Arquiteturas Peer-to-Peer

Bruna Almeida Osti
19 de Outubro de 2020

Resumo

Neste trabalho serão abordados conceitos a respeito da arquitetura Peer-to-Peer, no qual discutiremos a estrutura da arquitetura, suas vantagens e desvantagens, desafios e aplicações. Além disso, analisaremos suas principais diferenças quando comparada com a arquitetura Cliente-Servidor.

1 Introdução

A computação distribuída tem como objetivo lidar com diferentes formas de computação, acesso da informação e troca de informação entre diferentes plataformas usando a internet (1). Um dos tipos de sistemas mais utilizados é conhecido como Peer-to-Peer (P2P), um paradigma para a construção de aplicativos distribuídos de vários hosts de maneira uniforme que evita sistemas centralizados, ou seja, todos os pares são ao mesmo tempo servidores e clientes (2).

As arquiteturas peer-to-peer buscam distribuir o uso do recurso entre vários hosts mantendo a carga de equilíbrio de cada rede. Os aplicativos neste tipo de arquitetura são usados para fornecer compartilhamento de arquivos, uso de cache web, distribuição de informações entre outros serviços que exploram os recursos de milhares de máquinas na internet (2).

De acordo com vários provedores de internet, mais da metade das conexões da internet são de conexões do tipo Peer-to-Peer. Isso acontece devido ao crescimento contínuo da quantidade de aplicações e dos usuários, portanto a quantidade de recursos requeridos está cada vez maior (3). Além disso, há alguns requisitos fundamentais para o pleno funcionamento das aplicações baseadas em internet, como: escalabilidade; segurança e confiabilidade; flexibilidade e qualidade do serviço.

O uso desse tipo de arquitetura ficou conhecido depois da criação do Napster (1999) que era um serviço de troca de músicas, no qual fornecia um meio para os usuários compartilharem suas músicas. Entretanto, a aplicação feria os direitos autorais dos arquivos distribuídos e foi desativado depois de várias ações jurídicas contra os operadores do serviço (2).

Neste trabalho serão abordados conceitos essenciais para o entendimento das arquiteturas peer-to-peer, no qual discutiremos a estrutura da arquitetura, suas vantagens e desvantagens, desafios e aplicações. Além disso, analisaremos suas principais diferenças quando comparada com a arquitetura de cliente-servidor.

2 Peer to Peer (P2P)

As arquiteturas peer-to-peer utilizam um sistema de mapeamento dos nós chamado de rede sobreposta (*rede overlay*), no qual os nós são formados virtualmente por cima de uma rede física já existente. Esse tipo de prática passou a ser comum dado a disponibilidade de equipamentos de comutação digital. Por outro lado, existem dois tipos de rede sobrepostas: as estruturadas e as não estruturadas (4).

Segundo Dejan S. Milojicic et al. (5), algumas características desse tipo de arquitetura são:

- **Auto-organização:** não há um coordenador do organização, não há um coordenador do grupo, toda a coordenação é distribuída;
- **Descentralização:** Em um modelo descentralizado, cada nó é um participante igualitário. O que torna o sistema tolerante a falhas, já que não existe apenas um ponto de falha. Na prática isso torna o sistema mais complexo de ser implementado, este é o motivo para a maioria das aplicações optarem por um modelo híbrido;
- **Escalabilidade:** É um benefício decorrente da descentralização. Pois não depende do desempenho de apenas um servidor, e por isso conseguem manter mais usuários simultâneos; é um requisito necessário para suprir a demanda de recursos (largura de banda, capacidade de armazenamento e potência de processamento causado

pela grande quantidade de usuários). Dessa forma, é possível evitar gargalos e prováveis perdas de eficiência ao escalar o serviço (3).

- **Anonimato:** Permite os usuários utilizarem o sistemas sem preocupações legais, e sem possibilidade de censura. Existem três tipos de anonimato: Por parte do remetente, pelo destinatário e mútuo;
- **Custo de propriedade e performance:** É possível obter uma enorme performance por um custo menor, utilizando os computadores de seus usuários;
- **Conectividade ad-hoc:** A configuração da rede em um certo momento é difícil de ser prevista, pois a computação distribuída não será executada em todos os nós participantes o tempo todo. Alguns não estarão disponíveis em alguns momentos, enquanto outros estarão. É preciso estar cientes desse comportamento ad-hoc e lidar com sistemas constantemente mudando a lista de nós disponíveis.

