SMA-Torrent: Um Cliente BitTorrent com Uso de Sistema Multiagente

Manoel Campos da Silva Filho[1], **Célia Ghedini Ralha**[2]

[1]Coordenação de Informática
Instituto Federal de Educação, Ciência e Tecnologia do Tocantins
AE 310 SUL, Av. LO 05 S/N, CEP 77.021-090 – Palmas, TO – Brazil

mcampos@ifto.edu.br

[2]Departamento de Ciência da Computação – Universidade de Brasília (UnB)
Caixa Postal 4466 – 70.904-970 – Brasília – DF – Brasil

ghedini@cic.unb.br

Abstract. *This paper presents SMA-Torrent, a BitTorrent cliente developed on multiagent approach using the JADE framework. SMA-Torrent implemented most of BitTorrent resources based on the multiagent theory. During this work, no comparative studies with other traditional BitTorrent proposals were possible, but we believe that the development of a BitTorrent client based on distributed characteristics, through the multiagent approach, will bring advantages over a traditional BitTorrent client, specially related to the reduction of execution time and better use of computational resources.*

Resumo. *O presente artigo apresenta um cliente BitTorrent denominado SMA-Torrent, o qual foi desenvolvido com a abordagem de sistema multiagente utilizando o framework JADE. O SMA-Torrent implementa grande parte dos recursos de um protocolo BitTorrent tradiconal baseando-se na teoria de sistema multiagente. Durante o desenvolvimento deste trabalho não foi possível realizar estudos comparativos entre a solução proposta e outras implementações de BitTorrent tradiconais, mas acreditamos que o desenvolvimento de um cliente BitTorrent baseado em características distribuídas com uso da abordagem de sistema multiagente trará benefícios sobre as propostas de cliente BitTorrent tradicionais devido a redução de tempo de execução e melhor utilização dos recursos computacionais.*

1. Introdução

O crescimento das redes de computadores e principalmente da Internet, assim como o avanço da computação, faz surgir, cada vez mais, sistemas distribuídos interoperáveis e dotados de comportamento cooperativo. Nesse sentido, a tecnologia de sistemas multiagente (SMA) facilita a construção de aplicações distribuídas, que podem exibir comportamento cooperativo e inteligente. Um SMA é composto por um grupo de agentes, que são entidades de software, capazes de serem independentes, tendo comportamento próprio, em nome de seu usuário ou dono [Weiss 2000]. Desta forma, neste artigo, será demonstrada uma aplicação desenvolvida utilizando o framework JADE, o Java Agent DEvelopment Framework [JADE 2009], que implementa um SMA de compartilhamento de arquivos utilizando o protocolo BitTorrent [Bittorrent.org 2009].

A utilização do protocolo proporciona um compartilhamento de recursos entre computadores [Costa-Montenegro et al. 2008], como memória, armazenamento e largura de banda, reduzindo a necessidade de infra-estrutura de grande porte para distribuição de arquivos entre uma grande quantidade de usuários. O compartilhamento de recursos permite uma utilização otimizada dos mesmos, garantindo maior desempenho para os sitemas de informação que utilizam a infraestrutura da rede, permitindo menor tempo de resposta. Além disso, sistemas que utilizam tal modelo para compartilhamento de arquivos, passam a ter uma infra-estrutura distribuída tolerante a falhas.

Este modelo é bastante utilizado por empresas que precisam distribuir grandes arquivos para vários usuários. Com o torrent, é criada uma rede distribuída para compartilhamento de tais arquivos, não sobrecarregando a infra-estrutura de servidores da empresa, além de distribuir o tráfego de dados entre os usuários. Exemplos desta abordagem são as distribuições de sistema operacional GNU/Linux que são fornecidas via torrent, como a Ubuntu [Ubuntu 2009].

Neste trabalho realizamos um estudo do protocolo BitTorrent com foco em seus recursos associados aos conceitos de SMA, com uso da plataforma de desenvolvimento JADE. O objetivo do trabalho foi a definição do projeto de SMA, incluindo aspectos arquiteturais e de modelagem dos agentes, resultando no desenvolvimento de um protótipo funcional, o qual pode ser utilizado como prova de conceito da proposta de utilização da abordagem de SMA em um cliente BitTorrent.

O artigo está organizado como segue. Na Seção 2 é apresentado o framework JADE, sua arquitetura, características e biblioteca de classes; na Seção 3, o protocolo BitTorrent, arquitetura de rede e trocas de mensagens entre os clientes; na Seção 4, a arquitetura da aplicação desenvolvida, os recursos usados do framework JADE, a troca de mensagens entre os agentes, os recursos existentes na aplicação desenvolvida, a documentação do projeto e os desafios de implementação superados; na Seção 5 as limitações do sistema; na Seção 6 são tecidas conclusões e tratados de trabalhos futuros.

