colloquium exactarum

FERRAMENTA DE BACKUP TRANSPARENTE BASEADA EM P2P

P2P-BASED TRANSPARENT BACKUP TOOL

Ronaldo Albertini; Caetano Bocchi Pedroso

Universidade do Oeste Paulista – UNOESTE, Faculdade de Informática de Presidente Prudente, Presidente Prudente, SP
tomatealbertini@gmail.com; pedroso@unoeste.br

RESUMO - Com o avanço da tecnologia houve um aumento significativo na quantidade de arquivos armazenados em formatos digitais, como fotos, áudio ou mesmo documentos de importância comercial. A falta de uma rotina de backup pode gerar sérios problemas para os donos destes arquivos. Este artigo apresenta uma ferramenta de backup transparente ao usuário utilizando a tecnologia de rede P2P que, aproveitando a conectividade existente entre computadores, simplifique os trabalhos de backup.
Palavras-chave: sistemas distribuídos; Peer-to-Peer; backup; JXTA; XML; middleware.

ABSTRACT - With advances in technology came a significant increase in the number of stored files in digital format, like photos, audio files or even commercial related documents. The lack of a proper backup routine may causes serious problems to file owners. This paper presents a user transparent P2P backup tool that, using the existent connectivity between computers, simplifies the backup jobs.
Keywords: distributed systems; Peer-to-Peer; backup; JXTA; XML; middleware.

Recebido em: 02/07/2015
Revisado em: 02/12/2015
Aprovado em: 02/12/2015

1 INTRODUÇÃO

Com o avanço tecnológico, houve um aumento significativo do número de computadores em uso em todos os lugares. A tecnologia está presente em todas as áreas, e cada vez mais são utilizados documentos em formato digital e multimídia.

Os computadores como qualquer aparelho eletrônico, podem apresentar defeitos, seja por mau uso, ou por falhas de alguns de seus sistemas internos de software ou hardware. Tais falhas podem ocasionar perda de informações importantes aos seus usuários. Apesar disso, mesmo sabendo da importância de suas informações, usuários sempre adiam ou não realizam a rotina de cópia de segurança em seus arquivos, o que pode ocasionar a perda de seus documentos, sejam eles pessoais ou de negócios.

Outro fator a se considerar é a grande popularização das redes locais, assim, possuir mais de um computador conectado em rede já pode ser considerado termo comum em médias e pequenas empresas, e até mesmo em algumas residências.

Tendo isso em mente, foi proposto o desenvolvimento de uma ferramenta de backup, transparente ao usuário, que faz proveito da conectividade presente entre vários equipamentos. Ela deveria ser multiplataforma, ou seja, que possa ser executada em diferentes arquiteturas e baseada na tecnologia Peer-to-Peer (P2P).

Este artigo apresenta o protótipo de tal ferramenta, descrevendo seus módulos componentes e a interação entre eles. Para tanto, nas seções seguintes serão abordados os conceitos de Sistemas Distribuídos, principalmente os sistemas P2P. Na sequência será abordado o *middleware* usado na implementação da ferramenta. Esta é apresentada em seguida. Considerações finais encerram o artigo.

2 SISTEMAS DISTRIBUÍDOS

Um Sistema Distribuído é aquele no qual os componentes localizados em computadores interligados em rede se comunicam e coordenam suas ações apenas trocando mensagens (COLOURIS, 2007).

Desta forma, o compartilhamento de recursos é um dos principais motivos para a construção deste tipo de sistema, que podem ser utilizados por servidores ou por outros clientes. Estes recursos podem ser definidos por dispositivos de hardware, objetos de software remotos e compartilhamento de arquivos.

Sistemas distribuídos funcionam em computadores conectados em rede, não importando a localização física dos seus membros, pois eles podem estar separados, por distâncias continentais, ou estar no mesmo prédio ou sala. A seção a seguir

detalha o compartilhamento de recursos neste tipo de sistemas, parte importante no desenvolvimento e funcionamento da ferramenta desenvolvida.

2.1 COMPARTILHAMENTO DE RECURSOS

Ainda segundo Coulouris (2007), rotineiramente, compartilhamos recursos, tanto de hardware como de software, sem perceber que estamos fazendo isso.

