Avaliação de Mecanismos Avançados de Recuperação de Conteúdo em Sistemas P2P

Fabrício Benevenuto, José Ismael Júnior, Jussara Almeida

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627, Belo Horizonte, MG, 31270-901

{fabricio, ismaeljr, jussara}@dcc.ufmg.br

Abstract. *This paper provides a performance evaluation of four mechanisms applied in current unstructured P2P systems, quantifying the main benefits of each feature over the previous community-based message flooding protocol. Our main results show that a hierarchical super-peer architecture and a content-aware query routing mechanism are each responsible for significant reductions on system load with no impact on the number of successful queries. Furthermore, user-controlled query retransmission gives users the flexibility to trade higher query success rates and shorter download times for longer latency. Finally, the benefit of downloading a file from multiple sources can be limited if they are not carefully selected.*

Resumo. *Este artigo apresenta uma avaliação de desempenho de quatro mecanismos utilizados nas atuais arquiteturas P2P não estruturadas, quantificando os principais benefícios de cada mecanismo sobre um protocolo previamente proposto baseado no uso de comunidades e inundação de mensagens. Os principais resultados mostram que a arquitetura hierárquica de super-nós e o mecanismo de roteamento de consultas são responsáveis por uma grande redução na carga do sistema, sem impacto no número de consultas respondidas com sucesso. Além disso, a retransmissão de consultas controlada pelo usuário permite um compromisso entre altas taxas de consultas bem-sucedidas e longa latência. Finalmente, o benefício de transferir um arquivo de múltiplas fontes pode ser limitado se elas não forem criteriosamente selecionadas.*

1. Introdução

Sistemas Par-a-Par, ou simplesmente P2P, para compartilhamento de arquivos têm crescido em popularidade e, como conseqüência, têm evoluído em complexidade a fim de prover uma melhor qualidade de serviço para seus milhões de usuários. Como exemplos de sistemas de primeira geração podemos citar o Napster [Napster,], centralizado, e o sistema Gnutella original [Gnutella,], totalmente descentralizado.

O sistema Gnutella original, em particular, é baseado na inundação controlada de mensagens pela rede para localização de conteúdo. Vários trabalhos propuseram otimizações para mitigar o problema de escalabilidade, conseqüente desta inundação [Lv

et al., 2002, Liu et al., 2004, Chawathe et al., 2003, Barbosa et al., 2004]. Apesar de tanto esforço, o sistema Gnutella original tem decaído em popularidade, motivando o lançamento de uma nova versão do protocolo, totalmente diferente.

A nova versão do protocolo Gnutella (versão 2.0) é baseada em uma arquitetura hierárquica semi-descentralizada e emprega vários mecanismos avançados também utilizados em outros sistemas P2P atuais como o KaZaA [KaZaA,]. Estes sistemas exploram a heterogeneidade previamente observada entre os nós [Saroiu et al., 2002]. Diferentemente do Gnutella original, a nova versão Gnutella assinala responsabilidades especiais aos nós com mais recursos e/ou que permanecem ativos no sistema por períodos mais longos. Estes nós, chamados super-nós formam um backbone central. Os demais nós, chamados folhas, se conectam ao sistema através de um ou mais super-nós.

As arquiteturas P2P baseadas em super-nós empregam vários mecanismos avançados para otimização da recuperação de conteúdo. No sistema Gnutella 2.0, por exemplo, cuja especificação é de acesso público [Gnutella2,], a inundação de mensagens na rede é evitada, pois um super-nó repassa uma mensagem de consulta apenas para seus super-nós vizinhos. Além disto, um mecanismo de roteamento otimizado utiliza tabelas mantidas por cada super-nó com os arquivos armazenados em suas folhas e em seus vizinhos para evitar repassar consultas para nós onde o arquivo requisitado não está armazenado. Os sistemas atuais também permitem ao usuário reiniciar uma consulta a partir de super-nós que não foram contactados nas buscas anteriores, aumentando assim a chance de encontrar o arquivo na rede. Finalmente, estes sistemas também permitem a transferência de um arquivo de múltiplas fontes, isto é, a transferência de segmentos diferentes do arquivo a partir nós diferentes.

Apesar de intuitivos, os ganhos de desempenho propiciados pelos mecanismos empregados nos sistemas P2P atuais ainda não foram quantificados. Este trabalho visa preencher esta lacuna. Através de simulação, ele avalia quantitativamente os benefícios de cada um dos mecanismos acima citados, isto é, da arquitetura hierárquica de super-nós e folhas, do roteamento otimizado de consultas baseado no conhecimento do conteúdo armazenado nos nós, da retransmissão de consultas controlada por usuários e da transferência de arquivos de múltiplas fontes.

Podemos citar duas principais contribuições deste artigo. A primeira é o projeto e implementação de um simulador de sistema P2P baseado em super-nós que emprega mecanismos utilizados em sistemas populares atuais além de características de carga e de nós realistas. Não estamos cientes de nenhuma outra ferramenta semelhante disponível. A segunda grande contribuição é uma extensa avaliação dos ganhos quantitativos de desempenho providos pelos mecanismos empregados em arquiteturas P2P populares.

Nossos resultados mostram que a arquitetura de super-nós por si só reduz a carga no sistema em aproximadamente 95% sem reduzir a taxa de consultas bem sucedidas. Uma redução maior na carga pode ser atingida com o roteamento otimizado de consultas. A retransmissão de consultas provê ao usuário a flexibilidade de um compromisso entre latência e taxa de sucesso. Duas retransmissões são suficientes para atingir uma taxa de sucesso de 98%, reduzir o tempo médio de transferência em até 40% para arquivos pequenos, mantendo a carga no sistema e a latência em níveis competitivos. Finalmente, a transferência de múltiplas fontes pode na verdade degradar o tempo de transferência,

se os nós fontes não forem cuidadosamente escolhidos. Esperamos que estes resultados possam guiar o projeto de sistemas P2P mais escaláveis no futuro.

