Tony Morelli
CPE701
4/14/2009
Unstructured Peer To Peer Networks

Peer to Peer networks in their purist form are networks where there is no concept of client and server. All nodes connect to each other either directly, or through other nodes on the network. This type of peer to peer network is decentralized and unstructured, and is what most of this paper focusses on. One of the issues with the unstructured Peer to Peer networks is the difficulty in contacting another node on the network, or even knowing that a particular node exists. Because of this issue, the Peer to Peer networks have utilized methods of structure and centralization in order to more quickly connect nodes with each other. Central servers can keep a list of the contents and capabilities of all nodes on the network, and when a client needs a resource, it can query the central server and quickly be put in direct contact with the desired node. Hybrid solutions are also available where the network is primarily a true Peer to Peer network, however certain nodes cache information about the other nodes on the network. That way when a particular resource is needed, an individual member of the Peer to Peer network has a minimum number of servers to search through, instead of searching the entire network, or simply connecting to one server.

Peer to Peer networks have many uses. They are most commonly used to pirate software and media. However they do have other uses. Skype, is an internet phone/video phone service where Peers on the network connect to each other. Although a central service maintains the status of whether or not certain individuals are online, the actual communications between users is performed without the aid of Skype's servers. Television shows can be streamed from other peers using programs such as Joost. Structured Peer to Peer networks using the bit torrent protocol are also popular where each individual user can shares pieces of a file, so that an person downloading can grab pieces from several different other users which would speed up the download. Programs such as Seti@Home utilize peers for CPU cycles, instead of files. This distributed computing network is hoping to one day detect intelligent life outside of Earth.

The focus of this paper is to explore unstructured Peer to Peer networks. The most famous Peer To Peer network is the Gnutella network.

Unstructured Peer To Peer Networks are comprised of computers connecting to each other instead of connecting to a central server. Peer To Peer based networks are very stable as they contain no centralized server. With this feature, nodes of the network can come and go as they wish with the overall structure of the network remaining unchanged. There is no central computer that could crash the entire network. The distributed nature of the network keeps it running. One problem with this structure is that without one central system, knowing about all the other nodes on the network and their capabilities is extremely difficult. To query other nodes, the nodes in the network are forced to flood the network hoping that another node will respond with a hit on the query. This causes lots of network usage as well as many delays as the search is propagated throughout the network.

Although searching through the network can be time consuming and difficult, Unstructured Peer To Peer systems can heal themselves very quickly in the event of a node failure. This is because the overall design of the network does not depend on any one node to perform a specific task.

Unstructured peer to peer based networks are generally used for the proliferation of pirated software. This is the primary reason the networks being used for pirating software are unstructured

peer to peer networks. There is no one person for the authorities to target. It is much harder for the authorities to go after a whole bunch of poor people instead of targeting one rich organization. It seems that for a network to be efficient there needs to be some kind of structure to it. In the distribution of pirated goods, the search aspect of unstructured peer to peer networks leaves more to be desired, but that same limitation in searching for material is also an aid for the pirates as the authorities are limited to the same search issues when searching for criminals. Think about why big software houses are not using unstructured peer to peer networks for distribution, having some structure makes it easier for the users to find what they are looking for.

The main issue surrounding unstructured peer to peer networks lies in the search. Since the actual transfer of the data is using standard internet protocols (http), the bottleneck is in the search to find a servent that has the requested files. A servent as used in this paper is a node on the unstructured peer to peer network who can function as both a SERVer and a cliENT. Different approaches to search have been experimented with, but they all involve some kind of caching servers who cache indexes which contains all the files the peers have. That way queries can be sent to the caching servers only and flooding the network is not required. This makes sense, but the main issue with this style is the legality of it.

The most popular unstructured peer to peer network in use is the Gnutella protocol. This is primarily used for the distribution of pirated materials. The original version of Gnutella was completely unstructured, but after dealing with the issues surrounding a totally unstructured network, future versions of the protocol added a little bit of structure. The next few sections describe the Gnutella protocol, how it works, and how it evolved to have a little bit of structure to it.

