Understanding Peer-to-Peer Technology

Choopan Rattanapoka*

College of Industrial Technology, King Mongkut's University of Technology North Bangkok
1518 Pibulsongkram Rd., Bangsue, Bangkok, 10800, Thailand

Abstract

Peer-to-peer (P2P) networks such as Gnutella and Bittorrent have revolutionized Internet-based applications. A peer-to-peer network offers high-availability, high-reliability, and fault tolerance. The use of a peer-to-peer network can be applied to many kinds of applications. It is not only limited to file sharing applications. Many projects have adopted peer-to-peer networks for their applications, such as SETI@home, a CPU sharing application, Oceanstore, a storage sharing application, and OverCite, a distributed version of the Citeseer digital library. An understanding of peer-to-peer technology will allow us to adapt the existing client/server model applications into a peer-to-peer model which may give us an alternative solution that is better than the existing one.

Keywords: Peer-to-Peer topology, Peer-to-Peer network, Peer-to-Peer computing

บทคัดย่อ

เครือข่ายเพียร์ทูเพียร์ เช่น กนูเทลล่าและบิตเทอร์เรนได้ปฏิวัติวงการของแอพพลิเคชั่นบนอินเทอร์เน็ต เนื่องด้วยเครือข่ายเพียร์ทูเพียร์มีทรัพยากรบนเครือข่ายอยู่มากและมีความเสถียรภาพของเครือข่ายสูง อีกทั้งเครือข่ายยังมีความทนทานสูงต่อความผิดพลาดที่เกิดขึ้นในระบบ การใช้งานของเครือข่ายเพียร์ทูเพียร์สามารถนำไปประยุกต์ได้หลายอย่างไม่ได้จำกัดอยู่เพียงแต่การแบ่งปันแฟ้มข้อมูล เช่น เซติแอทโฮมที่เป็นแอพพลิเคชั่นสำหรับแบ่งปันหน่วยประมวลผลกลางเพื่อร่วมกันประมวลผล โอเชียนสโตร์ที่เป็นแอพพลิเคชั่นสำหรับแบ่งปันเนื้อที่ในการเก็บข้อมูล หรือโอเวอร์ไซต์ที่เป็นการพัฒนาต่อยอดจากห้องสมุดดิจิตอลไซต์เซียร์ให้อยู่ในรูปของการเก็บข้อมูลแบบกระจายบนเครือข่ายเพียร์ทูเพียร์ การที่เรามีความรู้และเข้าใจกับเทคโนโลยีเพียร์ทูเพียร์มากขึ้นจะทำให้สามารถมองเห็นทางเลือกใหม่ ๆ ซึ่งอาจจะสามารถนำมาทดแทนแอพพลิเคชั่นที่ใช้งานกันอยู่ในปัจจุบันให้มีประสิทธิภาพมากขึ้น

Introduction

Most network applications are based on a client/server model. A client/server model assumes a client which handles the main program sequence of instructions, and initiates requests to a server when needed. This model is characterized by its communication scheme, which is typically limited to communication between clients

* *Corresponding author.*
E-mail address: choopanr@kmutnb.ac.th

and servers. Thus, when clients become numerous, a problem arises on the server side when it becomes a bottleneck for service. The cost of upgrading a server side machine to support client requests is high. In the past decade, several projects tried to use the computing power of clients to solve the bottleneck problem of the server. Peer-to-Peer is one of the key technologies that solves the server bottleneck problem. Peer-to-peer or P2P is not only limited by the idea of file sharing applications. Popular examples of peer-to-peer file sharing networks are Gnutella, and Bittorrent. Generally, a peer-to-peer computer network refers to any network that does not have fixed clients and servers, but a number of peer nodes that function as both clients and servers to the other nodes in the network. This model of network arrangement is contrasted with the client/server model. Any node is able to initiate or complete any supported transaction. Peer nodes may differ in local configuration, processing speed, network bandwidth, and storage quantity. A peer-to-peer network can be formed by using a different topology. These topologies refer to how each peer in the network is linked to others.

