Um estudo sobre sistemas P2P

Emilio de Camargo Francesquini

25 de junho de 2004

Resumo

Sistemas peer-to-peer[1] (P2P) são sistemas distribuídos que funcionam sem um ponto central de organização. Em uma rede P2P nenhuma máquina participante é considerada diferente das demais, ou seja, não existem servidores centrais para cuidar das requisições dos clientes. Existem vários sistemas P2P implementados para os mais diversos fins. Neste relatório serão mostradas algumas aplicações e características destes sistemas.

[1] A palavra inglesa '*peer*´ se refere a alguém que está à mesma altura de alguma outra pessoa no que diz respeito a algum círculo social tal como o grau de instrução ou posição financeira. Duas palavras semelhantes em português são companheiro ou colega.

Sumário

1 Introdução

Considere os seguintes requisitos para a construção de uma rede:

Disponibilidade da rede A rede deve estar disponível mesmo que ocorra a falha de alguns de seus componentes. Isso é essencial para sistemas críticos.

Interoperabilidade entre diversas máquinas Isto implica que a rede é independente das várias arquiteturas de suas máquinas componentes. Todos os seus nós 'falam a língua' da rede, e não o contrário.

Segurança da informação A rede deve oferecer uma certa proteção contra a interceptação das transmissões.

Estes três requisitos foram as linhas mestras para a definição do protocolo TCP/IP. Em uma rede TCP/IP todos os nós atuam da mesma forma, com os mesmos direitos (como peers), não existe nenhum ponto central de controle para monitorar as comunicações. Entretanto, com o tempo, o uso de peers na internet acabou entrando em decadência. Grande parte da internet hoje é baseada na arquitetura cliente/servidor (eg. web e e-mail). Neste sentido, o grande aumento da utilização de programas P2P é como uma volta no tempo, onde a descentralização é levada ao extremo.

Pode-se também classificar as redes em três categorias:

Centralizadas Uma rede centralizada (figura 1 a) é uma rede onde os clientes se conectam a um servidor. É uma estrutura parecida com uma estrela, onde o centro é o servidor e as pontas são os clientes. Em uma rede deste tipo, caso o servidor falhe, toda a rede falha. Existe um único ponto central de falha.

Descentralizadas Uma rede descentralizada(figura 1 b) é um aglomerado de estrelas com os seus centros conectados. Neste caso existem diversos pontos centrais de falha, e a falha de um deles compromete apenas parte da rede.

Distribuídas Em uma rede distribuída (figura 1 c) não existem servidores pré-definidos. Qualquer nó pode ser um servidor ou um cliente. Não existem pontos centrais de falha.

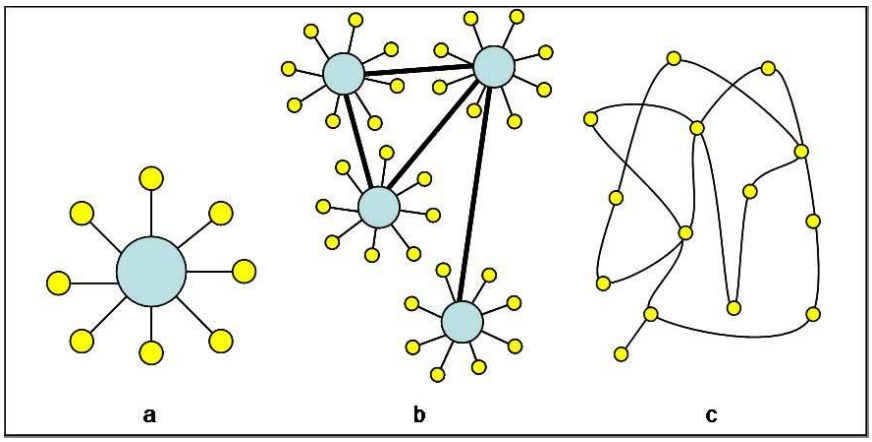


Figura 1: Categoria das redes. a) mostra uma rede centralizada, b) mostra uma rede descentralizada e c) uma rede distribuída.

Dizemos que uma rede é P2P quando ela é puramente distribuída.

Redes P2P trazem diversas vantagens quando comparadas com redes centralizadas ou descentralizadas. Dois exemplos são a menor suscetibilidade a falhas (não há pontos de falha centrais) e distribuição da carga (o serviço fica distribuído pelos nós e não concentrado em alguns poucos servidores). Maiores comparações serão feitas nas seções que seguem.