The Gnutella protocol is relatively simple. A servent connects to a known servent in the network, and then begins communicating with the network. Initially a servent has no knowledge of anyone on the network, so the use of a host cache network is used. This is where the servent connects to a known host who has a list of nodes within the network. This adds a little bit of structure to the network, but once this initial seed of known hosts is given, the servent can participate in the network without the use of any centralized nodes. When a node leaves the network, it will store off the nodes it was directly communicating with, with the hopes that at least one of them will be available when it rejoins the network. If none of the previous neighbors are alive when the node comes back, it will once again utilize the host cache server.

Once the servent has contacted its neighbors, there are four basic messages that can be sent between the nodes through the peer to peer network – *Ping, Pong, Query,* and *QueryHit*.

Ping – A servent will send out a ping command to its neighbors. There is no data sent along with this message, just the command.

Pong – A servent will send out a pong command in response to a ping command. The response will contain an address of another servent and the amount of data it is sharing. The address and the amount of data does not necessarily need to belong to the machine responding to the ping. This is the method that new hosts can be discovered throughout the network. A servent can send any number of pong messages in response to one ping message. The amount of data piece of the

message is broken down into two parts. One part is the number of files shared, and the other part is the number of kiloBytes shared.

Query – A servent will send out a query message to its neighbors when a file is desired. There are two pieces to the query. The first part is obvious, a string containing the file the servent is looking for, the other part is less obvious. This piece is the minimum speed required to transfer the file. Servents can only respond to the query if they contain the file, and they can meet or exceed the minimum speed requirements. This message also includes a TTL (time to live) field. If the servent does not have a match, it will decrement the TTL and forward the query on to all of its neighbors. If the TTL is zero, the query will not be forwarded, and the servent will not respond. Although the query creates a lot of traffic, this TTL field helps to minimize the amount of traffic and prevents total flooding.

Query Hit – A servent will send the query hit in response to a query if it has one or more matches. The response will contain a list of file names and their file sizes that match the query. It also contains the speed that it can serve the files, and its own ip address. The ip address is necessary because the actual download of the file is a direct http connection between the two nodes. The peer to peer network is only used for searching for files, not for transferring the files.

Initially the number of nodes each servent was connected to was quite small. Each node would be connected to about 5 other nodes. If the query was not matched at any particular node, it would forward on the request to all of its direct neighbors. The initial TTL field was set to seven, which meant if a request was not met within seven hops, it was dropped. The ability to find resources on the network was very time consuming so with version 0.6 of the Gnutella protocol they introduced the concept of ultrapeers. The leaf nodes are directly connected to a lesser amount of direct neighbors, however the direct neighbors are ultrapeers who are connected to many other ultrapeers. The ultrapeers cache indexes from the leafs that it is connected to. So the query is usually matched much faster, and because of this the TTL was reduced to 4. When a servent makes a query request to an ultrapeer, the ultrapeer checks its own cache of indexes first before forwarding the request to the other ultrapeers or any of its leaf nodes. Once a match is found, the ip address of the servent containing the files is sent to the node making the request, and the node makes a direct connection to the host servent and begins downloading the file. Ultrapeers start out as a leaf when they first enter the network. As a leaf maintains participation in the network, it can be promoted to an ultrapeer.

Another issue with the Gnutella network is the overlay network may not be that accurate. In *Mapping the Gnutella Network: Properties of Large Scale Peer to Peer Systems and Implications for System Design,* they demonstrated this issue by the following mapping:

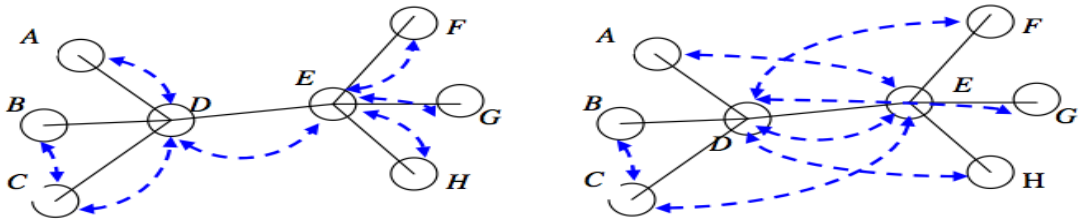

Figure 7: Two different mappings of Gnutella's virtual network topology (blue, dotted arrows) to the underlying network infrastructure (black, solid lines). Left picture: perfect mapping. A message inserted into the network by node A travels physical link D-E only once to reach all other nodes. Right picture: inefficient mapping. The same distribution requires that the message traverse physical link D-E six times.