O restante deste artigo está organizado como a seguir. Os trabalhos relacionados são discutidos na seção 2. A seção 3 apresenta brevemente as arquiteturas P2P avaliadas neste artigo. A metodologia de avaliação é discutida na seção 4 e os principais resultados são apresentados na seção 5. A seção 6 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Sistemas P2P para compartilhamento de arquivos podem ser categorizados em três grupos, com base no mecanismo de localização de conteúdo. O primeiro deles, utilizado no Napster [Napster,], é baseado em um servidor central com a localização de todos os arquivos compartilhados no sistema. Em sistemas descentralizados estruturados, tais como Chord [Stoica et al., 2001], Pastry [Castro et al., 2002], e CAN [Ratnasamy et al., 2001], os índices dos arquivos são armazenados em todos os nós participantes, que são organizados em uma estrutura bem definida utilizada para roteamento de consultas. O terceiro grupo consiste de sistemas descentralizados não estruturados, tais como Gnutella [Gnutella, ,Gnutella2,] e KaZaA [KaZaA,]. Em particular, o protocolo original do Gnutella (versão 0.4) utiliza inundação de mensagens na rede.

Vários estudos analisaram o desempenho de sistemas P2P para compartilhamento de arquivos. Em [Figueiredo et al., 2003], os autores propõem modelos analíticos para explorar os principais compromissos de desempenho nas três arquiteturas P2P básicas. De forma similar, vários trabalhos tiveram seu foco em melhorar a recuperação de conteúdo no protocolo original do Gnutella, baseado em inundação de mensagens [Lv et al., 2002, Liu et al., 2004, Chawathe et al., 2003, Barbosa et al., 2004]. Em [Barbosa et al., 2004], o uso de comunidades de pares que compartilham interesses mostrou ser bastante eficaz, contribuindo para uma melhora significativa do desempenho do protocolo original e sendo superior a outras otimizações previamente propostas.

Uma grande limitação dos trabalhos anteriores é que eles não abordam vários aspectos e mecanismos importantes implementados em sistemas mais recentes, como o KaZaA [KaZaA,] e a versão mais recente do Gnutella [Gnutella2,]. Estes sistemas possuem uma arquitetura hierárquica, onde os nós com mais recursos (banda, processamento), chamados *super-nós* compõem um backbone central, otimizando a localização de conteúdo.

Poucos trabalhos anteriores trataram de sistemas baseados em super-nós. Em [Yang and Garcia-Molina, 2001], os autores comparam o desempenho de sistemas com super-nós com diferentes abordagens de replicação e organização. Em [Yang and Garcia-Molina, 2003], são propostas estratégias de construção da topologia com super-nós. Estes estudos enfatizam a rede lógica (*overlay*) criada por arquiteturas P2P. O foco do nosso trabalho não está apenas na hierarquia dos nós em sistemas com super-nós, mas também nos mecanismos implementados por estes sistemas.

3. Arquiteturas P2P Não Estruturadas para Compartilhamento de Arquivos

Esta seção apresenta um resumo do funcionamento das duas arquiteturas de P2P para avaliadas neste artigo. A seção 3.1 descreve os principais aspectos da arquitetura Gnutella original, baseada em inundação de mensagens. A seção 3.2 descreve a arquitetura com super-nós e seus principais mecanismos, enfatizando a implementação do protocolo Gnutella 2.0, cuja especificação é de acesso público [Gnutella2,].

3.1. Arquitetura Baseada em Inundação de Mensagens

Os protocolos P2P, tais como o Gnutella original (versão 0.4), são arquiteturas descentralizadas, onde todos os nós, agindo tanto como clientes quanto como servidores, possuem as mesmas responsabilidades na construção da rede lógica e interagem através de mensagens. Há três tipos diferentes de mensagens. As mensagens de *ping* e *pong* são trocadas periodicamente entre nós vizinhos para checar a permanência dos mesmos no sistema. As mensagens de *query* e *query hit* são enviadas como parte do mecanismo de localização de conteúdo, descrito a seguir. As mensagens de *get* e *push* são usadas para a transferência de arquivos entre dois nós, após localização.

Para localizar um arquivo, um nó (origem) envia uma *query* para todos os seus nós vizinhos. Quando um nó recebe uma *query*, ele verifica se ele possui o arquivo procurado. Se possui, este nó envia uma mensagem de *query hit* como resposta para o nó origem. Independente do arquivo ser ou não encontrado, a *query* é propagada pela rede, dos vizinhos para seus vizinhos, por um determinado período (*time-to-live*) pré-definido. Este mecanismo não é escalável devido à sobrecarga imposta em cada nó para o processamento de uma quantidade excessiva de mensagens.

Em [Barbosa et al., 2004], os autores propuseram e avaliaram dois mecanismos que exploram a localidade de interesses entre os nós para melhorar a escalabilidade do protocolo Gnutella. Nós que compartilham interesses, avaliados através do número de arquivos em comum, formam uma *comunidade*. Para localizar um arquivo, um nó procura primeiro nos nós de sua comunidade, executando o mecanismo de inundação apenas se o arquivo não for encontrado em sua comunidade. Os resultados em [Barbosa et al., 2004] mostram que os mecanismos propostos garantem uma melhoria significativa de desempenho sobre o protocolo original bem como outras otimizações.

