Doyle P. Hunt May 6, 2005

Free Riding in a Peer to Peer Networked Environment

Chapter 1: History of P2P and the Free Rider Problem

The History

Peer to Peer, or P2P, networks, have existed since the mid-1990's, with the Napster phenomenon birth just before Y2K as a marked point in the development of P2P networks. P2P networks are essentially a marketplace for people to share and trade digital files, with little regulation or oversight. As the internet emerged, the ability for any user to quickly connect to any other user, quickly became available and heavily utilized. The term 'P2P' does not explicitly denote any sort of use regarding file sharing, but file sharing quickly became a main hobby for P2P enthusiasts, along with instant messaging. The premise behind the concept, is quite a noble one, fulfilling the slogan of the early web days of living in a "truly connected world." The Internet is not inherently a P2P environment, but it does provide the tools to create an efficient one; one that can connect any two peers together to form their own private and secure connection. These connections, and the ability to create byte streams between the two peers are what have made this technology as controversial as any other social dilemma that society currently faces. The problem is not the technology, but rather its implementations.

Peer to Peer networks are now interchangeably being called 'File Sharing Networks', mostly in part because of the rise of Napster and Napster-like applications which allow for easy file transfers amongst a large group of peers. That in and of itself, is not harmful, but it does allow for the easy and uninterrupted exchange of files that may be digital copies of copyrighted material. This ability quickly gained the interest of many,

and fueled the fire for P2P application development. Napster's success was a product of Network Effects. Networks Effects are a cyclical situation whereby more users provide more content, which in return, attracts more users. This effect quickly spawned a tremendous interest in Napster, which was allowing users to freely share and download music files, which otherwise would have to be paid for or obtained by physically recording a compact disk into a digital recording format. Because of these effects, Napster became by far the cheapest (free), fastest, and most convenient way of building a tremendous music library. This phenomenon has had tremendous ramifications, but also some large benefits, depending on the industry in question. According to the record industry, music sales are down, but at the same time, the large distribution of unprotected copyrighted music led to the huge surge in growth of the MP3 player market. So clearly, P2P networks have had effects that were not foreseen in the early days of these file sharing networks.

As these applications became as commonplace to have on a computer as Microsoft Word, some old-world problems began to hit the P2P application world. 'Network Congestion' and 'Free Riding' arose as two problems that have existed in other social dilemmas that were now harming P2P networks as well. As both of these problems have been realized and studied, solutions have been made to try and forcefully remove the negativities that they may cause, through different application solutions that have tried to eliminate the problems that were inherent with Napster. Since Napster was first started by Shawn Fanning in his dorm room at Northeastern University, many other file sharing applications have come and gone, and many lawsuits have been filed against P2P application providers by those who feel that P2P applications are nothing more than an

outlet for outright theft. The current status of the legality of P2P networks is pending in the United States Supreme Court, and a ruling is not expected until June of 2005. Although there are obvious legal issues with infringement upon copyrighted materials, it is very difficult to make a case against the technology behind P2P applications, since the technology itself is not inherently illegal. Very similar legal issues were raised with the Xerox photocopier and Betamax video recorder. Both can be used for illegal means, but the technologies themselves are not altogether illegal, so long as there is evidence that the principal use of the technology does not violate any laws.

The legalities of P2P networks and the decisions that are yet to be made by the Supreme Court and potentially the Legislative Branches of our government, may have profound effects upon the future existence and development of P2P networks. In the current case pending in the Supreme Court, with twenty eight entertainment companies suing Grokster, a P2P application provider, for running a business where its sole profit comes from the exploitation of copyrighted works, although just about everyone agrees that file sharing should be condemned, there is a very broad consensus that the technology itself must not be stifled, otherwise further innovation may be thwarted. Dozens of companies have filed briefs supporting P2P technologies, including Intel, AT&T, MCI, Verizon, and Sun Microsystems along with organizations such as the ACLU, the National Library Associations and the Educational Defense Fund. All of these companies and organizations feel that P2P systems are inherently good, and that misuse by some does not give cause to limit the functionality of those systems. Also, even though many may argue that companies have an obligation to shut down a system if they know that it is being used to precipitate illegal activities, the Digital Millennium

Copyright Act of 1998 (the DMCA) places the burden on reasonable enforcement upon the owners of the material by stating that,

> "Contracting Parties shall provide adequate legal protection and effective legal remedies against the circumvention of effective technological measures that are used by authors in connection with the exercise of their rights under this Treaty or the Berne Convention and that restrict acts, in respect of their works, which are not authorized by the authors concerned or permitted by law." [1]

This clause in the DMCA forces media providers to implement security measures that could only be defeated by a malicious hacker. The problem with this for the Recording Industry though, is that there are no security measures encoded in a compact disc like there are on dvd's. Music cds are recorded according to the Red Book Standard that was developed by Phillips Electronics and the Sony Corp in the early 1980's. This being the case, it has been very difficult to prosecute offenders of file sharing, and penalties have usually only been small monetary fines. So as file sharing networks have grown to gigantic sizes, lawsuits have been unable to stomp out these activities, but have also caused worry amongst the software development community, as members fear legal action against those who provide the means for the networks. In fact in an interview with Bram Cohen, the inventor of BitTorent, the writer writes,

> "The one person who hasn't joined the plundering (file sharing) is Cohen himself. He says he has never downloaded a single pirated file using BitTorrent. Why? He suspects the MPAA would love to make a legal example of him, and he doesn't want to give them an opening." [2]

This is a clear example of the vulnerabilities that developers face when developing P2P applications. BitTorrent was originally developed as a way to distribute Linux applications (open source, not copyrighted) that were of significant file size, to many users, more quickly than a direct connection between two peers. This technology was

quickly picked up (BitTorrent itself is open source) by people interested in sharing large movies across a network. But it can be proven that the original inattentions were not to create a means of distributing copyrighted material.