As you can see by this map, the picture on the left shows the optimal routes, and on the right it shows inefficient routing. The Gnutella protocol has no specifications for making routes dependent on the internet topologies. The grouping of neighbors is based on the set of neighbors the node last talked to, not who it is closest to based on any method whatsoever. For example, a node in Reno, might be directly to a node in China, and when it needs to find a file on another computer in Reno, the search route might take the query through China and back to Reno, which is a waste. However, as the authors of the paper pointed out, determining the network topology, and where to insert yourself is a very costly procedure. In an ideal situation, where the network topology is known beforehand, it makes a lot of sense to group nodes by their physical location, but when thousands of nodes are joining and leaving the network all the time, it cost prohibitive to figure out the network topology first, and then join.

The main issue with Gnutella networks is their flooding mechanism for searching the network. In the 2002 paper *Replication Strategies in Unstructured Peer To Peer Networks* the authors (Cohen and Shenker) look for searching alternatives. They considered two different replication strategies uniform, and proportional.

Uniform replication indicates that indexes for all items on the network are replicated equally. This is a very primitive indexing idea where all items are treated identically regardless of their popularity. A very good reason to use uniform indexing is where there are very few queries that are not available on the network and there is a limited amount of unique files. Since the network knows about all items, and there are not many files, the responses are quick, and the network is not flooded with updates to the uniform indexes. The network traffic would be low as there are not many files to replicate indexes. When a new item first enters the system its index is replicated a fixed number of times across the network.

Proportional replication indicates that indexes for items on the network are replicated based on the popularity of the items. This makes it very easy to find popular items, but very difficult to find rare items. Items that have been searched for in the past, but are then not searched for are removed from the indexes.

In *Efficient and Scalable Query Routing for Unstructured Peer to Peer Networks*(Kumar et al) they discuss a new method for quickly searching an unstructured peer to peer network. They do not propose a tiered system of ultrapeers but they do introduce a method that will achieve a faster

search result than simply a blind query. What they do is instead of passing all the information about what files a node contains to everyone, they pass indexes and attach a weight in a radial pattern. So for example, if a node contains a file with a particular name, all of its direct peers contain the query string relating to that file and attach a high weight to it. Those peers pass on the index string to their other peers and decrement the weighting. So when searching for which node has a particular file, the query is sent and the route is determined by finding the path of increasing weight for the particular query string. Their experiments were positive, although they concluded that they needed more time to solidify their findings. I think the idea behind this is good it seems on the surface that a routing path would be found with ease. However, since the indexes are being passed around, it seems like it is not that much more traffic or overhead to pass along the ip address of the host that has the files. If that were the case, there would be no need to find a route, just ask your neighbor, or check your cache of indexes and directly contact the node with the desired file.

Most (all) unstructured peer to peer networks are used to pirate movies, sounds, programs and everything else. In the Napster case, Napster was found guilty because although the actual pirated files resided on individual computers, since their server provided an index and a nice way to access those files, they were guilty of aiding in the proliferation of pirated content. That being the case, all of the research done to increase the performance of unstructured peer to peer networks is nice, however it does not enhance the overall usability of the network. If I am involved in an unstructured peer to peer network, I do not want my computer to be involved in the distribution of any indexes of pirated software living on another machine as that puts me in the same position (guilty!) as Napster.

Although pirating software is the primary purpose of unstructured peer to peer networks, there has been at least 1 other documented experimental use of an unstructured peer to peer network. Awan et al (*Unstructured Peer To Peer Networks For Sharing Processor Cycles 2005)* experimented with using unstructured peer to peer networks to share computing power. They looked at systems like Seti@home, and noticed that although the computing was distributed, the actual distributing of the individual jobs was done by a central server (or cluster of central servers). This creates a nice distributed computing environment, but if the node responsible for distributing the jobs goes out to lunch, the entire system comes to a screeching halt. This paper wanted to modify that approach such that there is no central server responsible for handing out jobs. The jobs are handed out to the network by nodes as needed. Using an unstructured peer to peer network creates a very tolerant network of computers where individual nodes can come and go as they please, but the overall network remains fully functional. They also weighted nodes. Nodes that contributed more CPU cycles would be able to utilize more of the network's CPU cycles. This makes a lot of sense. The more you share, the more you get back, and it has a way of prioritizing nodes within the network.