2. O Framework JADE

O Java Agent DEvelopment Framework - JADE [JADE 2009] é um framework para desenvolvimento de SMA utilizando Java. Ele é uma implementação da especificação da Foundation for Intelligent and Physical Agentes – FIPA [Bellifemine et al. 2008]. A FIPA é uma fundação iniciada em 1995 para criar padrões de construção de SMA, tendo como alguns resultados, a especificação de um arquitetura para tais sistemas, que é implementada pelo JADE, e a Agente Communication Language – ACL, uma linguagem utilizada para comunicação entre os agentes, semelhante a linguagem Knowledge Query Manipulation Language - KQML [Michael 2002]. O grande problema da KQML é que existem várias implementações da mesma, muitas impossíveis de interoperarem [Michael 2002].

A especificação da FIPA possui elementos para permitir o gerenciamento dos agentes, consistindo de operações como criação, registro, localização, comunicação, migração e operação de agentes [Bellifemine et al. 2008], como mostrada na Figura 1.

Os elementos principais da arquitetura são a plataforma de agentes, o AMS, o DF e os próprios agentes.

A plataforma de agentes consiste de uma infraestrutura física em que os agentes

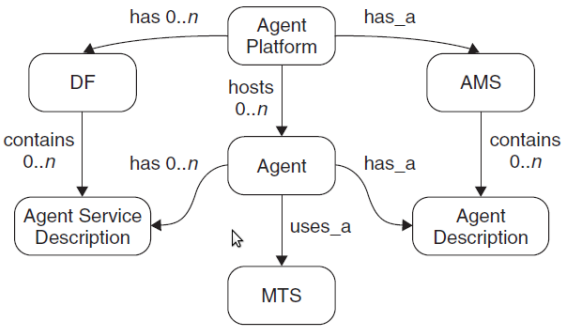

Figura 1. Arquitetura da FIPA para implementação de sistemas multiagente [Bellifemine et al. 2008]

são distribuídos, incluindo máquinas e sistemas operacionais e todos os elementos da arquitetura [Bellifemine et al. 2008]. A seguir são apresentados cada um destes elementos.

- O AMS é o Agent Management Subsystem, responsável pelo gerenciamento das operações na plataforma, como a criação e exclusão de agentes [Bellifemine et al. 2008]. Ele é um componente única em cada plataforma [Maíra 2007].
- Os agentes são aplicações que se registram no AMS e que pode fornecer serviços para outros agentes.
- O Directory Facilitator (DF) é responsável por prover recursos para localização dos agentes e dos serviços que eles disponibilizam. Um agente pode utilizar o serviço de páginas amarelas do DF para encontrar outros agentes que disponibilizem um determinado serviço, baseado na sua descrição [Bellifemine et al. 2008].

A Figura 2 mostra um exemplo da arquitetura FIPA implementada pelo JADE, onde existem 2 plataformas, cada uma composta por um conjunto de computadores. Cada plataforma possui pelo menos um contêiner, o Main Container, ambiente onde os agentes executam [Maíra 2007]. Os contêineres são as instâncias do ambiente de execução do JADE, que podem conter vários agentes [Giovanni 2003].

2.1. Classes básicas para construção de agentes JADE

O JADE possui um conjunto de classes e ferramentas para construção e gerenciamento de SMA, que facilita o desenvolvimento de aplicações. A principal, utilizada na construção de agentes, é a classe Agent, do pacote jade.core. Os agentes são construídos criando-se classes Java que estendem a classe Agent. Esta classe possui métodos para controle do ciclo de vida dos agentes, como os métodos setup e takeDown, que, respectivamente, permitem executar tarefas de inicialização e finalização do agente [Giovanni 2003].

A classe base Behaviour e suas descendentes, contidas no pacote jade.core.behaviours, permitem adicionar comportamentos aos agentes. Um comportamento é uma tarefa que o agente tem que fazer [Giovanni 2003]. Todos os comportamentos do agente devem ser criados estendendo-se uma das classes do pacote behaviours. Algumas classes que permitem implementar comportamentos básicos são as que seguem:

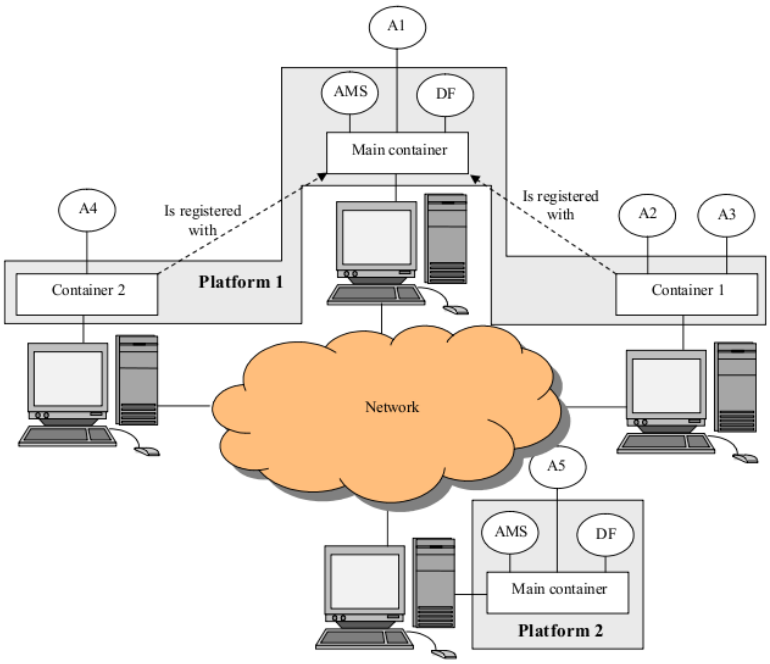

Figura 2. Arquitetura FIPA implementada pelo JADE [Giovanni 2003]

- OneShotBehaviour – comportamento que executa apenas uma vez e automaticamente termina.
- TickerBehaviour – comportamento que executa periodicamente, em intervalos de tempos definidos.
- WakerBehaviour – comportamento semelhante ao TickerBehaviour, porém, ele aguarda o tempo definido antes de executar a ação.
- CyclicBehaviour – comportamento que executa repetidamente, porém, sem um intervalo de tempo definido. Após a ação ser terminada, ela entra novamente na fila de comportamentos a executar, não havendo um tempo de espera determinado.

3. O Protocolo de BitTorrent

Nesta Seção explicar-se-á sucintamente o funcionamento de cada elemento de uma rede BitTorrent e como ocorrem as interações entre eles. Para maiores detalhes, sugere-se consultar a especificação em [theory.org 2009].

O BitTorrent é um protocolo p2p utilizado para compartilhamento de arquivos, amplamente utilizado na Internet. As aplicações clientes de BitTorrent permitem o download e upload de partes de um arquivo simultaneamente, o que faz com que essas pequenas partes sejam distribuídas rapidamente por uma rede de computadores. O protocolo utiliza uma abordagem denominada tit-for-tat (dente-por-dente), onde cada cliente procura maximizar sua taxa de download [Pouwelse et al. 2005]. Enquanto ele baixa partes, ele envia outras, ao mesmo tempo, para outros clientes.

3.1. Arquitetura de uma rede BitTorrent

O componente inicial, necessário para um usuário usar uma rede BitTorrent, é um programa cliente. Este é responsável por compartilhar e obter arquivos. Existem várias

implementações de clientes, gratuitamente disponíveis na Internet. Cada cliente em execução é denominado peer.

Para o usuário iniciar o download de um arquivo, ele precisa obter um arquivo com extensão torrent. Este arquivo contém meta-dados a respeito do arquivo que o usuário deseja baixar. Os arquivos torrent são normalmente disponibilizados em servidores web chamados de Torrent Indexer, que apenas armazenam estes e permitem ao usuário localizar torrents por diversos campos como descrição do arquivo compartilhado, categoria, tipo do arquivo e outros [Pouwelse et al. 2004]. Os torrent são pequenos arquivos que contém informações como o tamanho do arquivo e de cada parte do arquivo compartilhado, o endereço do tracker (que será explicado a seguir), a descrição do arquivo compartilhado, o hash SHA1 de cada parte do arquivo compartilhado, entre outros [theory.org 2009] [Costa-Montenegro et al. 2008]. Os dados nos arquivos torrent são organizados em formato b-encoding [theory.org 2009], um formato para representação simplificada de dados.

O Tracker é um servidor web responsável por receber requisições HTTP de clientes de BitTorrent, solicitando o endereço de outros clientes que possuam o arquivo indicado [Costa-Montenegro et al. 2008]. O arquivo compartilhado é identificado por meio do hash SHA1 do campo info do arquivo torrent, que contém um hash SHA1 de cada parte do arquivo e outras informações [theory.org 2009], como já citado.

Cada cliente (peer) que possui o arquivo completo é denominado seeder (semeador), caso contrário, são chamados de leechers (sangue-sugas). A Figura 3 mostra um exemplo da arquitetura de uma rede de BitTorrent.

3.2. Funcionamento de uma rede de BitTorrent

Para que um usuário possa iniciar o compartilhamento de um novo arquivo, um usuário seeder, que possui todo o arquivo, deve criar um torrent e registrá-lo em um Torrent Indexer, para que outros clientes possam obtê-lo e começarem a baixar o arquivo associado a ele. Em algum momento estes clientes (leechers) se tornarão seeders também, o que aumenta a eficiência de compartilhamento do arquivo, pois mais clientes passam a tê-lo por completo, e assim, passando a somente enviar partes dele para outros clientes.