Why P2P Applications Work

P2P applications work so well because it is possible to limit network congestion, processing power continues to double per dollar spent every eighteen months, bandwidth is becoming cheaper and cheaper, and compression mechanisms are creating smaller files while preserving original quality (Lossless codecs). With home personal computers and even laptops having the processing power of what high-end servers had only a few years ago, the role of client and server has began to diminish. Almost every computer has been used in some form or another of a server, unlike the early days of computing where the distinction between client and server was much clearer, with every dumb terminal having a physical connection to its server. P2P applications cause whatever machine is running the app to act as both a server and a client. A peer in a P2P network must act both as a client and server in order to live up to its obligation of providing and using other's resources. P2P applications also create a self-fulfilling attraction in that more users that are attracted to the service, the faster the growth. Growth of a P2P network appears to be exponential and not linear, judging from the fast growth of popular P2P networks such as KaZaa and Napster. However, as these networks grow larger and larger, it gives users the ability to 'shirk' from their responsibilities by hiding amongst the masses, and consuming more than they produce. This can happen in many forms, but all are considered a form of 'Free Riding'.

Congestion is limited in P2P networks in an ideal situation, since there is no central source for a single file. Ideally, as soon as one user has a file, multiple people can download the file simultaneously, and from there, multiple people can download from each of those peers and so forth. This exponential growth makes reliance on a single peer, such as in a traditional client/server model, much less important, and prevents a single peer or server from having the sole responsibility of providing the requested files. This sharing of responsibility can limit the amount of congestion on a single network or peer. As bandwidth and processing power have increased dramatically as well, connections tend to last shorter amounts of time, and network equipment can handle the requests more quickly as well. One thing that is not greatly changing though is the actual size of files that are being transferred. Music files have typically, and most likely remain, in a compressed form taking approximately $3x10^6$ bytes of disk space. Video however, which has become ever more popular to share after the P2P craze that was started with Napster, takes up considerable more space, but tends to be consumed in much smaller quantities. While some people may consume 2,000 music files on their personal computer, users will typically only have a few movies or television episodes. Nevertheless, the MPAA (Motion Picture Association of America) is taking the lead in prosecuting file swappers, as they feel that the music industry has already lost their battle, and they do not want to end up in the same position as them. As John Malcolm of the MPAA said in an interview with *Wired Magazine* "We consider it (lawsuits and billboards urging users not to download movies) a regrettable but necessary step. We saw the devastating effect that peer-to-peer piracy had on the record industry."[2]

One of the problems the movie industry is facing that is difficult to combat, is the consumer's desire for on-demand availability. Downloading individual episodes of television shows or movies before they are released to dvd or vhs, allows users to have exactly what they want when they want it. Movie studios use windowing as a technique to maximize profits, by staging the release of movies in theatres, home video, and cable television. This demand by consumers and lack of remedy by the Motion Picture Industry has caused file swapping of movies and television shows to be the fastest growing use for P2P networks. BitTorrent, the P2P network that has revolutionized the file sharing methods used for large files has skyrocketed this problem to become a real concern in the industry. One report believes that one third of all internet traffic is related to the BitTorrent network.

Network Congestion

P2P networks go to the extreme by increasing processing power dynamically, by using the resources of a large number of users as a collective whole. As the number of users increases, ideally the amount of processing power and resources increase along with it, however, the *Free Rider* issue can hamper that perfect solution. But even with a small percentage of users with fairly fast connections actually providing their bandwidth, processing power and other resources, P2P networks can provide the same power as very well-built server farms. Problems, such as cracking 40-bit encryption and finding large Mersenne prime numbers (a prime number in the form of 2^p-1), have been accredited to large distributed systems, which can be developed through a P2P network, instead of using single or clustered servers. This is a main argument for the uninhibited development of P2P technologies. The academic and research possibilities that could

result from large scale P2P networks are endless. The amount of processing power that is unused in a simple personal computer is valuable, but all of those unused resources collectively are indescribably valuable.

Network Congestion has been a problem that has plagued file servers since their inception, and the solution has been to increase bandwidth and increase overall processing power (speed of servers in addition to the number of servers). Since Network Congestion is not an easily solvable problem, and an expensive one to minimize, P2P networks seem to have the upper hand when it comes to cost ($0) and effectiveness. The cost associated with P2P networks is consumed by the user, with their fees they pay for internet access, in whatever form they receive it, as the true cost of a P2P network. In a P2P network, free of free riding, network congestion is diminished by spreading out data transfers. If peer A wants to send X to peer B, a connection can be established between the two peers, which is completely independent of a connection between Peer B and Peer C. With a client server model, all traffic would have to travel to Server Z, and then out of Server Z to the destination peer. Now, there are issues that arise with P2P networks since typical households have asynchronous connections, in that their upload speeds are slower than their download speeds. This problem was addressed by Bram Cohen's *BitTorrent* application, which allows users to download a single file by downloading pieces (torrents) from different users, so that a user's download speed can be maximized instead of being held to another peer's upload speed. This is why *BitTorrent* has become a favorite amongst users who share and swap large files (such as movies and games). Anyway you look at it though, P2P networks seem to be addressing all of the problems, more efficiently in some cases, that the client – server model faces as well.

Free Riding

The purpose of this paper is to address the *Free Riding* problem that is inherent in a system whereby the benefits are maximized and the risk is minimized when a user 'free rides' on the system. Free riding is the social dilemma whereby one uses a community's (P2P network) resources, but does not contribute to the cause. Free riding is a problem that occurs in different aspects of society. Anytime someone tries to obtain something for free or with minimal effort, that person can be thought of as a free rider. Free riders typically exist in large communities where their lack of participation can go unnoticed without repercussions in respect to the benefits available. Free riding has become a big problem in P2P networks because of various reasons. In academic research and other legal activities, there can be a tendency to desire the use of another's machine for speed purposes, but when it comes to allowing others to use your processing power, there really is no benefit for the person who is giving up clock cycles, unless there is a mutual agreement for reciprocal time when the user needs the other user's clock cycles for their own research. With security always being an issue, users may be weary of intentional or unintentional vulnerabilities in the P2P application, which can be a very legitimate cause for concern, especially within the Windows operating systems. With P2P networks that are based around illegal file sharing, the Free Rider problem is magnified due to potential legal consequences for providing illegal material over a network. Although peers who *free ride* are also potentially liable for copyright infringement, peers feel less likely to be caught if they are only downloading and not supplying the illegal files to others. This has caused the free rider problem to have a profound effect on these illegal file sharing networks, by diminishing the value of a P2P network, where a majority of users are not

contributing, but only using the community's resources. This could potentially lead to self-destruction of these file sharing networks that the Movie and Music Industry have so greatly feared.