One issue with this is who stores the information regarding participation amounts. If the node stores its own participation information, it would be easy for that node to manipulate the data such that it appears more favorable to the network. In order to prevent this, the network itself stores the record sets containing the participation information for the nodes on the network. The information is stored on random nodes and duplicated throughout the network to allow nodes to come and go. Nodes who submit malicious code or too many jobs will be marked as dangerous and when a node receives a job request from a dangerous node, it will have the ability to deny the request, or forward

onto another random node. This will protect any node from an overloading style of attack.

Messages between two nodes on the network pass through the peer to peer overlay network. This means that neighbors in the network may not be neighbors in the underlying internet graph, and any message between the two nodes might pass through any number of other nodes in the network on its way to the destination. They chose the node to perform the job by selecting it at random throughout the network. This proved to be successful, however the topology of the network did have an effect on the performance. For example if there are 3 hops between the source node and the destination node, the job could be completed quickly, but if there are 1000 hops between the source node and the destination node, the delay in sending and receiving commands could have a negative effect on performance.

They utilize random walks with uniform sampling to determine the node to perform the job. The piece of the paper describing this method is buried with math and is very complicated, however they claim it to be a good method and it looks like they certainly have enough equations to back it up. On the other hand, I always subscribe to the mentality that the best solution is a simple solution, and this looks like its fairly complex to explain, so maybe there is a better way.

LEGAL

Due to the constant use of unstructured peer to peer networks for the proliferation of copyrighted materials, it is important to discuss the various legal battles that have occurred over the years. This can provide a basis in determining the current legal status of modern peer to peer file sharing networks. The legal issues begin back in 1984 where a lawsuit was filed against Sony over their Betamax home video recorder. Betamax was a VCR type device which initially competed with VHS, and in the end VHS became the standard leaving Betamax in the dust. People were concerned with the Betamax as users could utilize this new device to copy copyrighted materials. The idea was that since Sony was selling the device capable of committing a crime that Sony was responsible for committing the crime. In the end the judgment stated that since the Betamax had real non-illegal purposes, Sony was in the clear. This important ruling became the basis of the Peer to Peer lawsuits.

The first big Peer to Peer lawsuit was the Napster case. Napster busted onto the internet in the late 1990's, and by the time the case was heard, in 2001, Napster had become a memory. However, the rulings were important. Napster was widely used to download music free of charge. The idea was that the users were simply “sharing” the material which was properly bought. For example if I purchased a cassette tape and then copied it for a friend, I was sharing that material. This sharing mentality was not something the record companies were in favor of. It became so easy to download any song or complete album quickly, that there was no point to actually buying the CD. When users were downloading a song file, they were directly connected to another user, Napster had nothing to do with the transmission of the copyrighted material. That was handled user to user. But the problem was how do you find files on this massive network. Napster's solution was to create giant indexing servers which would index the contents of all of the files each user was sharing. When a user would connect to the network, its list of files would be sent to the central server. When a user searched for a file, it would be done on Napster's central server. This made it quick and easy to find

files. When the Napster user wanted to download a file, the two user's computers would talk to each other be the only nodes involved in the file transfer. Napster felt they were in the clear, as their software broke no laws. No copyrighted material ever touched their servers, so the felt they were in the clear. The courts, however, did not see it that way. Since Napster's servers encouraged the proliferation of copyrighted materials, Napster profited from this, and Napster had the technology to stop the infringements, the courts found Napster responsible.