2 Redes P2P

As redes P2P são redes de computadores que não necessitam de servidores centrais para funcionar. Todos os participantes da rede são tratados como iguais (peers) e cada um destes assume, a cada instante, um papel diferente. Quando estão acessando informações são clientes, quando estão servindo informações eles são servidores e quando estão repassando informações eles são roteadores. O grande desafio é modelar uma rede P2P robusta e escalável que seja composta de computadores baratos, individualmente não confiáveis e arbitrariamente espalhados pela rede. [18]

2.1 Motivação

Redes P2P são muito atraentes, principalmente, pelos seguintes motivos [12]:

- As barreiras para a criação de uma rede deste tipo são mínimas já que, ao contrário dos sistemas centralizados/descentralizados, este tipo de rede não necessita nenhum tipo de administração ou instalações específicas;
- Sistemas P2P oferecem uma maneira de utilizar, de forma conjunta, o enorme potencial de armazenamento e processamento espalhado pelos computadores conectados a Internet;
- A natureza distribuída deste tipo de rede lhe dá o potencial de ser resistente a falhas ou ataques intencionais, o que faz com que seja o ambiente ideal para o armazenamento de informações a longo prazo ou computações demoradas.

Redes P2P trazem à tona uma grande quantidade de problemas interessantes para a pesquisa no campo de sistemas distribuídos. Neste relatório concentraremo-nos no problema de busca: como encontrar uma informação dentro de uma grande rede de forma escalável sem o auxílio de servidores centrais ou de hierarquia? Este problema está presente em praticamente todos os sistemas P2P e, atualmente, não é tratado de maneira satisfatória.

2.2 Principais exemplos de caso de uso

Aqui estão listadas algumas das aplicações P2P mais freqüentemente utilizadas.

2.2.1 Compartilhamento de arquivos

Talvez este seja o caso de uso mais difundido tanto que o termo 'programa P2P' virou um sinônimo para programas de compartilhamento de arquivos. Exemplos deste tipo de programa são: o Gnutella [5], Napster [8], KaZaA [7] e E-Donkey [2]. O uso mais comum para este tipo de programa é o compartilhamento de arquivos MP3 e de vídeo. Ou seja, este tipo de programa incentiva a pirataria digital: pessoas conseguem baixar álbuns inteiros da internet sem pagar nada por isso. Este tipo de problema já alcançou tribunais em diversas ocasiões sendo que o caso mais conhecido é o do Napster que foi obrigado a fechar as suas portas.

2.2.2 Compartilhamento de recursos

Existem diversos sistema que se utilizam deste tipo de rede para o processamento ou armazenamento de informações. A idéia aqui é aproveitar melhor os recursos que normalmente seriam desperdiçados. Grande parte destes sistemas utiliza uma arquitetura centralizada ou descentralizada. Exemplos disto são: SETI@Home [9] e Integrade [6]. Outros entretanto utilizam-se de uma arquitetura totalmente descentralizada como o Freenet [14], o OceanStore [17].

2.3 Arquiteturas

Aqui estão listadas as arquiteturas utilizadas na construção de sistemas P2P juntamente com as suas principais características.

2.3.1 Arquiteturas com serviço de localização centralizado

Este tipo de arquitetura não faz uso de uma rede totalmente distribuída e, portanto, este tipo de sistema não é um sistema totalmente P2P.

Quando o usuário se conecta na rede ele, na verdade, está se conectando a um ou mais servidores. Estes servidores servem como um índice para busca dos dados disponíveis, ou seja, quando um usuário se conecta ele informa ao servidor todos os dados que ele está dispondo na rede e o servidor atualiza os seus índices para que fiquem disponíveis para consulta para os outros usuários.

Quando um dos usuários deseja fazer uma busca por um determinado ítem, ele contacta um dos servidores e busca a informação desejada. A resposta do servidor é o endereço IP de uma das máquinas que tem o dado desejado (essa máquina obtida na pesquisa também já passou por todo o processo de conexão descrito no parágrafo anterior). Deste instante em diante as duas máquinas passam a se falar, diretamente uma com a outra, para efetuar a troca de dados. A figura 2 ilustra este processo.