Entretanto, apesar da grande quantidade de benefícios os desafios encontrados na sua implementação são proporcionais.

2.1 Arquiteturas P2P estruturados

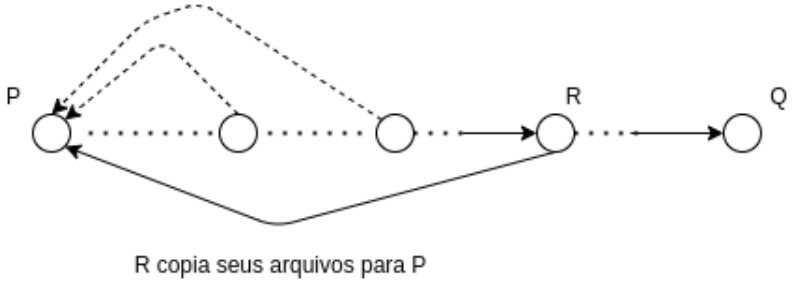
Numa rede peer-to-peer estruturada, os nós são organizados em um grafo estruturado. Através do grafo podemos descobrir eficientemente onde estão os dados pela rede sobreposta. Entretanto, uma rede simples não dá suporte a buscas complexas e portanto é necessário guardar uma cópia ou ponteiro para cada objeto existente nos nós (4).

O procedimento mais utilizado, é organizar os processos através de uma tabela hash distribuída (DHT). Neste caso, os dados recebem uma chave aleatória de um espaço de identificadores, o grande desafio é implementar um sistema eficiente que mapeia unicamente a chave de um item para o identificador responsável pelo item desejado. Sendo possível o retorno do endereço do nó responsável pelo item (4).

Por outro lado, a replicação é disponibilizada para balancear a carga entre os nós da rede. Um método muito utilizado é replicar o arquivo ao longo do caminho desde a fonte do arquivo distribuindo a carga entre os nós próximos, o que alivia a carga do nó responsável por disponibilizar o arquivo. Entretanto, esse tipo de sistema não leva em consideração a carga dos outros nós, o que pode gerar desequilíbrio na rede (6).

Para evitar esses problemas foi proposto um sistema que leva em conta a carga atual dos outros nós ao longo da rota. O objetivo principal é distribuir ponteiros em cache para os outros nós, se o nó estiver sobrecarregado ele pode pedir que o nó destino instale réplicas dos arquivos mais requisitados como mostra a Figura 1. Se o nó destino puder manter uma cópia do arquivo, todos os nós intermediários terão um ponteiro para o destino, indicando que há uma cópia do arquivo neste lugar.

Figura 1: Balanceamento de Carga P2P (6)



2.2 Arquiteturas P2P não estruturados

Os sistemas não estruturados geralmente se baseiam em algoritmos aleatórios para a construção da rede sobreposta. Geralmente suas consultas são feitas através de broadcast, que pode ser retransmitida entre os nós, inundando a rede e prejudicando o desempenho do sistema (6).

Além disso, esse tipo de consulta pode não ser respondida, pois há mecanismos que impedem que mensagens se repliquem infinitamente na rede através de mecanismos como TTL (Time to live) que faz o controle de por quantos nós o pacote já passou, e o descarta caso tenha passado por um número limite de nós (7).

Entretanto, se os arquivos forem replicados para todos os nós a busca seria muito rápida, mas os computadores tem capacidade limitada. Portanto, a replicação completa fica fora de questão e estratégias com um certo grau de otimização são buscadas (6).

A replicação nesses tipos de sistema geralmente acontece quando usuários fazem o uso de arquivos de outros usuários, e logo em seguida os disponibilizam para a comunidade. Dentro desse tipo de arquitetura (6).