Em mais baixo nível, o cliente que deseja baixar um arquivo, obtém o endereço do tracker, existente dentro do arquivo torrent, e envia uma solicitação HTTP ao mesmo para obter uma lista de outros clientes (peers) que possuem o arquivo desejado. Após o cliente ter essa lista, ele envia uma mensagem handshake para iniciar a comunicação com cada peer. Esta mensagem possui o hash SHA1 do campo info do arquivo torrent. Cada cliente que recebe uma mensagem handshake responde com uma mensagem bitfield, indicando as partes do arquivo que ele possui. O cliente solicitante escolhe qual parte do arquivo solicitar ao peer remoto e envia uma mensagem request para fazer a solicitação. O peer remoto envia a parte solicitada dentro de uma mensagem piece. [theory.org 2009]

4. Proposta de Solução

A solução foi desenvolvida utilizando-se o framework JADE, como pode ser visto na Figura 4. Cada cliente (peer) é um agente, que pode ter vários torrents registrados, e pode agir como um seeder, contendo todas as partes de um arquivo para enviar a outros agentes,

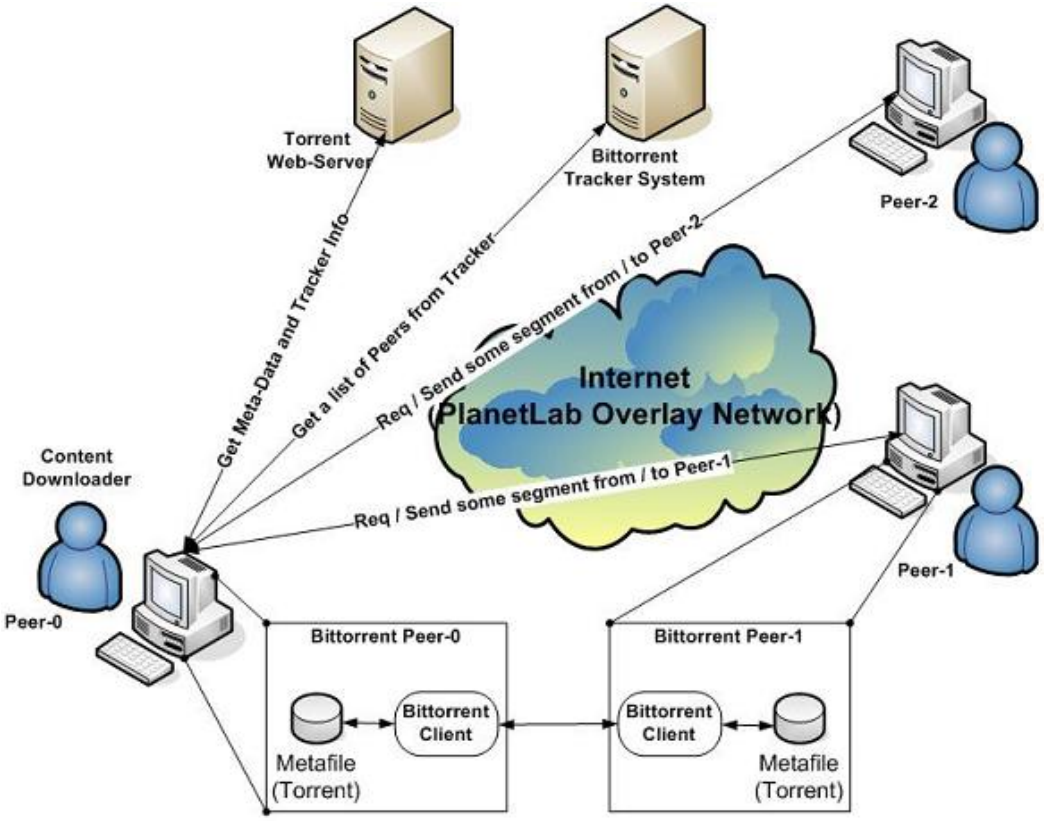

Figura 3. Arquitetura de uma rede BitTorrent [Wiki 2009]

e também como leecher, que não contém o arquivo completo e solicita de outros agentes partes deste arquivo.

Observe na Figura 4 que o Torrent Web Server (Torrent Indexer) é o mesmo componente da arquitetura tradicional. O Torrent Tracker foi implementado utilizando os serviços do Directory Facilitator (DF) do JADE. Cada agente que tem um arquivo para compartilhar, registra esse arquivo no Directory Facilitator, para que outros agentes utilizem o serviço de busca de agentes do DF e assim encontrar outros agentes que tenham o arquivo que ele deseja. Cada arquivo compartilhado é registrado com o hash SHA1 do campo info do arquivo torrent. Logo, o nome do serviço é o valor desse hash. Desta forma, os agentes, ao usarem o recurso de páginas amarelas do DF, para encontrarem outros agentes que tenham um determinado arquivo, procuram pelos serviços cujo nome seja o valor do hash SHA1 do torrent do qual eles desejam obter partes. O serviço de páginas amarelas do DF funciona como as páginas amarelas das listas telefônicas, permitindo que os agentes registrem os serviços que eles fornecem, e que outros agentes possam buscar por estes serviços nessa lista [Bellifemine et al. 2008].