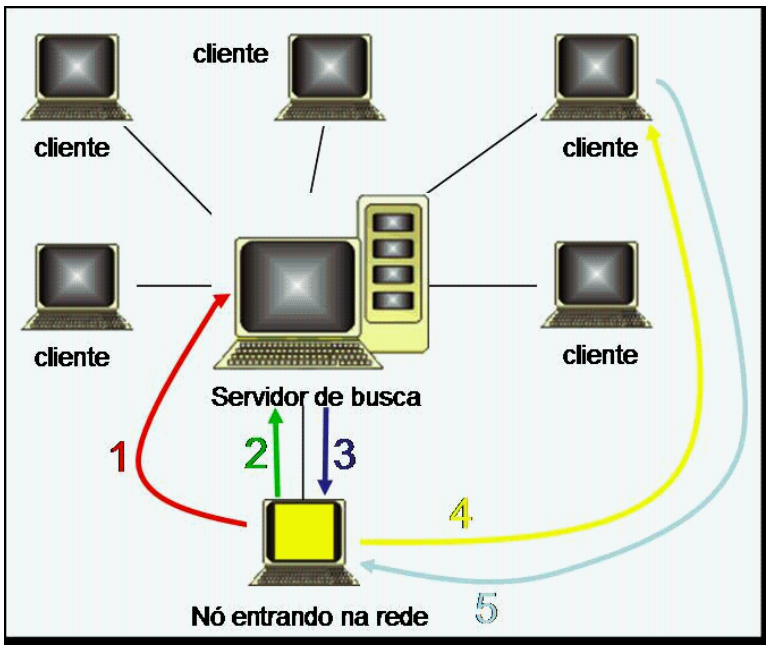


Figura 2: Funcionamento de uma rede centralizada. 1. O novo cliente se conecta ao servidor e envia uma lista dos seus dados compartilhados. 2. O cliente já registrado no servidor solicita a pesquisa de dados. 3. O servidor retorna uma lista com o endereço de todos os nós que contém a informação desejada. 4. O cliente contacta o nó que contém o arquivo desejado e o requisita. 5. O nó contactado transfere o arquivo para o nó que o requisitou.

O serviço de localização centralizado traz vantagens e desvantagens. Um dos grandes ganhos é a garantia de que se o dado está disponível em algum dos nós da rede então ele será encontrado. Esta garantia não está presente em todos os sistemas P2P, como veremos adiante. Além disto, buscas com palavras chave e múltiplos critérios são facilmente implementadas. A desvantagem, entretanto, é a alta suscetibiliade a falhas pois existem pontos centrais de falha, se estes pontos falharem a rede toda falha.

Abaixo citamos dois exemplos de sistemas que usam este tipo de arquitetura junto com algumas de suas características.

Exemplo 1 - Napster A rede do Napster é muito parecida, senão idêntica, ao esquema apresentado acima. Justamente por isso foi tão fácil fechar as suas portas quando a justiça assim o determinou.

O componente crítico que compunha a sua rede (a pesquisa) era centralizado. A parte P2P de seu protocolo não funcionava sem a parte centralizada o que fez da rede um alvo fácil.

Exemplo 2 - Edonkey A robustez da rede do Edonkey é, em muitos aspectos, superior à rede do Napster. Ela é mais resistente a ataques e a comunicação, entre os peers, se dá de uma forma mais inteligente. A rede do Edonkey veio algum tempo depois do Napster e aprendeu algumas de suas lições. Primeiramente o servidor não é centralizado e sim descentralizado. Qualquer um pode, tendo uma conexão à Internet, iniciar um servidor na própria máquina. Isto protege, até certo ponto, a rede como um todo: a queda de alguns servidores não compromete a rede toda. Para acabar com a rede ter-se-ia que acabar com todos os servidores, uma tarefa bem difícil já que eles estão espalhados pelo mundo todo.

O esquema de busca é bem parecido com o esquema já apresentado. O cliente se conecta a um dos servidores e lá publica a lista de seus arquivos compartilhados. Quando um nó deseja efetuar uma pesquisa, ele itera sobre a lista de servidores enviando requisições de pesquisa. Cada servidor responde a essa requisição com uma lista dos IPs dos nós de sua responsabilidade que contém o ítem pesquisado. O cliente recebe as listas dos servidores e as percorre contactanto os nós devolvidos na pesquisa. Os nós contatados então respondem enviando o dado requisitado.

Quando um nó recebe a requisição por um dado, ele também envia, junto com os dado propriamente dito, uma lista dos outros nós conhecidos que possuem o mesmo dado. Isso faz com que (mesmo que a lista de servidores do nó que está efetuando a pesquisa não esteja atualizada) sejam encontrados praticamente todos os nós da rede que têm o arquivo. A única restrição é que todos os nós tenham na sua lista de servidores um único servidor em comum. Essa restrição é facilmente cumprida já que a lista de servidores ativos é publicada regularmente na própria página do Edonkey.

2.3.2 Arquiteturas baseadas em inundação

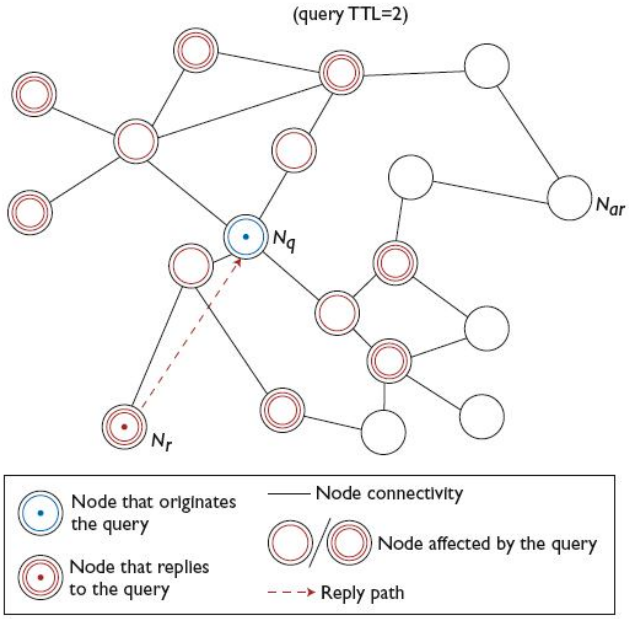
Arquiteturas de rede baseadas em inundação (Flooding Style Networks) são redes 100% P2P. Estas redes são totalmente distribuídas e portanto não possuem um ponto central de falha. A única maneira de desmantelar a rede seria acabar com todos os seus nós, uma tarefa que é, para não dizer impossível, extremamente difícil. A maneira pela qual os nós se ligam a rede não é estruturada. Um nó que deseja entrar na rede simplesmente descobre alguns outros nós já presentes na rede (por broadcast ou qualquer outro meio) e se conecta a eles.

