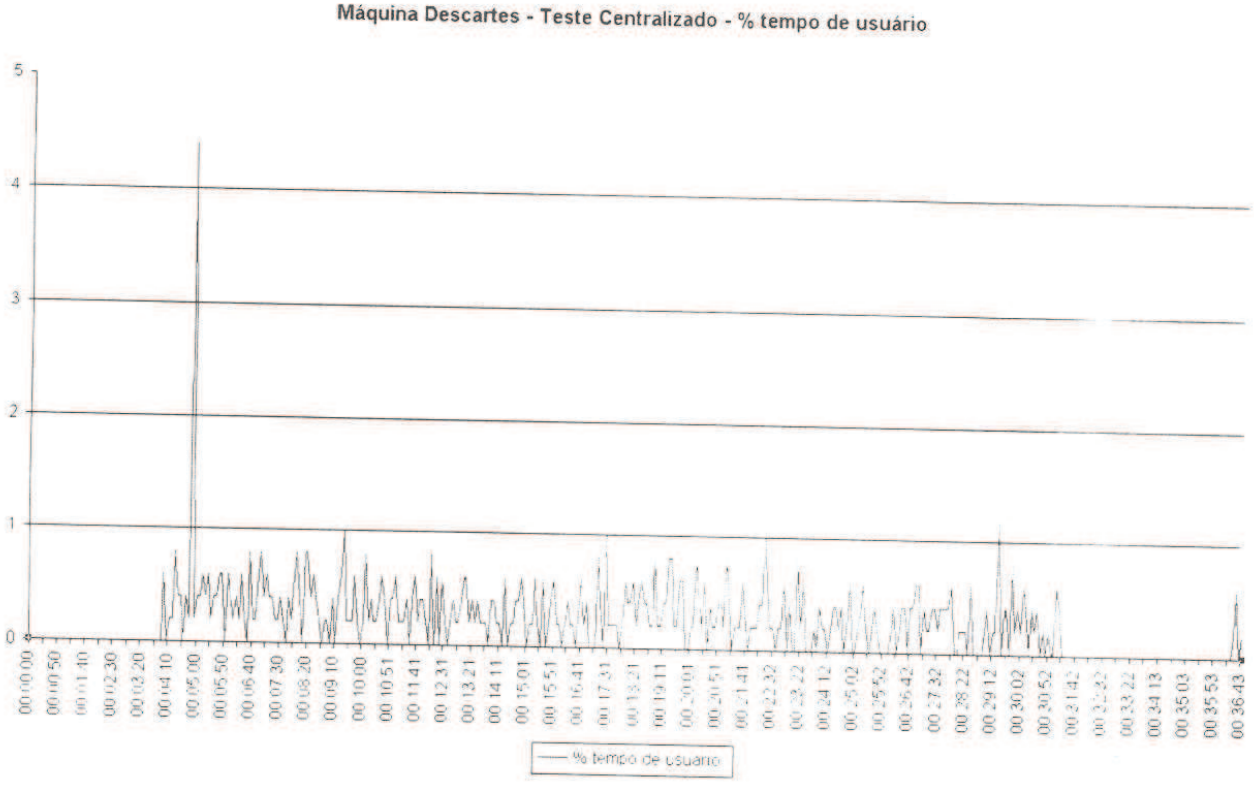

Figura B.8: Máquina Descartes – % do tempo de usuário no processador

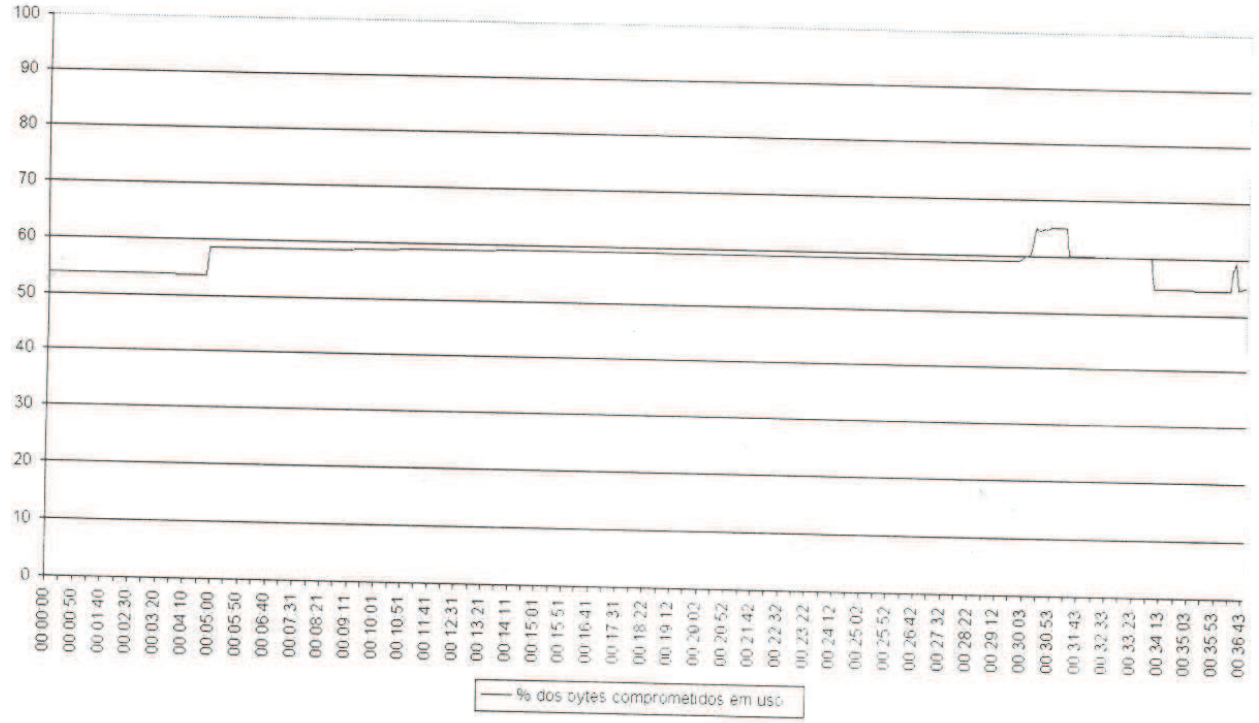

Figura B.9: Máquina Platao – % dos bytes comprometidos em uso

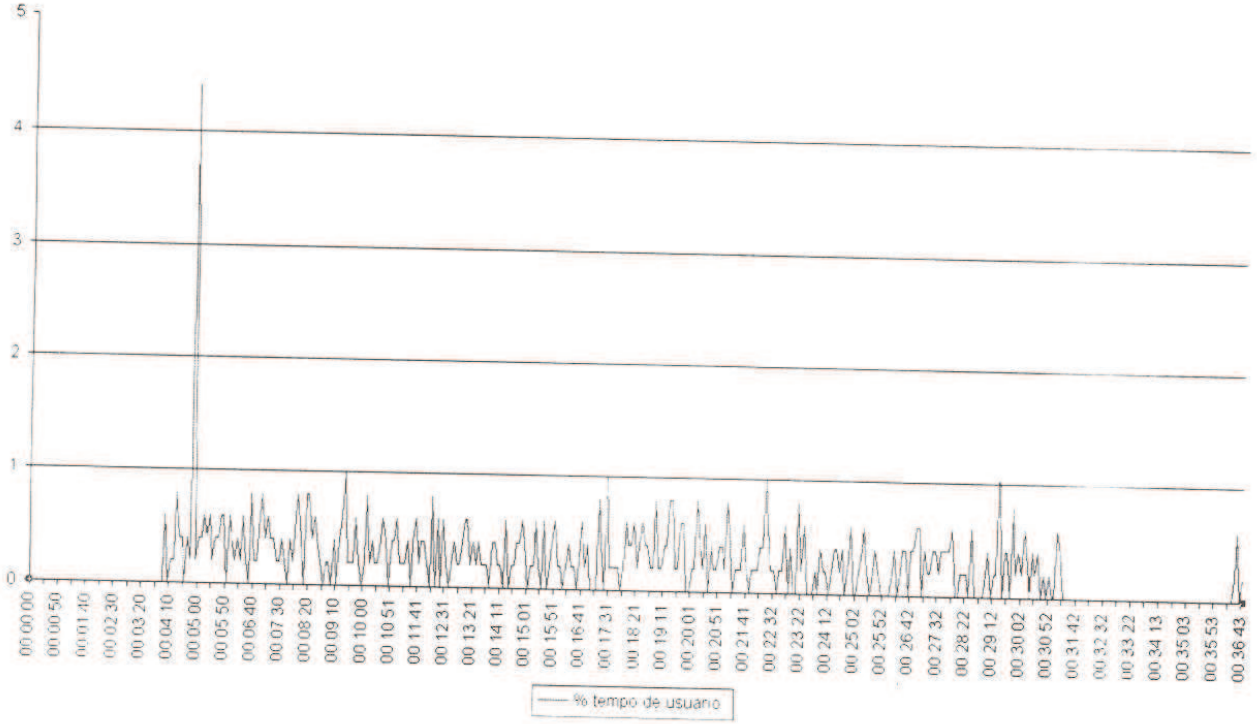

Figura B.10: Máquina Platao – % do tempo de usuário no processador

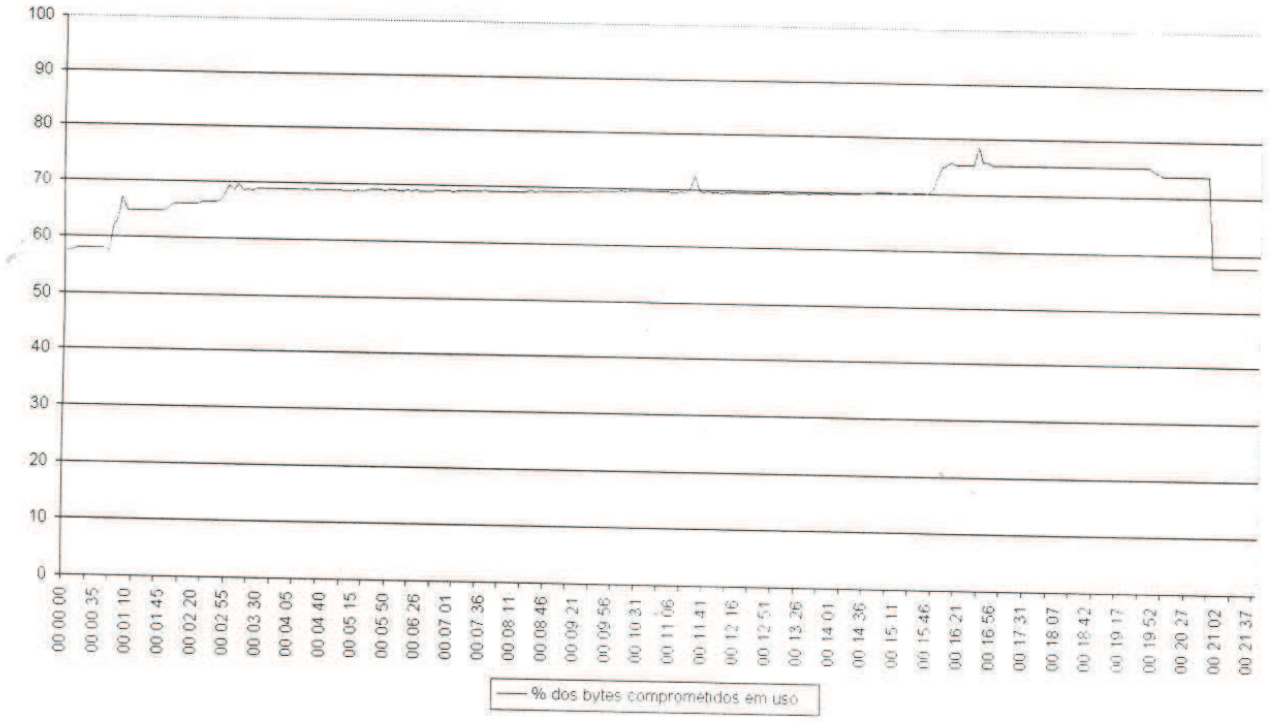

Figura B.11: Máquina Laplace – % dos bytes comprometidos em uso

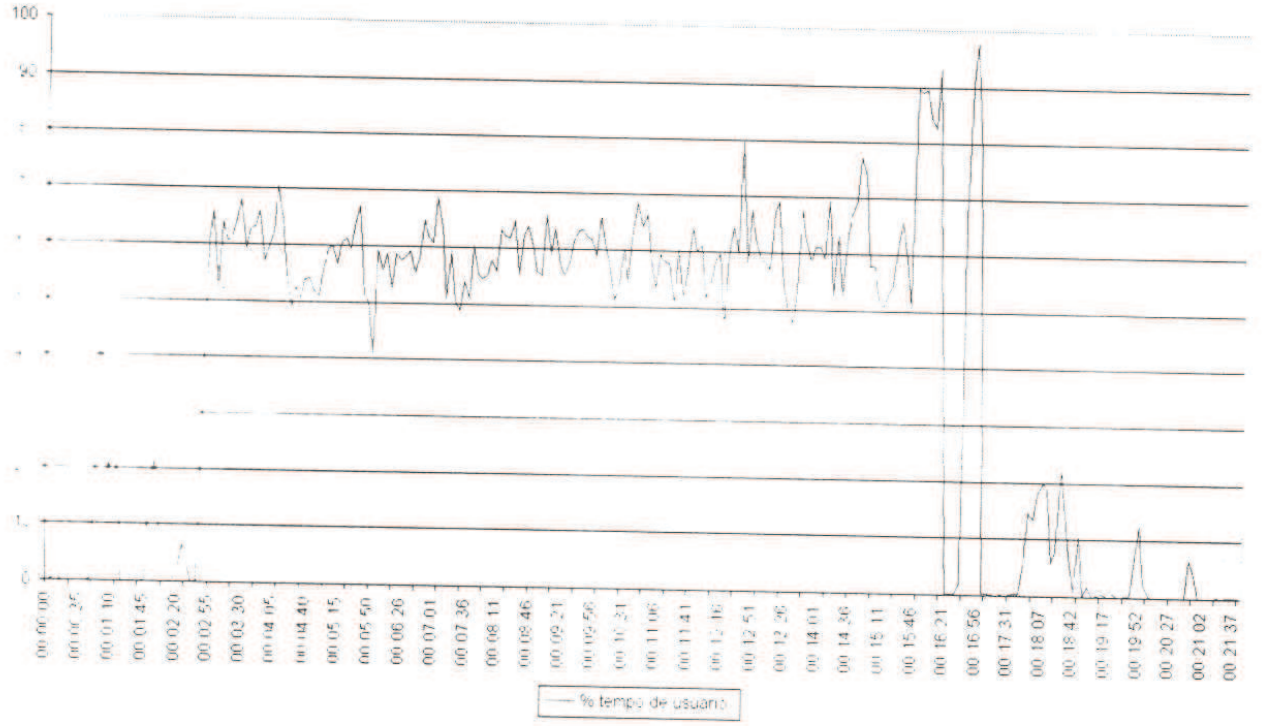

Figura B.12: Máquina Laplace – % do tempo de usuário no processador

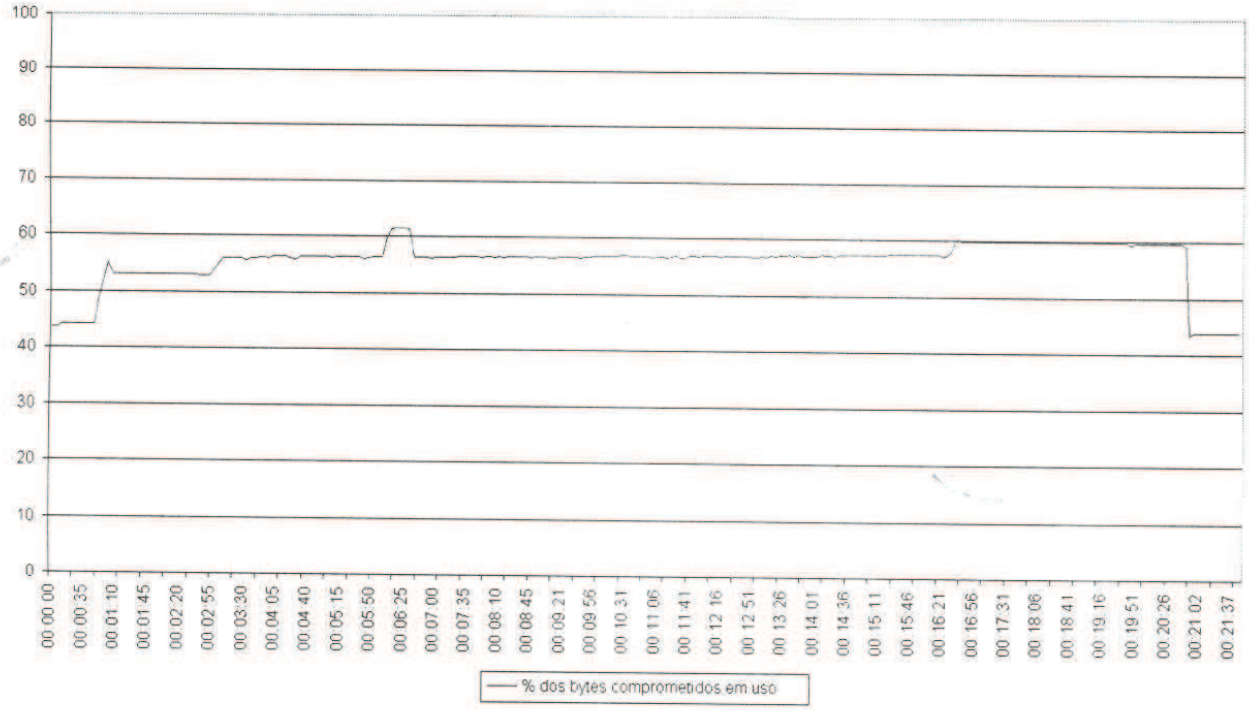

