X-Peer: Um Middleware para Aplicações Peer-to-Peer[∂]

João B. da Rocha Júnior[1], Joseane F. Fidalgo[2], Ramide A. S. Dantas[2], Luciana P. Oliveira[2], Carlos A. Kamienski[3], Judith Kelner[2], Djamel F. H. Sadok[2]

[1]Departamento de Exatas – Universidade Estadual de Feira de Santana – UEFS
[2]Centro de Informática – Universidade Federal de Pernambuco – UFPE
[3] Centro Federal de Educação Tecnológica da Paraíba – CEFET-PB

joao@uefs.br, {joseane, ramide, lpo, cak, jk, jamel}@gprt.ufpe.br

Abstract. *Popular P2P applications may be built using pure or hybrid models. In an attempt to offer the best of both worlds, a new middleware has been designed. It adopts the concepts of super-nodes running DHT search in order to guarantee scalability. X-Peer configuration ranges from centralized to pure DHT. It is currently being deployed and tested over the Brazilian academic network (RNP). This paper describes the X-Peer architecture, the applications developed for its testing and provides a preliminary performance evaluation.*

Resumo. *As aplicações P2P, atualmente em grande evidência, podem ser implementadas de acordo com os modelos puro e híbrido. Com o objetivo de extrair as maiores vantagens de cada um dos modelos, foi proposto o middleware X-Peer, que utiliza o conceito de super-nós executando sobre um modelo estruturado DHT para garantir escalabilidade. Dessa forma, o X-Peer pode ser configurado para ser desde um sistema P2P centralizado até um DHT puro. O X-Peer atualmente está em fase de testes em PoPs da RNP. Este artigo apresenta a arquitetura do X-Peer, aplicações desenvolvidas para testá-lo e uma avaliação preliminar de seu desempenho.*

1. Introdução

A computação peer-to-peer (*P2P*) tem promovido uma grande modificação nos padrões de uso da Internet nos últimos anos. Os sistemas P2P possuem algumas vantagens em relação à computação cliente/servidor. Entre elas, a possibilidade de colaboração e acesso diretos entre os usuários e seus recursos, independentes de uma organização central ou hierárquica e do uso de *firewalls* e *NATs*. Uma característica básica desses sistemas é dispor aos seus integrantes, chamados de *peers* (pares) ou nós, as mesmas capacidades e responsabilidades [1].

Os sistemas *P2P* podem ser implementados de maneira parcialmente centralizada (sistemas híbridos) ou totalmente descentralizada (sistemas puros). Um sistema P2P híbrido contém alguns elementos centralizados, que são nós com um papel diferenciado dos demais. Sistemas híbridos podem ser subdivididos em totalmente centralizados e hierárquicos. Exemplos de sistemas híbridos centralizados são o Napster

[∂] O X-Peer está sendo desenvolvido pelo GPRT/CIN/UFPE, com financiamento da RNP.

[7] e os sistemas de mensagens instantâneas, como o Yahoo! Messenger e o MSN Messenger [8]. Estes sistemas têm problemas semelhantes aos de sistemas cliente/servidor, como falta de robustez e escalabilidade. Os sistemas híbridos hierárquicos possuem nós especiais (chamados de super-nós, super-peers ou refletores), que atuam como servidores para um pequeno grupo de nós. A comunicação entre esses super-nós é realizada através de um mecanismo dependente de implementação. Por exemplo, os super-nós do KaZaA [9] se comunicam entre si de maneira totalmente não estruturada, o que dificulta a recuperação de todas as informações existentes na rede.

Os sistemas puros subdividem-se em sistemas estruturados e não estruturados, que se diferenciam, respectivamente, pela capacidade de recuperar todas as informações disponíveis na rede ou apenas parte delas. Todavia, nem sempre é desejável implementar sistemas puros, por motivos como desempenho, especialização de funções e robustez. Como exemplos de sistemas estruturados tem-se os baseados em mecanismos como Chord [2], Pastry [3], Tapestry [4] e CAN [5], que são sistemas estruturados na forma de tabelas *hash* distribuídas (*DHT – Distributed Hash Tables*) [8]. Como exemplo de sistema não estruturado tem-se o Gnutella [6].

Em geral, uma limitação dos atuais sistemas P2P é a existência de uma rede P2P distinta para suportar cada aplicação específica. Por exemplo, no caso de aplicações de compartilhamento de arquivos, existem as redes FastTrack (KaZaA), eDonkey e BitTorrent [8]. Para mensagens instantâneas existem as redes Yahoo!, MSN e ICQ e para VoIP existe a rede do Skype [10]. Uma vez que os sistemas P2P modernos são relativamente recentes, ainda não se desenvolveu uma plataforma (rede) global que suporte vários tipos de aplicações, possibilitando a redução de custos devido ao compartilhamento de recursos.