Users who have been free riding typically do so as a way to prevent viruses, security holes, loss of computing power and criminal prosecution. Some of these problems though are being addressed, and criminal prosecution tends to be the main deterrent from participating as a true peer in a P2P network. The idea of being shielded from prosecution or security vulnerabilities by free riding, is a false notion. Viruses can easily spread by downloading corrupted files which can then alter the current application and add additional security threats. Prosecution of file swappers is not limited to providers either. Those who contribute in any manner shape or form to the illegal distribution

Current and Former P2P Applications

Below are some popular P2P applications, along with their purpose, their history and what they have developed into.

Napster- First developed by Shawn Fanning in 1999, after he dropped out of Northeastern University in Boston, MA. Napster has been said to have had the most profound effect on P2P networks of any P2P application every developed. Napster was not a true P2P network, since there was a centralized server which registered IP addresses of users along with lists of files which the users were offering to share. Napster was the P2P network that sparked the Recording Industry Association of America (RIAA) to start prosecuting P2P network providers. Napster was easy to shut down because of its

centralized nature. By the time Napster was shut down, it had over 50 million users who were sharing files every day. By this time, the P2P craze had exploded, and there would be no way to stop the future development of other P2P file sharing networks. Since Napster, P2P applications have tried to adopt a less centralized nature, which is the way a true P2P network should operate. Without multicasting capabilities, a completely decentralized service is difficult to develop, but a less centralized solution than Napster is possible.

BitTorrent – BitTorrent is currently the largest threat to the Movie, Game, and Software industry because of its ability to handle large files, and its ability to satisfy courts that its purpose is for something other than file swapping of copyrighted material. BitTorrent was developed by Bram Cohen because he found the problem of file sharing large files to be a difficult one, but one that he could solve.

BitTorrent addresses the problems of congestion and Free Riding in the following ways. First off, the way BitTorrent works is that a single file must first be made available by a single user, known as a 'seed'. This user must upload at least one complete version of the file to connected peers. Peers are found by trackers, which are simple applications that run over HTTP, that help peers identify other peers that have the pieces, or torrents, of the files they need. These pieces are typically 250 kilobytes and are reassembled into one file once all pieces have been downloaded. Once all of the pieces have been assembled, a hash function verifies each torrent's integrity, comparing the hash value to the value kept by the tracker. As a user downloads pieces of a file from another user, the user must also upload torrents that other users may need. At anytime, a user may be

uploading or downloading from multiple peers, with a queue of torrent requests, as a way to operate the most efficiently.

According to Bram Cohen,

> "Each peer is responsible for attempting to maximize its own download rate. Peers do this by downloading from whoever they can and deciding which peers to upload to via a variant of tit-for-tat. To cooperate, peers upload, and to not cooperate, they 'choke' peers." [3]

In order to have an efficient algorithm, the resources of connected peers must be reevaluated in a timely fashion. BitTorrent recalculates the current transfer rate every 20 seconds for each connection and reevaluates whether or not it is connected to the best peers every 30 seconds. If the algorithm finds other peers who would create a faster download, it will drop its slowest connection and connect to the new peer in order to facilitate the fastest download. Along with connection speed, the algorithm also uses a 'rarest first' approach, which attempts to download torrents which are least available first, so that its availability will eventually increase.

Lastly, Cohen expresses his interest in distributed collaboration as a way to create a very powerful decentralized service. In order to accomplish this, the system must become pareto efficient. A pareto efficient system is a system in which the system is so optimal, that the only way to further advance any node in the system, could only come at a cost to another node in the same system.

> "In computer science terms, seeking pareto efficiency is a local optimization algorithm in which pairs of counterparties see if they can improve their lot together, and such algorithms tend to lead to global optima." [3]

Clearly, collaboration towards efficiency will ultimately lead to a fast, highly scalable and distributable P2P network.

Gnutella- Initially released in March of 2000 by Nullsoft, a subsidiary of America Online, it was quickly pulled back as it was quickly realized that this application had the same abilities as Napster, which was currently in the process of being shut down by the judicial system. At this time, AOL and Timer Warner were in the process of merging to form AOL Time Warner. As Gnutella had been released for a short while before it was recalled, it was enough time for the application to spread rather quickly.

Shortly after its release, the protocol, the Gnutella Protocol, was deciphered, and additional applications began to use the protocol for their own uses. Applications such as Morpheus and Limewire were developed using these standards, which themselves became hosts to very large P2P networks. The Gnutella Protocol however did not scale well. Once released to the entire internet, bottlenecks and bandwidth issues quickly became very apparent. Users would find the first peers from a list on the internet, but from there, all peers would be found through a sort of depth-fist search algorithm. Once an additional peer was found, the user could then find all of that user's peers, and so forth. Once a peer was found with the proper resources, queries would be sent to request the resource. However, these requests could sometimes take up large portions of bandwidth, which would hamper the entire usability of the network. If one peer knew of 4,000 peers, which was the application's maximum, that peer alone would generate 4,000 request queries Congestion caused by large numbers of request packets could cause Denial of Service attacks on users that had rare resources. The life of these packets was also a problem, as some packets did not die, and only cluttered the network, eating up usable bandwidth.

Clearly the Gnutella application will work in a small closed environment, but large scale deployment brings about many additional concerns. One of the issues was that peers were trying to contact every peer directly for a request, instead of having request messages forwarded, which could have preserved bandwidth. Free riding also played a huge role in this application, because according to the *First Monday Journal*, "…a large portion of the user population, upwards of 70%, enjoy the benefits of the system without contributing to its content." [5] This large free riding issue is what requires so many requests to be made, and peers to be searched. So if a user is connected to 4000 peers only 1200 of those users are actually providing content which may be desirable.

