

The Survey Of The Technologies Of Peer-To-Peer

YIMA

College of Computer, Georgia Tech

mikema@cc.gatech.edu

1. Introduction

Peer-to-Peer (P2P) systems and applications are distributed systems without any centralized control or hierarchical organization, where the software running at each node is equivalent in functionality. We can see that the Internet was a P2P system when it came into being in the late 1960s. The goal of the original ARPANET was to share computing resources around the U.S. The first few hosts on the ARPANET ---- UCLA, SRI, UCSB, and the University of Utah ---- were already independent computing sites with equal status. The ARPANET connected them together not in a master/slave or client/server relationship, but rather as equal computing peers. As the years gone, the scale of Internet also developed. Some client/server applications appeared, such as FTP, Telnet etc. The Internet is also not as safe as before, so the firewall, NAT came forth to protect LAN. At this time, the Internet had not been the P2P application. Recently, the P2P become a “hot” research topic again. There are many P2P applications now, such as Napster (the famous and immensely popular music exchange system), SETI@home (the distributed computation system), Freenet (symmetric file exchange system), Gnutella (another file exchange system), Jabber (instant messaging system) etc.

A review of the features of recent P2P applications yields a long list: redundant storage, permanence, selection of nearby servers, anonymity, search, authentication, and hierarchical naming. And implementing these features, we need consider following technical topics: metadata, performance, trust, accountability, reputation, security and interoperability through gateways etc.

Next, we survey the performance of P2P application: file-sharing systems.

2. Performance of P2P File-Sharing Systems

Today, thanks to the development of a number of advanced P2P file sharing applications, the reach and scope of peer networks has increased dramatically. The two main models that have evolved are the centralized model, such as that used by Napster, and the decentralized model, used by Gnutella.

2.1 The Centralized Model and The Decentralized Model

2.1.1 Centralized Model of P2P File Sharing---Napster

The Client/Server Structure

One model of P2P file sharing is based on the use of a central server system that directs traffic between individual registered users. This is the model used by Napster and certain other P2P networks. The central servers maintain directories of the shared files stored on the respective PCs of registered users of the network. These directories are updated every time a user logs on or off the Napster network, meaning a node add into or leave the Napster network.

Each time a user of a centralized P2P file sharing system submits a request or search for a particular file, the central server creates a list of files matching the search request, by crosschecking the request with the server's database of files belonging to users who are currently connected to the network. The central server then displays that list to the requesting user. The requesting user can then select the desired file from the list and open a direct HTTP link with the individual computer which currently owns that file. The download of the actual file takes place directly, from one network user to the other. The actual MP3 file is never stored on the central server or on any intermediate point on the network.

Advantages of the Centralized Client/Server Structure

One of the centralized Client/Server Model's main advantages is its central index that locates files quickly and efficiently. Because the central directory constantly updates the index, files that users find through their searches are immediately available for download. Another advantage lies in the fact that all individual users, or clients, must be registered to be on the server's network. As a result, search requests reach all logged-on users, which ensures that all searches are as comprehensive as possible.

Problems with the Centralized Client/Server Model

While a centralized architecture allows the most efficient, comprehensive search possible, the system also has only a few points of entry. As a result, the network has the Slashdot effect, that popular data becomes less accessible because of the load of the requests on a central server. The centralized Client/Server systems are also vulnerable to censorship and technical failure because they rely on a small number of very large servers. The system could completely collapse if one or several of the servers were to be incapacitated. Furthermore, the server-client model may provide out of date information or broken links, as the central server's database is refreshed only periodically.

2.1.2 The Decentralized Model of P2P File-Sharing ----Freenet and Gnutella

The Freenet and Gnutella system are all Decentralized P2P File-Sharing system. But the principle of their structure is not the same. The request mode of Freenet system is, I call, Chain mode, whereas the request mode of Gnutella is broadcast mode.