Nesse contexto foi proposto o middleware X-Peer, que define um novo modelo de arquitetura que utiliza o melhor dos modelos P2P citados. O X-Peer utiliza o conceito de super-nós que se comunicam através de uma rede estruturada baseada em DHT, o que lhe garante os benefícios da escalabilidade e robustez. O modelo do X-Peer pode ser configurado para se comportar como um sistema totalmente centralizado, caso exista apenas um super-nó, até um sistema totalmente descentralizado, caso o X-Peer esteja em execução em cada nó da rede. Neste caso, o X-Peer se assemelha ao Freenet [8], onde a aplicação se comunica com uma instância local do sistema P2P. Com um número intermediário de super-nós, o X-Peer funciona de maneira semelhante ao KaZaA (modelo híbrido), sendo que neste caso os super-nós se comunicam de maneira estruturada, através de um sistema DHT.

Outra característica distinta do X-Peer é sua capacidade de suportar várias aplicações P2P em uma única infra-estrutura de rede. Neste artigo são apresentadas quatro aplicações que foram desenvolvidas para o X-Peer, como um exemplo das funcionalidades suportadas. Na seqüência deste artigo, a seção 2 apresenta o X-Peer, incluindo a arquitetura do X-Peer e seus componentes, o mecanismo de replicação, as aplicações que foram desenvolvidas no X-Peer, um exemplo de seu funcionamento e os resultados de algumas avaliações de desempenho feitas no X-Peer. A seção 3 aborda os trabalhos relacionados, comparando-os com o X-Peer. A seção 4 apresenta as considerações finais, focando as contribuições e os trabalhos futuros do X-Peer.

2. X-Peer

O X-Peer é um middleware para a construção de aplicações P2P que utiliza o conceito de super-nós que se comunicam através de uma rede estruturada baseada em DHT, o que lhe permite recuperar todas as informações disponíveis na rede X-Peer[11]. Essa garantia de localização em uma rede distribuída e hierárquica é um dos grandes diferenciais dessa solução. Além disso, o X-Peer é capaz de suportar várias aplicações P2P em uma única infra-estrutura de rede. A arquitetura distribuída da plataforma X-Peer é apresentada na Figura 1, onde observam-se três nós X-Peer, que se comunicam via DHT. Também pode-se observar algumas aplicações utilizando os serviços dos nós X-Peer através de comunicação via Sockets/TCP e a comunicação direta (P2P) entre as aplicações.

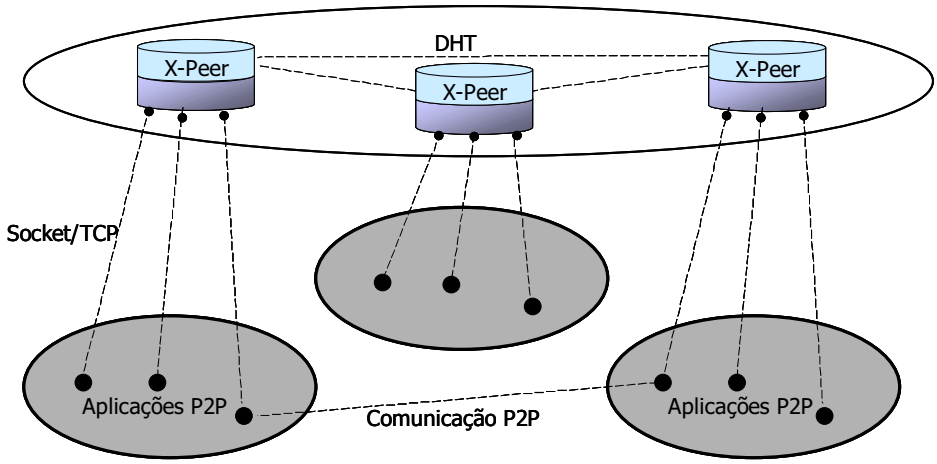

Figura 1: Arquitetura distribuída da plataforma X-Peer.

A rede X-Peer pode ser usada para publicar e recuperar diversos tipos informações, incluindo meta-informações. Uma meta-informação no X-Peer pode ser vista como uma informação sobre os diversos tipos de dados que podem ser armazenados numa rede X-Peer. Por exemplo, no caso de compartilhamento de arquivos, as meta-informações de um arquivo podem ser referentes às propriedades e palavras-chave associadas a esse arquivo. Essas informações são armazenadas de forma distribuída entre os super-nós que compõem a rede X-Peer e são categorizadas em campos (tipos) e valores. Os campos são utilizados pelo DHT para indicar o local (nó) em que as informações referentes a esse campo estarão armazenadas. Com essa estruturação é possível realizar buscas por palavras-chave e melhorar o desempenho na recuperação das informações.

A estruturação dos super-nós e das informações no X-Peer permite utilizar o melhor dos modelos DHT e baseado em super-nós. Nesse contexto, o X-Peer pode ser visto como uma rede (quando se observam os nós X-Peer), como arquitetura P2P (considerando sua estruturação) e como *middleware* (considerando a infra-estrutura oferecida para o desenvolvimento de aplicações P2P). Assim, para evitar ambigüidades, será adotado como padrão nesse artigo o termo X-Peer para denominar o *middleware* X-Peer e referência explícita para os demais termos.