Dos mecanismos propostos em [Barbosa et al., 2004] o algoritmo que obteve os melhores resultados é chamado de *algoritmo estendido*. Este algoritmo cria uma comunidade entre os pares que compartilham o maior número de arquivos. Como base de comparação na avaliação apresentada neste trabalho usaremos este algoritmo que será referenciado simplesmente como mecanismo de inundação baseado em comunidades.

3.2. Arquitetura Hierárquica Baseada em Super-Nós

Em sistemas P2P mais recentes, a rede forma uma arquitetura hierárquica onde os nós são classificados como folhas ou super-nós. Folhas, que são tipicamente nós com recursos limitados e disponibilidade baixa ou imprevisível, não possuem nenhuma responsabilidade na infra-estrutura da rede, agindo apenas como clientes requisitando conteúdo. Por outro lado, os super-nós, também chamados de *hubs*, são nós com muitos recursos disponíveis e um histórico de longos períodos de disponibilidade e que possuem responsabilidades cruciais na manutenção da rede e localização de conteúdo. Em particular, os

super-nós formam um backbone central ao qual as folhas se conectam (através de um ou mais super-nós) a fim de participar do sistema. A Figura 1 ilustra a topologia da rede em uma arquitetura com super-nós. Super-nós são representados por quadrados e as folhas por círculos.

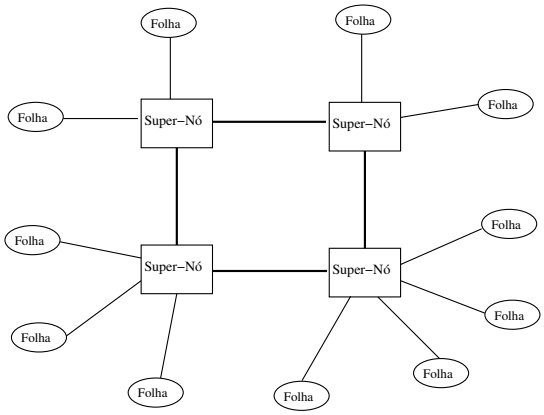

Figura 1: Topologia da Rede com Super-Nós

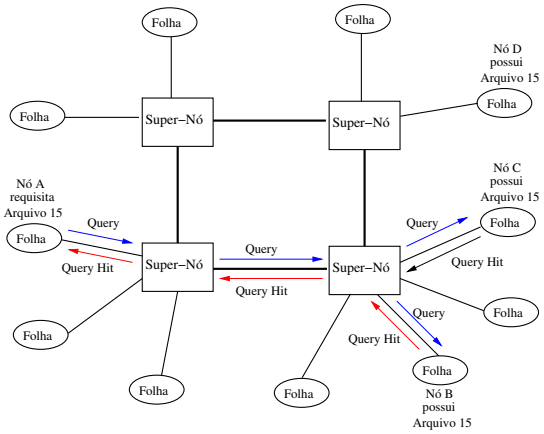

Figura 2: Localização de Conteúdo em Sistema com Super-Nós

A próxima seção discute as mensagens mais importantes da arquitetura com super-nós. A localização de conteúdo e os componentes do sistema são descritos na seção 3.2.2.

3.2.1. Mensagens

Além de *ping*, *pong*, *query*, *query hit*, *get* e *push*, que funcionam de maneira similar no protocolo original, o novo protocolo Gnutella 2.0 utiliza mensagens de *query ack*, *known hub list* e *query hash table*.

A mensagem de *query ack* é enviada por um super-nó como resposta a uma mensagem *query* de uma folha e contém a lista de super-nós vizinhos ao pai e a lista de vizinhos dos vizinhos (mantida atualizada através de mensagens KHL). Esta mensagem é enviada como parte do algoritmo de localização de conteúdo, descrito na próxima seção.

A mensagem de *known hub list* ou KHL é enviada periodicamente por cada super-nó para suas folhas e super-nós vizinhos. Cada mensagem KHL contém uma lista de super-nós conhecidos e uma de super-nós vizinhos. A primeira lista é utilizada quando um nó precisa se reconectar à rede após um período inativo. A lista de vizinhos é trocada apenas entre super-nós e será utilizada no mecanismo de localização de conteúdo.

A mensagem de *query hash table* ou QHT é enviada por um nó quando há alteração no conteúdo armazenado localmente. Ela contém a lista de arquivos que foram adicionados ou removidos do nó. Quando um super-nó recebe uma mensagem QHT de uma folha ou quando ocorre alguma alteração local, ele atualiza sua tabela e repassa a mensagem para seus super-nós vizinhos, indicando as alterações no conteúdo que o cluster composto por aquele super-nó e suas folhas sofreu. Para a execução do algoritmo de localização de conteúdo, descrito a seguir, cada super-nó mantém, além de uma tabela com os arquivos armazenados localmente, tabelas com os arquivos armazenados por

cada uma de suas folhas e por cada um de seus clusters vizinhos (super-nós vizinhos e suas respectivas folhas). As mensagens de QHT trocadas resultam em atualizações destas tabelas.

3.2.2. Localização de Conteúdo

A localização de conteúdo inicia com o envio de uma mensagem *query* de um cliente para seus super-nós. Note que este cliente pode ser o próprio super-nó. Quando um super-nó recebe uma *query*, ele retorna uma *query ack* para o cliente informando que sua consulta está sendo processada. Logo a seguir, ele verifica se o arquivo requisitado está armazenado localmente e propaga a mensagem *query* para suas folhas e para seus super-nós vizinhos.