Kazaa- A very similar P2P network, owned and operated by Sherman Networks of Australia. KaZaA is also a decentralized network that dynamically appoints nodes as 'Supernodes' which are typical users that have high bandwidth and processing power. Every supernode indexes all of the resources that it and its peers have. When a peer sends a request, the request is first sent to its nearest supernode. That supernode will check its index to see if it knows of a peer with the requested resource. If the supernode does not have the resource in its index, it will send back the address of another supernode, where the requesting peer can re-request its resource.

The Lawsuits

Undoubtedly, every week there are cases in the news regarding P2P networks, copyright infringement, and twelve year olds being subpoenaed by the Recording Industry Association of America (RIAA). Since Napster, the Motion Picture Industry and the Recording Industry have been trying to close P2P networks, as they believe the effect

of users downloading material for free has had a disastrous effect on their businesses. It is a very difficult situation because it is difficult to monitor, difficult to prevent, and deterrents can have additional consequences that the industries did not intend. As Bram Cohen says about executives in the Music and Movie industries at an awards show in Los Angeles last November,

> "the content people have no clue. I mean no clue. The cost of bandwidth is going down to nothing. And the size of hard drives is getting so big, and they're so cheap that pretty soon you'll have every song you own on one hard drive. The content distribution industry is going to evaporate." [2]

The industries have difficulty in trying to stop these P2P networks for several reasons. First, as P2P networks evolve, they become more and more decentralized. True P2P networks are made up of nodes with no hierarchy. Intentionally or not. With no center, there is no way to shut it down. Second, it is not possible to determine entirely who is using a P2P network, so the best chances the prosecution have are to find a few 'super nodes' who are providing the majority of content, shut them down, and hope that that deters others from doing the same. Lastly, and the most important reason that some P2P networks cannot be shut down is because they have a very legitimate use besides copyrighted file sharing. It has been determined in courts that providers of technology cannot be held liable for misuse of their product, so long as the technology was developed for a legal and meaningful purpose.

Chapter 2: Free Riding: A Social Dilemma

The Free Rider problem has existed in society for a very long time, and has been studied in various contexts. Free Riding in a P2P network is not unexpected, especially

since the increase in criminal prosecutions of P2P file sharers. The problem itself is not related to P2P networks; rather it is attributed to a natural tendency for people to obtain the largest benefit with the lowest cost. Realizing that free riding in a P2P network is a result of a deeper sociological problem, allows researchers and developers a better insight as to how to quell the issue. The problem stems from the lack of central authority to govern the group. In a sense, a P2P network is an anarchical community; no rules, and no hierarchy. Without this central authority, there is little fear of repercussions from minimal participation in the community, but there is also not authority to limit when an individual participates in that community. This ability to freely come and go, and pick and choose, is what has lead to the free riding problem in P2P networks.

Free riding can be seen elsewhere, and during other chronological periods in time. People who cheat on taxes can be seen as free riders. People who cheat on taxes are part of a very large community, the entire country, and the central authority, the IRS might not catch on to them, and any form of repercussions may be avoided. People who cheat on their taxes, even though they may feel justified in doing so, use the same resources as just about anyone else, yet feel that their contribution should be smaller.

All forms of social welfare can suffer from free riding, as for some people, they are much happier consuming resources provided by others, instead of contributing themselves. People on welfare for extended periods of time that are capable of supporting themselves and their families, people collecting unemployment benefits for lengthy periods of time, and people who file frivolous lawsuits are all examples of individuals who would rather consume than produce. Clearly another set of examples of free riding in a large system. This problem demonstrates strong differences in personal choice when

an individual is acting alone, and when that same individual is part of a group. The most famous story to demonstrate this problem is the *Prisoner's Dilemma*, and this is the story:

> "Tanya and Cinque have been arrested for robbing the Hibernia Savings Bank and placed in separate isolation cells. Both care much more about their personal freedom than about the welfare of their accomplice. A clever prosecutor makes the following offer to each. You may choose to confess or remain silent. If you confess and your accomplice remains silent I will drop all charges against you and use your testimony to ensure that your accomplice does serious time. Likewise, if your accomplice confesses while you remain silent, they will go free while you do the time. If you both confess I get two convictions, but I'll see to it that you both get early parole. If you both remain silent, I'll have to settle for token sentences on firearms possession charges. If you wish to confess, you must leave a note with the jailer before my return tomorrow morning." [6]

Users of P2P networks suffer from the same dilemma that the two prisoners do because users of P2P networks can only achieve the best result if they cause harm to one other, while the other does no harm to the first. If both people 'free ride' (testify against the other), then their worst possible outcome is still better than not free riding (not testifying, with the possibility that the other will testify). The worst possible outcome for either individual will come from not free riding, while the other does. It is a difficult situation, and causes the individual to decide which is more important; themselves or the group. There is no real solution, and is left up for individuals to decide their own action. Likewise, this is also comparable to the Voter's Paradox. The Voter's Paradox is as follows:

> "We are all better off if most people vote, but my vote will make no impact and it does cost me to vote." [7]

This situation becomes more and more apparent as the size of the group grows larger and larger. With a possible cost for participation, an individual feels that their participation is

negligible, and is not required, because of their trivial role in the group. The costs that are associated with P2P networking, like Tanya or Cinque going to Jail, are explained more thoroughly.

How Does Participation in a P2P Network Harm Individuals? The Costs

One of the obvious questions in dealing with the Free Rider problem in a P2P network is, what is the cost? In a theoretical, legal, copyright material free, P2P network, the costs of participation are almost none. At least monetary costs are very close to zero, and labor time is pretty minimal as well. Essentially your computer is allowed to act as a daemon, handling requests from others on its own, and sending out requests at your convenience. So what harm can come from allowing others to use your resources that would create no cost for you?