2.1. Arquitetura do X-Peer

O objetivo do X-Peer é possibilitar a utilização de recursos distribuídos de forma simples, através de um conjunto básico de serviços disponíveis para aplicações cadastradas. A arquitetura do X-Peer é ilustrada pela Figura 2.

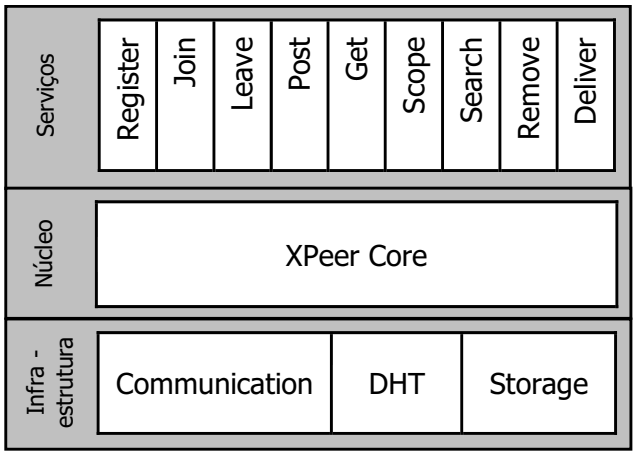

Figura 2: Arquitetura do X-Peer

A arquitetura do X-Peer é composta por três níveis principais, infra-estrutura, núcleo e interface de serviços. A infra-estrutura é composta pelos módulos *Communication*, DHT e *Storage*, que provêem serviços ao núcleo.

O módulo de comunicação (*Communication*) tem a função de gerenciar as conexões dos clientes com o nó X-Peer, assim como permitir a comunicação direta entre *peers*, inclusive usando a rede X-Peer para intermediar essa comunicação caso os *peers* estejam sob NAT ou *firewall*. O módulo DHT define uma interface genérica para mecanismos baseados em tabela *hash* distribuída (*Distributed Hash Table*, DHT) tais como o Pastry e Chord. Este módulo realiza o roteamento das mensagens entre os nós que compõem a rede X-Peer e foi construído utilizando o FreePastry (implementação livre do Pastry). Todavia, qualquer outro mecanismo DHT pode substituí-lo desde que forneça a mesma API, como o Chord. O módulo *Storage* compreende o sistema de armazenamento em arquivo desenvolvido especificamente para o X-Peer e realiza a persistência das informações publicadas na rede X-Peer.

O núcleo (X-Peer Core) tem a função de atender as requisições de serviços dos *peers* clientes. Essas requisições de alto nível são transformadas em operações de armazenamento e recuperação de informações, realizadas através da troca de mensagens internas DHT, entre o nós que compõem a rede X-Peer. Também é função do núcleo a replicação e a redistribuição das informações quando ocorre a entrada de novos nós X-Peer na rede. O núcleo faz uso dos serviços da camada de infra-estrutura no roteamento das mensagens internas (módulo DHT), no armazenamento das informações (módulo *Storage*) e na gerência das conexões com as aplicações (módulo *Communication*).

A interface de serviços contém os serviços disponibilizados pelo X-Peer para a construção de aplicações P2P. Esses serviços são disponibilizados na forma de um conjunto de mensagens e uma API fornecida às aplicações para a comunicação entre clientes e os nós X-Peer.

2.1.1. Serviços disponibilizados pelo X-Peer

A parte mais importante do *middleware* X-Peer do ponto de vista do desenvolvimento de sistemas são os serviços oferecidos para a construção de aplicações P2P, descritos em seguida:

- Register –Serve para cadastrar um usuário ou uma aplicação – um usuário só pode ter acesso aos serviços do X-Peer se tiver sido previamente cadastrado;
- Join – Possibilita ao usuário solicitar o seu ingresso na rede. Para que um usuário seja aceito pela rede, ele deve possuir um identificador único e uma senha. Esse serviço também é responsável por autenticar e garantir a identidade de um usuário, podendo ser utilizado a partir de qualquer dispositivo capaz de se conectar a rede, como *laptops* e *PDAs*;
- Post – Permite ao usuário/aplicação solicitar o armazenamento de dados na rede X-Peer, incluindo meta-informações. Cada informação a ser armazenada é composta por nome e valor do campo e os tipos de acesso (public, protected ou private) e de armazenameto (volátil ou persistente);
- Scope – Especifica a restrição de acesso pertinente a cada informação a ser publicada, composto pelos tipos public, protected ou private. No tipo public o acesso é permitido a todos os usuários, no protected o acesso é apenas aos usuários cadastrados no escopo do usuário que publicou a informação e no private o acesso é exclusivo ao usuário que publicou a informação;
- Get – Recupera informações armazenadas na rede X-Peer através do serviço Post, de acordo com o escopo definido no serviço Scope;
- Search – Permite realizar pesquisas direcionadas aos valores dos campos especificados com base em palavras-chaves fornecidas pelo usuário. Essa localização pode ser feita através do nome do usuário ou de uma parte da informação publicada por ele, ex: palavra-chave;
- Remove – Remove um campo publicado através do Post;
- Deliver – Permite enviar uma mensagem para outra aplicação conectada na mesma rede X-Peer, fornecendo um mecanismo que possibilita a troca de mensagens entre nós que estão sob *NAT* ou *firewall*;
- Leave – Representa a mensagem de desconexão do usuário, incluindo uma requisição do usuário à rede X-Peer para remoção das informações voláteis publicadas pelo usuário. As informações persistentes de um usuário permanecem armazenadas no X-Peer enquanto o usuário estiver cadastrado.