As pesquisas nestas redes são feitas por mecanismos de inundação controlada por TTL (time-to-live). Um nó que deseja pesquisar por dados manda uma mensagem para todos os nós nos quais ele está conectado: seus vizinhos. Cada um de seus vizinhos avalia a mensagem e verifica se tem o dado pesquisado. Caso possua, ele envia uma mensagem de volta ao nó de onde se originou a pesquisa informando o seu endereço IP. Caso não possua, ele incrimenta o número de saltos da mensagem (hop count) e a repassa para todos os seus vizinhos. Quando o número de saltos ultrapassar o TTL o repasse de mensagens é interrompido. Logo, o TTL define o horizonte, ou o raio de ação da pesquisa. Sistemas deste tipo têm mecanismos específicos para lidar com mensagens presas em laços.

Redes deste tipo, entretanto, não são escaláveis pois elas requerem um grande poder de processamento e largura de banda. Além disto, elas não oferecem garantia alguma quanto a acessibilidade dos dados (o nó que contém o dado pode estar mais longe que o TTL - ex. requisição de Nq por um dado que está em Nar na figura 3) ou do tempo necessário para encontrá-lo [15] [5]. A figura 3 [15] ilustra este esquema.

Exemplo 1 - Gnutella [5] O protocolo do Gnutella segue o esquema descrito acima com algumas poucas alterações e, portanto, possui os mesmos pontos fracos. De fato, existem relatos de que no dia seguinte ao fechamento do Napster, a rede Gnutella parou de funcionar devido a carga

Figura 3:



Uma representação do funcionamento de uma rede P2P baseada em TTL. O nó Nq transmite uma consulta requisitando o valor de uma chave que está localizada em Nr. Os círculos concêntricos representam o número de saltos da mensagem.

criada pela grande quantidade de usuários migrando do Napster para o Gnutella [12]. Este tipo de problema poderia ser resolvido através do uso de *superpeers* como é feito pela plataforma P2P Fasttrack [3]. Entretanto, o uso de superpeers traz consigo a perda de resistência a falhas nos superpeers nos pontos mais altos da hierarquia, além de não resolver o problema de acessibilidade.

O anonimato dos usuários é praticamente inexistente. Qualquer um na rede consegue facilmente listar as pesquisas (e os pesquisadores) que estão sendo feitas. Outro ponto fraco é a proteção de quem está fornecendo os arquivos, os seus endereços IP também podem ser facilmente listados, já que o protocolo utilizado é o HTTP.

Gnutella, entretanto, não tem apenas pontos fracos. Um dos pontos fortes é a liberdade de pesquisa. Cada um dos nós da rede mantém uma lista dos seus arquivos compartilhados. Uma mensagem de pesquisa, dependendo da implementação utilizada, pode ser composta de uma expressão lógica ou caracteres coringas. Neste sentido as pesquisas por arquivos no Gnutella não deixam nada a desejar (obviamente levando em conta o horizonte das mensagens).

Exemplo 2 - Freenet [14] [4] Freenet é um programa de código fonte aberto que foi originalmente desenvolvido por Ian Clarke. Freenet é um sistema bem mais radical que o Napster ou o Gnutella. Não há servidor central. Não existe a possibilidade de rastrear a origem de um arquivo, quem o está baixando ou quem salvou este arquivo em disco. O objetivo é simples: anonimato. Ele promete criar uma internet sem censura e anônima dentro da internet. Arquivos, uma vez disponíveis, não podem ser (facilmente) retirados da rede. As metas do Freenet são bem claras: liberdade social e política. Entretanto, toda esta liberdade de expressão e anonimato podem ser utilizadas para fins subversivos. Isto é discutido em [14] e [13]. Existem relatos da utilização deste sistema na China e no Oriente Médio para espalhar informações censuradas pelo governo.

