Hermes: Um Arcabouço para Programação de Aplicações P2P

Emilio Francesquini, Francisco Reverbel

Departamento de Ciência da Computação
Instituto de Matemática e Estatística da Universidade de São Paulo

{emilio,reverbel}@ime.usp.br

Abstract. *In the early stages of the development of a distributed application, information on its final size or utilization profile is often unknown. At an early point, deciding which overlay network implementation suits the application needs can be a complex task. Hermes provides a simple, yet powerful, environment for the development of P2P applications. It gives the application developer the possibility of choosing the overlay network implementation and replacing such implementation at any point, from development to production, without changes on the application code.*

Resumo. *Geralmente, no início do desenvolvimento de uma aplicação distribuída, tem-se poucas informações sobre o seu tamanho final ou perfil de utilização. A escolha de uma implementação de rede de sobreposição adequada às necessidades da aplicação nessa fase do desenvolvimento pode ser uma tarefa complexa. O Hermes oferece um ambiente simples, porém poderoso, para a criação de aplicações P2P. Ele possibilita a escolha e a troca da implementação de rede de sobreposição em qualquer uma das fases do ciclo de vida da aplicação, desde o desenvolvimento até a produção, sem alterações no código.*

1. Introdução

De uma forma geral é possível classificar as redes de sobreposição atuais como estruturadas ou não estruturadas. Dependendo da aplicação, cada uma delas tem vantagens e desvantagens [Castro et al. 2004]. As redes estruturadas, por exemplo, tendem a ser mais eficientes no que diz respeito ao consumo de banda, enquanto as buscas feitas nas redes não estruturadas tendem a ser mais robustas. Mesmo entre redes do mesmo tipo, existem diferenças significativas na maneira pela qual elas realizam suas tarefas, como por exemplo buscas [Risson and Moors 2004]. No entanto, não é muito fácil distinguir, *a priori*, qual solução é a mais apropriada para a construção de um sistema P2P. Antes da efetiva implantação do sistema, é muito difícil obter dados precisos sobre o perfil de utilização e sobre as proporções da rede. A obtenção de uma estimativa do desempenho real que uma certa solução pode alcançar também não é simples. Muitos modelos de simulação foram propostos mas, em qualquer simulação, é sempre necessário fazer diversas suposições, algumas delas discutíveis, e aproximações para a simplificação do modelo. Por isso, apesar dos simuladores disponíveis estarem cada vez mais completos e robustos, não existe uma maneira segura de avaliar se o comportamento de um sistema feito para executar em milhões de nós será o mesmo tanto no simulador quanto no ambiente real [Gribble 2005].

A escolha de uma solução incorreta pode comprometer o bom funcionamento da aplicação e, dependendo do seu grau de acoplamento com a solução escolhida,

uma mudança em um estágio avançado do projeto pode ser extremamente custosa. Muitos sistemas, em busca de desempenho e simplicidade de implementação, acabam não colocando uma divisão clara entre a aplicação e a implementação da rede de sobreposição. Várias redes sofrem deste problema. Um exemplo muito conhecido é a rede Gnutella. A solução adotada durante o seu desenvolvimento, apesar de ter funcionado perfeitamente enquanto a rede era pequena, é comprovadamente não-escalável [Balakrishnan et al. 2003]. Estima-se que, em junho de 2001, um dos períodos de auge de utilização desta rede, apenas para manter a rede de 50.000 nós conectada e difundir as pesquisas feitas pelos usuários, fossem trafegados 330TB por mês, o que equivalia a 1.7% do tráfego total nos *backbones* dos EUA. Isso se deve predominantemente ao fato da topologia da rede Gnutella não mapear a infra-estrutura da rede física e à utilização de técnicas de inundação para a busca de informações [Ripeanu et al. 2002].

Para evitar problemas como os enfrentados pela rede Gnutella, propomos não escolher a solução *a priori*, mas sim testar diversas soluções *a posteriori* e quiçá alterar a solução adotada com a aplicação já em produção. Para isto, os diversos sistemas P2P, ao invés de acessarem a camada de comunicação diretamente, fariam-no através de uma fachada que contém a definição de uma API geral. Para cada implementação de rede de sobreposição, haveria uma fachada disponível. Assim sendo, a única ação a ser tomada para que o sistema utilize uma solução alternativa é a alteração de um arquivo de configurações que define a implementação a ser utilizada. Este tipo de arquitetura não é inédito [Conde and Francesquini 2002]. Arquiteturas similares aparecem nas interfaces de portabilidade de um ORB Java [Object Management Group 2002], bem como em APIs como JNDI [Sun Microsystems 2007], ODBC (*Open Database Connectivity*), JDBC (*Java Database Connectivity*) e JMS (*Java Message Service*).

O Hermes é uma implementação da proposta acima. Apesar de acreditarmos que essa proposta seja também adequada aos sistemas P2P não puros, ou seja, aqueles que contam com algum tipo de elemento centralizador para efetuar alguma tarefa de rede, aqui nos concentraremos apenas nos sistemas puros, ou totalmente distribuídos (tanto estruturados como não estruturados), pois é neles que a implementação é não trivial.