O Gnutella 2.0 utiliza um mecanismo de roteamento de consultas para evitar o tráfego excessivo de mensagens na rede. Ao invés de enviar a *query* para todas as suas folhas, o super-nó primeiramente verifica as tabelas, mantidas localmente, com as consultas previamente respondidas por suas folhas. A *query* é enviada para uma folha somente se a tabela mantida pelo super-nó indica que aquela folha *pode* possuir o arquivo procurado. Da mesma forma, o super-nó verifica se determinado cluster vizinho, pode atender aquela query antes de repassá-la para um super-nó vizinho.

A figura 2 ilustra este mecanismo. O nó A está procurando o arquivo 15. Ele envia uma *query* para o super-nó ao qual ele está conectado (super-nó X), que retorna uma *query ack*, contendo a lista de seus vizinhos (super-nós Y e Z) e a lista de vizinhos de vizinhos (super-nó W). O super-nó X verifica se a *query* está nas tabelas de conteúdo de cada um de seus filhos e não encontra o arquivo. Então, ele procura nas tabelas de seus clusters vizinhos e descobre que apenas um deles possui o arquivo. Ele repassa a *query* para o super-nó Y. Y verifica se algum de seus filhos possui o arquivo e repassa a *query* apenas para dois deles, os nós B e C, onde o arquivo é encontrado. Os nós B e C enviam um *query hit* de volta para o cliente pelo mesmo caminho seguido pela *query*. Note que a *query* não é repassada para nenhum outro cluster. Logo, durante uma busca, uma *query* visita apenas clusters vizinhos à origem. No exemplo da Figura 2, a cópia do arquivo localizada no nó D não será encontrada, pois a *query* não é enviada para o super-nó W.

Uma vez que um usuário recebeu os resultados de uma consulta, ele pode continuar o processo de busca para encontrar outras cópias do arquivo na rede. Nós nos referimos a este mecanismo como retransmissão de consultas controlada por usuário. Se o usuário decide continuar o processo, a *query* é então enviada para um número de super-nós não visitados na busca anterior. Uma mensagem *query ack* contém uma lista de vizinhos do super-nó e uma lista de vizinhos dos vizinhos deste super-nó (conhecidos a partir da troca de mensagens KHL). A primeira lista contém super-nós que serão visitados pela *query* na busca corrente e que não devem ser contactados novamente em retransmissões futuras. A segunda lista contém super-nós que estão a dois hops de distância e não receberão a *query*. O usuário pode recomeçar a busca, enviando a *query* para estes super-nós. Na figura 2, o usuário pode iniciar uma nova busca enviando a *query* diretamente para o super-nó W. Neste caso, uma nova cópia do arquivo será encontrada no nó D.

Finalmente, após o término da busca de conteúdo, a transferência do arquivo requisitado pode ser feita de múltiplos nós, isto é, transferindo segmentos do arquivo de nós

diferentes. Este mecanismo de transferência de múltiplos nós reduz a sobrecarga nos nós servidores e a probabilidade de transferências canceladas por desconexões.

4. Metodologia de Avaliação

A avaliação do desempenho dos mecanismos utilizados em sistemas P2P recentes é feita através de simulação. A seção 4.1 apresenta os simuladores de sistemas P2P utilizados nesta avaliação. As métricas de desempenho são apresentadas na seção 4.2.

4.1. Simuladores de Sistemas P2P

Neste trabalho foram utilizados dois simuladores de sistemas P2P: um simulador existente [Barbosa et al., 2004, Costa et al., 2004] do protocolo Gnutella original modificado com a localização de conteúdo baseada em comunidades [Barbosa et al., 2004] e um novo simulador do protocolo Gnutella 2.0, exemplo típico da arquitetura baseada em super-nós. Ambos os simuladores foram implementados em Java e compartilham vários aspectos em comum descritos a seguir. As características específicas de cada simulador são apresentadas nas seções 4.1.1 e 4.1.2.

Cada experimento da simulação executa uma série de passos. A cada passo, um ou mais dos seguintes eventos podem ocorrer com probabilidade pré-definida. Um arquivo pode ser *requisitado* com probabilidade 0.5. Neste caso, um nó é escolhido para iniciar a busca. Um novo arquivo pode ser *criado* e inserido em um nó com probabilidade 0.2, no início da simulação, e 0.05 alguns passos depois. Foram inseridos mais arquivos no início da simulação a fim de popular o sistema. Um nó selecionado pode enviar uma mensagem de *ping* a seus vizinhos com probabilidade 0.1. Finalmente, no simulador do Gnutella 2.0, um nó pode enviar uma mensagem *KHL* para seus super-nós ascendentes (se for uma folha) e super-nós vizinhos (se for um super-nó) com probabilidade 0.1.

Os simuladores reproduzem várias características dos sistemas reais, descritos em análises anteriores de sistemas P2P [Saroiu et al., 2002, Ripeanu and Foster, 2002]. Uma das características é a heterogeneidade dos nós, tanto em termos de capacidade de transmissão (banda) e armazenamento quanto em nível de atividade (alguns nós enviam mais requisições que outros). A tabela 1 mostra as probabilidades de um nó ter diferentes capacidades de armazenamento e banda de transmissão. Além disso, a seleção do nó que executa uma operação (requisição, inserção, etc) a cada passo segue uma lei de potência (uma distribuição Pareto), a fim de modelar o fato de que alguns nós são mais ativos do que outros. É importante observar que os nós mais ativos são os nós que possuem uma maior probabilidade de possuir maior largura de banda e capacidade de armazenamento.