Do ponto de vista do hardware, compartilhamos recursos como impressoras e espaço em disco para reduzir custos. Mas o que é mais importante para os usuários, é o compartilhamento dos recursos a um nível mais abstrato, como informações relacionadas ao seu trabalho diário e suas atividades sociais. Estes acessam os recursos, sem precisar saber sua localização física, ou como estão implementados.

O termo serviço é usado para designar uma parte distinta de um sistema de computador que gerencia um conjunto de recursos relacionados e apresenta sua funcionalidade para usuários e aplicativos. Por exemplo, o acesso a arquivos compartilhados é feito através de serviços de sistemas de arquivos, já o envio de documentos para impressoras é feito pelo serviço de impressão (TAYLOR, 2005).

Em sistemas distribuídos os recursos são fisicamente encapsulados nos computadores, permitindo acesso a eles somente através de um mecanismo de comunicação. Para se obter um compartilhamento eficiente, é necessário um mecanismo que tenha uma interface de comunicação, onde o recurso pode ser acessado e atualizado de forma consistente e confiável.

Um dos desafios encontrados no desenvolvimento de sistemas distribuídos é a heterogeneidade de computadores, redes, sistemas operacionais, linguagens de programação e formas de representação de dados.

Neste contexto, os tipos de dados podem ser representados de diferentes formas no hardware ou no software. Assim, essas diferenças devem ser consideradas, caso haja troca de mensagens entre programas que operem em plataformas distintas. Diferentes linguagens de programação usam diferentes representações para caracteres e estruturas de dados, como vetores e registros. Essas diferenças também devem ser consideradas, no caso de programas escritos em linguagens distintas precisem se comunicar (COULOURIS, 2007).

3 SISTEMAS PEER-TO-PEER

O termo P2P se refere ao conceito onde, em uma rede de nós semelhantes, utilizando sistemas de comunicação, dois ou mais têm a capacidade de colaborar espontaneamente uns com os outros, sem a

necessidade de uma coordenação central (SCHODER E FISCHBACH, 2003).

Existem várias topologias de redes possíveis, contudo, a utilizada no projeto é a topologia descentralizada.

A topologia descentralizada é a topologia de rede que mais se aproxima de ser verdadeiramente P2P (GRADECKI, 2002). Nela, não existe uma autoridade central, apenas computadores individuais, que são capazes de se conectar e comunicar com qualquer um dos outros computadores na rede.

Um exemplo desta topologia é a própria Internet. Neste caso, quando um pacote de informações trafega pela Internet, as informações dentro do pacote só dizem a cada computador para onde enviar o pacote.

A Figura 1 mostra um exemplo de uma rede descentralizada. Nela, todos os nós (pares) no sistema atuam, tanto como clientes, quanto como servidores, gerenciando consultas, pedidos de downloads e, ao mesmo tempo, fazendo buscas e transferindo mídias solicitadas (GRADECKI, 2002). Um exemplo de rede P2P que faz uso da topologia descentralizada, no caso particular do compartilhamento de arquivos, é o *Gnutella* (BARCELOS E GASPARY, 2006).

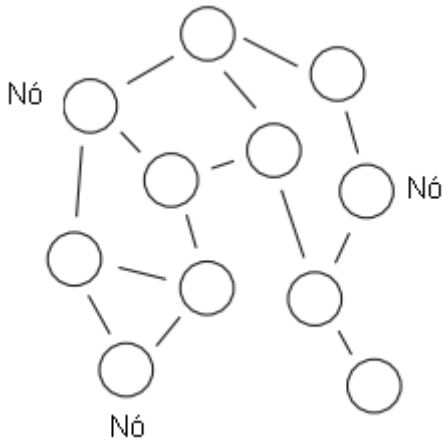

Figura 1. Uma figura normal.
Fonte: (Própria).

4 MIDDLEWARE

Para um sistema distribuído suportar uma variedade de computadores heterogêneos e redes, enquanto fornece uma visão única do sistema, a pilha de software, normalmente é dividida em duas partes. A parte mais alta é a aplicação (e usuários) e a parte inferior é o *middleware*, que interage com as camadas mais baixas de redes e computadores para fornecer a transparência necessária.

O *middleware* abstrai as características das máquinas e dos protocolos do desenvolvedor de aplicações, fornecendo uma variedade de facilidades de alto nível para os programadores desenvolvedores (TAYLOR, 2005). Desta forma, liberando-os da necessidade de conhecimentos específicos sobre a plataforma onde o sistema será executado, permitindo que eles se dediquem quase que exclusivamente as funcionalidades desejadas do sistema.