Figura B.13: Máquina Descartes – % dos bytes comprometidos em uso

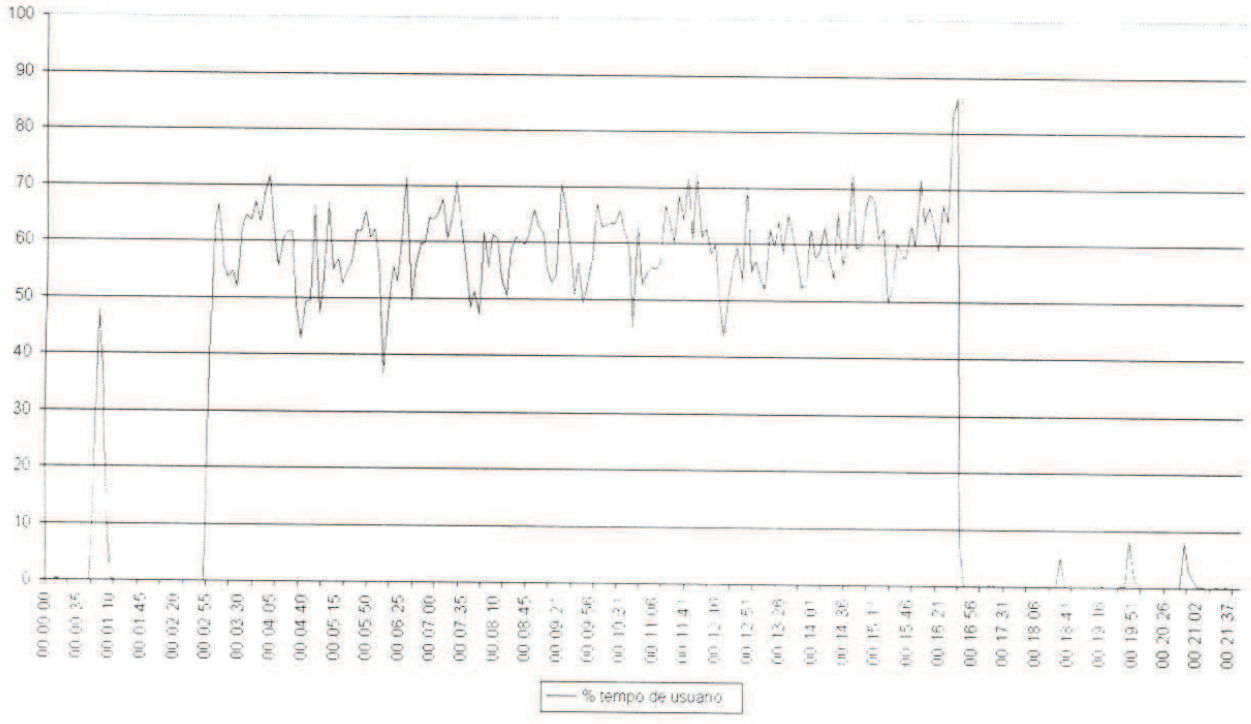

Figura B.14: Máquina Descartes – % do tempo de usuário no processador

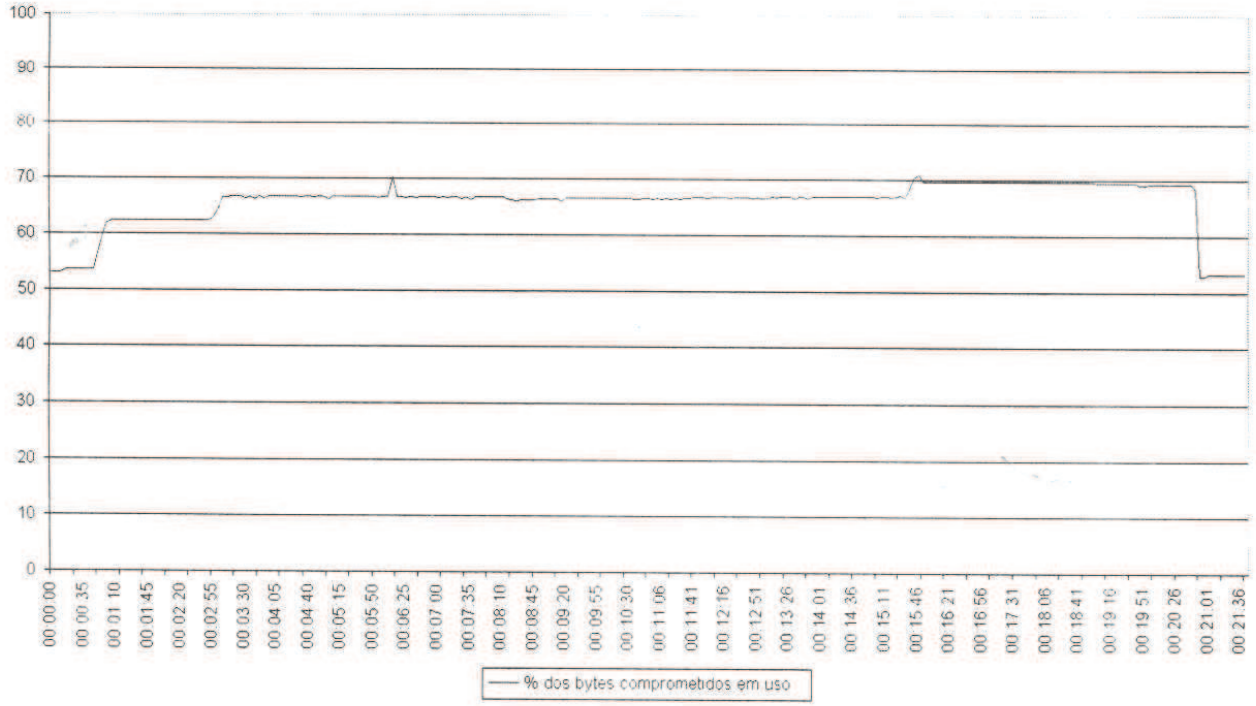

Figura B.15: Máquina Platao - % dos bytes comprometidos em uso

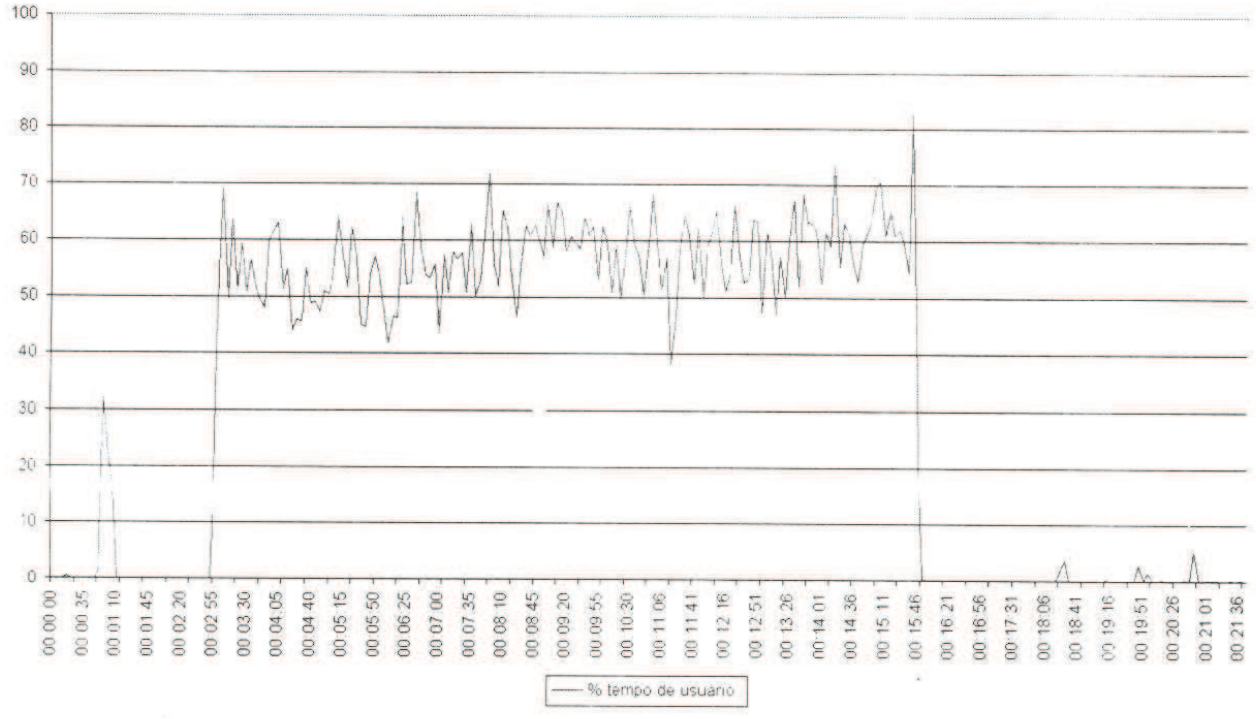

Figura B.16: Máquina Descartes - % do tempo de usuário no processador

Referências Bibliográficas

[3CO] 3COM. Palm Computing. http://www.palm.com.

[Alt] Altavista. http://www.altavista.com.

[B+97] J. Baumann et al. Mole - concepts of a mobile agent system. 1997.