2.1.2. Mecanismo de Comunicação do X-Peer

O mecanismo básico de comunicação do X-Peer é semelhante ao de uma rede P2P comum. Inicialmente uma aplicação conectada faz uma requisição de localização de recurso/informação ao X-Peer, que lhe retorna a localização do recurso. A partir desta informação, a aplicação inicia comunicação direta com o peer detentor do recurso. Entretanto, se este peer não puder ser contactado diretamente por estar sob *NAT* ou *firewall*, o X-Peer intermediará essa comunicação, operando como um mediador.

A comunicação entre os super-nós na rede X-Peer é realizada através de uma rede estruturada baseada em DHT e a comunicação entre os clientes e os nós X-Peer é realizada através de mensagens de serviço do X-Peer. Inicialmente foi utilizado *XML* (*EXtensible Markup Language*) na codificação das mensagens. Entretanto, através do uso de um Profiler (ferramenta usada para localizar problemas de desempenho) [15] e

testes de carga no X-Peer, observou-se que o desempenho geral com codificação em XML não era satisfatório e decidiu-se adotar uma codificação binária interna.

2.1.3. Armazenamento e Recuperação de Dados no X-Peer

Uma das limitações do modelo estruturado baseado em DHT é a baixa flexibilidade na recuperação das informações, uma vez que para recuperar a localização de uma informação armazenada deve-se fornecer exatamente a chave utilizada para gerar o índice na tabela hash (busca exata). Além desse mecanismo básico, o X-Peer adicionou uma busca por palavras-chave. No cadastro de uma informação a ser publicada na rede X-Peer, o usuário deve fornecer as palavras-chave associadas às informações e estas são estruturadas internamente no X-Peer. No momento da busca da informação, pode ser fornecida tanto a chave exata quanto as palavras-chave associadas no cadastro da informação. Neste caso é feita uma busca nos arquivos de estruturação para recuperar a chave original e posteriormente é realizada uma consulta nas tabelas hash do DHT.

2.2. Mecanismo de Replicação

Em uma estrutura em anel, quando um novo nó possuindo ID X entra na rede os vizinhos virtuais desse nó são notificados de sua presença, armazenam uma referência para esse nó e passam a enviar mensagens do tipo "ping" para saber se esse nó ainda está ativo. Caso o nó X não responda às mensagens durante um determinado período, seus vizinhos assumem que o nó deixou de funcionar e atualizam suas tabelas, removendo a referência para esse nó. Nessa estrutura em anel, cada nó é responsável por tratar um conjunto de chaves e os nós vizinhos de um dado nó figuram como seus substitutos naturais, uma vez que quando um nó sai do anel os seus vizinhos passam a responder automaticamente pelo nó que saiu, sem que a rede como um todo seja comprometida. Na Figura 3, por exemplo, o nó identificado pelo nó ID 0122 tem como vizinhos os nós 0112 e 0131.Caso o nó 0122 deixe de funcionar, o conjunto de chaves que eram tratadas por esse nó passarão a ser atendidas pelo nó 0112 ou pelo nó 0131 ou por ambos, dependendo da estrutura de DHT utilizada.

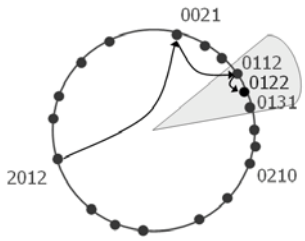

Figura 3: Nível de replicação igual a 1.

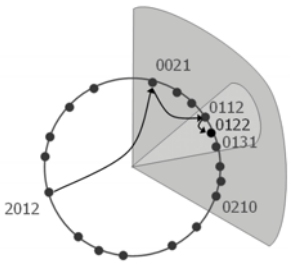

Figura 4: Nível de replicação igual a 4.

Baseado nessas considerações, o X-Peer criou o conceito de nível de replicação, que corresponde ao número de nós posteriores e anteriores que possuem cópias das informações de um nó e são capazes de substituí-lo caso esse nó venha a falhar. Na Figura 3, considerando o nó 0122 por exemplo, o nível de replicação é igual a 1, ou seja, existe uma máquina a frente desse nó (0112) e uma máquina atrás (0131) que armazenam uma cópia do conteúdo armazenado no nó 0122 e estão aptas a substituí-lo em caso de falha. Quanto maior o nível de replicação, maior a garantia de que as informações não serão perdidas. Todavia, como há uma degradação de desempenho em

níveis de replicação altos devido ao maior número de mensagens para manter essa estrutura atualizada, deve ser considerado o trade-off entre redundância e desempenho. Ainda considerando o nó 0122, a Figura 4 exibe uma estrutura com nível de replicação 4 para esse nó.