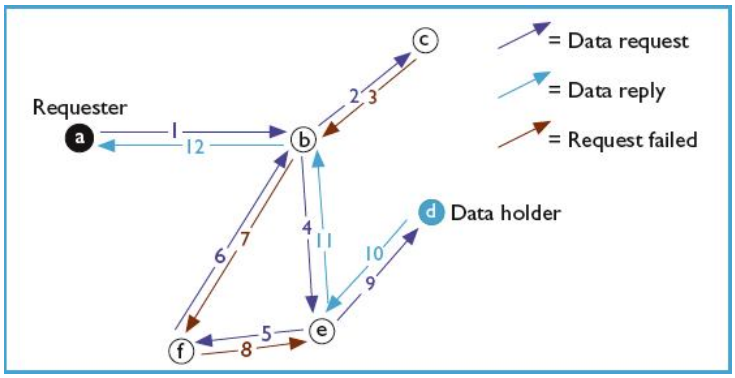
Os arquivos colocados na rede do Freenet não ficam arquivados na máquina da pessoa que os dispõe, mas sim em alguma outra máquina aleatória na rede.Quando a requisição por um certo arquivo cresce, ele é automaticamente copiado para os outros nós na vizinhança da sua localização. Através deste mecanismo o Freenet distribui por vários nós as informações mais populares na rede. Em contrapartida, como a quantidade de armazenamento é limitada, arquivos muito pouco

populares vão pouco a pouco dando lugar aos arquivos mais populares. Desta forma os arquivos menos populares tendem a desaparecer completamente da rede após um certo tempo.

Todos os arquivos, quando dispostos, são imediatamente encriptados com uma assinatura digital isso garante que ninguém será capaz de adulterar o seu conteúdo. Por isso, mesmo o dono de um nó, não sabe que tipo de informação está armazenada em sua máquina (isso evita que o dono de uma máquina participante seja responsabilizado por possuir informações).

Para garantir tudo o que promete os projetistas do Freenet tiveram que fazer algumas escolhas, que a princípio, podem parecer estranhas. Um exemplo é a pesquisa por dados. O Freenet usa uma arquitetura baseada em inundação controlada por TTL, assim como o Gnutella, entretanto a suas buscas são feitas em profundidade e não em largura - o que evita uma grande inundação de mensagens na rede (isto, no entanto, não o impede de responder a buscas em um tempo aceitável). Quando uma busca alcança algum nó da rede que possui a informação desejada, essa informação, diferentemente do Gnutella, volta pelo caminho por onde veio. Isso evita que o pesquisador seja identificado, já que em cada nó que a pesquisa passa, a sua origem é criptografada e enviada para o próximo vizinho. Este comportamento está representado na figura 4 [14].

Figura 4:



Seqüência tipica do funcionamento de uma requisição. O nó *a* quer um dado que está armazenado no nó *d*. A requisição sai de *a* e varre o grafo em profundidade (note que existe um mecanismo para evitar mensagens em laços infinitos no passo 7) até alcançar *d*. A resposta é então enviada a *a* através do mesmo caminho pela qual ela veio.

Como as mensagens são enviadas em profundidade, a escolha arbitrária de qualquer vizinho para o repasse de mensagens poderia deixar algumas buscas extremamente lentas. Para evitar este tipo de problema cada nó da rede mantém, juntamente com a lista de vizinhos, todos os identificadores dos arquivos que ele acha[2] que cada um dos vizinhos tem. Isso é feito através de *caching*. Assim, a primeira busca por um arquivo pode ser demorada, entretanto demais buscas pelo mesmo arquivo serão mais rápidas (o caminho, que provavelmente é o certo, já está demarcado).

O Freenet possibilitou [13]:

- A garantia do anonimato de seus participantes;
- Que nós, ainda que com uma pequena banda, distribuíssem documentos grandes e populares;
- Que os arquivos fossem distribuídos automaticamente para mais nós e para a parte da rede P2P na qual eles são mais requisitados. Na verdade o criador de um arquivo pode se preocupar menos com espaço para armazenamento e largura de banda. Ele traz a informação para perto de quem a deseja;

[2]O verbo *achar* foi empregado aqui propositalmente. O nó que guarda a lista não sabe ao certo se quem respondeu que tinha o arquivo procurado é o seu vizinho, ou algum outro nó que acabou sendo alcançado pela busca em profundidade através de seu vizinho. Para todos os efeitos, é como se o seu vizinho possuísse o dado, já que é dele que o dado virá.

- Arquivos populares se espalham pela rede, enquanto arquivos indesejados acabam desaparecendo silenciosamente. Isso é vantajoso pois isso previne que arquivos, que a maioria dos usuários considera como valiosos, sejam tirados da rede.

A fraqueza do Freenet está no seu mecanismo de busca. Assim como o Gnutella ele não traz garantias quanto a acessibilidade de um arquivo na rede, e além disto ele não é apropriado para buscas aleatórias por nomes (ou parte do nome) de arquivos. Se o fizesse então seria possível, para o operador de um nó, saber quais arquivos estão armazenados na sua máquina, isso por sua vez criaria a possibilidade de um de seus usuários ser responsabilizado pela posse dos mesmos. Entretanto é possível buscar arquivos pelo seu nome completo ou identificador (calculado aplicando a função de hash SHA-1 sobre o nome completo), ou seja, a busca pelo nome completo (e correto) dos arquivos deve ser feita de alguma outra forma fora da rede Freenet. Isso é feito, por exemplo, com a rede P2P BitTorrent [1] onde existem vários catálogos de arquivos disponíveis, veja por exemplo [10] [11].