Armazenamento		**Banda**	
Capacidade (GB)	**Probabilidade**	**Capacidade (Kbps)**	**Probabilidade**
1	0.2	64	0.1
5	0.4	256	0.6
10	0.4	512	0.3

Tabela 1: Distribuição de Armazenamento e Banda entre os Nós

A carga de arquivos criada e requisitada pelos nós também é heterogênea. Os arquivos podem ser categorizados em três classes baseadas em seus tamanhos: música,

shows de TV e filmes. Os tamanhos dos arquivos em cada uma destas três classes são distribuídos com média (desvio padrão) iguais a 4.5 (0.5) MB, 70 (15) MB e 700 (150) MB, respectivamente. Para nossos experimentos, os arquivos são distribuídos pelas três classes com probabilidades 0.7 para música, 0.15 para shows de TV e 0.15 para filmes. A popularidade dos arquivos no sistema também não é uniforme. Os 2000 arquivos únicos no sistema possuem popularidade relativa seguindo uma distribuição Pareto.

Nosso simulador modela um conjunto estático de nós ativos (3000 nós em nossos experimentos). Entre outras palavras, os nós não saem e entram nos sistema dinamicamente. A simulação do comportamento dinâmico dos nós foi deixado para um trabalho futuro. As características específicas de cada simulador seguem nas próximas seções.

4.1.1. Simulador do Sistema com Inundação e Comunidades

Este simulador implementa os mecanismo básicos do protocolo Gnutella 0.4 utilizando o algoritmo de busca de conteúdo baseado em comunidades, descrito na seção 3.1. Os parâmetros específicos deste simulador são o número máximo de vizinhos por nó que é estabelecido como 20, conforme [Cornelli et al., 2002], e o tamanho máximo da comunidade, igual a 10. Além disso, o *time-to-live* de todas as mensagens é igual a 7 hops [Ripeanu and Foster, 2002].

A topologia da rede foi construída de forma a modelar uma rede *small world*, previamente observada em estudos sobre o Gnutella [Saroiu et al., 2002, Ripeanu and Foster, 2002]. Em redes com topologia *small world*, os nós tendem a ficar agrupados, mas ao mesmo tempo, poucos links separam um nó de qualquer outro. Para representar este comportamento, os nós foram conectados em forma de anel adicionando-se atalhos entre pares de nós, com uma probabilidade pequena (0,05 nos nossos experimentos), como proposto em [Hayes, 2000]. Uma descrição mais bem detalhada deste simulador pode ser obtida em [Barbosa et al., 2004, Costa et al., 2004].

4.1.2. Simulador do Sistema de Super-Nós

Este simulador implementa a arquitetura e os principais mecanismos empregados no protocolo do Gnutella 2.0 [Gnutella2,] e descritos na seção 3. O número de super-nós corresponde a 10% do total de nós, ou seja, 300. Os super-nós são eleitos dentre todos os nós início da simulação e o critério utilizado para a eleição é a largura de banda dos nós. Desta forma, os super-nós escolhidos são, em geral, os nós com maior capacidade de armazenamento e atividade na rede. O projeto de políticas de eleição de super-nós é tópico de trabalho futuro.

O backbone entre os super-nós é construído de forma que o grau dos super-nó esteja uniformemente distribuído entre 2% a 10% do número de super-nós na rede, isto é, entre entre 6 e 30, de acordo com a especificação [Gnutella2,]. Cada folha se conecta a 3 super-nós, selecionados aleatoriamente. A conexão de um nó folha a mais de um super-nó garante uma maior disponibilidade do nós. Quando um super-nó se torna indisponível, se suas folhas não estão conectadas a outros super-nós, elas perderiam a conexão com o sistema. Em trabalhos futuros vamos avaliar o impacto do número de super-nós por folha no desempenho do sistema em um cenário no qual o nós possam sair e entrar no sistema

dinamicamente.

4.2. Métricas de Desempenho

O desempenho das arquiteturas P2P são avaliadas com as seguintes métricas:

- **Sobrecarga no sistema**: número médio de mensagens processadas por cada nó.
- **Latência da consulta**: tempo médio gasto na localização de arquivos, medido como o número de hops que uma mensagem *query* atravessa durante este processo.
- **Taxa de sucesso**: Porcentagem de consultas que são respondidas com sucesso.
- **Tempo de transferência**: Tempo médio gasto na transferência de arquivos

5. Resultados Experimentais

Esta seção apresenta os resultados da comparação entre a arquitetura de super-nós com a arquitetura de inundação com comunidades. A seção 5.1 apresenta os melhorias de desempenho considerando apenas a arquitetura hierárquica de super-nós sem nenhuma otimização. A seção 5.2 mostra os ganhos de desempenho do mecanismo de roteamento de consultas. O impacto de retransmissões de consultas é investigado na seção 5.3, e finalmente, o desempenho de transferências de múltiplas fontes é discutido na seção 5.4. Os resultados são a média de 10 execuções, cada uma com 5000 passos simulados, suficientes para cobrir um longo período de estabilidade do sistema.

5.1. Arquitetura Hierárquica

A primeira avaliação consiste na redução da carga no sistema promovida pela arquitetura com super-nós. A figura 3 mostra o número médio de mensagens processadas por cada nó durante a simulação, para diferentes tipos de mensagens. Nós omitimos os números de *query ack, KHL* e *get* porque eles totalizam menos que 1.3% de todas as mensagens processadas. O número médio de mensagens *query* processadas por cada nó na arquitetura com super-nó é aproximadamente 95% inferior ao valor correspondente na arquitetura que utiliza inundação de mensagens com comunidades. Se compararmos o volume total de mensagens trocadas como parte do mecanismo de busca (isto é, *q*uery, *q*uery hit e *q*uery ack), a redução é de 91%. Além disso, a redução do número de mensagens de *ping* e *pong* também é muito significativa (99%), pois o protocolo Gnutella original também utiliza inundação de mensagens para manutenção da rede.