Segundo a UFRJ (8) existem três tipos de arquiteturas não estruturadas:

- **P2P pura:** É completamente descentralizada, todos os nós possuem papel equivalente o que torna seu sistema de busca por inundação (broadcast). Entretanto, esse tipo de abordagem é rara. Exemplo: Gnutella, Freenet.
- **P2P híbrida:** Existem supernós que concedem o acesso a nós da rede e liberam a busca por esses recursos. A falha em um supernó é facilmente tolerada elegendo outro supernó. Exemplo: Kazza e BitTorrent.
- **P2P centralizada:** São redes híbridas, que possuem servidores centrais que controlam as entradas e saídas dos nós. Os nós registram os recursos que vão compartilhar na rede com o servidor central. O servidor disponibiliza banda e processamento e direciona os recursos entre os pares (transferência de dados é P2P). Exemplo: eMule, Napster.

2.3 Segurança

Por se tratar de um sistema de compartilhamento de arquivos descentralizado há brechas para alguns tipos de ataques. Além dos riscos já conhecidos pela arquitetura cliente-servidor, como mascarar arquivos maliciosos com nomes de arquivos inofensivos (ex: malware com o nome “Harry Potter e a Pedra Filosofal”), a arquitetura Peer-to-Peer dificulta o combate a pirataria. Sendo preciso monitorar cada rede Peer-to-Peer e processar diretamente os usuários que compartilham esse tipo de conteúdo (9).

Além disso, como cada nó da aplicação também possui funções de servidor, a vulnerabilidade a ataques remotos aumenta. Problemas de segurança nas aplicações podem levar a investidas maliciosas, incluindo ataques de roteamento (ex: encaminhar pedidos incorretamente) e de negação de serviço (*denial of service*) (2).

2.4 Middlewares

Um problema muito recorrente das aplicações peer-to-peer é o fornecimento de recursos de dados rapidamente e com segurança quando os dados estão espalhados por toda a rede. Os middlewares surgiram com a ideia de atender a necessidade de disposição automática e localização dos objetos distribuídos (2).

Portanto, a função de um middleware para aplicações peer-to-peer é de simplificar a implementação de sistemas em muitos hosts. Para isso, é necessário que os clientes se localizarem e se comunicarem com os recursos individuais disponibilizados para o serviço, mesmo que estes estejam distribuídos entre muitos hosts. Além disso, é necessário que o sistema esteja preparado para a adição ou remoção de hosts no sistema, pois é bem dinâmico e não há como prever quais hosts estarão disponíveis. (2).

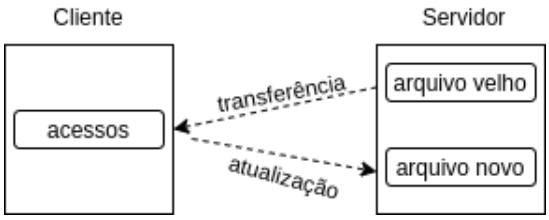
Por outro lado, o middleware deve prover: escalabilidade global, balanceamento de carga, otimização das interações locais entre peers vizinhos, lidar com a disponibilidade dinâmica dos hosts, segurança dos dados, anonimato capacidade de negação e resistência a censura. Portanto, para a camada de middleware suportar sistemas com escalabilidade global é muito complicado, pois torna-se praticamente impossível manter um banco de dados com a localização de todos os recursos existentes (2).

3 Arquitetura Cliente-Servidor x P2P

A diferença entre os sistemas é bem nítida pelas peculiaridades em cada forma de gerenciamento dos recursos, como mostra a Tabela 1. Portanto, os sistemas cliente servidor tem mais chances de ter apenas um ponto de falha como

descrito na Figura 2, e requerem muitos recursos pois a informação está concentrada. Por outro lado, o sistema peer-to-peer não garante disponibilidade visto que várias pessoas apenas liberam o computador pelo tempo de *download* de um arquivo (6).

Figura 2: Modelo de Carga Cliente-Servidor (6)



Em ambos os casos há vantagens e desvantagens, no caso da arquitetura cliente-servidor, a manutenção é muito mais simples pois há a possibilidade de troca de servidores caso algum dê problema, além da segurança aos dados (garantir que apenas as pessoas com permissão podem acessar os dados). Entretanto, se o número de requisições é muito alto, o servidor pode ficar sobrecarregado (6) (2).