2.3.3 Arquiteturas baseadas em redes de superposição

Um problema muito comum em aplicações P2P é como localizar eficientemente um nó que contém uma informação desejada. Vimos em 2.3.1 e 2.3.2 algumas maneiras de resolver este problema. A desvantagem do método apresentado em 2.3.1 é a sua grande suscetibilidade a ataques enquanto que a desvantagem apresentada em 2.3.2 é a falta de escalabilidade e a falta de garantias quanto a acessibilidade aos dados da rede. As redes de superposição (*overlay networks*) foram criadas para tentar preencher as lacunas deixadas pelas outras arquiteturas.

Redes de superposição como CAN [16], Chord [22], Pastry [21], Kademlia [20] e Viceroy [19] criam uma topologia virtual sobre a topologia física da rede[3]. A conexão entre os nós da rede Gnutella é feita de maneira arbitrária. Isto não acontece com redes de superposição (o Chord, por exemplo, utiliza o endereço IP das máquinas participantes para organizar a rede).

As redes de superposição têm as seguintes características:

- Garantia de acessibilidade dos dados
- Provas para limites superiores para o tempo de procura (geralmente O(log n) onde n é o número de nós da rede)
- Balanceamento automático de carga
- Auto organização

Essas características merecem uma melhor explicação. A conexão entre os nós da rede é feita com base no conteúdo de cada nó, ou seja, as conexões entre cada um dos nós podem ser expressadas em termos de funções matemáticas. Isso possibilita que a pesquisa por dados seja modelada através de um processo iterativo determinístico ao invés de uma busca baseada em inundação (por profundidade ou largura) no grafo de conexões. A análise deste processo permite, portanto, que sejam definidos os limites superiores para o tempo gasto em um processo de busca além da garantia de acessibilidade dos dados. Em termos abstratos podemos olhar uma rede de superposição como uma tabela de hash distribuída que suporta as operações de inserção, remoção e consulta de chaves. Tais chaves são, tipicamente, obtidas através de alguma função segura de hash (tal como o SHA-1). Um exemplo concreto deste esquema pode ser visto no exemplo mostrado abaixo.

[3]Neste sentido, redes P2P baseadas em técnicas de inundação controlada por TTL, também são redes de superposição, entretanto, utilizamos o termo aqui para nos referirmos apenas às redes que criam uma topologia virtual baseada nos atributos e arquivos de cada um dos nós.

Todas as redes de superposição citadas definem algoritmos de estabilização para tratar a entrada e a saída (sendo por falha ou não) de nós da rede. A responsabilidade destes algoritmos é manter a topologia (definida pelas funções matemáticas de cada uma das redes) intacta. Essas redes são muito úteis para a criação de aplicações que requerem buscas rápidas, escaláveis, confiáveis e auto organizadas por pares únicos de chave-valor.

Exemplo 1 - Chord O protocolo Chord define apenas uma operação: dada uma chave mapeá-la para um dado nó. O Chord, na verdade, é um arcabouço que aplicações P2P podem utilizar para efetuar as buscas e estruturar a rede.

Cada um dos nós possui uma identificação (ID). A identificação de um dado nó é obtida através da aplicação de uma função de hash no seu endereço IP. A rede é estruturada em forma de anel. Os nós são colocados neste anel em ordem crescente de ID[4]. Cada nó i guarda uma tabela de apontadores para outros O(log n) nós, onde n é o número de nós da rede. Esta tabela guarda os nós vizinhos de i. São eles: $i + 2^0$, $i + 2^1$, $i + 2^2$, $i + 2^3$ $\ldots i + 2^m$ onde m é o número de bits máximo que o ID pode assumir do ID (logo 2^m define o número máximo de nós que a rede suporta). Formalmente os vizinhos do nó i são $(i + 2^{k-1})$ mod 2^m, $1 < k < m$. A figura 5 mostra um exemplo de vizinhança.

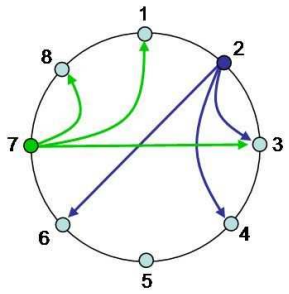


Figura 5: Vizinhança dos nós. As setas azuis apontam para os vizinhos do nó 2. As setas verdes apontam para os vizinhos do nó 7. Neste caso, $m = 3$, então, o número máximo de nós é $2^3 = 8$. Logo, cada nó tem apenas log(n) = m = 3 vizinhos na sua tabela de apontadores.