[BGS98] Flávia A. Barros, Pedro F. Gonçalves, and Thiago L. V. L. Santos. Ontologies for Enhancing Web Searches Precision and Recall. *Anais do XXV Seminário Integrado de Software e Hardware (SEMISH'98)*, 1998.

[BSY95] Marko Balabanovic, Yoav Shohan, and Yeogirl Yun. An Adaptative Agent for Automated Web Browsing. *Journal of Visual Communications and Image Representation 6(4)*, 1995.

[BYR99] Ricardo Baeza-Yates and Berthier de Araújo Neto Ribeiro. *Modern Information Retrieval.* Addison Wesley, Harlow, England, 1999.

[CZ97] William Cockayne and Michael Zyda. *Mobile Agents.* Manning Publications Co., Greenwich, 1997.

[D+86] J. L. Deneubourg et al. Random Behaviour, Amplification Processes and Number of Participants: How they Contribute to the Foraging Properties of Ants. *Physicia, pp. 176-186*, 1986.

[D+91] J. L. Deneubourg et al. The Dynamics of Collective Sorting Robot-Like ants and Ant-Like robots. *From Animals to Animals, p. 356-363, MIT Press*, 1991.

[D+98] Prithviraj Dasgupta et al. A Supplier Driven Electronic Marketplace Using Mobile Agents. *Proceedings of the First International Conference on Telecommunications and E-Commerce*, November 1998.

[D+99] Prithviraj Dasgupta et al. MAgNET : Mobile Agents for Networked Electronic Trading. *IEEE Transactions on Knowledge and Data Engineering, Special Issue on Web Applications*, March 1999.