Foi utilizado o protocolo de interação Contract Net, com troca de mensagens utilizando a Agent Communication Language (ACL). As aplicações de torrent são um exemplo de sistemas de resolução de problemas distribuídos, onde cada indivíduo (agente) na rede é responsável pela resolução do problema final: o compartilhamento de arquivos entre todos. No entanto, os agentes precisam trabalhar juntos para alcançar esse objetivo

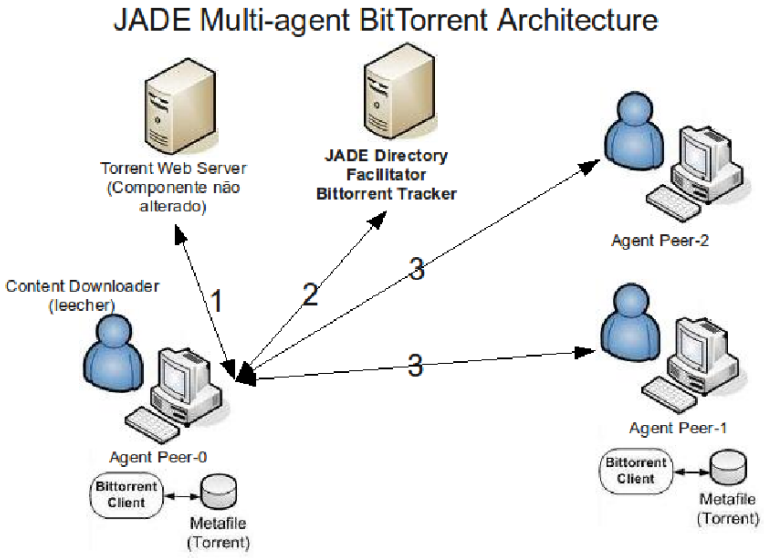

Figura 4. Arquitetura da solução implementada (adaptada de [Wiki 2009])

comum. Desta forma, eles precisam cooperar para que todos alcancem seus objetivos. Assim, um agente que recebe partes de um arquivo também deve disponibilizar essas e outras partes para outros agentes. O Contract Net vai ao alcance desses objetivos pois é um protocolo para permitir uma eficiente cooperação entre os agentes, definindo toda a troca de mensagens a ser realizada entre os agentes [Michael 2002]. O protocolo especifica que, quando um agente deseja que outro execute uma tarefa, ele envia uma mensagem de Call For Proposal (CFP) solicitando que os agentes enviem propostas para execução da tarefa. As propostas são enviadas por mensagens PROPOSE. O agente emissor avalia as propostas recebidas e escolhe a mais adequada para o contexto. Em seguida, envia uma mensagem de ACCEPT_PROPOSAL para o agente da proposta escolhida, informando a ele que sua proposta foi aceita e que ele deve executar a tarefa para a qual se propôs [Giovanni 2003]. A troca de mensagens do protocolo BitTorrent funciona de maneira similar ao Contract Net, provendo trabalho em conjunto. Por isso o Contract Net foi escolhido como protocolo de interação entre os agentes.

A Figura 5 a seguir, demonstra as aplicações (agentes) atualmente registrados no Directory Facilitator do JADE. Como pode ser visto, existem três clientes registrados que podem trocar partes de arquivos.

A sequência de operações do sistema, para obtenção de partes de um arquivo, é apresentada no diagrama de atividades da Figura 6.

A atividade 11, no diagrama da Figura 6, é executada automaticamente quando a aplicação é iniciada. Esta atividade representa outro comportamento do agente. Enquanto ele estiver recebendo partes, pode também servir outros clientes (agentes) remotos, enviados partes de arquivos que possui. Para isso, cada cliente tem um comportamento CyclicBehaviour, de nome ReceiveRequestsBehaviour, que fica constantemente aguardando requisições. Ao receber uma requisição para envio de uma parte de um arquivo, este comportamento inicia uma outra sequência de atividades para atender

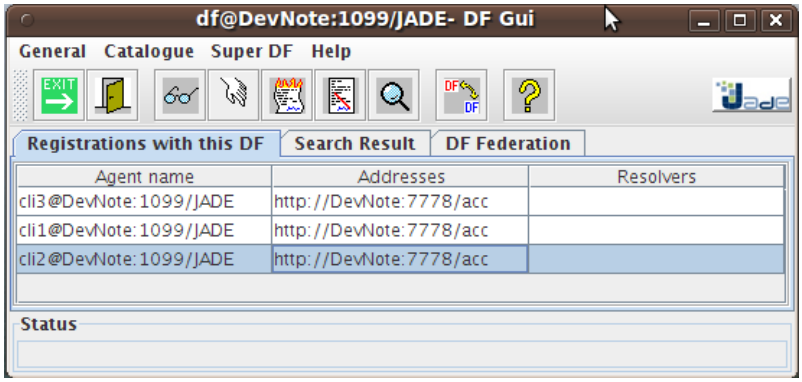

Figura 5. Aplicações (agentes) registrados no Directory Facilitator do JADE

a solicitação, enviando a parte solicitada de acordo com as especificações do protocolo BitTorrent.