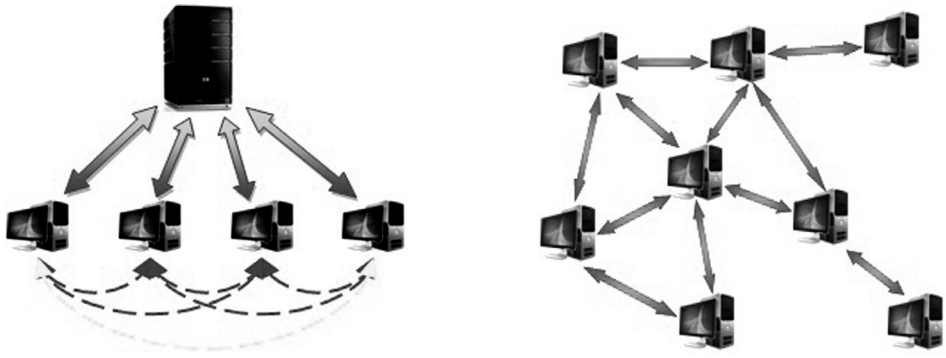
Peer-to-Peer Topologies

The peer-to-peer model can be divided into several topologies. Schollmeier (2001) divides the peer-to-peer networks into two topologies. Figure 1 shows two main topologies of the peer-to-peer model: (a) a centralized topology, and (b) a decentralized topology.

A Centralized Topology

The centralized systems or hybrid peer-to-peer systems are the most familiar form of topology, typically seen as the client/server pattern used by databases, web servers, and other simple distributed systems. All functions and information are centralized into one server with many clients connecting directly to the server in order to send and receive information. Many applications called *peer-to-peer* also have a centralized component. SETI@Home (Anderson, 2001) is a fully centralized architecture with the job dispatcher as the server. Similarly, the original Napster's search architecture (Shirky, 2001) was centralized, although the file sharing was not. The advantage of this topology is that searching other peers or services that other peers provide is efficient since the centralized server maintains all the information.

However, the disadvantage is the bottleneck of the centralized server.



(a) a centralized topology (b) a decentralized topology

Figure 1. Two main peer-to-peer topologies.

A Decentralized Topology

Decentralized systems or pure peer-to-peer systems are where all peers communicate symmetrically and have equal roles. Gnutella (Kan, 2001) is probably the purest decentralized system used in practice today, with only a small centralized function to bootstrap a new host. Many other file-sharing systems are also designed to be decentralized, such as Freenet or OceanStore. There is no bottleneck in this topology because there is no special centralized server. We can categorize the decentralized P2P network into two types of system: unstructured peer-to-peer systems and structured peer-to-peer systems.

Unstructured peer-to-peer systems are formed when the overlay links are established arbitrarily. The networks can be easily constructed. The most popular query method is *flooding*. If a peer wants to find a desired piece of data in the network, the query has to be flooded through the network to find as many peers as possible that share the data. This system is highly reliable and supports range queries and wild-card queries. However, the main disadvantages of the flooding method are the bandwidth usage and the rare piece of data in the system may not be discovered by some peers.

Structured peer-to-peer systems ensure that any node can efficiently route a search to some peer that has the desired piece of data in the network, even if the piece of data is rare. Most systems use a distributed hash table (DHT) that provides a look-up service similar to a hash table. The (key, value) pairs are stored in the DHT, and any participating node can efficiently retrieve the value associated with a given key. This system provides good performance for retrieving data from the network and the load-balancing. However, the cost of maintaining overlay links is quite high.

Peer-to-Peer Infrastructure

The peer-to-peer infrastructure is the main key to develop peer-to-peer applications. The peer-to-peer infrastructures should provide some basic facilities such as:

- joining the peer-to-peer network
- discovering the other peers
- self-configuring and insuring robustness
- providing application scalability.