Uma pergunta em aberto é se a redução da carga no sistema observada vem ao custo de uma baixa taxa de consultas respondidas com sucesso. A figura 4 mostra a taxa de sucesso obtido para ambos os protocolos ao longo da simulação. Note que, como observado em [Barbosa et al., 2004], a taxa de sucesso para o mecanismo de inundação com comunidades melhora com o passar do tempo, quando as comunidades se tornam mais efetivas. A taxa de acerto para a arquitetura com super-nós é bem menos sensível. Entretanto, ambas as arquiteturas convergem para a mesma taxa de sucesso, cerca de 90%. Sendo assim, a arquitetura com super-nós, mesmo sem otimizações, consegue alcançar uma melhor escalabilidade para sistemas P2P para compartilhamento de arquivos sem penalizar a taxa de sucesso das consultas.

A latência média de consultas no protocolo original do Gnutella corresponde a duas vezes o *time-to-live* da *query*, isto é, 14 em nossos experimentos, de forma a permitir

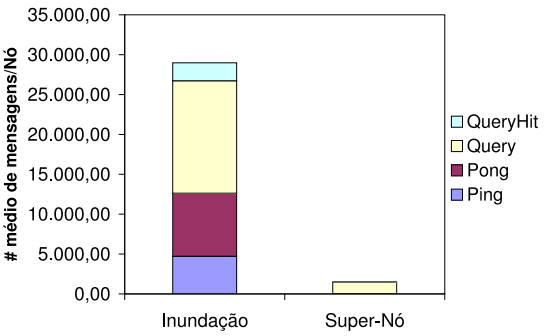

Figura 3: Carga no Sistema

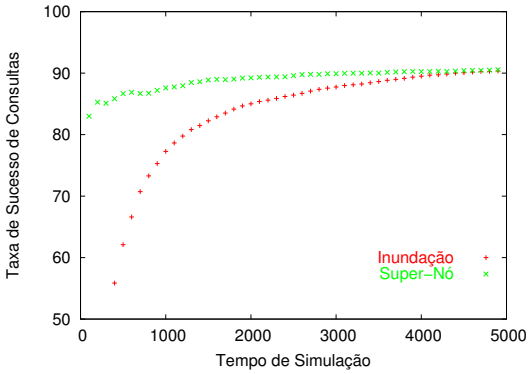

Figura 4: Evolução da Taxa de Sucesso de Consultas

que cada nó possa enviar e receber a resposta em tempo. Com o uso do algoritmo de comunidades, algumas consultas são respondidas pela comunidade, diminuindo o tempo médio de latência para 12.5 hops. De forma similar, na arquitetura com super-nós, a latência é igual a 6 hops, que corresponde a duas vezes o tamanho do maior caminho que uma *query* pode percorrer durante uma busca (ignorando possíveis retransmissões). Logo, a latência média é cerca de 50% inferior na arquitetura com super-nós.

Arquitetura P2P	Música	Show de TV	Filme
Inundação com Comunidades	407 seg	879 seg	2624 seg
Super-nó	277 seg	811 seg	2647 seg

Tabela 2: Tempo Médio de Transferência

Finalmente, a tabela 2 mostra o tempo médio de transferência para música, shows de TV e filmes, em cada arquitetura. Note que os resultados são para a transferência de uma única fonte. O impacto da transferência de múltiplas fontes na arquitetura de super-nós é avaliado a seguir. O tempo de transferência é muito maior para a arquitetura que utiliza inundação, especificamente para arquivos pequenos. Isto ocorre devido ao número de consultas atendidas pela comunidade. Alguns nós ficam sobre-carregados, realizando muitas transferências simultaneamente.

Portanto, mesmo sem três mecanismos importantes, a arquitetura hierárquica de sistemas P2P consegue prover uma plataforma para compartilhamento de arquivos mais escalável, reduzindo carga no sistema, latência da busca e tempo de transferência, sem degradar a taxa de consultas bem sucedidas.

5.2. Roteamento Otimizado de Consultas

O principal objetivo do mecanismo de roteamento de mensagens *query* baseado nas tabelas de conteúdo armazenadas nos super-nós é reduzir o número de mensagens trocadas durante a busca por conteúdo. A figura 5 mostra o número médio de mensagens processadas por cada nó, com e sem o mecanismo de roteamento. No último caso, uma *query* é repassada a todas as folhas e super-nós vizinhos independente deles terem

ou não o arquivo requisitado. As duas colunas da esquerda mostram os resultados sem retransmissão de mensagens *query*. Neste caso, o mecanismo de roteamento é capaz de reduzir o total de mensagens em cerca de 97%. Comparando com o número de mensagens do protocolo de inundação de mensagens, isto corresponde a uma redução de tráfego de 99.9%. As duas colunas da direita mostram resultados com duas repetições da busca (duas retransmissões). Estes resultados são discutidos abaixo.