Segundo Steinmetz e Wehrle (2005) alguns benefícios do uso do *middleware* são:

- Detalhes de módulos que não interessam ao programador são encapsulados.
- Os desenvolvedores não precisam se preocupar com a complexidade do *middleware*, somente com os módulos menos complexos.
- Facilita a manutenção, no caso de adicionar novos módulos de programação.

O *middleware* escolhido para o desenvolvimento da ferramenta proposta foi o JXTA, apresentado na próxima seção.

5 JXTA

A sigla JXTA vem da palavra inglesa *juxtapose*, que significa lado a lado, considerando-se assim que essa tecnologia caminha lado a lado com os modelos mais conhecidos hoje como WEB e cliente-servidor (DEVMEDIA, 2015).

O *middleware* JXTA é definido em (TAYLOR, 2005) como um conjunto de protocolos abertos e generalizados que permitem a qualquer dispositivo conectado (de telefone celular até PDA, de PC até Servidor) em rede se comunicar e colaborar. JXTA é um projeto de código aberto que é desenvolvido por vários contribuidores.

Segundo Brookshier *et al*. (2002), algumas das metas principais do JXTA, são:

- Independência de sistemas operacionais;
- Independência de linguagem de programação; e,
- Fornecer serviços e infraestrutura para aplicações P2P.

Assim, tais metas dão suporte a programação P2P para os mais variados tipos de dispositivos, como computadores *desktop* ou computadores de mão.

O JXTA é um *framework* com um conjunto de padrões que suportam aplicações P2P. Assim, ele não é uma aplicação e não define o tipo de aplicações que podem ser desenvolvidas através de sua utilização.

É importante lembrar que os protocolos e a especificação JXTA não especificam quais linguagens devem ser usadas na implementação, contudo uma implementação de referência foi feita utilizando a linguagem Java.

Segundo Gradecki (2002), a modelagem deste *middleware* para programação P2P em Java é feita usando um pequeno número de protocolos utilizados nos serviços do JXTA. Os protocolos podem ser implementados usando qualquer linguagem, assim permitindo a dispositivos heterogêneos se comunicarem com outros em um sistema P2P. Em (FLENNER *et al*., 2002) são citados os seis protocolos que trabalham juntos para habilitar a descoberta,

organização, monitoração e comunicação entre os nós de um sistema, são eles:

- Protocolo de consulta de nó: usado para enviar uma consulta a qualquer número de outros nós e receber respostas;
- Protocolo de descoberta de nó: usado para propagar conteúdo e descobrir conteúdo;
- Protocolo de Informação: usado para obter informações do status do nó;
- Protocolo de Ligação: usado para criar comunicação entre nós;
- Protocolo de Roteamento: usado para encontrar uma rota de um nó até outro;
- Protocolo *Rendezvous* (encontro): usado para propagar mensagens na rede;

Estes seis protocolos são os necessários para que um dado nó possa existir em uma rede P2P descentralizada, ou seja, sem a necessidade de um servidor central. Os nós têm capacidade de existir em redes privadas, através de *firewalls*, e podem ser associados a um endereço IP.

Com o uso desde protocolos os nós do sistema são capazes, de:

- Propagar conteúdo que é disponibilizado;
- Descobrir conteúdo que acham interessante;
- Formar ou entrar em grupos, privados ou públicos;
- Auxiliar o roteamento e encaminhamento das mensagens de forma transparente.

5.1 ANÚNCIO (ADVERTISEMENT)

Segundo Gradecki (2002) o anúncio é a ferramenta principal que os protocolos JXTA usam para criar um nó, grupo de nós, e informações de configurações disponíveis na rede. O anúncio é um arquivo que pode ser passado de um nó para outro usando um formato comum. O formato utilizado no anúncio é o XML, pois fornece uma representação hierárquica facilmente expansível de informações que os nós necessitam para operar em uma rede JXTA. Um exemplo de anúncio pode ser visto da Figura 2.

```
<?xml version="1.0"?>
<jxta:PipeAdvertisement>
  <Id>UUID</Id>
  <Type>Type of the pipe</Type>
  <Name>Example</Name>
</jxta:PipeAdvertisement>
```

Figura 2. Exemplo de anúncio de pipe (canal de comunicação).
Fonte: (Própria).