The research projects in this domain aim to develop peer-to-peer infrastructures that are easy for users to work with and also provide some more advantageous features over other implementations. The list below gives some descriptions of existing projects working on the improvement of peer-to-peer infrastructure.

CAN (Ratnasamy *et al.*, 2001) or the Content Addressable Networks work is being done at AT&T Center for the Internet Research at ICSI (ACIRI). In the CAN model, nodes are mapped onto a N–dimensional coordinate space on top of TCP/IP. The space is divided up into N-dimensional blocks based on server density and load information, where each block keeps information on its immediate neighbors. Because addresses are points inside the coordinate space, each node simply routes to the neighbor which makes the most progress towards the destination coordinate. Object location works by the object server pushing copies of location information back in the direction of the most frequent incoming queries.

Chord (Stoica *et al.*, 2001) aims to build scalable, robust distributed systems

using peer-to-peer ideas. The basis for much of its work is the Chord distributed hash look-up primitive. Chord is completely decentralized and symmetric, and can find data using only log (N) messages, where N is the number of nodes in the system. Chord's look-up mechanism is provably robust in the face of frequent node failures and re-joins. Some research projects take advantage of Chord to distribute the load of serving data widely to achieve high performance despite flash crowds such as CFS (Cooperative File System) (Dabek *et al.*, 2001) which allows anyone to publish and update their own file system, and provides read-only access to others, and OverCite (Stribling *et al.*, 2005) which is a distributed version of the CiteSeer digital library.

JXTA (Traversat *et al.*, 2003) is a set of open, generalized peer-to-peer protocols, defined as XML messages. Using the JXTA protocols, peers can cooperate to form self-organized and self-configured peer groups independently of their positions in the network and without the need of a centralized management infrastructure. Peers may use the JXTA protocols to advertise their resources and to discover available network resources (service, pipes, etc.) from other peers. Peers form and join peer groups to create special relationships. Peers cooperate to route messages allowing for full peer connectivity. The JXTA protocols allow peers to communicate without needing to understand or manage the potentially complex and dynamic network topologies which are becoming common. Many research projects use JXTA as their peer-to-peer infrastructure such as P3 (Kazuyuki *et al.*, 2005) which is middleware for distributed computing using volatile PCs, in which participants provide and also use other's computers.

Pastry (Rowstron and Druschel, 2001) is a generic, scalable and efficient substrate for peer-to-peer applications. Pastry nodes form a decentralized, self-organizing and fault-tolerant overlay network within the Internet. Pastry provides efficient request routing, deterministic object location, and load balancing in an application-independent manner. Furthermore, Pastry provides mechanisms that support and facilitate application-specific object replication, caching, and fault recovery. Peer-to-peer applications based on Pastry such as SQUIRREL (a decentralized, peer-to-peer web cache), PAST (a large-scale, persistent peer-to-peer storage utility), SCRIBE (a large-scale and decentralized application-level multicast infrastructure), and SplitStream (a high-bandwidth content distribution in a cooperative environment).

Peer-to-Peer Applications

Whereas some research projects are interested in developing peer-to-peer infrastructure that can support general types of applications, there are some research projects which try to tackle peer-to-peer application directly and even propose their own frameworks together with their application such as Bittorrent, BOINC, P2P-MPI, and Zorilla.