The Freenet Structure ---- Chain mode

In order to make use of Freenet's distributed resources, a user must initiate a request. Requests are messages that can be forwarded through many different nodes. Initially the user forwards the request to a node that he or she knows about and trusts (usually one running on his or her own computer). If a node doesn't have the document that the requestor is looking for, it forwards the request to another node that, according to its information, is more likely to have the document. The messages form a chain as each node forwards the request to the next node. Message time out after passing through a certain number of nodes, so that huge chains don't form. The chain ends when the message times out or when a node replies with the data.

The reply is passed back through each node that forwarded the request, back to the original node that started the chain. Each node in the chain may cache the reply locally, so that it can reply immediately to any further requests for that particular document. This means that commonly requested documents are cached on more nodes, and thus there is no Slashdot effect if one node becomes overloaded.

The reply contains an address of one of the nodes that it came through, so that nodes can learn about other nodes over time. This means that Freenet becomes increasingly connected. Thus, you may end up getting data from a node you didn't even know about. In fact, you still might not know that that node exists after you get the answer to the request---each node knows only the ones it communicates with directly and possibly one other node in the chain.

The Gnutella Structure ---- Broadcast mode

Unlike the Freenet, the Gnutella use the broadcast mode to request a file. To share files using the Gnutella model, a user starts with a networked computer, which we'll call "A," equipped with a Gnutella "servent" (so called because the program acts as a combination of a "server" and a "client"). Computer "A" will connect to another Gnutella-networked computer, "B." A will then announce that it is "alive" to B, which will in turn announce to all the computers that it is connected to, "C," "D," "E," and "F," that A is alive. The computers C, D, E, and F will then announce to all computers to which they are connected that A is alive; those computers will continue the pattern and announce to the computers they are connected to that computer A is alive. Although the reach of this network is potentially infinite, in reality it is limited by "time-to-live" ("TTL") constraints; that is, the number of layers of computers that the request will reach. Most Gnutella servents will reject any network messages that have TTL's that are excessively high.

Once "A" has announced that it is "alive" to the various members of the peer network, it can then search the contents of the shared directories of the peer network members. The search request will send the request to all members of the network, starting with, B, then to C, D, E, F, who will in turn send the request to the computers to which they are connected, and so forth. If one of the computers in the peer network, say for example, computer D, has a file which that matches the request, it transmits the file information (name, size, etc.) back through all the computers in the pathway towards A, where a list of files matching the search request will then appear on computer A's Gnutella server display. A will then be able to open a direct connection with computer D and will be able to download that file directly from computer D. The Gnutella model enables file sharing without using servers that do not actually directly serve content themselves.

The Gnutella Network (gNet) has a number of distinct advantages over other methods of file sharing. Namely, the Gnutella Network is decentralized and hence more robust than a centralized model because it eliminates reliance on centralized servers that are potential critical points of failure. Messages are transmitted over Gnutella Network in a decentralized manner: One user sends a search request to his "friends," who in turn pass that request along to their "friends," and so on. If one user, or even several users, in the network stop working, search requests would still get passed along.

2.2 The Performance of Decentralized P2P File Sharing System

I will discuss and compare the performance, (such as: how long will it to retrieve this file? or how much bandwidth will this query consume?), fault tolerance and scaling of the two different request mode: chain mode and broadcast mode.

2.2.1 the Small-World Effect

It's a famous phenomenon found by Stanley Milgram, a Harvard professor, in 1967. He mailed 60 letters to a set of randomly chosen people living in Omaha, Nebraska. He asked them to participate in an unusual social experiment in which they were to try to pass these letters to a given target person, a stockbroker working in Boston, Massachusetts, using only intermediaries known to one another on a first-name basis. That is, each person would pass her letter to a friend whom she thought might bring the letter closest to the target; the friend would then pass it on to another friend, and so on until the letter reached the target person. The result is surprising. In all, 42 letters made it through, via a median number of just 5.5 intermediaries. This is similar with the file request process. It can be seen as a graph problem. Peoples are vertices, and the relationship between peoples is link. Question is finding a shortest path between two people.