5.2 ARQUITETURA JXTA

A arquitetura do JXTA é dividida em três camadas, que são mostradas na Figura 3.

A camada de núcleo (core) é a camada mais baixa da arquitetura, e é a responsável pelo estabelecimento de conexão, comunicação e gerenciamento (tal como roteamento) (TAYLOR, 2005). Entre os

protocolos existe um grupo universal, chamado de WorldPeerGroup. Quando um nó começa sua execução, ele automaticamente entra para esse grupo, que fornece todas as funcionalidades já implementadas (GRADECKI, 2002).

A camada de Serviços (*services*) é a camada intermediária, que fornece funcionalidades de mais alto nível, como procura e compartilhamento de arquivos. Estes serviços fazem uso pesado de recursos de ligação fornecidos pelo núcleo, que podem ser usados como componentes em sistemas P2P.

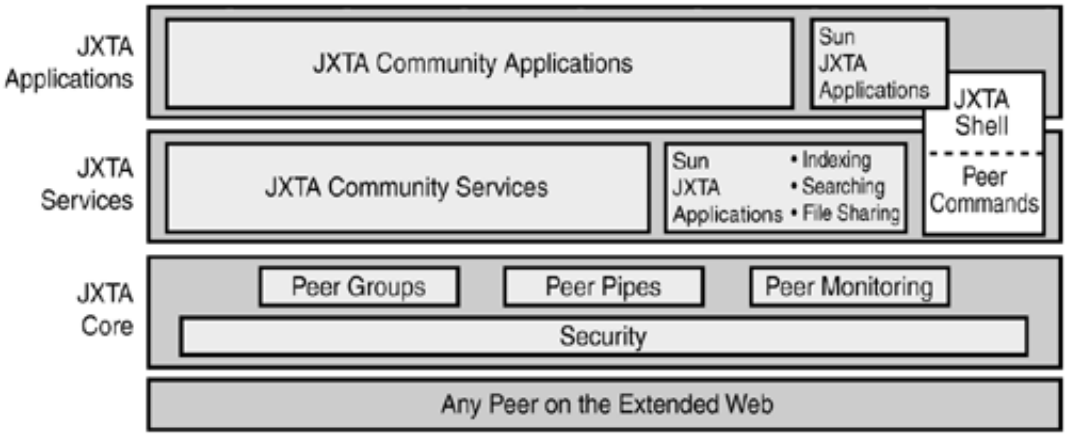

Figura 3. Arquitetura do JXTA.
Fonte: (FLENNER *et al.*, 2002).

A última camada é a de Aplicação (*applications*), é nela que o desenvolvedor cria seus sistemas P2P para a que outros possam usá-los em rede, seja na Internet ou em redes locais. Essa camada auxilia no desenvolvimento de vários tipos de aplicações, como compartilhamento de arquivos, armazenamento distribuído, e mensagens instantâneas (FLENNER *et al.*, 2002). Um ponto importante é que a linha de divisão entre as camadas não é rígida. Se o desenvolvedor cria um nó que fornece funcionalidades, outro nó deveria enxergar a funcionalidade fornecida como um serviço que pode ser utilizado pelo outro, enxergando como uma aplicação completa, sem necessidade de adicionar outras partes.

5.3 SERVIÇO DE GERENCIAMENTO DE CONTEÚDO

O Serviço Gerenciador de Conteúdo (CMS) habilita para um nó o compartilhamento de arquivos, como documentos, fotos, áudio ou outros tipos com nós remotos. Para manter a consistência com a especificação, o CMS utiliza anúncios para fornecer informações relacionadas ao conteúdo que será compartilhado pelo nó, e utiliza JXTA pipes para a transferência do conteúdo.

Segundo Gradecki (2002) o CMS mantém um registro dos arquivos compartilhados pelo nó local. Nele são armazenadas informações relacionadas ao conteúdo dos arquivos publicados. Estas informações são usadas para agilizar o acesso ao conteúdo do nó local.

Os canais de comunicação (pipes) do JXTA são usados para receber requisições, consultas e conteúdo de outros nós; para cada instância do CMS é criado um canal de comunicação. Quando uma consulta é realizada, uma mensagem LIST_REQ é enviada para nós remotos para obter a lista de arquivos compartilhados. Todos os nós remotos responderão com uma mensagem LIST_RES contendo um ou mais anúncios de conteúdos compartilhados.