Bittorrent (Cohen, 2003) allows users to distribute large amounts of data without the heavy demands on their computers that would be needed for standard Internet hosting. A standard host's servers can easily be brought to a halt if high levels of simultaneous data flow are reached. The protocol works as an alternative data distribution method that makes even small computers (e.g. mobile phones) with low bandwidth capable of participating in large data transfers. The mechanism of Bittorrent can be described as follows: First, a user playing the role of file-provider makes a file available to the network. This first user's file is called a seed and its availability on the network allows other users, called *peers*, to connect and begin to download the seed file. As new peers connect to the network and request the same file, their computer receives a different piece of the data from the seed. Once multiple peers have multiple pieces of the seed, BitTorrent allows each to become a source for that portion of the file. The effect of this is to take on a small part of the task and relieve the initial user, distributing the file download task among the seed and many peers. With BitTorrent, no one computer needs to supply data in quantities which could jeopardize the task by overwhelming all resources, yet with the same final result. After the file is successfully and completely downloaded by a given peer, the peer is able to shift roles and become an additional seed, helping the remaining peers to receive the entire file. This eventual shift from peers to seeders determines the overall *health* of the file (as determined by the number of times a file is available in its complete form).

BOINC (Anderson, 2004) or Berkeley Open Infrastructure for Network Computing is a non-commercial middleware system for volunteer and grid computing. It was originally developed to support the SETI@home project before it became useful as a platform for other distributed applications in areas as diverse as mathematics,

medicine, molecular biology, climatology, and astrophysics. The intent of BOINC is to make it possible for researchers to tap into the enormous processing power of personal computers around the world. BOINC has about 589,000 active computers (hosts) worldwide processing on average 4.931 petaFLOPS as of April 10th 2010[1]. The framework is supported by various operating systems, including Microsoft Windows, Mac OS X and various Unix-like systems including Linux and FreeBSD. In essence, BOINC is software that can use the unused CPU and GPU cycles on a computer to do scientific computing, what one individual does not use of his/her computer, BOINC uses. In late 2008, BOINC's had applications that run on NVIDIA GPUs using CUDA. Beginning in October 2009, BOINC added support for the ATI/ AMD family of GPUs also. These applications run from 2X to 10X faster than the former CPU-only versions. BOINC consists of a server system and client software that communicate with each other to distribute, process, and return workunits.

P2P-MPI (Genaud and Rattanapoka, 2007) is a middleware aimed to execute parallel programs, in particular in a grid computing environment. The P2P-MPI design adheres to the peer-to-peer paradigm: when a peer requests the execution of a parallel computation, the middleware will search for peers to form a group of volunteering computers to perform the computation. This dynamic selection of hosts before runtime is well suited to volatile environments such as grids. Similar to peer-to-peer file sharing systems, a P2P-MPI user must agree to share its CPU before being able to request the use of other peer CPUs. P2P-MPI enables the development of message passing parallel programs: the system provides a communication library in Java following the MPJ specification, which is basically MPI for Java. P2P-MPI also integrates a fault-tolerance mechanism based on process replication, aimed at increasing the *robustness* of executions.

Zorilla (Niels, 2006) is a prototype peer-to-peer grid middleware system. It strives to implement all functionality needed to run applications on a grid in a fully distributed manner, such as scheduling, file transfer and security. Zorilla is designed to be used in situations where a full-blown grid environment is not needed, or simply not possible. Deployment of Zorilla is easy; only a single application needs to be

[1] *http://boincstats.com/stats/project_graph.php?pr=bo*

installed on the participating machines. Zorilla requires little configuration, since machines automatically organize themselves into a grid. Due to its peer-to-peer design, Zorilla can be scaled to handle large numbers of machines.

Conclusion

This article introduces the basic concept of peer-to-peer technology. To understand the advantages and the disadvantages of each peer-to-peer topology, we can choose and apply them to existing applications to offer them a better performance. The centralized topology offers good peers a service searching performance, however, it suffers from a bottle neck with the centralized peer. The decentralized topology offers load-balancing and high-reliability to the network, however, it requires a lot of traffic when searching for peers or service because of its flooding method. Many research projects aim to facilitate peer-to-peer application developers by offering a library that includes the peer-to-peer infrastructure and API to operate on the peer-to-peer network. CAN, Chord, Pastry are implemented by the concept of structured decentralized peer-to-peer topology. They use DHT to locate objects in peer-to-peer network space. All of them have stable releases of the library which several recent research projects are using. JXTA is quite a new implementation that offers the hybrid topology. During the last five years, it has become quite popular in the peer-to-peer research field because it is an open-source system and it is open to anyone who wishes to join the team of developers. However, JXTA suffers from performance issues and peer-to-peer application developers need to fine tune the library itself to solve the performance issues.