O restante deste artigo é organizado da seguinte maneira: a seção 2 apresenta o Hermes, a seção 3 compara a presente proposta com outras que aparecem na literatura, a seção 4 discute trabalhos futuros e, finalmente, a seção 5 contém nossas conclusões.

2. O Hermes

O Hermes é um arcabouço para a programação de aplicações P2P. Ele conta com uma API concisa e simples, mas ao mesmo tempo poderosa o suficiente para atender à maioria das necessidades que uma aplicação P2P possa ter. O Hermes foi implementado em Java[1] e possui atualmente mapeamentos para quatro diferentes redes de sobreposição: Bamboo [Bamboo 2007]; FreePastry [FreePastry 2007]; SimpleFlood[2] e JXTA [Gong 2002].

[1] O Hermes pode ser obtido em `http://sourceforge.net/projects/hermesp2p`.

[2] O SimpleFlood é uma implementação de rede de sobreposição criada juntamente com o Hermes. Trata-se de uma rede de inundação simples, que não tem o objetivo de servir de base para aplicações reais. O funcionamento do SimpleFlood é semelhante ao de outras redes não estruturadas, como a Gnutella. Diferentemente desta rede, entretanto, o SimpleFlood não possui quaisquer otimizações, o que o torna ainda menos escalável. A motivação para a criação do SimpleFlood foi demonstrar e testar o funcionamento do Hermes sem a sobrecarga de uma rede mais robusta e completa, como o Bamboo ou o FreePastry.

As duas primeiras são implementações de redes estruturadas, enquanto as duas últimas são de redes não estruturadas. O Hermes permite que o desenvolvedor de uma aplicação troque uma implementação de rede por outra, bastando para isso alterar um arquivo de configuração que define a rede a ser utilizada. Essa troca é totalmente transparente para a aplicação. Assim, as aplicações que utilizarem o Hermes poderão, sem mudanças em seu código, se beneficiar de novas implementações de rede que possam ser criadas e de novas versões das redes já utilizadas. Utilizando o Hermes, o desenvolvedor ganha a liberdade de escolher a qualquer momento, desde o desenvolvimento até a produção, a implementação de rede que melhor se adequa ao seu caso.

Os serviços oferecidos pelo Hermes estão descritos abaixo.

Troca de mensagens O Hermes fornece duas maneiras diferentes para um nó se comunicar com outro. A primeira maneira, mais simples e rápida, não garante a entrega nem a ordem de chegada das mensagens. O seu funcionamento é bem semelhante ao do UDP/IP. Nesta modalidade, o Hermes envia mensagens quase sem nenhuma sobrecarga para controle, funcionando com a política do melhor esforço. A segunda maneira garante a entrega e a ordem de chegada das mensagens. Para isso, ela gera uma sobrecarga, tal como o TCP/IP quando comparado ao UDP/IP.

Publicação e busca A cada recurso ou informação publicada na rede o Hermes associa um ou mais identificadores (chaves únicas), bem como uma descrição textual. Ele provê busca exata por esses identificadores e busca aproximada de texto. Uma busca exata recebe como parâmetro um conjunto de identificadores e localiza os recursos previamente publicados cujos conjuntos de identificadores sejam superconjuntos daquele recebido como parâmetro. Por busca aproximada entende-se a procura por trechos de texto que descrevem um dado recurso. Por exemplo, em uma busca aproximada por “cate”, as palavras “alicate”,“abacateiro” e “catedral” são resultados aceitáveis, mas “alfaiate” e “catinga” não são.

Comunicação em grupo O Hermes disponibiliza um serviço de comunicação em grupo baseado em assinaturas. A criação e a assinatura de um grupo são efetuadas através de uma única operação, que recebe apenas o nome do grupo. O cancelamento da assinatura é igualmente simples. A entrega das mensagens enviadas para o grupo é feita com a política do melhor esforço, o que é relativamente comum nos serviços de comunicação em grupo oferecidos por redes de sobreposição.

Armazenamento distribuído O serviço de armazenamento distribuído do Hermes oferece uma forma fácil para as aplicações salvarem dados na rede com algumas garantias quanto à persistência dos dados salvos. O dado a ser armazenado é fragmentado e espalhado entre alguns dos nós participantes da rede. Dependendo da rede de sobreposição escolhida, a aplicação pode contar com *caching* e replicação automática desses dados. O armazenamento distribuído serve principalmente a dois propósitos: salvamento para posterior recuperação de dados (possivelmente em um nó diferente do nó que efetuou o armazenamento) ou distribuição da carga gerada pelo acesso a dados muito populares.

Alguns dos serviços oferecidos pelo Hermes não estão disponíveis em todas as redes mapeadas. Assim, cada um dos mapeamentos das redes tem a responsabilidade de implementar os serviços não disponíveis na rede mapeada, de modo a oferecer às

aplicações toda a API do Hermes. Para tanto, o mapeamento utiliza apenas os serviços da rede mapeada. A tabela 1 compara os serviços oferecidos pelo Hermes com os serviços oferecidos pelas redes de sobreposição mapeadas.