2.3. Aplicações Desenvolvidas

Para demonstrar a viabilidade do X-Peer foram desenvolvidas quatro aplicações simples, Xat, XBall, XBrain e XatMobile, que utilizam os principais serviços do X-Peer.

O XBall é uma aplicação simples de jogo da velha, enquanto o Xat é uma aplicação de troca de mensagens instantâneas. As aplicações utilizam a infra-estrutura para autenticar usuários, localizar oponentes (XBall) e amigos (Xat), armazenar dados e realizar também a comunicação fim a fim.

O XBrain é uma aplicação de educação à distância onde os usuários possuem dois papéis distintos, colaboradores (figura do professor) e participantes (figura do aluno). Os colaboradores cadastram suas áreas de conhecimento e os usuários participantes pesquisam quem são os colaboradores para a área de seu interesse e, a partir dessa informação, entram em contato direto (*P2P*) com os colaboradores. Essa aplicação foi feita usando-se as APIs do X-Peer e tanto a pesquisa quanto o contato entre colaborador e participante são feitos usando-se a infra-estrutura da rede X-Peer.

O XatMobile é um aplicativo simples de troca de mensagens entre dispositivos móveis desenvolvido pelo grupo do X-Peer com o objetivo principal de identificar a viabilidade em desenvolver aplicações móveis para o XPeer utilizando a tecnologia J2ME. O XatMobile foi desenvolvido utilizando dois Palm Top's Sony Clié PEG-TJ37, com interface IEEE 802.11 e processador de 200Mhz. Para adaptar o X-Peer às restrições impostas por ambientes móveis, foi criado um novo módulo responsável pela comunicação com esses dispositivos. Como resultado, foi obtido um aplicativo compatível com uma vasta gama de dispositivos que suportam o perfil MIDP (Mobile Information Device Profile) 2.0 [16], que possibilita a troca de mensagens entre duas aplicações executando em dispositivos móveis. Todavia, devido às restrições naturalmente impostas por ambientes móveis, como capacidade de processamento e memória, continuam sendo realizadas melhorias, relacionadas ao desempenho na recuperação de informações e uso do X-Peer.

2.4. Exemplo de Funcionamento

Para facilitar a compreensão do X-Peer, segue uma breve descrição sobre um de seus serviços, o Register, cuja função é cadastrar novos usuários na rede. O cadastro pode ser realizado a partir de qualquer aplicação conectada à rede X-Peer. Para efetuar o cadastro, o usuário deve fornecer seu login e senha, além de uma descrição. Essas informações são codificadas no formato de mensagem do X-Peer e submetidas ao nó no qual a aplicação está conectada.

No nó X-Peer, essa mensagem é quebrada em um conjunto de mensagens internas, que realizam o acesso a tabela *DHT*, incluindo uma mensagem de retorno, informando se a operação foi realizada com sucesso ou não. A Figura 5 contém um exemplo de mensagem interna PUT, responsável por armazenar uma informação na

rede X-Peer. Como exemplo, é ilustrado o cadastro do usuário com login “Boanerge”, senha “Belinha”, aplicação “Xat” e descrição “Coronel de Vila da Mata”.

Os dados submetidos à rede X-Peer são identificados pelo nome do campo que é utilizado como chave para a função de *hash,* que gera um identificador a partir de seu valor. Na Figura 5 pode-se observar o funcionamento desse mecanismo, assumindo uma função *hash* que gera, como identificadores, números entre 0 e 100. Cada super-nó X-Peer é responsável por armazenar as informações dos campos cujo número produzido pela função *hash* esteja contido na faixa de valores administrada pelo nó, como ocorre tipicamente nos mecanismos baseados em *DHT*.

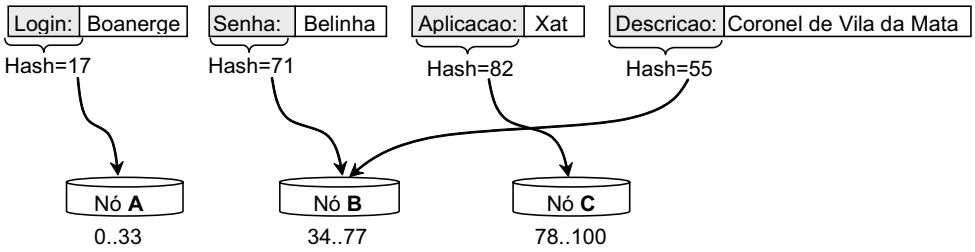

Figura 5: Distribuição das meta-informações entre os super-nós X-Peer.

Considerando que a função hash é aplicada somente sobre o nome do campo, todos os valores referentes a um mesmo campo permanecerão armazenados em um mesmo nó e replicados em outros nós, de acordo com o mecanismo de replicação utilizado no X-Peer descrito anteriormente. Por exemplo, no sistema ilustrado na Figura 5, todos os *logins* de usuários serão armazenados no nó A. Esse esquema de armazenamento pode ser utilizado para aumentar o desempenho e permitir a recuperação rápida de informações, visto que possivelmente um único nó é acessado para recuperar as informações de um campo. Também permite realizar consultas mais sofisticadas nos valores. Entretanto, dependendo da granularidade das chaves fornecidas ao X-Peer, alguns problemas na distribuição das chaves podem ocorrer devido à concentração de chaves em um determinado nó.