[F+93] Tim Finin et al. Draft specification of the kqml agent-communication language plus example agent ploicies and architectures. http://www.cs.umbc.edu/kqml/kqmlspec/ spec.html, 1993.

[FD92] J. Ferber and A. Drogoul. Using Reactive Multi-Agent systems in Simulation and Problem Solving. *Distributed Artificial Intelligence: Theory and Praxis, Kluwer Academic Publishers, pp. 53-80*, 1992.

[Fer95] Jacques Ferber. *Les Systèmes Multi-Agents – Vers une intelligence collective.* InterEditions, Paris, 1995.

[FG96] S. Franlkin and A. Graesser. Is It an Agent, or Just a Program? a Taxonomy for Autonomous Agents. http://www.msci.memphis.edu:80/franklin/ Agent-Prog.html, March 1996.

[Fou] The Apache Software Foundation. Apache http server. http://www.apache.org.

[Gas88] Mauro Gaspari. Concurrency and knowledge-level communication in agent languages. *Artificial Intelligence 105 (1998) 1-45*, 1988.

[GBH88] L. Gasser, C. Braganza, and N. Herman. MACE: a Flexible Testbed for Distributed AI Research. *Readings in Distributed Artificial Intelligence*, 1988.

[GI97] GMD FOKUS and International Business Machines Coporation. Mobile Agent System Interoperability Facilities Specification. http://www.omg.org, November 1997.

[Hal97] David Allan Halls. *Applying Mobile Code to Distributed Systems.* PhD thesis, Computer Laboratory - University of Cambridge. http://www.cl.cam.ac.uk/users/dah28/, June 1997.

[Kus] Lain Kusel. MANTA: A-Lif Ants. http://pigeon.csd.abdn.ac.uk/teaching msc_projec

[L+97] S. Luke et al. Ontology-based Web-Agents. *Proceedings of the First international Conference on Autonomous Agents (AA-97)*, 1997.

[Lan98] Danny B. Lange. Mobile Agents : The Future of Distributed Computing? *Lecture Notes in Computer Science 1445*, 1998.