	SimpleFlood	**JXTA**	**FreePastry**	**Bamboo**	**Hermes**
Troca de mensagens	•	•	•	•	•
Busca exata	•	•	•	•	•
Busca aproximada		•			•
Comunicação em grupo	•	•	•	•	•
Armazenamento distribuído			•	•	•
Autenticação de usuários		•			

Tabela 1. Serviços oferecidos pelo Hermes e pelas quatro redes mapeadas

Para validar a completude dos serviços disponibilizados pelo Hermes, foram desenvolvidas duas aplicações de demonstração: o Hermes Messenger, um IM construído sobre o Hermes, e o TiWarriorsP2P, adaptação de um jogo em rede simples, que utilizava TCP/IP como camada de comunicação e foi modificado para utilizar a API do Hermes. Em cada uma dessas aplicações pode-se, sem nenhuma alteração no código da aplicação, experimentar a troca da rede de sobreposição, escolhida entre as quatro redes mapeadas para o Hermes.

A figura 1 mostra a arquitetura do Hermes, bem como a organização em camadas de uma aplicação que o utiliza. Na camada mais alta fica a aplicação do usuário, a qual se comunica apenas com a camada inferior a ela, o Hermes. Este, por sua vez, emprega um módulo de mapeamento, que é específico para uma certa rede de sobreposição. O módulo de mapeamento converte chamadas à API do Hermes em chamadas à API da rede de sobreposição escolhida. Se esta rede não oferecer suporte a todas as operações

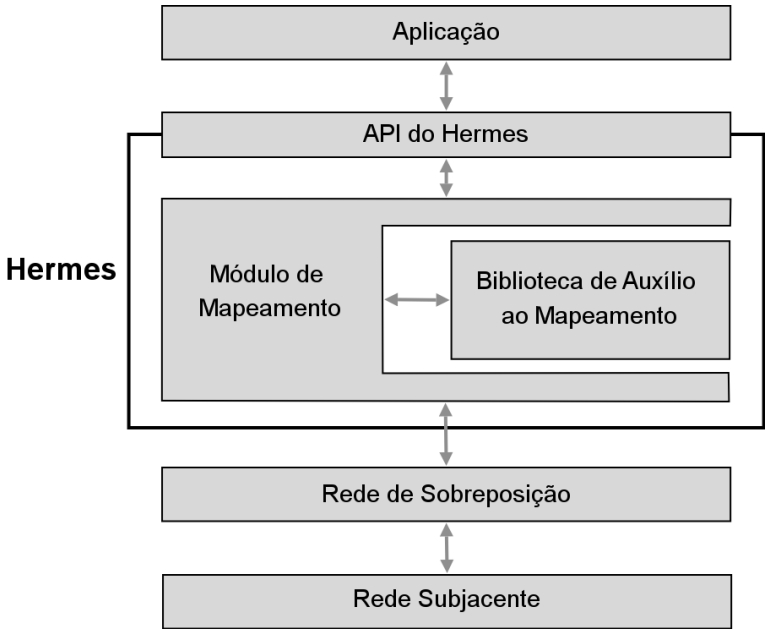

Figura 1. Arquitetura do Hermes

disponíveis na API do Hermes (e.g. busca aproximada), um mapeamento direto não poderá ser efetuado. Nesse caso, o módulo de mapeamento faz uso de uma biblioteca de classes auxiliares, que implementa soluções para os problemas mais comuns encontrados durante o mapeamento de uma nova rede de sobreposição para o Hermes. A biblioteca de auxílio ao mapeamento utiliza os serviços básicos oferecidos por todas as redes de sobreposição para implementar certas funcionalidades não disponíveis em algumas dessas redes. O acesso à rede subjacente (e.g. TCP/IP ou UDP/IP) fica exclusivamente a cargo da rede de sobreposição escolhida.

2.1. A API do Hermes

A API do Hermes se baseia em seis conceitos principais, descritos a seguir. Em Java, esses conceitos são representados por interfaces.

`Network` Representa a rede sobre a qual a aplicação está sendo executada. Aqui ficam definidas as operações necessárias para a conexão e desconexão, para a assinatura e o cancelamento de assinaturas de grupos e para a publicação, retirada e busca de recursos publicados na rede.

`ID` Toda informação que é publicada na rede através da interface `Network` deve ter ao menos um identificador. Em certas aplicações, é comum que um recurso possua mais de um identificador. Cada um destes identificadores é representado por esta interface e pode ser criado utilizando-se a interface `IDFactory`. Na maior parte das vezes esse identificador acaba sendo, na verdade, um vetor de bytes gerado por uma função de *hash* segura. A função desta interface é esconder esse tipo de detalhe do desenvolvedor da aplicação, de modo que ele veja estes `ID`s como objetos opacos.

`ResourceDescriptor` Todas as publicações e resultados de busca na rede feitas através das operações definidas pela interface `Network` se utilizam desta interface. Um `ResourceDescriptor` traz consigo informações sobre um recurso. Essas informações incluem os `ID`s do recurso, seu tipo e um texto com uma breve descrição.