A situação exposta no parágrafo anterior é apenas uma possível solução para tratar do *trade-off* entre o grau de distribuição das chaves e a eficiência da busca. Neste aspecto, a contribuição do X-Peer é deixar esse compromisso sob a responsabilidade do desenvolver de aplicações. Outra maneira de modelar o registro do login desse exemplo seria incluir o nome do usuário na chave, que geraria uma distribuição mais uniforme.

2.5. Avaliação

Foram realizadas avaliações preliminares do comportamento do *X-Peer*, onde foi possível coletar algumas informações sobre o seu funcionamento. Nessa avaliação, três nós *X-Peer* foram configurados em três PoPs da RNP, localizados em Pernambuco, Minas Gerais e Paraná. Para a avaliação do X-Peer, foi utilizada a aplicação XBrain com seis usuários distintos e conectados a PoPs diferentes, dois em cada PoP. Foram coletados os dados que compreendem os tempos de execução e carga de mensagens internas DHT na rede para as seguintes operações: conexão dos usuários ao X-Peer, cadastro do usuário e aplicação, cadastro dos usuários em diferentes áreas de conhecimento, localização de informações e saída dos usuários da rede X-Peer. A execução dessas operações envolveu a utilização de um subconjunto dos serviços do X-Peer de uso mais comum pelas aplicações (Join, Leave, Register, Post, Get, Search).

Um problema identificado durante os testes, que também é recorrente em soluções baseadas em tabelas *hash* distribuídas, é o envio de um número de mensagens para atender a uma única requisição de serviço de mais alto nível. A Figura 6 ilustra a quantidade média de mensagens internas (mensagens enviadas entre os nós X-Peers) necessárias para atender os principais serviços ofertados pelo X-Peer (mensagens externas solicitada pelo cliente). Analisando-se a Figura 6, é possível perceber que o nó X-Peer, ao atender a uma requisição de serviço de um cliente, envia em média cinco mensagens internas de gerência do DHT para outros nós X-Peer. Observa-se também que o Join é o serviço que propaga o maior número de mensagens, porque no momento em que o usuário se conecta ao X-Peer é necessário armazenar uma grande quantidade de meta-informações sobre o usuário, necessárias enquanto o usuário estiver conectado.

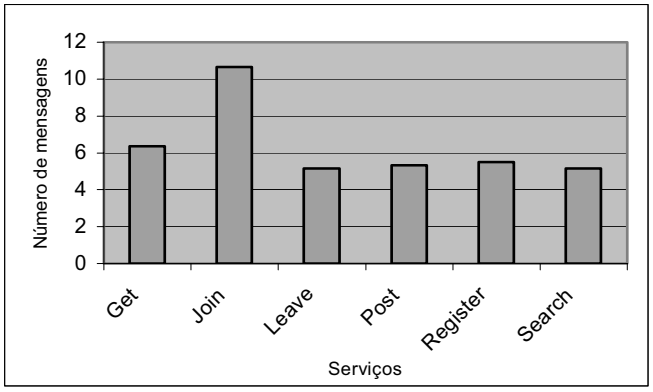

Figura 6: Número de mensagens internas por serviços do X-Peer

A Figura 7 apresenta o tempo gasto entre o recebimento de um serviço por um nó *X-Peer* e o seu processamento, incluindo o tempo gasto entre a solicitação das várias mensagens internas. A análise dessa figura mostra que os serviços que consomem mais tempo são o *Join*, o *Register* e o *Leave*. Isso ocorre principalmente porque esses serviços alteram, inserem e removem várias informações do usuário de uma única vez. O *Join*, por exemplo, é o serviço que gera maior número de mensagens internas, enquanto que o *Register* e o *Leave* são os serviços que armazenam e removem mais informações no *X-Peer*. O desvio padrão acentuado apresentado pelo serviço *Register* se deve ao fato de que algumas vezes foram geradas requisições na tentativa de registrar uma aplicação que já estava registrada, e, nesse caso o tempo para processá-las é bastante baixo.

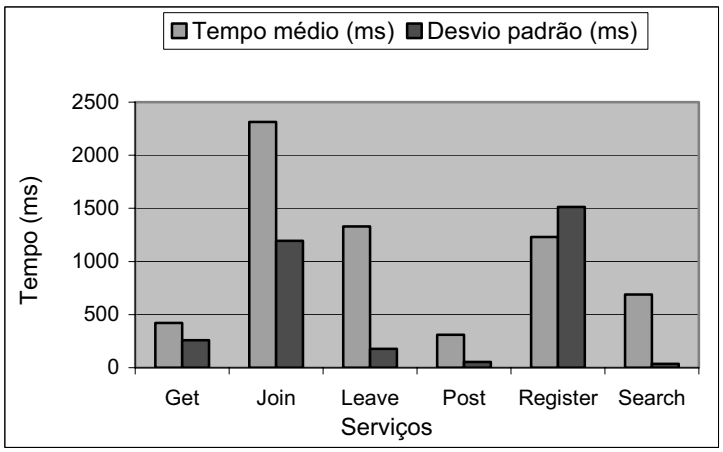

Figura 7: Tempo necessário para processar os principais serviços do X-Peer.