Por outro lado, apesar do sistema peer-to-peer ser mais difícil de ser administrado devido a falta de centralidade, sacrificando muitas vezes a segurança, esse tipo de sistema oferece um custo menor com rede e a simplicidade do funcionamento. Além disso oferece vários pontos de falha e não apenas um, o que faz com que o sistema seja mais robusto e esteja sempre disponível (6).

A Tabela 1 explicita as principais diferenças entre as abordagens P2P e Cliente-Servidor, é importante observar que as duas arquiteturas são válidas, apenas é válido a utilização de cada uma em aplicações com fins específicos, afim de aumentar a eficiência do sistema.

Tabela 1: Diferença entre duas abordagens de sistemas: P2P x Cliente-Servidor (8)

Peer-to-Peer	Cliente-Servidor
A informação está distribuída	A informação está concentrada
Conecta-se a partir da rede	Conecta-se através de um servidor
Transferência de informações ativa, pois possui vários nós	Pode ter um ponto de falha se o sistema não estiver operando
Não requer servidores e clientes fixos	Requer um servidor e um cliente pelo menos
Todos os nós são iguais e podem ter as mesmas funções	As funções são limitadas para o servidor e para cada cliente

4 Aplicações

Nesta seção abordaremos a estrutura de cada uma das aplicações e seu método de funcionamento. Serão abordados arquiteturas P2P dos tipos: pura, híbrida e centralizada.

4.1 Gnutella e FreeNet

Nesta seção ambas as aplicações (Gnutella e FreeNet) são arquiteturas p2p do tipo pura, pois tem os seus sistemas completamente descentralizados e fazem que seus usuários sejam tanto clientes como servidores.

O Gnutella foi desenvolvido pela empresa Nullsoft (2000) com a finalidade de ser um protocolo de comunicação para compartilhar arquivos. Por ser descentralizado oferece anonimidade aos usuários, o que também causa um certo grau de incerteza. Para buscar um arquivo, o usuário deve fazer uma consulta a uma página que lista os usuários e seus respectivos endereços IP. Depois, usando esses endereços é feito uma consulta para cada usuário perguntando se estes possuem o arquivo desejado. Os usuários que possuem o arquivo respondem com os resultados;

caso contrário, encaminham a consulta a outros nós. Funciona de forma parecida ao algoritmo de inundação mas existe um mecanismo de Time-to-Live que define até aonde a consulta será encaminhada (10).

De mesma forma, o FreeNet (2000) é um software gratuito que permite compartilhar arquivos anonimamente, navegar e publicar "freesites"(sites acessíveis apenas através da Freenet) e conversar em fóruns, sem medo de censura pois as informações são criptografadas e roteadas por meio de outros nós para tornar extremamente difícil determinar quem está solicitando as informações e qual é o seu conteúdo (11).

4.2 Kazza e BitTorrent

Nesta seção ambas as aplicações (Kazaa e BitTorrent) são arquiteturas p2p do tipo híbrida, pois tem supernós que concedem o acesso a nós da rede e liberam a busca por esses recursos. A falha em um supernó é facilmente tolerada elegendo outro supernó.

O Kazaa (2000) é um serviço de compartilhamento de arquivos, no qual permite a troca de música, imagens e outros arquivos do gênero. Muito perseguido pela RIAA, a associação das gravadoras norte-americanas, pela troca de arquivos ilegais (geralmente canções protegidas por direitos de autor). O seu criador também é autor dos programas Skype e Joost (12).

O BitTorrent (2001) foi criado pela BitTorrent, inc e têm sido utilizado para o compartilhamento de arquivos pela rede, principalmente filmes, álbuns, jogos eletrônicos e programas de computador (13).

4.3 eMule, Napster

Nesta seção ambas as aplicações (eMule e Napster) são arquiteturas p2p centralizadas, são redes híbridas, que possuem servidores centrais que controlam as entradas e saídas dos nós. Os nós registram os recursos que vão compartilhar na rede com o servidor central. O servidor disponibiliza banda e processamento e direciona os recursos entre os pares (transferência de dados é P2P).

O eMule (2002) é um serviço de compartilhamento de arquivos, e seus diferenciais são a troca de links entre os clientes, rápida recuperação de downloads corrompidos e o uso de um sistema de créditos para premiar os usuários que fazem mais uploads (14).