There are two potential costs; viruses and criminal prosecution. I believe the latter is the more expensive and justified cost, since there is no evidence that only using a P2P network for its resources, and then free riding on the system, is more likely to make you safe from viruses, and thus prevent whatever costs you might incur from recovering from it. The idea of criminal prosecution though has been made a very real threat to individuals of P2P networks that are used for the sharing of copyrighted materials. Obviously sharing copyrighted material is illegal and wrong, but this leads to the development of a stereotype that P2P networks are only used for illegal activity. The potential power in a distributed P2P network is immense, and if this stereotype continues to develop, it will hinder further P2P developments. One of the inherent problems with this issue, is the inability to filter out illegal materials from traveling over the network. The pure definition of a P2P network calls for no central authority, so without a governing body, it is not

really possible to monitor all distributed content. Whether it is a newly released top 10 hit song, or an open-source plug in for a Linux application, they all cross the network as byte streams, and cannot be filtered without some form of central filter. This is probably the biggest problem facing developers of P2P software. P2P software developers fear possible legal repercussions if their software is found to contribute to this illegal activity. This debate is currently in the Supreme Court, and a final ruling is not expected until June of 2005. This ruling could potentially have a tremendous impact on the future of P2P development. If the ruling's make developers liable for the uses of their products, the future of P2P networks might be in serious doubt.

The Legal Ramifications for P2P Software Developers

One of the most important things that must be realized in understanding the technology and power behind P2P networks is that the future of this technology is in doubt because of the stigma that has been associated with the technology because of copyrighted file sharing. As the role of P2P software and network developers hangs in the balance in the U.S. Supreme court, firms are reluctant to embrace the technology before the legal responsibility questions are answered in full. The responsibility of the P2P application developer is unclear, because history could swing the decision either way.

The most important case that has ever been decided in the Supreme Court that is relevant to the *MGM Studios v. Grokster Inc.* case, the case currently pending in the Supreme Court, is the *Sony Corp of America V. Universal City Studios Inc.* case of 1984. In this case, Universal Studios was suing Sony because Sony was the manufacturer and distributor of the Betamax video recorder. Universal Studios claimed that Sony was

infringing upon its copyrights, because the video recorder allowed anyone to time-shift any broadcast and view it at a later time; effectively copying the original broadcast. Sony's defense was that the Betamax video recorder itself was "capable of substantial non-infringing uses"[8] The Supreme Court agreed with Sony, but only with a 5-4 decision in favor of Sony. Sony was found not to be responsible because they had no way of knowing what its consumers were doing with their products, and would have no way of enforcing copyright laws with the available technology. However, another point that was clearly made, and what separates the Sony case from the Grokster case, is that Sony did not maintain an ongoing relationship with the consumer after the initial purchase. After the sale of the Betamax recorder, the relationship between Sony and the consumer was completely severed. With Grokster, the same reasoning cannot be applied. Although Grokster may not know what is happening directly between members of its network, it is necessary to maintain a relationship with its consumers (users) in order for the product (P2P application) to function properly. In addition to the Sony case, the Napster case also applies, but in this case, could hurt Grokster. Napster was a very clear cut case of secondary copyright liability, whereby Napster clearly knew about and supported acts of copyright infringement. When it can be proven that a product has the capability to monitor a user's usage, and those monitoring do nothing to prevent that sort of illegal activity, regardless of whether or not the product can be used for legitimate reasons, the company in charge of the service can be held liable for secondary copyright infringement.

This next ruling in the Supreme Court may very well have a very dramatic effect on the direction of P2P technologies in the very near future. If companies may be held liable for their users' uses, it may be too big of a gamble to produce P2P software. One of

the technical differences between P2P technologies though involves the actual role of the central service. If the central service, which is required on non-multicast networks, does not hold any information, and simply passes user addresses to other users, there may be no way of that service knowing what its users are using the network for. If however, like in the case of Napster, the service acted as a central depository for filenames, it is much easier to determine whether or not the service could have in fact known what was transpiring on its service.

Free Riding: The Numbers

Obviously Free Riding is a problem on P2P networks, and is only getting worse, as potential costs rise. With that being the case, Mr. Eytan Adar and Mr. Bernardo A. Huberman, of the Xerox Palo Alto Research Center, did a study of users on the Gnutella network, and quantified their results, pertaining to percentages of users who provided either minimal, undesirable, or no resources at all. In their published report, they found that "nearly 70% of Gnutella users share no files, and nearly 50% of all responses are returned by the top 1% of sharing hosts." In addition, they note that "the free riding leads to degradation of the system performance and adds vulnerability to the system. If this trend continues copyright issues might become moot compared to the possible collapse of such systems." [5] One of the possible reasons for this situation is that users who have no resources at all, (ie. files) will not be able to participate openly on the network, and will use the network to respond to their needs. Once their needs are met, they will no longer participate on the network, and the burden will continue on that top 1% of users. In the report, this reasoning is what contributes to "rampant free riding" in a large system like Gnutella.

In this study, a sampling pool of 7,349 peers was monitored for traffic. Of those users, only 37% ever provided a response to a query. A query is sent whenever someone else on the network is requesting someone else's resources. Of those responses, the top 25% of all sharing hosts provided request responses to 98% of all queries that were passed through this network. With that being the case, another hypothesis was verified, that even though 98% of all responses come from the top 25% of users, many of the other users are contributing resources to the network, but only share undesirable content, which is why they are unable to respond to query messages in the first place. This reluctance to share desirable resources is another form of free riding.

The inherent danger of this free riding situation is that rampant free riding centralizes P2P networks, and these networks begin to fall back towards a client server model, and away from the decentralization which is critical in a P2P system. If 25% of peers control 98% of the resources of a network, then we no longer have a decentralized system. This makes the system very vulnerable, because now only that 25% needs to be shut down in order to render the network inoperable. In a true P2P system, an elimination of any 25% of users should have a minimal effect, whereas in a system with rampant free riding, that 25% would have a catastrophic effect, and render the system useless.

Possible Solutions to the problem

Eliminating free riding from a non-centralized network is almost impossible, but measures can be installed to help make free-riding more costly to the user. Increasing the cost of free riding is the most effective way of retarding the problem. The best way of increasing the cost for a user is by limiting the services that that peer can use without

providing returns to the group. In a perfect network, only users who provide resources may consume them. With that however, it must be possible for users who have nothing to provide, to obtain something first, so that they will then have something to share with the rest of the peer group. In order to enforce some form of regulation, peers must be forced to prove to other peers that they are indeed proving a service to the rest of the group. Failure to provide this evidence must be met with disdain by the other peers, and force the non-participating peer either to participate, or to leave the group entirely.