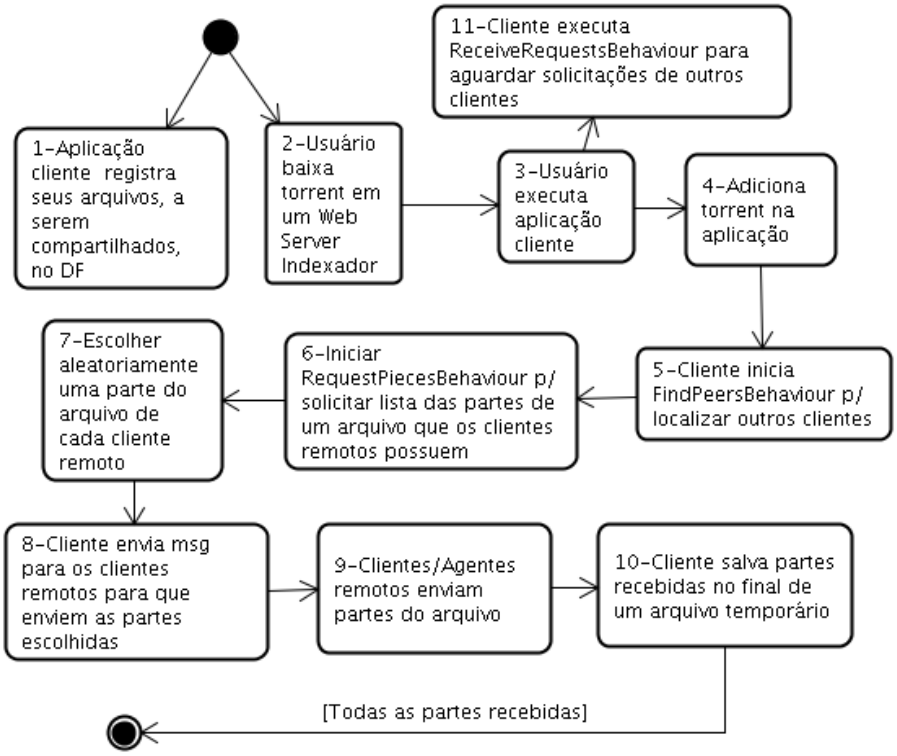

Figura 6. Diagrama de Atividades mostrando a sequência de operações do sistema, incluindo os comportamentos dos agentes

Para cada arquivo compartilhado, é adicionado um conjunto dos behaviours citados, para fazer requisições por novas partes desses arquivos, exceto o behaviour que aguarda requisições, que é único para cada cliente e não finaliza.

4.1. Recursos da Aplicação

A seguir são listados os principais recursos da aplicação desenvolvida:

- adicionar, remover, pausar e reiniciar torrents (arquivos compartilhados);

- adicionar torrents para obter um determinado arquivo;
- adicionar torrents para um arquivo que o usuário possui e que deseja compartilhar com outros;
- continuar a baixar um arquivo do mesmo ponto onde parou, mesmo depois de reabrir a aplicação.

A Figura 7 a seguir, mostra a janela principal da aplicação desenvolvida, contendo um torrent registrado, cujas partes do arquivo compartilhado, de nome ”frejat-segredos.mpg”, estão sendo baixadas de outros clientes.

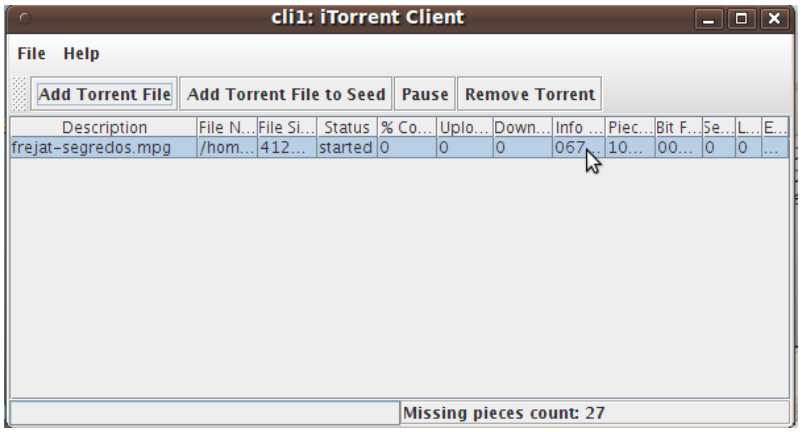

Figura 7. Janela principal da aplicação

Estes recursos tornam a aplicação bem próxima das implementações existentes, apesar de existirem alguns problemas conhecidos que podem fazer a mesma se comportar de forma inesperada, como informado no arquivo “LEIAME.txt”, disponível com os fontes e binários do sistema. Todos esses arquivos podem ser baixados em http://smatorrent.manoelcampos.com.