Mesmo considerando a simplicidade do cenário utilizado, esta avaliação permitiu identificar, dentre os serviços do X-Peer, aqueles passíveis de melhorias, no sentido de reduzir a quantidade de mensagens internas geradas e o tempo total de atendimento à requisições. Salienta-se que os serviços que demonstraram pior desempenho — Join, Leave e Register — são, tipicamente, requisitados com freqüência inferior a dos serviços Post, Get e Search, o que provoca pouco impacto no desempenho geral das aplicações. Avaliações mais complexas estão em andamento, todavia, os resultados coletados até o momento são compatíveis com os apresentados nessa seção.

3. Trabalhos Relacionados

Existem disponíveis na literatura diversos *middlewares* P2P, todavia, foi feita uma seleção desses *middlewares* no sentido de escolher, para serem abordados nessa seção, aqueles de maior destaque na literatura e/ou mais diretamente relacionados com X-Peer.

Um desses *middlewares é o* SAINT, cuja proposta é gerenciar a distribuição e acesso à meta-dados sobre uma estrutura P2P, que são extraídos automaticamente do conteúdo que o usuário deseja compartilhar. O SAINT fornece um conjunto de serviços genéricos para a publicação e recuperação de meta-dados, com uma semântica bem definida. Os meta-dados podem ser interpretados em qualquer plataforma, uma vez que o SAINT faz uso de *XML* (com algumas extensões) e do *framework RDF* (*Resource Definition Framework*) [12].

Do ponto de vista dos serviços, a proposta do X-Peer é similar a do SAINT, porém mais genérica, uma vez que permite a publicação de informações de forma geral, inclusive meta-dados. Quanto ao problema da localização de recursos, a arquitetura do SAINT não especifica o mecanismo utilizado na obtenção dos meta-dados. O X-Peer, por sua vez, implementa uma estrutura *DHT*, atualmente implementada sobre o FreePastry, que permite a recuperação confiável de informações. Além disso, o X-Peer também se mostra mais flexível, permitindo a adoção de modelos *P2P* híbridos ou puros.

O XMIDDLE [13] é um *middleware* voltado à elaboração de aplicações para dispositivos móveis, as quais realizam o compartilhamento de informações sobre uma estrutura de comunicação bastante dinâmica, típica dos ambientes com mobilidade. O XMIDDLE permite o compartilhamento de documentos XML entre as aplicações móveis, oferecendo também as funções de replicação e reconciliação destes documentos. Esta última função, bastante específica, porém importante para as aplicações móveis, realiza a consolidação e resolução de conflitos das modificações feitas *off-line* em um documento publicado na rede.

No XMIDDLE, os nós são estruturados em forma de árvore, permitindo a recuperação de informações de forma precisa. Esta estrutura, intrinsecamente hierárquica, é tolerável quando a quantidade de nós é pequena, porém, não é adequada quando esta quantidade aumenta, uma vez que também aumenta a carga sobre os nós próximos da raiz. O X-Peer, apesar de não enfocar ambientes com mobilidade, oferece algumas alternativas, por meio de sua estrutura P2P híbrida, para a construção de aplicações como as suportadas pelo XMIDDLE. Em um ambiente móvel heterogêneo, por exemplo, nós com maior capacidade de processamento e/ou memória poderiam ser eleitos como nós X-Peer, os quais atenderiam a requisições de nós mais “leves”.

O DERMI (*Decentralized Event Remote Method Invocation*) [14] é um *middleware* para o desenvolvimento de aplicações descentralizadas sobre uma infra-estrutura de rede P2P. É baseado em uma estrutura *DHT*, utilizando o Pastry como mecanismo de roteamento de mensagens, o Scribe como serviço de notificação, e o Past para os serviços de *caching* e replicação de objetos. O DERMI oferece os serviços tradicionais de um *middleware* orientado a objetos e alguns serviços novos. Como serviços tradicionais, o DERMI disponibiliza serviços de nomes – localização de objetos descentralizada, que permite encontrar as referências dos objetos —, e invocação síncrona de chamadas remotas sobre objetos. O DERMI também oferece serviços de mobilidade, replicação e *cache* de objetos (via Past). Seu objetivo é proporcionar um nível de abstração que possibilite o desenvolvimento de aplicações distribuídas utilizando uma rede P2P estruturada de grande escala.

O DERMI se baseia no conceito de objetos distribuídos e na invocação remota de procedimentos. O X-Peer oferece serviços de publicação e recuperação de informações e roteamento de mensagens, sobre os quais uma aplicação pode implementar os serviços do DERMI. O DERMI ainda está na sua fase inicial, não possuindo aplicações implementadas, enquanto o X-Peer já está em fase de implantação e testes na rede RNP, e já possui um conjunto de aplicações que fazem uso dos seus serviços.