In a pure Peer to Peer network, there is no one individual for the courts to go after. Everyone is sharing and everyone is responsible. In the Napster case it was easy to point the finger at Napster themselves because they held all the indexes. In an unstructured peer to peer network, no one user holds the indexes of other users contents. This makes it very difficult for the authorities to prosecute anyone as they would have to have separate lawsuits for each individual and that becomes more costly than the lost revenues. The difficulty in searching an unstructured peer to peer network has created the necessity of super nodes, or caching servers to speed up the searches. These nodes start out as normal nodes, but get “promoted” to have a more important role. Since these are indexing servers, they fulfill similar functions that Napster's central servers served, which in the opinion of this author means they can be found just as responsible as Napster was. So if you are participating in a Peer to Peer network with some kind of structure to it, it would be in the best interest of that user to ensure he is not a super node, or a caching server. Under those circumstances the federal authorities would have some basis to charge that person with a crime.

My Take

In *Optimizing Peer To Peer Content Discovery Over Wireless Mobile Ad Hoc Networks (*Mecado et al 2007) they discuss Peer to Peer technologies in MANETs. They discuss a few different strategies for searching on ad hoc networks, but it brought up an interesting thought. Most (if not all) of the unstructured peer to peer networks are used to download files. There has not been much documentation on the use of unstructured peer to peer for much else, but this paper touched on a few ideas that I think need further investigation.

Unstructured Peer to Peer networks are best suited when nodes leave and appear frequently. The research in these networks has mainly focused on how to search the network looking for which node currently holds a piece of information. What I was thinking was, especially in ad hoc networks, the information is replicated across the network. In other words the knowledge contained within the network is stored within the network, not just the location of the information. For example, an ad hoc network created by cars on a freeway could contain information about the state of the road. This could be used for inter-car communications. If cars can one day drive themselves they will need to communicate with each other about what the other car is doing. Especially while driving on the road we do not have the time to wait out lengthy queries probing the network, we need results immediately. Cars will be joining and leaving the network all the time, but some information will need to be known all the time.

Lets say most cars travel on the highway at the speed limit. That means that the cars will appear to drive in a pack. A faster driver will approach this pack and when it connects to these peers it will

have immediate access to all information relating to this pack. It will not need to query each car, the state of the pack will be known by all the cars. So if a specific car has a tendency to speed up and slow down, that will be known by the group. If a particular car changes lanes quite often, that will be known by the group. This information will be used by the approaching car's navigation system to be more aware of its surroundings, and to be more aware of important changes in the surroundings.

In n *Optimizing Peer To Peer Content Discovery Over Wireless Mobile Ad Hoc Networks*, they also suggest the use of unstructured peer to peer networks for MANets in military situations. In these situations, ad hoc networks are set up, and there is no central server. They claim nodes are entering and leaving the network frequently, which they might be, but it seems as though a client/server architecture would work better in this area. They did not go into details about what types of communications are necessary in this military environment. If the ad hoc network is there to provide communications between the nodes, such as a Skype environment, a structured peer to peer network seems more fitting, although the server containing the status of the nodes could be a roaming server, that is any peer could take over the duties of being the server.

To sum things up, most Peer to Peer networks have some structure to them. Even the best example of an unstructured Peer to Peer network, Gnutella, now has some structure to it with the addition of ultrapeers. The unstructured Peer to Peer networks are mainly used to distribute pirated materials, however there have been attempts to distribute jobs across an unstructured Peer to Peer network to share and better utilize CPU cycles. The best part of an unstructured Peer to Peer network is that there is no central server which makes the network very robust and hard to shut down. Finding nodes on the network proves difficult, but there has been a lot of research done on how to better search the network. The future of unstructured Peer to Peer networks will be in networks where there is a high turnover rate and a great need for robustness, such as mobile networks that will drive our cars.

References

(1) Ripeanu, Foster, Iamnitchi - Mapping the Gnutella Network: Properties of Large Scale Peer To Peer Systems and Implications for System Design

(2) Kumar, Xu, Zegura - Efficient and Scalable Query Routing for Unstructured Peer to Peer Networks

(3) Chou, Wei, Kou - Performance Comparison of Unstructured Peer to Peer Content Discovery Techniques Over Mobile Ad Hoc Networks

(4) The Gnutella Protocol Specification v0.4 http://www.clip2.com

(5) Cohen, Shenker - Replication in Unstructured Peer to Peer Networks

(6) Negrin – Improving Search In Unstructured Peer to Peer Networks

(7) Awan, Ferreira, Jagannathan, Grama - Unstructured Peer to Peer Networks for Sharing Processor Cycles

(8) RFC4981