4.2. Documentação do Projeto

Todo o código fonte do sistema está comentado, e a documentação em Java DOC está disponível, juntamente com os fontes, no endereço http://smatorrent.manoelcampos.com. Diagramas de classes e de sequência UML, criados na ferramenta Jude Community 5.5, também estão disponíveis, e alguns não foram incluída neste artigo por questões de espaço. Imagens dos pacotes com.manoelcampos.smatorrent e com.manoelcampos.smatorrent.behaviours, com o mesmos nomes citados, também estão disponíveis, e representam o diagrama de classes destes pacotes. Para os diagramas de sequência também existem os arquivos “Baixar partes de um arquivo.png” e “Enviar partes de um arquivo.png”.

A Figura 8 a seguir, mostra um diagrama de sequência das operações de upload de partes de um arquivo, mostrando a interação entre os agentes e a troca de mensagens geradas pelos comportamentos dos mesmos.

4.3. Desafios de Implementação

Durante a implementação do sistema, alguns desafios foram encarados e superados. Abaixo são citados alguns destes:

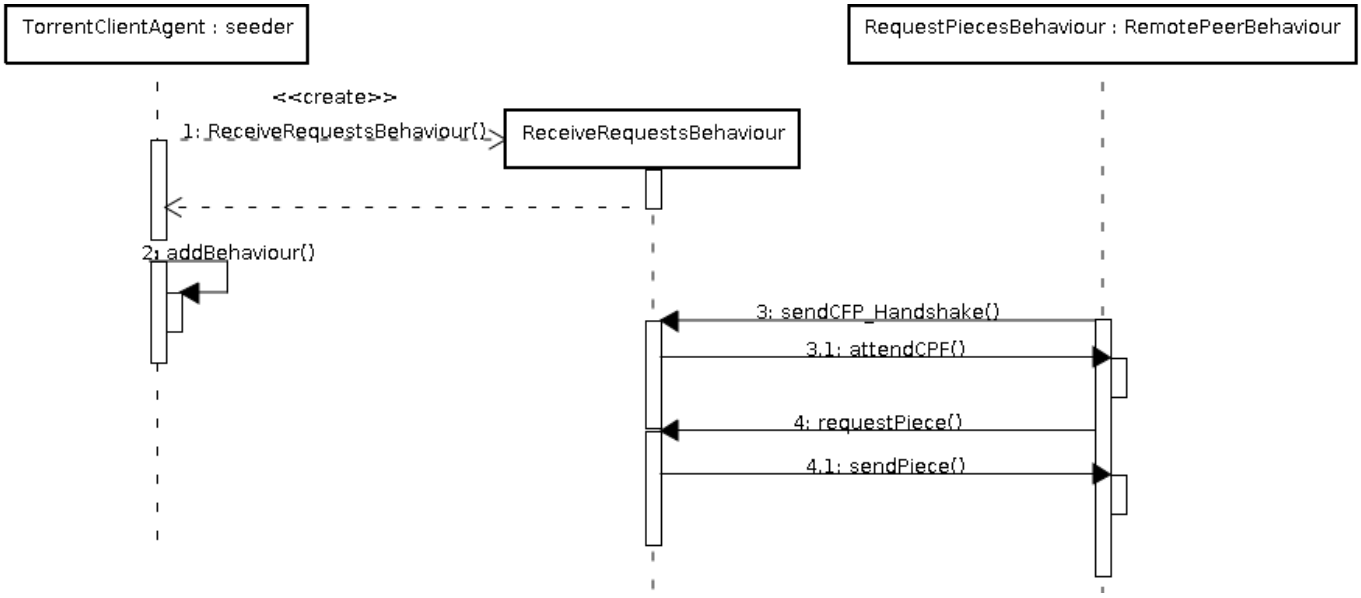

Figura 8. Diagrama de sequência mostrando a troca de mensagens geradas pelos comportamentos, para envio de partes de um arquivo

- Compreender a estrutura do arquivo torrent e o formato de codificação b-encoding utilizado nele;
- Implementar uma classe java para permitir a leitura e geração de arquivos torrent;
- Compreender e implementar a troca de mensagens de acordo com o protocolo de BitTorrent;
- Obtenção de partes do arquivo;
- Montagem de arquivo com as partes aleatoriamente recebidas do arquivo compartilhado e ordenação das partes, gerando o arquivo final.

5. Limitações do Sistema

O sistema possui algumas limitações, devido a forma de implementação e por nem todos os recursos do protocolo BitTorrent e da estrutura do arquivo torrent terem sido implementados. Algumas dessas limitações são listadas a seguir:

- A aplicação não lê arquivos torrent que tenham mais de um tracker registrado, apesar de este campo não ser usado, pois o tracker passou a ser o Directory Facilitator do JADE. Isto é devido à estrutura de arquivos torrent contendo mais de um tracker, requerer maior detalhamento da implementação da classe java criada para manipular tais arquivos;
- os clientes (agentes JADE) não podem interoperar com outros clientes em uma rede BitTorrent tradicional, pois a arquitetura é diferente, devido ao fato de que os clientes não se comunicam com um tracker em um servidor web (o servidor que clientes de BitTorrent tradicionais usam) e sim com o Directory Facilitator do JADE.