Peer-to-peer infrastructure is still an active subject in research fields. How to implement a peer-to-peer infrastructure that is not costly in terms of network traffic using a link maintenance to searach for peers and services; how to apply locality awareness in the peer-to-peer network; how to improve performance in the search for peers or services in the peer-to-peer network: all of these subjects are of interest in the research fields and waiting to be solved.

In Thailand, we still need to improve our research facilities, especially in science, technology and engineering domains. However, we lack the computing platform to

calculate and solve problems in these fields. Buying supercomputers is not an option for small or medium-sized organizations or for other institutes. However, in the universities, there are many personal computers available (lecturers' PCs, PCs in computer laboratories, etc.) and they are not fully utilized. Thus, we can apply the peer-to-peer technology and form these personal computers into a large-scale computing platform to solve problems in several research domains which will reduce the cost of buying expensive computer platforms, like supercomputers.

References

Anderson, D. (2001). **Peer-to-Peer: Harnessing the Power of Disruptive Technologies**, chapter 5, page 45-50, O'Reilly.

Anderson, D. (2004). BOINC: A System for Public-Resource Computing and Storage. In **Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing**.

Cohen, B. (2003). **Incentives build robustness in bittorrent** [On-line]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.1911

Dabek, F., Kaashoek, F., Karger, D., Morris, R., and Stoica, I. (2001). Wide-area cooperative storage with CFS. In **Proceedings of the 18th ACM Symposium on Operating Systems Principles**, Chateau Lake Louis, Banff, Canada.

Genaud, S., and Rattanapoka C. (2007). P2P-MPI: A peer-to-peer framework for robust execution of message passing parallel programs on Grids. **Journal of Grid Computing**, volume 5(1), pages 27-42, Springer.

Kan, G. (2001). **Peer-to-Peer: Harnessing the Power of Disruptive Technologies**, chapter 8, page 62-79, O'Reilly.

Kazuyuki, S., Yoshio, T., and Satoshi, S. (2005). P3: P2P-based Middleware Enabling Transfer and Aggregation of Computational Resources. In **Proceedings of Cluster Computing and Grid**.

Niels, D., Rob, V. N., and Henri, E. B. (2006). Simple locality-aware co-allocation in peer-to-peer supercomputing. In **GP2P: Sixth International Workshop on Global and Peer-2-Peer Computing**, Singapore.

Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In **Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)**, pages 329-350.

Ratnasamy, S., Francis, P., Shenker, S., Karp R., Handley, M. (2001). A scalable content-addressable network. In **Proceedings of ACM SIGCOMM**, pages 161-172.

Schollmeier, R. (2001). A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In **Proceedings of the First International Conference on Peer-to-Peer Computing**.

Shirky, C. (2001). **Peer-to-Peer: Harnessing the Power of Disruptive Technologies**, chapter 2, page 19-28, O'Reilly.

Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. In **Proceedings of ACM SIGCOMM**, pages 149-160.

Stribling, J., Councill, I., Li, J., Kaashoek, F., Karger, D., Morris, R., and Shenker, S. (2005). Overcite: A cooperative digital research library. In **Proceedings of the 4th International Workshop on Peer-to-Peer Systems**.

Traversat, B., Arora, A., Abdelaziz, M., Duigou, M., Haywood, C., Hugly, J-C., Pouyoul, E., Yeager, B. (2003). **Project JXTA 2.0 Super-Peer Virtual Network** [On-line]. Available: http://research.sun.com/spotlight/misc/jxta.pdf