4. Considerações Finais

O X-Peer está sendo desenvolvido pelo GPRT/CIN/UFPE (GT-P2P), com o patrocínio da RNP, com o objetivo de construir um mecanismo para facilitar o desenvolvimento de aplicações P2P por instituições ligadas a RNP. Entretanto, o X-Peer é um *middleware* que pode ser utilizado em corporações de pequeno, médio e grande porte como substrato básico para a construção de aplicações P2P. Uma grande vantagem do X-Peer é ocultar do desenvolvedor de aplicações uma série de complexidades inerentes a esse tipo de sistema.

O X-Peer é focado na escalabilidade e flexibilidade, utilizando super-nós sobre um modelo estruturado DHT e podendo ser configurado para ser desde um sistema P2P centralizado até um DHT puro. Além disso, devido à forma como as informações são estruturadas, amplia-se os tipos de consulta possíveis. Uma característica distinta do X-Peer é sua capacidade de suportar várias aplicações em uma única infra-estrutura P2P. Uma outra vantagem do X-Peer, inerente aos sistemas baseados em DHT, é a garantia de localização da informação em log(n) saltos, onde n é o número de super-nós.

O protótipo do X-Peer ainda está em fase de implantação e testes no *backbone* da RNP. Novos requisitos ainda estão sendo identificados e devem ser implementados para que o X-Peer se torne uma solução operacional e robusta. Estão em andamento testes de carga mais elaborados, bem como a construção de módulos do X-Peer no network simulator (ns-2), para simulá-lo sob diferentes cenários, especialmente aqueles difíceis de reproduzir em ambientes reais.

Além disso, serão adicionados ao X-Peer, na camada de núcleo, requisitos não funcionais como gerenciamento dos nós e segurança. Em relação à segurança, pretende-se usar criptografia para determinadas mensagens trafegadas na rede e adicionar mecanismos para evitar ataques de negação de serviço, uma vez que uma aplicação cadastrada no X-Peer pode solicitar inúmeras requisições em um curto período e comprometer todo o sistema. Para o gerenciamento uma possibilidade é que cada nó X-

Peer periodicamente publique informações sobre o seu funcionamento, tais como número de requisições atendidas, memória disponível e processamento médio. Essas informações seriam disponibilizadas para qualquer outro nó X-Peer e visualizadas por uma interface gráfica amigável exibindo gráficos que ilustram o funcionamento em tempo real de todos os nós.

Referências

[1] Parameswaran, M.; *et al.*, "P2P Networking: An Information-Sharing Alternative". IEEE Computer ,Volume 34 ,Issue 7 ,July 2001, pages31 - 38.

[2] Stoica, I *et al.* "Chord: A scalable peer-to-peer lookup protocol for internet applications". IEEE/ACM Transactions on Networking, Vol 11, Issue 1, Feb 2003, pages 17 – 32.

[3] Rowstron, A. *et al.*, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, 2001.

[4] Zhao, B. Y. *et al.* "Tapestry: An infrastructure for fault-tolerant wide-are location and routing". Technical Report UCB/CSD-01-1141, UC at Berkeley, CA, 2001.

[5] Ratnasamy, S. *et al.* "A scalable content-addressable network". SIGCOMM, Conference on Applications, technologies, architectures, and protocols for computer communications, San Diego, CA, USA, Pages: 161 – 172, 2001.

[6] Gnutella, Web Site, http://www.gnutella.com, último acesso em Novembro de 2004.

[7] Napster, Web Site, http://www.napster.com, último acesso em Fevereiro de 2005.

[8] Rocha Junior, J. B. *et al.*; "Peer-to-Peer: Computação colaborativa na Internet", Minicursos, XXII SBRC, Gramado-Rs, Maio 2004.

[9] Kazaa, Web Site, http://www.kazaa.com, último acesso em Janeiro de 2005.

[10] Skype, Web Site, http://www.skype.com, último acesso em Dezembro de 2004.

[11] Milojicic, Dejan S., *et al.*, "Peer-to-Peer Computing", technical report, HPL-2002-57R1, HP Labs, Março de 2002.

[12] Park, J.S. *et al.* "A Middleware Approach for SAINT (Secure, Automatic, Interoperable, and Transparent) Peer-to-Peer Content Services". IEEE International Symposium on Computers and Communication, July 2003, vol.2, pages 1047 - 1052.

[13] Mascolo, C. *et al.* "An XML based Middleware for Peer-to-Peer Computing", First International Conference on Peer-to-Peer Computing, August 2001, Pages:69 - 74.

[14] Gavalda, C. P., *et al.* "Dermi: A New Distributed Hash Table-Based Middleware Framework". IEEE Internet Computing, Vol 8 ,Issue 3,May-Jun 2004 Pages:74 - 84.

[15] Sun M. Inc, Web Site, http://developers.sun.com/techtopics/mobility/midp/ttips/optimize, último acesso em Março de 2005.

[16] Sun M. Inc., J2ME MIDP, Web Site, http://java.sun.com/products/midp/, último acesso em Março de 2005.