`Node` Cada um dos participantes da rede é representado por um objeto `Node`. É através desses objetos que se efetuam as operações de troca de mensagens e de requisição de recursos. A obtenção de objetos deste tipo pode ser feita através de uma busca, do recebimento de uma mensagem, ou através da obtenção do nó local através da interface `Network`.

`Group` Cada objeto deste tipo representa uma assinatura, cancelada ou não, de um grupo. Um objeto `Group` pode ser utilizado para enviar mensagens aos assinantes do grupo e efetuar o cancelamento da assinatura.

`ApplicationDataReceiver` Esta é uma interface *callback* para comunicação entre o Hermes e a aplicação do usuário. Quando a aplicação se conecta à rede, ela deve fornecer ao Hermes um objeto deste tipo. Esse objeto é notificado toda vez que uma mensagem ou requisição para o nó local for recebida.

As aplicações construídas com o Hermes têm três estágios: inicialização, execução e finalização. A inicialização e finalização são estágios passageiros, enquanto a execução é um estágio mais duradouro. Durante a inicialização são carregados arquivos de configuração que especificam a rede de sobreposição a ser utilizada e definem os

parâmetros configuráveis dessa rede[3]. Em seguida, a rede é inicializada e a conexão com a rede é efetuada. Passado o estágio de inicialização, entra-se no estágio de execução. Nesse estágio é possível efetuar a publicação de recursos na rede, fazer buscas e trocar mensagens com os outros nós. É nele que a aplicação faz o seu trabalho. Quando a aplicação, por qualquer motivo, decidir que é hora de finalizar ou se desconectar, ela entra no estágio de finalização. Algumas redes provêem mecanismos para desconexão voluntária; o Hermes disponibiliza esses mecanismos para a aplicação.

2.2. Mapeamentos para as redes de sobreposição

O mapeamento de uma implementação de rede de sobreposição para o Hermes consiste na criação de classes que implementam os conceitos básicos apresentados na seção anterior. Essas classes são as responsáveis por efetuar a tradução das APIs disponibilizadas pelo Hermes para as APIs disponibilizadas pela rede em questão. Os conceitos mais importantes a serem levados em consideração durante o mapeamento são aqueles representados pelas interfaces `ID`, `Network`, `Node`, `ResourceDescriptor` e `Group`.

O mapeamento do objeto `ID` se faz necessário pois o espaço de `ID`s (além dos algoritmos utilizados para a sua construção) de cada uma das implementações é diferente. Um mapeamento, através da API do Hermes, permite que a aplicação gere e utilize `ID`s de modo independente da rede de sobreposição. Essa parte do mapeamento é geralmente direta, pois grande parte das redes já dispõem de classes para a construção desse tipo de objeto.

A criação de uma implementação para a interface `Network` já não é tão direta. Nesta classe devem estar contidas as regras de inicialização, execução e finalização para a rede sendo mapeada. Algumas das operações presentes nesta interface, como a obtenção de uma referência para um nó local e a busca exata, são normalmente de implementação trivial, enquanto outras, como a publicação e a retirada de publicações, não raramente demandam esforço do implementador para manter a semântica definida pela API do Hermes. Um exemplo claro disto é a implementação da retirada de publicações. Em algumas das redes de sobreposição disponíveis, nelas incluídas o Bamboo e o JXTA, não existe uma interface para a retirada de algo que foi publicado[4]. Esse tratamento tem que ser feito totalmente pelo mapeamento.

A implementação da interface `Node` merece mais atenção. Ela é, na verdade, um dos pontos centrais da API – juntamente com a interface `Network`. Naquela interface estão contidos os métodos que possibilitam a troca de mensagens entre os nós da rede. Muitas redes de sobreposição possuem limites para o tamanho das mensagens enviadas e é papel da implementação do `Node` fazer o tratamento da fragmentação dos dados para o envio de forma transparente à aplicação. Também faz parte de das tarefas do `Node` garantir a entrega, a ordem e consistência dos dados enviados.

[3]O Hermes já vem com configurações padrão para as quatro redes de sobreposição mapeadas, evitando assim que o desenvolvedor da aplicação tenha que lidar com as idiossincrasias de cada uma dessas redes. Entretanto, ele dá ao desenvolvedor a possibilidade de alterar tais configurações caso necessário.

[4]Apesar de tanto o Bamboo quanto o JXTA não possuirem interfaces explícitas para a retirada de um par chave-valor publicado em suas redes, ambos têm outros mecanismos para lidar com este problema. Essencialmente, para cada publicação é especificado um intervalo de tempo durante o qual a publicação é válida. Através da utilização controlada deste mecanismo, é possível implementar a operação de retirada de publicação simplesmente fazendo com que esta não seja renovada automaticamente.