Quanto um nó local solicita o download de um arquivo, uma mensagem GET_REQ é enviada ao nó específico baseada na informação encontrada no anúncio recebido. Toda comunicação é realizada pelos protocolos JXTA.

6 FERRAMENTA DE BACKUP

A ferramenta desenvolvida é divida em dois módulos, um de configuração, e outro de backup que fará uso das configurações geradas pelo primeiro.

6.1 MÓDULO DE CONFIGURAÇÃO

Este módulo tem a função de gerar um arquivo XML que será utilizado pelo módulo de backup. Este arquivo contém informações relevantes à execução do módulo de backup. Um exemplo de arquivo pode ser observado na Figura 4.

```
<configuration>
  <nomepeer>nome do peer</nomepeer>
  <pastacompartilhada>G:\Pasta Compartilhada</pastacompartilhada>
  <repositorio>C:\Repositorio Peer</repositorio>
  <pastasistema>C:\p2p</pastasistema>
  <pastapeer>C:\Peer</pastapeer>
</configuration>
```

Figura 4. Exemplo de arquivo de configuração.
Fonte: (Própria).

As informações descritas nos elementos XML são as seguintes:

❿ <nomepeer>: Nome que identifica o nó local, é utilizado para indicar aos nós remotos o dono dos arquivos publicados na rede.

❿ <pastacompartilhada>: Caminho da pasta do nó local onde estão os arquivos para backup. O módulo de backup sempre procurará os arquivos a serem publicados nesta pasta.

- <repositorio>: Indica no nó local, onde serão guardados os arquivos recebidos de nós remotos na rede.
- <pastasistema>: Caminho onde serão armazenados os arquivos com informações de conexão da ferramenta.
- <pastapeer>: Caminho de onde serão armazenados os arquivos com informações relacionadas ao nó local.

É importante salientar que todos os atributos no arquivo XML são obrigatórios.

O módulo de configuração foi desenvolvido para permitir que o de backup seja compatível com a estrutura do sistema de arquivos do sistema operacional onde ele será executado. Ou seja, o padrão de nomenclatura do sistema de arquivos não fica vinculado ao programa.

6.1 MÓDULO DE BACKUP

A Figura 5 mostra os passos principais de execução dos módulos de backup, que serão detalhados a seguir.

O módulo de backup é iniciado com a leitura do arquivo XML gerado pelo módulo de configuração. Primeiramente verifica se há arquivos anteriores nas pastas definidas em <pastasistema> e <pastapeer>, se houver são todos apagados. Assim, toda vez que o módulo é iniciado, os arquivos de informação de conexão e os de informações locais são recriados.

É importante notar que cada computador com o módulo em execução mantém um servidor, e um nó local.

O servidor recebe tanto conexões do nó local quanto de nós remotos. Depois de estabelecida a conexão do nó com o servidor local, é iniciada a busca de arquivos na pasta definida para publicação (<pastacompartilhada>), estes serão publicados pelo CMS do JXTA.

A publicação dos arquivos é feita pelo CMS, que armazena informações sobre os arquivos publicados. Para cada arquivo que é publicado, é gravado uma referência a ele, e outras informações adicionais referentes a sua origem e integridade.

O controle da integridade do arquivo é feito utilizando-se uma assinatura *hash*. O método escolhido é o algoritmo MD5. Já para as informações pertinentes à origem do arquivo, é utilizado o nome do nó local, definido em <nomepeer>. Este nome é utilizado para que os nós remotos identifiquem o dono do arquivo.

Foi definido um tempo limite para que os arquivos na pasta de backup sejam republicados, este tempo foi especificado em 1 (uma) hora. Assim, a cada hora todos os arquivos que estão na pasta compartilhada são republicados, para que sejam identificados por outros nós os novos arquivos ou os que sofreram modificações desde a última publicação.

Uma vez publicados os arquivos locais, é enviada uma requisição de arquivos remotos aos demais nós parceiros da ferramenta. São analisados os anúncios recebidos após a requisição para verificar a existência de arquivos remotos.

Se nos anúncios recebidos dos nós remotos não forem encontradas informações de arquivos, o módulo de backup aguarda o tempo pré-determinado antes de publicar novamente seus arquivos locais e realizar uma nova busca de arquivos nos anúncios.