A busca pelo nó X com o identificador $ID(X)$ é muito simples. Primeiramente a tabela de apontadores é investigada para encontrar o nó Y tal que $ID(Y)$ é máximo e que $ID(Y) \leq ID(X)$. Se $ID(X) = ID(Y)$, o nó foi encontrado. Senão uma requisição é enviada para o nó Y. O nó Y repete os mesmos passos feitos acima. Até encontrar o nó desejado. Como a distância entre os apontadores cresce exponencialmente, em no máximo O(log(n)) passos o nó procurado vai ser encontrado ou será determinada a sua inexistência na rede. As figuras 6 e 7 ilustram este processo.

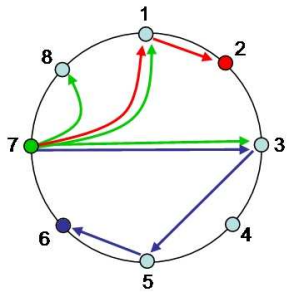


Figura 6: Buscas com sucesso por um nó. As setas verdes apontam para os vizinhos do nó 7. Em vermelho está indicado o caminho percorrido pelo processo para achar o nó com identificador 2 e em azul o caminho para se encontrar o nó com identificador 6.

[4]Por motivos de simplicidade vamos convencionar que a varredura do anel sempre se dá no sentido horário.

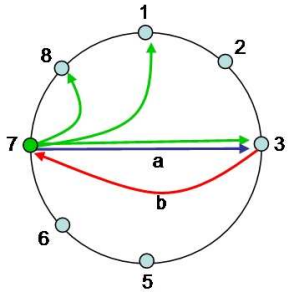


Figura 7: Buscas sem sucesso. As setas verdes apontam para os vizinhos do nó 7. Nó 7 tenta localizar o nó 4. O nó 4 não está na lista de apontadores do nó 7 e a pesquisa é repassada para o nó 3 (a). O menor apontador do nó 3 aponta para o nó 5 que é maior que o nó procurado. É devolvido como resposta (b) o maior nó que é menor que o nó procurado, no caso o nó 3.

Já falamos como os nós são localizados no Chord, falaremos agora como o Chord lida com o problema de busca de dados.

A localização dos dados no Chord é extremamente simples e por vezes até se confunde com o conceito de localização de um nó visto acima.

Para cada dado que for ser disposto na rede é calculado um ID. O cálculo deste ID é feito utilizando-se a mesma função de hash utilizada para calcular cada um dos IDs dos nós (portanto o espaço de IDs dos nós é o mesmo espaço dos IDs dos dados). Os dados disponíveis na rede, juntamente com a sua localização (tipicamente os endereços IP dos nós que o possuem), são guardados em uma estrutura semelhante a uma tabela de hash distribuída. Esta tabela é implementada da seguinte maneira: um nó é responsável por guardar uma lista dos arquivos cujos IDs são maiores ou iguais ao seu próprio ID e que são menores que o ID do primeiro nó da sua lista de apontadores (ou seja, ele guarda todos as localizações dos arquivos cujos IDs estão entre o seu próprio ID e o ID de seu primeiro vizinho). A figura 8 ilustra o processo de busca por dados em uma rede com 6 nós com o espaço de IDs de 3 bits ($m = 3$).

O Chord garante que os resultados da pesquisa são corretos ainda que apenas uma entrada da sua tabela de apontadores esteja correta. O desempenho do sistema degrada conforme o número de entradas incorretas na tabela de apontadores cresce. Como a tabela de apontadores é bem pequena (tipicamente O(log n) entradas) é muito fácil mantê-la atualizada através de algoritmos de estabilização. É o que o Chord faz.

Testes mostraram que o Chord foi capaz de manter o tempo de pesquisa em O(log n) mesmo com 50% de probabilidade de falhas por nó [15] [22].

O Chord, além de rodar os algoritmos de estabilização a intervalos regulares de tempo, também funciona com múltiplos nós virtuais por nó real. Isto ajudou no balanceamento de carga dos nós pois propiciou uma distribuição mais uniforme da DHT pela rede [15].

2.3.4 Comparação entre as arquiteturas

Aqui fazemos uma breve comparação entre as arquiteturas apresentadas. Abaixo cada um dos critérios utilizados juntamente com uma breve descrição:

Tolerância a falhas Quanto a rede é resistente a falhas e ataques. A remoção de algum nó especial traz grandes consequências a rede?

Escalabilidade A rede consegue suportar um grande número de usuários?.

Pesquisa de dados Suporta pesquisa por palavras chave e coringas? É bom para pesquisas aleatórias? Tem garantias quanto a acessibilidade dos dados?