We can consider two extreme conditions: regular graph and random graph. Suppose that a graph has n vertices, each of which is connected to k other vertices (regular graph's one connects to the nearest k vertices, whereas random graph's one connects to random k vertices). In the case of regular graph, the path length is approximately $N/2k$. In the case of random graph, the path length is approximately $\log N / \log k$, which is much better than the $N/2k$.

2.2.2 Freenet

Path Length

The operation of Freenet has also the small world effect. Freenet's path length approximates the random graph's path length. However, from the simulation result, we can see that there are some requests that take a disproportionately long time. That is, Freenet has good average performance but poor worst-case performance, because Freenet's local routing cannot always choose the globally optimal route, a few bad routing choices can throw a request completely off the track.

Indeed, local routing decisions are extremely important. Although the small-world effect tells us that short routes exist between any pair of vertices in a small-world network, the tricky part is actually finding these short routes. So we can do some research on this point.

Growth

When a new node wants to join Freenet, it must first find an initial introductory node that is already in the network. The new node then sends an announcement message to the introductory node, which forwards it into Freenet. Each node contacted adds a reference to the new node to its data store and sends back a reply containing its own address, before forwarding the announcement on to another node chosen randomly from its data store. In turn, the new node adds all of these replies to its data store. Simulating this process, we can see that the network is performing even better than the fixed-size simulation having the same number of nodes. Meanwhile, the characteristic path length is not appreciably different from the fixed case.

Fault Tolerance

There are two kinds of fault: one is stations turned off or disconnected from the network at random, we call this random failure, and the other is host content disapproved of by some group is the possibility of a deliberate attempt to bring the network down through technical or legal attacks, we call this targeted attack.

Because with targeted attack, the nodes with more links are down first, so the random

failure is more fault tolerance than targeted attack. From the simulation, we can see that the Freenet's network remains surprisingly usable, with the median request pathlength remaining below 20 even when up to 30% of node fail. But in the case of targeted attack, the median request path length pass 20 at the 18% failure level. We can see that the stations with more links are more important than others, so we should protect them well.

Scalability

In small-world graphs, the characteristic path length scales logarithmically with the size of the network, since it follows the random-graph path length of $\log N / \log K$, the same as the Freenet. Freenet scale logarithmically

2.2.3 Gnutella

Gnutella uses a broadcast model to conduct queries, so it does not invoke the small-world effect.

Path length

Using the broadcast model, Gnutella queries are satisfied extremely quickly, under both average-case and worst-case conditions. Indeed, the broadcast is breadth-first search, which guarantees that the optimal shortest path to the data will always be found. But, the price paid for a quick result is a large expenditure of effort to search the network. Gnutella makes a trade-off of much greater search effort in return for optimal paths and better worst-case performance.

Fault tolerance

Gnutella performs about the same under both random failure and targeted attack. Here again is a trade-off: Gnutella responds equally to failure and attack, since all of its nodes are roughly equivalent. Freenet's highly connected nodes enable it to better cope with random failure, but these then become points of vulnerability for targeted attack.

Scalability

As a random graph, its path length scales logarithmically with the size of the network. Since it's breadth-first search finds optimal paths, the request path length always equals the shortest path length and also scales logarithmically. But the bandwidth used by a query is proportional to the number of messages sent, which in turn is proportional to the number of nodes that must be contacted before data. This will pose a serious scalability problem. To solve this problem, we can limit the search depths or moving toward a hierarchical p2p model.

2.2.4 conclusion

From the comparison, we can see that there are many trade-offs between Chain mode and Broadcast mode. And these two modes encompass a diverse set of approaches. So analyzing them and presenting new ideas, we can exploit the power of p2p well.

Also there are other interesting topic about P2P, such as trust and accountability.

Reference

- Andy Oram eds. Peer-to-peer: Harnessing the Power of Disruptive Technologies, O'Reilly, March 2001.
- <http://www.openp2p.com/>
- <http://freenet.sourceforge.net/index.php?page=publications>
- Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, ACM SIGCOMM 2001, San Deigo, CA, August 2001
- A Scalable Content-Addressable Network In Proceedings of ACM SIGCOMM 2001