Após a análise dos anúncios trocados pelos nós da rede, é verificada a existência de arquivos remotos publicados na rede. Para cada arquivo remoto encontrado, é feita a varredura no repositório local, definido em <repositorio>, para verificar se há neste um arquivo com o mesmo nome e extensão do remoto que está sendo analisado.

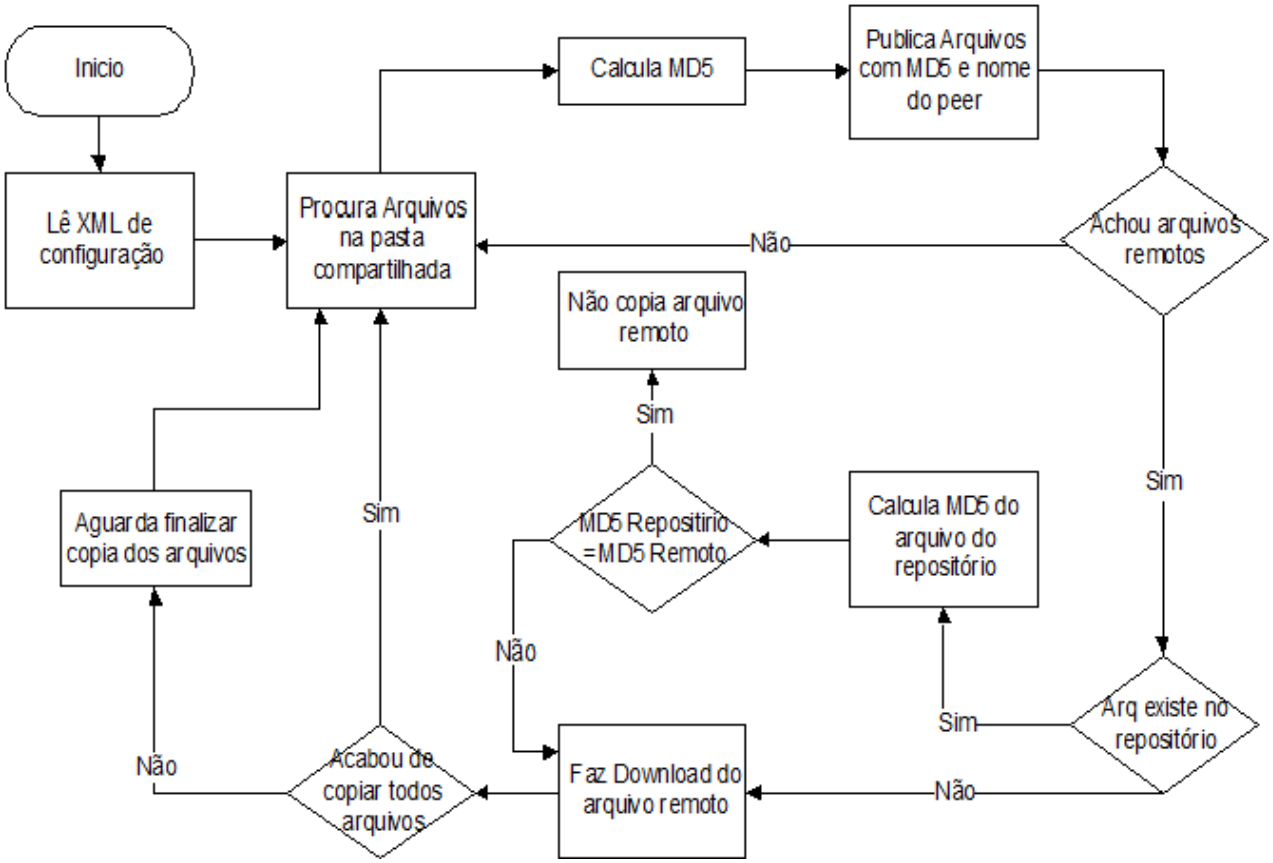

Figura 5. Fluxograma com os passos da execução do módulo de Backup.
Fonte: (Próprio).

Não sendo encontrado um arquivo correspondente, ele é solicitado ao nó remoto, ou seja, é feita a requisição de download do arquivo a partir do nó remoto.