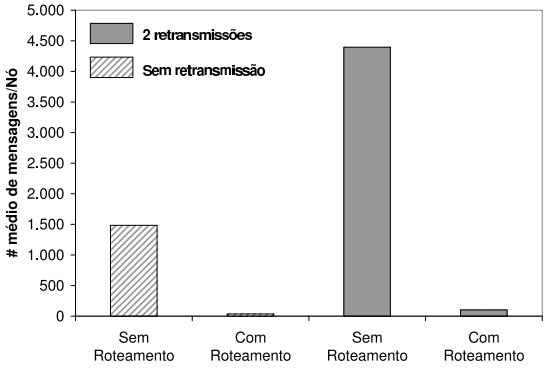

Figura 5: Carga no Sistema Com e Sem Roteamento de Consultas

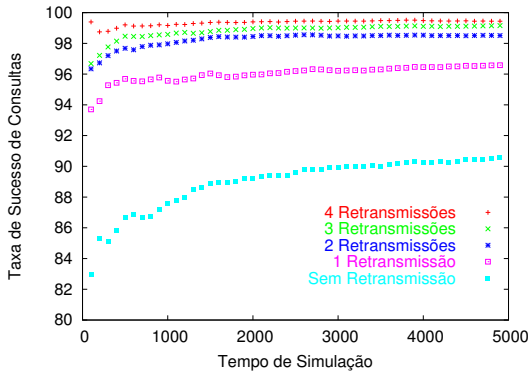

Figura 6: Taxa de Sucesso de Consultas em Função do Número de Retransmissões

5.3. Retransmissão de Consultas Controlada por Usuários

As arquiteturas com super-nós limitam a retransmissão de mensagens *query* a um pequeno raio ao redor do nó emissor, mas permite que o usuário retransmita a *query* para outras partes da rede. A retransmissão é necessária se uma busca não foi bem sucedida ou trouxe poucos resultados. Além disto, em algumas situações, o aumento óbvio na latência pode ser irrelevante se o arquivo for muito grande (ex: 600 MB). O usuário pode preferir esperar um pouco mais repetindo a busca para encontrar um maior número de fontes para transferências. Esta seção quantifica os benefícios deste mecanismo.

A figura 6 mostra que a taxa de sucesso de consultas ao final da simulação aumenta de 90% para aproximadamente 96%, se cada query é retransmitida uma vez. A segunda retransmissão aumenta a taxa para 98%. Sendo assim, com poucas retransmissões é possível cobrir quase toda rede devido à alta conectividade dos super-nós.

Outro aspecto de desempenho que pode ser melhorado com o número de retransmissões é o tempo médio de transferência. Em ambos simuladores, a escolha do par de onde a transferência de um arquivo deve ser realizada, caso múltiplas cópias sejam encontradas, é feita pelo critério de maior banda disponível. Logo, um número maior de retransmissões pode resultar em um número maior (e melhor) de opções de transferência. A figura 7 mostra as reduções no tempo médio de transferência obtidas com diferentes números de retransmissões para os três tipos de arquivos. Os melhores resultados são para arquivos de música, mais populares em nossa simulação.

Por outro lado, o mecanismo de retransmissão de *queries* aumenta o número médio de mensagens processadas por cada nó. As duas colunas da direita da figura 5 mostram o número médio de mensagens relacionadas ao mecanismo de busca processadas por cada nó, com e sem roteamento e com duas retransmissões. Em ambos os casos,

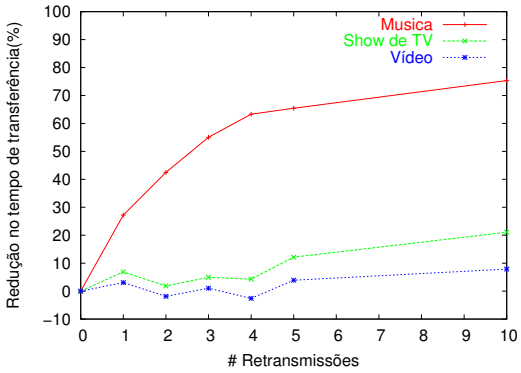

Figura 7: Redução no Tempo de Transferência em Função do Número de Retransmissões

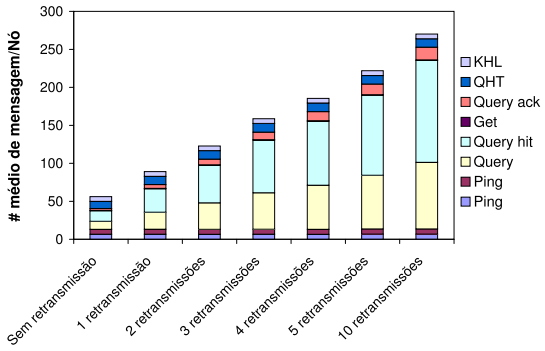

Figura 8: Carga no Sistema como Função do Número de Retransmissões

o número de mensagens aumenta por um fator de 2.9 se comparado com o caso sem retransmissão. Apesar disso, mesmo com duas retransmissões, a carga no sistema é mínima quando o roteamento otimizado de consultas é realizado.

Arquitetura P2P	Latência (# hops)
Inundação com Comunidades	12.5
Super-nós sem Retransmissão	6
Super-nós com 1 Retransmissão	12
Super-nós com n Retransmissões	$6(n+1)$

Tabela 3: Latência Média em Função do Número de Retransmissões

É interessante notar que o aumento da carga no sistema *não* é linear com o número de retransmissões, como mostra a figura 8. O número total de mensagens processadas por um nó quando cada *query* é retransmitida 10 vezes é cerca de 5 vezes maior que o número de mensagens quando não há retransmissão. Se compararmos apenas o número de *queries*, este fator é 8. Isto acontece devido a dois fatores: (1) ao mecanismo de roteamento de *queries*, que só repassa as mensagens para nós onde há probabilidade do arquivo ser encontrado, e (2) à alta conectividade do super-nós na rede (uma *query* é enviada apenas uma vez a cada super-nó).

Finalmente, a latência da busca aumenta linearmente com o número de retransmissões, como esperado. Entretanto, a tabela 3 mostra que a latência para uma ou duas retransmissões ainda é competitiva. Logo, a retransmissão de consultas é um mecanismo eficiente e escalável que dá flexibilidade aos usuários para decidir por um compromisso entre aumentar a latência e carga no sistema em troca de um aumento de consultas bem sucedidas e redução no tempo de transferência.