A comunicação em grupo funciona de maneira diferente em cada uma das redes de sobreposição. Algumas exigem que um nó efetue a criação do grupo antes de assiná-lo, enquanto em outras a simples assinatura implica a criação do grupo. É parte das responsabilidades do implementador de um mapeamento para o Hermes esconder da aplicação esse tipo de peculiaridade através da criação de uma implementação da interface `Group`. Aqui devem ficar encapsulados também todos os trâmites necessários para se efetuar o envio de uma mensagem para os assinantes de um grupo.

Freqüentemente, a criação de classes auxiliares se faz necessária para que a implementação do mapeamento seja completa. Para facilitar o trabalho do implementador de um novo módulo de mapeamento, o Hermes fornece uma biblioteca com algumas destas classes auxiliares, evitando assim que o implementador gaste seu tempo em tarefas já resolvidas. Nesta biblioteca estão disponíveis classes para auxílio à fragmentação e ao envio de mensagens longas, para empacotamento e seriação de objetos, para cálculo e verificação de filtros Bloom [Bloom 1970], para segmentação e consolidação de dados utilizando árvores Merkle [Merkle 1988] e para implementação de buscas aproximadas em redes não estruturadas. Essa implementação utiliza um algoritmo de busca aproximada baseado em filtros Bloom, em n-gramas e em busca exata de pares chave-valor. Em [Francesquini 2007] é apresentada uma descrição completa do algoritmo de busca aproximada fornecido por essa biblioteca, bem como detalhes sobre a implementação dos mapeamentos para as várias redes de sobreposição.

Abaixo descrevemos brevemente como os serviços disponíveis no Hermes foram mapeados para as seguintes redes: SimpleFlood, JXTA, FreePastry e Bamboo.

Troca de mensagens No caso de mensagens curtas a implementação foi direta. Todas as redes dispunham de um serviço de troca de mensagens curtas. As classes de auxílio à fragmentação e ao envio de mensagens longas, fornecidas pela biblioteca de classes auxiliares do Hermes, foram utilizadas para tratar o envio deste tipo de mensagem nos quatro mapeamentos.

Publicação e busca Exceto pelo caso do FreePastry, a publicação e busca exata de recursos foi feita utilizando-se diretamente os mecanismos oferecidos pelas redes de sobreposição. A razão do FreePastry não poder ser mapeado diretamente se deve ao fato da semântica das APIs do Hermes permitir a publicação na rede de recursos que, apesar de diferentes, têm o mesmo identificador. O SimpleFlood, o JXTA e o Bamboo são capazes de lidar corretamente com essas colisões. Já a DHT oferecida pelo FreePastry não permite a utilização do mesmo identificador para duas publicações distintas. Em outras palavras, a DHT do FreePastry não possui um tratamento para colisões de chaves nas operações de inserção. Cada chave deve apontar para um único valor. Uma alternativa, então, seria criar um objeto lista contendo os objetos com a mesma chave e publicar a lista no lugar dos objetos propriamente ditos. A cada vez que se fizesse uma publicação, bastaria verificar se já existe um objeto com a mesma chave publicada, e caso existisse, adicionar o objeto na lista do objeto já publicado. Essa abordagem, entretanto, traz sérios problemas de acesso concorrente ao objeto lista. O *caching* e a replicação automática das publicações feitas na DHT também criam alguns obstáculos uma vez que a busca, a publicação e a retirada de objetos da DHT podem alcançar nós diferentes da rede e, portanto, podem acabar fazendo acesso a versões diferentes da lista. Ao invés de tentar contornar esses problemas, uma solução escalável mais simples foi implementada como alternativa ao

uso da DHT do FreePastry. A implementação utiliza o serviço de comunicação em grupo oferecido pelo FreePastry, o Scribe [Castro et al. 2002], para criar uma rede semântica de sobreposição [Crespo and Garcia-Molina 2004]. Assim como uma DHT, o Scribe se baseia na topologia da rede para efetuar a manutenção dos grupos criados na rede.

Dentre as redes de sobreposição mapeadas para o Hermes, somente o JXTA oferece busca exata por um conjunto de chaves. As demais redes oferecem apenas busca por uma só chave. Nos mapeamentos para essas redes, a toda publicação feita através do Hermes é associado um filtro Bloom calculado a partir dos identificadores do recurso. Para cada busca exata é criado e enviado junto com a requisição de busca um outro filtro Bloom, calculado a partir do conjunto de identificadores recebido como parâmetro para a busca. Cada um dos nós atingidos pela busca (que são os possíveis detentores de algum recurso desejado) avalia quais publicações mantidas por ele possuem filtros Bloom que contenham o filtro recebido com a requisição de busca. Por evitar falsos positivos nas buscas exatas por múltiplos identificadores, a utilização de filtros Bloom diminui significativamente o tráfego de rede e traz ganhos de desempenho.

Das redes de sobreposição mapeadas, apenas o JXTA trazia consigo o serviço de busca aproximada. Nela a implementação é trivial. Nas demais redes a busca aproximada foi feita através do algoritmo de busca aproximada existente na biblioteca de classes auxiliares do Hermes.

Comunicação em grupo O serviço de comunicação em grupo teve nessas quatro redes uma implementação direta, já que todas elas dispunham desse tipo de serviço.