Apesar de o sistema não interoperar com outros clientes em redes tradicionais de BitTorrent, é possível criar uma arquitetura de servidores distribuídos também com o JADE. Como pode ser visto na Figura 2, pode-se ter mais de uma plataforma JADE,

uma interoperando com a outra, não necessitando de nenhuma alteração no sistema. Para configuração de várias plataformas JADE, recomenda-se consultar o JADE Programmer's Guide [Bellifemine et al. 2007].

6. Conclusões e Trabalhos Futuros

O desenvolvimento do sistema permitiu um maior conhecimento da plataforma JADE, incluindo a compreensão de conceitos de agentes, serviços, negociação e comportamentos.

Conseguiu-se desenvolver as funcionalidades de compartilhamento de arquivos utilizando o protocolo BitTorrent, para otimização do uso dos recursos de rede, não sobrecarregando servidores para distribuição de arquivos e criando uma estrutura resiliente, como proposto na Seção 1. A aplicação permite o compartilhamento de arquivos, dividindo o mesmo em diversas pequenas partes, que são rapidamente transferidas pela rede. O sistema permite o recebimento e envio de várias partes de diferentes clientes/agentes simultaneamente, mostrando ser uma aplicação funcional e que prova o uso dos conceitos de SMA e BitTorrent.

Entende-se que o sistema desenvolvido serviu como prova de conceitos dos fundamentos de agentes e dos recursos disponíveis na plataforma JADE. Ele permitiu alcançar maior conhecimento dos detalhes de implementação de agentes, ao se trabalhar com comportamentos concorrentes, o protocolo de interação Contract Net, a linguagem ACL e os serviços do Directory Facilitator do JADE. Pôde-se mostrar a viabilidade do desenvolvimento de uma solução de torrent baseada em SMA, mas requer estudo e melhorias para averiguar se a solução pode se tornar realmente superior às implementações tradicionais.

Para trabalhos futuros, sugere-se a implementação de mais recursos do protocolo BitTorrent, como a leitura de arquivos torrent contendo mais de um tracker, implementação de todas as trocas de mensagens do protocolo e a comparação entre o desempenho da solução contra as redes tradicionais de BitTorrent. Contudo, a solução desenvolvida está funcional, dentro de uma plataforma JADE, e considera-se muito valiosos os desafios enfrentados no desenvolvimento da mesma, que permitiram um maior conhecimento de muitos detalhes do framework JADE.

Referências

Bellifemine, F., Caire, G., Greenwood, D., et al. (2008). *Developing multi-agent systems with JADE*. Springer.

Bellifemine, F., Caire, G., Trucco, T., and Rimassa, G. (2007). *Jade Programmer's Guide*. http://jade.tilab.com/doc/index.html.

Bittorrent.org (2009). *Bittorrent Forum*. http://www.bittorrent.org.

Costa-Montenegro, E., Burguillo-Rial, J., Rodriguez-Hernandez, P., Gonzalez-Castano, F., Curras-Parada, M., Gomez-Rana, P., and Rey-Souto, J. (2008). Multi-Agent System Model of a BitTorrent Network. In *Proceedings of the 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pages 586–591. IEEE Computer Society.

Giovanni, C. (2003). *Jade Programming Tutorial for Beginners*. http://jade.tilab.com/doc/index.html.

JADE (2009). *Java Agent Development Framework*. http://jade.tilab.com.

Maíra, G. (2007). *JADE/JADEX*. http://wiki.les.inf.puc-rio.br/uploads/e/ef/07-02-JADE-JADEX.ppt.

Michael, W. (2002). *An introduction to multiagent systems*. Wiley.

Pouwelse, J., Garbacki, P., Epema, D., and Sips, H. (2004). An introduction to the BitTorrent Peer-to-Peer File-Sharing System. *Citeseer*.

Pouwelse, J., Garbacki, P., Epema, D., and Sips, H. (2005). The bittorrent p2p file-sharing system: Measurements and analysis. *Lecture Notes in Computer Science*, 3640:205.

theory.org (2009). *Bittorrent Protocol Specification v1.0*. http://wiki.theory.org/BitTorrentSpecification.

Ubuntu (2009). *Ubuntu Home Page*. http://www.ubuntu.com.

Weiss, G. (2000). *Multiagent systems: a modern approach to distributed artificial intelligence*. The MIT press.

Wiki, P. S. (2009). *Measurement Study of Bittorrent*. https://slab1.engr.sjsu.edu/wiki/index.php?title=Measurement_Study_of_Bittorrent.