[Lie97] Henry Lieberman. Letizia: Assisting Web Browsing. http://lieber.www.media.mit.edu/people/lieber/Lieberary/Letizia-Intro.html, 1997.

[LO98] Danny B. Lange and Mitsuro Oshima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison Wesley Longman, Inc., Massachussets, 1998.

[LSR96] S. Luke, L. Spector, and D. L. Rager. Ontology-based Knowledge Discovery on the World-Wide Web. *Proceedings of the Workshop on Internet-based Information Systems/AAAI-96*, 1996.

[Mic95] Sun Microsystems. Java soft. http://www.javasoft.com, 1995.

[Min86] M. Minsky. *The Society of Mind*. Simon and Schuster, New York, 1986.

[ML97] Pattie Maes and Yezdi Lashkari. *Webdoggie (webhound)*. http://webhound.www.media.mit.edu/projects/webhound, 1997.

[MR00] Khalid A. Mughal and Rolf W. Rasmussen. *A Programmers Guide to Java Certification - A Comprehensive Primer*. Addinson-Wesley, London, 2000.

[NIS] NIST - National Institute of Standards & Technology. TREC - Text REtrieval Conference. http://trec.nist.gov.

[Obj99] Objectspace. Voyager. http://www.objectspace.com, 1999.

[Pap99] Todd Papaioanou. The aglet portal. http://luckyspc.lboro.ac.uk/Aglets/index.html, 1999.

[Paz97] Michael J. Pazzani. Machine Learning and Information Filtering on the Internet. *15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, 1997.

[PMB96] Michael J. Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill & Webert : Identifiying interesting web sites. http://www.ics.uci.edu/pazzani Syskill.html, 1996.

[Por93] M.F. Porter. An algorithm for suffix stripping. *Readings in Information Retrieval*, 1993.

[PS97] Michael J. Pazzani and Y. Shoham. Lira. http://robotics.stanford.edu/users/marko/lira/demo1.html, 1997.

[Psi] Psion. Psion palmtop. http://www.psion.com.

[Rad98] Paulo Vinícius Wolski Radtke. MOTHRA: Arquitetura Para Busca e Recuperação de Informações Empregando Agentes Móveis. http://www.ppgia.pucpr.br/~radtke, December 1998.

[Rij79] K. V. Rijsbergen. *Informations Retrieval.* Butterworths, Londres, 1979.

[RK98] Paulo V. W. Radtke and Celso A. A. Kaestner. MOTHRA: Uma Proposta de Arquitetura Baseada em Agentes Móveis para Recuperação de Textos e Hiperdocumentos. *Anais do ISDM'98*, November 1998.

[RKS99] Paulo V. W. Radtke, Celso A. A. Kaestner, and Edson E. Scalabrin. BATHRA: Agentes Móveis Interativos para Sistemas e Abertos. *Anais do ENIA'99*, July 1999.

[RKS00] Paulo V. W. Radtke, Celso A. A. Kaestner, and Edson E. Scalabrin. Pegadas: Uma Proposta para a Interação entre Agentes Móveis na Internet. *Proceedings of ISADS'2000*, March 2000.

[RN95] Stuart Russel and Peter Norvig. *Articial Intelligence - A Modern Approach.* Prentice-Hall, Inc., New Jersey, 1995.

[S+96a] Edson E. Scalabrin et al. A Generic Model of Cognitive Agent to Develop Open Systems. *Advances in Artificial Inteligence - 13th Brazilian Symposium on Artificial Intelligence / SBIA'96*, October 1996.

[S+96b] Katia Sycara et al. Distributed Intelligent Agents. *IEEE Expert: Intelligent Systems and Applications, VOL 11, No. 6*, pages 36–46, December 1996.

[SAP] B. Starr, M. Ackerman, and M. Pazzani. DICA. http://www.uci.edu/ ackerman.

[ST98] Tomas Sander and Christian F. Tschudin. Towards Mobile Criptography. *Proceedings of Security & Privacy'98*, May 1998.

[Tok98] Tokyo Research Labs – IBM Coporation. Aglets Software Development Kit. http://www.trl.ibm.co.jp/aglets, 1998.

[Uni] University of Pensylvania. Linguistic Data Consortium. http://www.ldc.upenn.edu.

[WAP] WAP. Wap online. http://www.wap.com.

[WMB94] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing GigaBytes - Compressing and Indexing Documents and Images.* Van Nostrand Reinhold, Nova Iorque, 1994.

[YAH] Yahoo! http://www.yahoo.com.