In a paper, "Enforcing Fair Sharing of Peer-to-Peer Resources" from Rice University, the authors provide some general methods for this type of group enforcement. Their examples looked at a P2P network that was used to provide remote storage space to other peers. The most logical and decentralized approach seems to be Peer Auditing. Peer Auditing basically requires full disclosure by peers as to what they are actively providing to the group. If Peer A requests to store a file on Peer B, Peer B will first request an audit from Peer A, and determine whether or not it is providing an appropriated amount of resources to the groups. If Peer A is not providing resources to the group, than it is not paying for the ability to store a file on Peer B, and the request for storage would be denied. This solution would scale well in a large network, and does not require a central authority to monitor usage, as some of their other suggestions require.

Clearly there is no perfect way of enforcing participation, and in the above proposal, it does not clearly outline how a new peer would be able to join the network and participate if it does not have anything to give. The initialization of P2P networks can often times be the hardest, as situations may appear that will only appear in the beginning of its existence, and need to be dealt with in a special manner. If peers are required to

provide resources before they can consume, you have a problem when the group is formed, and all you have are new members who cannot consume, because everyone else is also new, and cannot use another's resources until someone uses theirs. If no one is able to consume, than none of the new peers will be able to offer their resources to others. This is a problem with the initialization of Peer Auditing, and one that must be addressed in order for it to function properly. Exceptions must also be made for those who cannot legitimately contribute to the group, and must be given resources first before they can afford to provide resources to other peers.

Chapter 3: A Free-Rider Problem Solution With JXTA

My goal for this thesis was to develop a P2P application which inherently limited free-ridership in a way that differed from BitTorrent, which significantly reduces the number and strength of free riders. In order to develop a P2P application, I chose to use the already existing, but still developing, JXTA framework. JXTA is an open-source project, funded mainly by Sun Microsystems, that has all of the libraries needed to build and deploy a P2P application, which would create a P2P network. JXTA is written for the Java environment, and was started in 2001. In hindsight, this may not have been the best choice for a framework, but it worked out in the end, and much was learned from experience. One of the interesting things about JXTA and other P2P applications is that I could not find any examples of fully developed and functional P2P networks, that had been developed with JXTA as its core. Having studied some of the large P2P network applications, it seems as though most P2P software is built from the ground up; each using a different approach to connect peers. JXTA uses small XML documents that are passed around from peer to peer, as a way to locate other peers and learn about what

services they are offering, or which groups they are part of. In any P2P network, that is not entirely on a multicast network, at least one peer or host must be known before any peer information can be propagated. With Napster, there was a central server, whose address was contained within the application code. With a JXTA network, any peer on a network can act as a *Rendezvous Peer*, which is a peer like any other, but which has the ability to accept and send any other peer's relevant information. In JXTA, information about a peer or service is known as an *Advertisement.* An Advertisement is an XML document that contains information such as peer name, IP address, and group affiliation. Once a rendezvous peer receives this information, it passes the information to every other peer that it is connected to.

In testing the application on a multicast network (IPv6), such as any single subnet, the multicast features of JXTA seemed to operate very efficiently and require no pre-known information about any other peers. Because JXTA uses a specific multicast address and port which is common to all JXTA applications, all instances of a JXTA app will find other peers running on the same multicast network almost immediately. Although they will not initially know of any group membership or peer identification of the nearby peers, connections are made between the peer and all other reachable peers on the same multicast network so that those advertisements may be publicly advertised. The idea of having seeding peers, or predetermined rendezvous peers, is really just a way to allow a P2P application to function on an IPv4 network; a network which does not support multicasting.

The JXTA framework allows for independent group formation, which allows members to join groups, and allows groups to limit its members through password

protection. Every group is a sub-network of the *Default Peer Group* network, by which every peer is a member, by default. From this *super group*, all other groups are formed. When a group is formed, a *PeerGroup Advertisement* is created, and propogated through the network. PeerGroups are identified by a unique ID, and can be created by any peer. In order for two peers to join the same group, they must know the unique ID prior to joining the group. Group ID's can be found in one of two ways. First, the peer could have the ID embedded within the application, or it could be obtained from an outside source, such as a website, with the web address already embedded within the application. The other way to receive the ID is by receiving the PeerGroup Advertisement from one of the other peers that the peer is connected to, through the Rendezvous Service. Peers can create a new group, and then advertise its existence, by publishing the Advertisement to its directly connected peers, which will then recursively propagate through all other connected peers. Once other peers begin joining this group, an independent P2P network is created.

This notion of multiple P2P networks co-existing under the same umbrella, is a main distinction between JXTA and other P2P technologies. Many peergroups may exist within the Default Peer Group Network, all provide different services or have different membership requirements, and still be able to interoperate because of the standards set forth in JXTA. Once a peer joins a group by knowing the group's unique ID, the group's Discovery Service will be able to search other peers for additional advertisements. Only advertisements published by a Discovery Service of another peer in the same group, will be found. With the potential for thousands of small networks all co-existing, it would be

inefficient and un-secure to have to process advertisements belonging to another group, for which a membership has not yet already been created.

Once a group has been formed, and members have been accepted, members will send out information about their means of communication to all of the other members of the group, so that direct connections can be made between any two peers. Since this is a P2P network, all communication occurs directly between the two peers, with no other involvement from any other peer, once the initial connection has been established. As members join the group, their information is passed through the rendezvous service, and is received by all peers in the group. Because there is no centralized resource to manage group membership, it is up to the individual application to determine when a member of the group is no longer available.