Caso seja encontrado no repositório local um arquivo com o mesmo nome, é feita outra verificação para saber se o arquivo remoto é idêntico ao local. Para isso é calculada a assinatura MD5 do arquivo local. Esta, então, é comparada com a assinatura encontrada nas informações do anúncio onde foi publicado o arquivo remoto. Se as

assinaturas forem idênticas, não há necessidade de fazer uma requisição de download do arquivo ao nó remoto.

Se os nomes dos arquivos forem iguais, mas as assinaturas diferirem é identificado que o conteúdo do arquivo remoto pode ter sido modificado ou o local está corrompido, necessitando baixá-lo novamente. Quando isto ocorre, é realizada uma nova requisição ao nó que publicou o arquivo, para iniciar o novo download do arquivo.

Após a requisição de download, o efetivo controle da transmissão do arquivo é feito utilizando-se *threads*. Uma é gerada para cada arquivo solicitado, respeitando-se um limite máximo de transmissões simultâneas. Essa abordagem permite que mais de um arquivo seja recuperado de forma simultânea, sem impactar o desempenho do sistema local ou gerar uma sobrecarga na rede.

Cada *thread* faz a transmissão do arquivo de forma fragmentada, de forma alternada. Foi verificado que se todos os *threads* são iniciados e executados simultaneamente, não são transmitidos todos os fragmentos dos arquivos, deixando-os incompletos. Para solucionar o problema foi implementado um gerenciador de *threads* que limita em 5 (cinco) o número máximo de execuções simultâneas. Os *threads* que não estão em execução, aguardam em uma fila, e assim que uma é finalizada começa a execução de outra, até acabarem todos os *threads* da fila.

Enquanto existam arquivos sendo copiados, não são feitas novas publicações dos arquivos locais, e nem requisições de arquivos remotos. Apenas após o término da cópia dos arquivos é realizada a publicação dos arquivos locais e a busca de arquivos remotos.

7 CONSIDERAÇÕES FINAIS

Após o desenvolvimento da ferramenta foi possível concluir que JXTA é um *middleware* com uma grande variedade de recursos para a implementação de sistemas P2P. Na ferramenta foram utilizados massivamente os serviços de descoberta e o CMS para gerenciar a publicação e download dos arquivos na rede.

As maiores dificuldades no desenvolvimento foram a falta de material atualizado sobre JXTA, o gerenciamento das *threads*, para que todos os fragmentos dos arquivos fossem copiados e como as informações referentes ao arquivo fossem publicadas de forma que ficassem disponíveis aos outros nós da rede.

Como trabalhos futuros poderiam ser implementados, além do módulo de backup, um módulo que, em caso de perda de arquivos do nó local, recuperaria os arquivos que foram salvos em outros nós da rede.

Também seria interessante, implementar uma camada de segurança, de forma a garantir que apenas os proprietários dos arquivos sejam capazes de acessá-los nos repositórios remotos.

REFERÊNCIAS

BARCELOS, M. P.; GASPARY, L. P. Fundamentos, tecnologias e tendências rumo a redes P2P seguras. In: BREITMAN, K.; ANIDO, R. **Atualizações em informática**. Rio de Janeiro: PUC-Rio , SBC, 2006.

BROOKSHIER, D. et al. **JXTA: Java P2P Programming**. Indianapolis: Sams Publ., 2002.

COULOURIS, G. **Sistemas distribuídos: conceitos e projeto**. São Paulo: Pearson Education, 2007. v.4.

DEVMEDIA. **JXTA – Parte II**. 2015. Disponível em: <http://www.devmedia.com.br/jxta-parte-ii/6054>. Acesso em: 19 jun. 2015.

FLENNER, R. et al. **Java P2P Unleashed**. Indianapolis: Sams Publ., 2002.

GRADECKI, J. D. **Mastering JXTA: building java peer-to-peer applications**. John Wiley & Sons, 2002.

STEINMETZ, R.; WEHRLE, K. **Peer-to-peer systems and applications**. Berlin: Springer, 2005. https://doi.org/10.1007/11530657

SCHODER, D.; FISCHBACH, K. Peer-to-peer prospects. **Communications of the ACM**, v.46. n.2, p.27–29, 2003. https://doi.org/10.1145/606272.606294

TAYLOR, I. J. **From P2P to web services and grids – peers in a client/server world**. Unites States: Springer, 2005.