Por outro lado, o Napster (1999) é um serviço de streaming de música pertencente a Rhapsody Internatonal Inc, anteriormente era um serviço de troca de músicas, no qual fornecia um meio para os usuários compartilharem suas músicas. Entretanto, a aplicação feria os direitos autorais dos arquivos distribuídos e foi desativado depois de várias ações jurídicas contra os operadores do serviço (2).

5 Considerações finais

Neste trabalho podemos observar a importância das arquiteturas peer-to-peer para aplicações que demandam de muitos recursos para conseguir ser escalável, fazendo com que vários hosts forneçam recursos de um modo que mantenham a carga de equilíbrio de cada rede. Essa arquitetura é utilizada para fornecer compartilhamento de arquivos, uso de cache web, distribuição de informações entre outros serviços que exploram os recursos de diversas máquinas pela rede.

Há muitas vantagens em manter um sistema com esse tipo de arquitetura, pois ela fornece auto-organização, decentralização de recursos e consequentemente vários pontos de falha, escalabilidade, anonimato, custo de propriedade e performance mais baratos e conectividade ad-hoc.

Geralmente é um grande facilitador de atividades ilícitas, visto que garante anonimato. Portanto, as pessoas utilizam muito para compartilhamento de músicas, filmes, livros e programas que não tenham devida autorização. Várias aplicações acabam tendo problemas judiciários por permitirem esse tipo de compartilhamento que fere os direitos autorais.

Entretanto a arquitetura peer-to-peer fornece diversos benefícios perante a arquitetura cliente-servidor na qual exige uma quantidade bem maior de recursos, e tem apenas um ponto de falha. Sendo assim, se o servidor falhar os clientes não terão acesso por tempo indeterminado ao servidor, até que seja substituído ou normalizado.

Referências

[1] A. Kshemkalyani, *Distributed Computing : Principles, Algorithms, and Systems*. Leiden: Cambridge University Press, 2008.

[2] G. Coulouris, *Distributed systems : concepts and design*. Harlow, England New York: Addison-Wesley, 2005.

[3] R. Steinmetz, *Peer-to-peer systems and applications*. Berlin New York: Springer, 2005.

[4] P. S. L. Eng, Crowcroft, "A survey and comparison of peer-to-peer overlay network schemes," Acesso em: https://www.cl.cam.ac.uk/research/dtg/www/files/publications/public/mp431/ieee-survey.pdf. Acessado: 3 out. 2020.

[5] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-peer computing," 04 2002.

[6] A. Tanenbaum, *Sistemas distribuídos: Princípios e Paradigmas*. São Paulo: Pearson Educación, 2008.

[7] J. Kurose, *Redes de computadores e a internet : uma abordagem top-down*. São Paulo: Pearson, 2013.

[8] UFRJ, "Peer-to-peer model," out 2020. Acesso em: https://www.gta.ufrj.br/ensino/eel878/redes1-2018-1/trabalhos-v1/p2p/arquitetura.html. Acessado: 7 out. 2020.

[9] Dunamys, "Cuidado com arquivos perigosos," jun 2019. Acesso em: https://www.dunamys.inf.br/cuidado-com-arquivos-perigosos/. Acessado: 7 out. 2020.

[10] UFRJ, "Gnutella," out 2020. Acesso em: https://www.gta.ufrj.br/grad/06_1/p2p/gnutella.html. Acessado: 6 out. 2020.

[11] freenetproject, "O que é freenet?," out 2020. Acesso em: https://freenetproject.org/pages/about.html. Acessado: 6 out. 2020.

[12] Freenet, "Riaa wins battle to id kazaa user," out 2020. Acesso em: https://www.cnet.com/news/riaa-wins-battle-to-id-kazaa-user/. Acessado: 6 out. 2020.

[13] bittorrent, "Bittorrent web," out 2020. Acesso em: https://www.bittorrent.com/. Acessado: 6 out. 2020.

[14] emule, "O que é o emule?," out 2020. Acesso em: http://www.emule-project.net/home/perl/general.cgi?l=30. Acessado: 6 out. 2020.