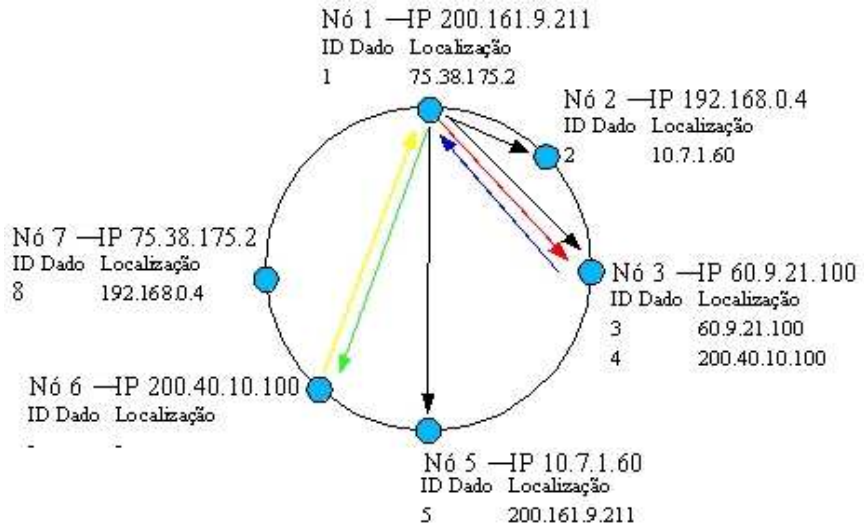


Figura 8: Busca por um dado na rede. Em cada um dos nós está indicado o seu identificador seguido pelo seu endereço IP (fictício). Logo abaixo destas informações está a parte da tabela de hash distribuída que compete a este nó. O nó 3, por exemplo, guarda as informações da localização dos arquivos com ID 3 e 4, já o nó 6 não tem informação nenhuma a respeito da localização de qualquer dado. A figura representa a busca efetuada pelo nó com ID 1 por um dado com ID 4. As setas negras indicam os vizinhos do nó 1. O processo de busca se dá da seguinte maneira: a. o nó 1 localiza na sua lista de apontadores o nó cujo ID é o mais próximo do ID procurado (4), neste caso o nó 3. b. O nó 1 envia uma requisição de busca ao nó 3 (seta vermelha). c. O nó 3 verifica que ele é o responsável pelo dado de ID 4 e que ele tem a sua localização, ele envia uma mensagem de volta ao nó 1 com o endereço do nó que possui o dado desejado (seta azul). d. O nó 1 com posse do endereço do nó 6 faz uma conexão direta e requisita o envio da informação (seta verde). e. O nó 6 responde a requisição enviando o dado requisitado (seta amarela).

Anonimato Tanto quem requisita quanto quem fornece a informação consegue manter a sua identidade em sigilo?

As notas variam de 0 a 3. Notas maiores são melhores.

	Tolerância a falhas	Escalabilidade	Pesquisa de dados	Anonimato
Centralizado	0	2	3	0
Descentralizado	1	3	3	0
Distribuído Redes de inundação	3	1	2 1 no caso do Freenet	3 usando Freenet 1 usando proxies [13]
Distribuído Redes de superposição	2	3	1	0

Referências

[1] Bittorrent. http://bitconjurer.org/BitTorrent/.

[2] E-donkey. http://www.edonkey2000.com.

[3] Fasttrack. http://www.fasttrack.nu.

[4] Freenet. http://freenetproject.org/.

[5] Gnutella. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[6] Integrade. http://gsd.ime.usp.br/integrade/.

[7] Kazaa. http://www.kazaa.com.

[8] Napster. http://www.napster.com.

[9] Seti@home. http://setiathome.ssl.berkeley.edu/.

[10] Suprnova. http://www.suprnova.org/.

[11] Torrent reactor. http://www.torrentreactor.net/.

[12] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking up data in p2p systems. *Communications of the ACM*, 46(2):43–48, February 2003.

[13] Albert Benschop. Peer-to-peer: Networks of unknown friends - the power of sharing. March 2004.

[14] Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, pages 40–49, February 2002.

[15] Diego Doval and Donal O'Mahony. Overlay networks - a scalable alternative for p2p. *IEEE Internet Computing*, pages 79–82, August 2003.

[16] S. Ratnasamy et al. A scalable content-addressable network. *Proc. ACM SIGCOMM*, pages 161–172, 2001.

[17] J. et al. Kubatowicz. Oceanstore: An architecture for global-scale persistent storage. *Proceedings of Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, 2000.

[18] John Kubiatowicz. Extracting guarantees from chaos. *Communications of the ACM*, 46(2):33–38, February 2003.

[19] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. *Proc. SIGCOMM, ACM Press*, pages 183–192, 2002.

[20] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. *Proceedings of the Twenty-First ACM Symposium on Principles of Distributed Computing (PODC 2002)*, pages 108–117, July 2002.

[21] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *Middleware'2001, Germany*, November 2001.

[22] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *Proc. ACM SIGCOMM*, pages 149–160, 2001.