Armazenamento distribuído Tanto o Bamboo quanto o FreePastry oferecem um serviço de armazenamento distribuído de dados mas impõem limites aos tamanhos dos dados armazenados. Como a API do Hermes não tem esse tipo de restrição, os módulos de mapeamento para essas redes usam árvores Merkle para implementar a fragmentação dos dados e o seu armazenamento em segmentos. Já o SimpleFlood e o JXTA não oferecem um serviço de armazenamento distribuído. Os mapeamentos dessas redes tiveram de implementar tal serviço sobre os serviços disponíveis. Nesses mapeamentos, quando um usuário requisita o armazenamento de algum dado na rede, o dado é salvo localmente e uma publicação é feita. Qualquer nó que requisitar esse dado faz uma busca na rede e, caso encontre a publicação, obtém uma cópia do dado desejado. Caso o nó que fez a busca seja um nó diferente do nó que fez o primeiro armazenamento, este nó também armazena localmente uma cópia do dado e em seguida o publica na rede. Isso faz com que dados muito requisitados sejam automaticamente distribuídos pela rede. Entretanto, no caso de dados não muito requisitados, essa técnica de replicação passiva tem algumas limitações. Se o dado publicado for uma informação pessoal, por exemplo, apenas o usuário do nó que armazenou a informação irá acessá-la. Caso ele faça isso de outro nó enquanto o nó onde foi feita a publicação estiver desconectado, o dado não será encontrado. Alguns esquemas de replicação e *caching* ativos para redes não estruturadas [Lv et al. 2002, Clarke et al. 2002] foram propostos como soluções para este problema.

3. Trabalhos Relacionados

Em [Dabek et al. 2003] é proposta a criação de uma API comum a todos os sistemas P2P baseados em redes de sobreposição estruturadas. Esse trabalho trata da criação de uma API comum tanto para o envio, roteamento e entrega de mensagens quanto para busca e

comunicação em grupo (*multicast*). Para a definição de tal API é utilizada uma linguagem de programação fictícia e facilmente mapeável a qualquer linguagem de programação real. Essa proposta, por ser dirigida exclusivamente às redes distribuídas estruturadas, diferencia-se do Hermes por deixar claro ao desenvolvedor da aplicação o tipo de rede utilizada. Por isso, a troca por outro tipo de rede não pode ser feita de forma direta. Além disso, a utilização de uma linguagem fictícia para a especificação da API, sem a definição de regras de mapeamento para uma linguagem real, dificulta que duas implementações, ainda que na mesma linguagem de programação, forneçam uma API comum.

Em [Ciaccio 2005] uma API mais flexível, genérica, simples e concisa do que a proposta por [Dabek et al. 2003] é apresentada. A proposta de Ciaccio tenta levar em consideração os diferentes tipos de rede P2P disponíveis. Esse trabalho parece uma lapidação da proposta feita por Dabek et alii: Ciaccio retira e simplifica diversas características (como o tratamento do encaminhamento de pacotes) e adiciona outras (como o conceito de portas de comunicação). O trabalho não especifica o tipo de rede. Nesse sentido, é bem próximo à API do Hermes: limita-se a especificar o envio, roteamento e entrega de mensagens. Para fazê-lo, utiliza uma linguagem fictícia, similar à utilizada por Dabek et al., para definição da API. As regras para o mapeamento dessa linguagem para uma linguagem real não foram definidas e, portanto, nesse aspecto a proposta de Ciaccio tem as mesmas limitações da proposta anterior.

O projeto cP2Pc [Kuz and van Steen 2003] é uma abordagem que visa a criação de um *ÜberClient*, conforme proposto em [Wiley 2001]. Com esse objetivo, os projetistas do cP2Pc criaram uma API comum de programação para a criação de aplicações de compartilhamento de arquivos utilizando uma linguagem real de programação, C. Este projeto conta com mapeamentos para as redes Gnutella e GDN. Ele se diferencia do Hermes por definir uma API voltada apenas para a criação de aplicações de compartilhamento de arquivos. Dessa forma, a API do cP2PC é limitada e não tem, por exemplo, suporte à comunicação em grupo ou à troca de mensagens entre os nós participantes da rede. Além disto, ela incentiva a aplicação a se conectar de forma simultânea em mais de uma rede, de modo a permitir o compartilhamento de arquivos entre as diferentes redes. Isso mostra uma distinção bem clara entre as intenções dos criadores do cP2Pc e o objetivo do Hermes, que é possibilitar a troca da rede de sobreposição utilizada sem alterações no código da aplicação.

Assim como o Hermes, o OMNIX [Kurmanowytsch et al. 2003] é um arcabouço para a criação de aplicações P2P que é independente da topologia da rede. Ele define, em Java, uma API geral que permite que a topologia de rede seja trocada sem alterações no código da aplicação. Diferentemente do Hermes, o OMNIX não é uma camada que envolve as implementações de rede já existentes, mas sim uma arquitetura extensível. Nessa arquitetura, novos descritores da topologia da rede podem ser acoplados para que as necessidades da aplicação sejam melhor atendidas. O OMNIX trata, inclusive, da comunicação. Para isso ele faz acesso à camada subjacente de rede, o que no caso do Hermes fica a cargo da rede de sobreposição escolhida.