Once a peer has published its communications advertisement, known as a PipeAdvertisement, other peers may receive that advertisement and create a direct connection with the information contained within the XML document. Connections can be made through Pipes or Sockets. Pipes are essentially an extension of the Discovery Service, and messages are sent the same way advertisements are published. The problem with pipes is the limited message size. BiDirectional Pipes are limited to about 64Kb in size, and there can be some delay in transmission. The other alternative, and my choice, was to use sockets. Sockets are from the Java library, and are independent of JXTA. JXTA provides the tools necessary to resolve socket connections from Pipe Advertisements, but once the connection has been established, Java handles the communication between peers. I chose this method because there is no size limitation and communication is virtually free of delay. I have had no issues with transferring files as

large as 10Mb, and the issues involving larger files stem from the Java Runtime Environment not being able to allocate enough memory to serialize an entire large file at once. When large files (large than the currently available amount of memory) need to be sent across sockets, they need to be serialized into a byte stream, written to disk, and then transferred in pieces, and reassembled at the endpoint. This ability to send infinite amounts of data from peer to peer, make a peer group a very powerful entity, similar to a cluster of servers.

Because of the decentralized nature of one of these groups, the application that is using this group must be very robust and be able to handle any form of exception, and heal any wounds in the group, so that if a single peer falls out of the group, the group will still be able to function properly. This requires a very solid protocol, imbedded inside the application, that will allow any peer to act as both a client and server, all at the same time, or as the need should arise for one of those particular roles. Another role of JXTA is to provide redundancy, should a peer fall out of the network. By providing all peer advertisements to all peers in a group, a web of connections is created, instead of a linear set of connections. This provides the redundancy required in a P2P network.

JXTA provides some level of security, but not all means of security function properly. For my application, security was at the bottom of the list, while the issues of the Free Rider problem were prioritized. Security is difficult to manage, again because of the decentralized nature. One of ways to provide security is to use a public and private key, but again, one of the keys must be known prior to being granted a group membership. This again proves to be difficult in a P2P network. Therefore, JXTA has an authentication process which may be used to authenticate members of a grou.

When a user joins a group, the user can either provide no, or bogus credentials, and still join a group, if the group does not have any means of authentication. If the group does contain a means for authentication, the user must be first verified before joining the group. The authentication process works as follows.

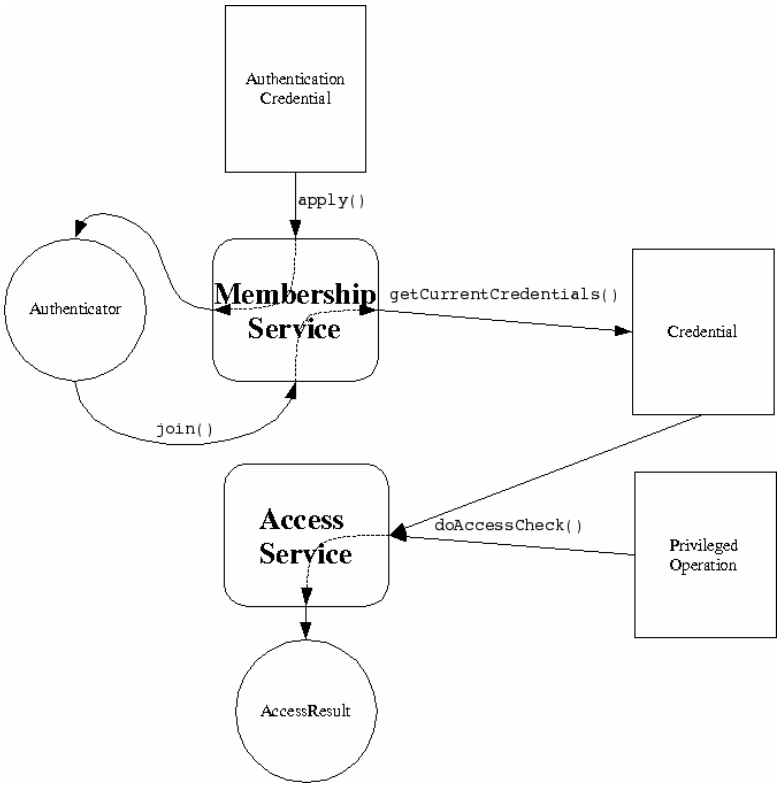

Diagram of an Authentication process in JXTA. (image: API: net.jxta.membership)

Minimizing Free Riding with JXTA

In order to minimize free riding in any application, the costs associated with free riding must outweigh the costs associated with active participation. BitTorrent achieves this relationship by relating download rates to upload rates. The more you upload, the more you can download. The problem with this is that it assumes that all users have something that others might want. If they don't they will download very slowly until they have something to offer. This worked well for BitTorrent because it was primarily

focused on large Linux files, and game applications. This later turned out to be a great way to distribute visual media, both copyrighted and non-copyrighted. But if the pool of applications and files were immense, there would be little chance that more than one or two other peers might have what is needed. In this case there must be some other way to negotiate upload and download rates, if it is highly likely that one peer might not have what another peer wants. A case of simple file transfers involving photographs, which are typically not copyrighted, would be a good example of legal file sharing whereby a user may not have something another user may want.

So my application uses simple chat as a way for users to negotiate file transfers. Assuming two friends are both using the application at the same time, one user might simply want to send the other user a few files, and that would be it. If the first user wanted something in return, they would negotiate through the chat feature until they reached an agreement. In this application, files are pushed, not gotten. Only the user who has the file may send the file. The application does not act like a daemon, and cannot accept incoming requests for files. Both users must be present during the file transfer, and only one file can be sent at a time. Currently there are no restrictions on file type, and the size of the file that can be sent is dependent upon available memory. In future editions of this application, large files would need to be broken into smaller byte arrays, and then sent and reassembled at the receiver's end.

When launching the application, users immediately join a unique group, called *PhotoGroup*. The unique ID of this group is:

urn:jxta:uuid-E3806F9067E142378FDBF2CD4962D26302

This PeerGroupID is theoretically unique, and the chances of having another group with the same ID is 1 in $8.7x10^{40}$. So the chances of a collision are pretty minimal. This ID is encoded in the application, and cannot be changed by the user. Only peers who know this ID may join the group. In order to prevent users who do not have the application from joining the group, the GroupAdvertisement is not published, and therefore other peers cannot see the GroupID. If the GroupAdvertisement were published in an XML document, by the DefaultNetPeerGroup's DiscoveryService, any user of any JXTA application would eventually receive that document, and therefore the unique ID.