5.4. Transferência de Arquivos de Múltiplas Fontes

Esta seção avalia o impacto da transferência de múltiplas fontes no tempo médio de transferência. Foram realizados uma série de experimentos nos quais o número máximo de fontes, isto é o número máximo de segmentos no qual o arquivo nos é subdividido é limitado. Note que este número já é limitado naturalmente pelo número de pares

# Transferências simultâneas	Música	Shows de TV	Filmes
1 fonte	278	811	2647
2 fontes	264	885	2405
3 fontes	235	936	2360
4 fontes	224	975	2508
10 fontes	174	1022	2393
Todas as fontes	164	1016	2372

Tabela 4: Tempo Médio de Transferência (em segundos)

que respondem a uma consulta com sucesso. A tabela 4 mostra os resultados obtidos. Em particular, a última linha da tabela mostra os resultados quando o número de fontes é limitado apenas pelo número de nós que responderam à consulta.

Como mostra a tabela 4, o tempo médio de transferência de arquivos pequenos, de música, melhora à medida que o número de transferências simultâneas aumenta. Porém, ele permanece estável e algumas vezes até aumenta para arquivos maiores. Isto ocorre porque à medida que se aumenta o número de fontes, aumenta-se também a probabilidade de se transferir um segmento de uma fonte com baixa capacidade de transmissão disponível. Sendo assim, a transferência de arquivos de múltiplas fontes só traz benefícios de desempenho se as fontes de transferência forem cuidadosamente selecionados. Políticas de seleção de nós para transferência serão abordadas em trabalhos futuros.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma extensa avaliação de desempenho de diferentes mecanismos empregados para recuperação de conteúdo em sistemas P2P: a arquitetura hierárquica baseada em super-nós, o roteamento otimizado de consultas, a retransmissão de consultadas controlada por usuários e a transferência de arquivos de múltiplas fontes. Ele quantifica os ganhos de desempenho propiciados por cada mecanismo sobre a arquitetura P2P baseada em inundação de mensagens otimizada com o uso de comunidades. Os resultados mostraram que reduções significativas na carga no sistema, na latência de consulta e no tempo médio de transferência bem como aumentos nas taxas de consultas bem sucedidas podem ser obtidos com cada um dos mecanismos individualmente.

Trabalhos futuros incluem experimentos com nós entrando e saindo dinamicamente da rede, avaliação do impacto de políticas para a eleição de super-nós e para a seleção de nós fontes de transferência.

7. Agradecimentos

Este trabalho foi desenvolvido em colaboração com a HP Brasil P&D.

Referências

Barbosa, M., Costa, M., Almeida, J., and Almeida, V. (Janeiro 2004). Using Locality of Reference to Improve Performance of Peer-to-Peer Aplications. In *Proc. 4th Workshop on Software Performance (WOSP)*, Redwood City, CA.

Castro, M., Druschel, P., Hu, Y., and Rowstron, A. (2002). Exploiting Network Proximity in Distributed Hash Tables. In *Proc. International Workshop on Future Directions in Distributed Computing*, Bertinoro, Italia.

Chawathe, Y., Ratnasamy, S., Breslay, L., and Shenker, S. (2003). Making Gnutella-like P2P Systems Scalable. In *Proc. ACM SIGCOMM*, Karlsruhe, Alemanha.

Cornelli, F., Damiani, E., and Capitani, S. (2002). Choosing Reputable Servents in a P2P Network. In *Proc. Eleventh International WWW Conference*, Honolulu, HI.

Costa, M., Barbosa, M., Almeida, J., and Almeida, V. (Agosto 2004). Otimizando a Recuperação de Conteúdo em Redes Par-a-Par. In *Proc. XIII SEMISH*, Salvador, Bahia.

Figueiredo, D., Jaiswal, S., Ge, Z., Towsley, D., and Kurose, J. (2003). Modeling Peer-to-Peer File Sharing Systems. In *Proc. IEEE INFOCOM*, San Francisco, CA.

Gnutella. http://www.gnutella.com.

Gnutella2. http://www.gnutella2.com/.

Hayes, B. (Janeiro 2000). *Graph Theory in Practice, Part II*, volume 88. American Scientist, Academic Press, New York.

KaZaA. http://www.kazaa.com.

Liu, Y., Liu, X., Xiao, L., Ni, L., and Zhang, X. (2004). Location-Aware Topology Matching in P2P Systems. In *Proc. INFOCOM*, Hong Kong.

Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S. (2002). Search and Replication in Unstructured Peer-to-Peer Networks. In *Proc. 16th International Conference on Supercomputing*, New York, NY.

Napster. http://www.napster.com.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. (2001). A Scalable Content-Addressable Network. In *Proc. ACM SIGCOM*, San Diego, CA.

Ripeanu, M. and Foster, I. (2002). Mapping the Gnutella Network: Macroscopic Properties of Large-Scale Peer-to-Peer systems. In *Proc. 1st International Workshop on Peer-to-Peer Systems*, Cambridge, MA.

Saroiu, S., Gummadi, P., and Gribble, S. (2002). A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proc. Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA.

Stoica, I., Morris, R., Karger, D., Kaashoek, M., and Balakrishnan, H. (2001). Chord: A Scalable Peer-to-peer Lookup Service for Internet. In *Proc. ACM SIGCOMM*, San Diego, CA.

Yang, B. and Garcia-Molina, H. (2001). Comparing Hybrid Peer-to-Peer Systems. In *The VLDB Journal*, pages 561–570.

Yang, B. and Garcia-Molina, H. (2003). Designing a Super-Peer Network. In *Proc. the 19th International Conference on Data Engineering*, Bangalore, Índia.