4. Trabalhos futuros

Devem ser realizados experimentos para avaliar a sobrecarga que o Hermes adiciona às aplicações P2P que o utilizam. Se, nos vários mapeamentos, essa sobrecarga for suficien-

temente pequena, então o Hermes poderá facilitar avaliações comparativas dos desempenhos das redes de sobreposição mapeadas, pois ele permite que uma mesma aplicação de testes seja executada nas diferentes redes.

Apesar das diferentes implementações de rede fornecerem, na sua maioria, apenas a funcionalidade de busca exata de chaves ou busca aproximada de cadeias de caracteres, por vezes as aplicações P2P podem requerer buscas mais elaboradas do que essas. Buscas na rede por faixas de valores, ou aproximadas através da utilização de vetores de características, são exemplos desse tipo de requisito. Enquanto tais buscas são facilmente implementáveis em redes não estruturadas, sua implementação em redes estruturadas é mais trabalhosa [Liu and Ryu 2006, Zhou et al. 2003, Bhattacharjee et al. 2003, Bauer et al. 2004]. Como a proposta do Hermes é facilitar o trabalho do implementador de uma aplicação P2P e isolá-lo da rede de sobreposição escolhida, o oferecimento de uma API para consultas avançadas se faz necessário.

As mensagens trocadas pelo Hermes não tem nenhum tipo de prioridade associada a elas. Com a adição de prioridades para o controle do envio de mensagens, tornar-se-ia possível o oferecimento de QoS. Isso possibilitaria a criação de canais contínuos de comunicação, como os que seriam necessários para a implementação de uma aplicação de telefonia, por exemplo.

O serviço de armazenamento distribuído do Hermes não é eficiente para a replicação de dados nas redes não estruturadas. O esquema passivo de replicação atualmente implementado não é apropriado para o armazenamento e posterior descarregamento de dados que são de baixa popularidade. Um esquema de replicação ativo para a melhoria da eficiência desse serviço está sendo implementado.

Apesar da construção de um sistema P2P totalmente seguro estar fora do escopo do Hermes, alguns serviços básicos de segurança seriam úteis se disponíveis. Um serviço de envio de mensagens assinadas e criptografadas pressupõe a existência de um serviço de autenticação de usuários. Ambos seriam serviços cuja importância se destacaria. Sem uma entidade certificadora, os nós precisariam confiar uns nos outros para efetuarem tarefas de autenticação. A figura de um nó certificador poderia surgir na rede através de um esquema de controle de reputação, como o proposto por [Kamvar et al. 2003]. A possibilidade de implementar um serviço básico de segurança nesses moldes está sendo estudada.

5. Conclusão

O Hermes oferece uma API definida numa linguagem de programação real (Java) e faz da implementação da rede de sobreposição uma parte das aplicações P2P que é facilmente substituível. Para usar uma certa rede de sobreposição através do Hermes é necessário um módulo de mapeamento para essa rede. Lembrando a similaridade entre o Hermes e camadas de portabilidade como ODBC (ou JDBC), JNDI e JMS, vemos que a função do módulo de mapeamento é similar à de um *driver* ODBC (ou JDBC), ou à de um *JNDI provider*, ou ainda à de um *JMS provider*. A existência de módulos de mapeamento para quatro redes de sobreposição distintas, a saber, Bamboo, FreePastry, JXTA e SimpleFlood, é uma forte indicação de que API do Hermes é mapeável para todas as redes de sobreposição usuais, sejam elas estruturadas ou não estruturadas. Até onde pudemos averiguar, esta é a única proposta com essas características.

Algumas redes de sobreposição não fornecem todos os serviços que são oferecidos pela API do Hermes. Por isso, nos mapeamentos dessas redes é necessário um passo além da simples tradução de chamadas à API do Hermes em chamadas à API da rede de sobreposição escolhida. Esse passo adicional seria um trabalho extra com o qual o implementador de uma aplicação P2P teria que se preocupar, caso sua aplicação precisasse desse serviço e fosse utilizar diretamente a API da rede de sobreposição. O Hermes elimina esse problema. Por isolar o desenvolvedor de aplicações P2P da rede de sobreposição, o Hermes remove do desenvolvedor a preocupação com certos serviços básicos e permite que ele se concentre na resolução dos problemas inerentes às aplicações.

A API do Hermes oferece todos os serviços das redes de sobreposição usuais, exceto funções de segurança (disponíveis apenas no JXTA), que devem ser tratadas externamente ao arcabouço. Sendo tão completa quanto as APIs das redes mapeadas, ela dá ao desenvolvedor de aplicações P2P tanto poder quanto a utilização direta das APIs dessas redes. O uso do Hermes, no entanto, tem a vantagem de evitar o acoplamento direto entre a aplicação P2P e a rede de sobreposição. Dessa forma, o Hermes dá ao criador de uma aplicação P2P a liberdade de adiar a decisão sobre qual a rede de sobreposição mais apropriada para a sua aplicação. Além de permitir que tal decisão seja tomada apenas no momento da implantação da aplicação, o Hermes permite que essa decisão seja revista com a aplicação já em produção, sem a necessidade de se alterar uma linha de código sequer.

Nos sistemas P2P atuais é relativamente comum a utilização de n-gramas, para a busca de padrões de cadeias de caracteres, de filtros Bloom, para filtrar conjuntos de dados, e de árvores Merkle, para a distribuição de um conjunto de dados. Entretanto, a implementação dos mapeamentos das redes de sobreposição exigiram a utilização dessas técnicas em conjunto e de forma distribuída, o que não é algo comum nos sistemas de hoje. Acreditamos ser essa uma das contribuições deste trabalho.

Referências

Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I. (2003). Looking up data in P2P systems. *Communications of the ACM*, 46(2):43–48.

Bamboo (2007). The Bamboo Distributed Hash Table. `http://bamboo-dht.org/`. Último acesso em abril de 2007.

Bauer, D. et al. (2004). Bringing efficient advanced queries to distributed hash tables. In *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, pages 6–14, Washington, DC. IEEE Comp. Society.

Bhattacharjee, B. et al. (2003). Efficient peer-to-peer searches using result-caching. In *The 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*.

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426.

Castro, M., Costa, M., and Rowstron, A. (2004). Peer-to-peer overlays: structured, unstructured, or both? Technical Report MSR-TR-2004-73, Microsoft Research.

Castro, M. et al. (2002). Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communication (JSAC)*, 20(8).

Ciaccio, G. (2005). A pretty flexible API for generic peer-to-peer programming. Technical Report DISI-TR-05-06, DISI, Università di Genova.

Clarke, I., Miller, S. G., Hong, T. W., Sandberg, O., and Wiley, B. (2002). Protecting free expression online with freenet. *IEEE Internet Computing*, 6(1):40–49.

Conde, D. and Francesquini, E. (2002). 4-ação. PLoP-IME, IME-USP, São Paulo, SP.

Crespo, A. and Garcia-Molina, H. (2004). Semantic overlay networks for p2p systems. In Moro, G. et al., editors, *AP2PC*, volume 3601 of *LNCS*, pages 1–13. Springer.

Dabek, F. et al. (2003). Towards a common API for structured peer-to-peer overlays. In *Proc. of the 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS03)*, Berkeley, CA.

Francesquini, E. (2007). Hermes: um arcabouço para a programação de aplicações P2P. Master's thesis, Instituto de Matemática e Estatística da USP, São Paulo, SP.

FreePastry (2007). Pastry - a substrate for peer-to-peer applications. `http://freepastry.org/`. Último acesso em abril de 2007.

Gong, L. (2002). Project JXTA: A technology overview. `http://www.jxta.org/project/www/docs/jxtaview_01nov02.pdf`.

Gribble, S. (2005). Public review for "Design choices for content distribution in P2P networks", by A. Al Hamra and P. A. Felber. *ACM SIGCOMM Comput. Commun. Rev.*, 35(5):29–40.

Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. (2003). The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proc. of the 12th Int. Conf. on World Wide Web*, pages 640–651, New York, NY. ACM Press.

Kurmanowytsch, R., Kirda, E., Kerer, C., and Dustdar, S. (2003). OMNIX: A topology-independent P2P middleware. In *15th Conference on Advanced Information Systems Engineering (CAiSE '03), Workshops Proceedings*, Klagenfurt/Velden, Austria.

Kuz, I. and van Steen, M. (2003). cP2Pc: Integrating P2P networks. *Linux Journal*, 110.

Liu, L. and Ryu, K. (2006). RHT: Supporting range queries in DHT-based P2P systems. In *Proceedings of the 2006 IASTED Conference on Parallel and Distributed Computing and Systems*. Acta Press.

Lv, Q. et al. (2002). Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proc. of the 16th Int. Conf. on Supercomputing*, pages 84–95. ACM Press.

Merkle, R. C. (1988). A digital signature based on a conventional encryption function. In *CRYPTO '87: A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, pages 369–378, London, UK. Springer-Verlag.

Object Management Group (2002). IDL to Java Language Mapping Specification. Version 1.2, formal/02-08-05.

Ripeanu, M., Foster, I., and Iamnitchi, A. (2002). Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1).

Risson, J. and Moors, T. (2004). Survey of research towards robust peer-to-peer networks: search methods. Technical Report UNSW-EE-P2P-1-1, University of South Wales.

Sun Microsystems (2007). Java Naming and Directory Interface (JNDI). `http://java.sun.com/products/jndi/`. Último acesso em abril de 2007.

Wiley, B. (2001). Interoperability through gateways. In Oram, A., editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc.

Zhou, F. et al. (2003). Approximate object location and spam filtering on peer-to-peer systems. In *Middleware 2003 — ACM/IFIP/USENIX International Middleware Conference*, volume 2672 of *LNCS*, pages 1–20. Springer-Verlag.