So only users that have the application can join the group, and once a user has joined, he or she will eventually receive all of the other peer's PipeAdvertisements. Once a peer has received another peer's PipeAdvertisement, a socket connection is made between the two peers, and chat and file transfers can begin. The amount of time it takes to discover other peers in a group varies by rendezvous setup. The more hops to the peer through rendezvous points, the longer it will take to receive their advertisements. These connections will stay alive indefinitely.

This method seems to be a simple way of preventing free riding, and could be used in many different ways. When a file is transferred, it is simply a byte stream. That byte stream could contain a photograph, a serialized object, or any array of bytes. This would be a perfect medium for a distributed application, or another form of a file sharing application. I compare this system to the way kids trade baseball cards, and that was the initial idea that I used to derive this protocol. When kids trade baseball cards, they cannot free ride. Each kid holds his own cards and only trades when he has reached an

agreement with the other. They do not leave their cards out for others to take, and verbal communication is essential to making the deal.

Since a user really does not have the ability to free ride, the cost of free riding is immense. The cost would be the inability to use the network or any services it would provide. Since one who free rides would not be able to obtain anything from another peer, unless the other peer made a conscious decision to allow the other peer to free ride on their own file library, it would be a long and arduous process to continually ask others to provide them with something for free. It might work for a short time, but is in no way a sustainable method for using the network. There might be an instance where one peer would not want anything from another peer, but would be willing to share their own library with another. File transfers are made through two separate peers, and broadcasting is not an option, although implementing broadcasting to create group chat or group file sharing would be possible to implement in the application, if such a feature were desired.

This inability to free ride, I believe, would greatly reduce the rate of transfer of copyrighted materials, because of the time and energy required to negotiate a transfer. Only files that were explicitly desired would be sought after, and without a daemon, a full list of available files would not be available for others to browse. One of the problems with other P2P file sharing applications was the ability for a user to see every available file from another user, and without the other user's knowledge, download every file that that one user had. This accessibility greatly contributed to the distribution of copyrighted materials across P2P networks. Therefore, Free Ridership and easy accessibility, although not required in a P2P system, are what led to the widespread use of P2P networks for exchanges of copyrighted material.

Conclusion

Free riding on P2P networks is a concern for the general usability and future of this technology. Free riding occurs when users can receive the same benefit regardless of whether they bear any cost. With P2P networks, those costs could be potential criminal prosecution, computer viruses, or a slowdown of processing power. Either way, there are considerable reasons as to why any individual would like to be able to consume the resources of others, and receive all of those benefits free of cost. In order to circumvent this, there must either be costs associated with free riding, or make free riding an unavailable choice. P2P network applications such as BitTorrent addressed this issue by requiring users to upload at the same time they download. This increases the overall download rate for all users on the network. This approach works well for large files, and files that are generally wanted by large numbers of other users. This does not work well for instance with personal photographs, but does for software and gaming modules.

In order to attempt a solution to the problem, the underlying reasons for the dilemma must be understood. Free riding has occurred in many aspects of society, and is not limited to P2P networks. It is a natural phenomenon in any group setting without a central authority. With P2P networks, and their lack of a centralized management service, it can be very difficult to enforce group policies. This leaves the enforcement up to the application itself, and must forcefully limit a user's ability to harm the system by leaching off of others.

My proposed solution was based off of the *baseball card trading* idea used by kids. The solution was implemented in Java, using the JXTA framework, which contains all necessary libraries to build a robust P2P network along with peer services. All users

must come to an agreement with one another, before any file transfer will be made. Instead of the traditional daemon approach, peers *put* files onto another's machine, instead of the requesting peer going and retrieving it. This virtually limits the possibility of any free riding from taking place. Without a central management service, this enforcement must be done from within the application itself. Once the application is launched, it will connect to a predetermined group, and must know of at least one other peer in the group, unless the group is contained within a multicast environment. Once at least one peer, or rendezvous, is located, other peers are subsequently found, and connections between the peers can be made. Once these socket connections are made, any byte stream may be sent from one peer to another, but only after an agreement between the peers has been formed.

Although it is a simple solution in principle, it will help protect the integrity of P2P networks. As other P2P networks grow larger and larger, users feel more obscure, and do not feel as though their lack of participation will go noticed. This is similar to the Voter's paradox, where one vote may not make a difference, but if everyone had the same mentality, there could be no election. Therefore it is imperative that the free riding issue be taken seriously, and directly addressed.

P2P networks have much potential, whether it be for file sharing applications, or distributed applications. The reputation of these networks has been infamous, as the early networks thrived and grew due to the demand for copyrighted media. File sharing is only a specific use for P2P networks, and should not be seen as the most impressive of such. Unfortunately, lawsuits have thwarted development of P2P technologies, and until they are settled, developers will be hesitant to develop any future products, regardless of their

use. The technology though must be able to develop, as networks of P2P communication would result in efficient network usage, powerful distributed systems, and the ability to not rely on a central server computing model. This technology can be very successful, but only if developers are left unhindered, and problems such as free riding are openly addressed, so as to not cause the self destruction of those systems, and the technology itself.

References

1) Digital Millennium Copyright Act of 1998 – U.S. Copyright Office Summary
http://www.copyright.gov/legislation/dmca.pdf

2) The BitTorrent Effect, *Wired Magazine*
http://wired-vig.wired.com/wired/archive/13.01/bittorrent.html

3) Incentives Build Robustness in BitTorrent, *Bram Cohen,* May 22, 2003

4) http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/p5.html

5) Free Riding on Gnutella, Eytan Adar and Bernardo A. Huberman
http://firstmonday.org/issues/issue5_10/adar/index.html

6) Kuhn, Steven, "Prisoner's Dilemma", *The Stanford Encyclopedia of Philosophy (Fall 2003 Edition)*, Edward N. Zalta (ed.),
http://plato.stanford.edu/archives/fall2003/entries/prisoner-dilemma

7) Felkins, Leon, "An Introduction to the Theory of Social Dilemmas", Nov 1994.
http://www.spectacle.org/995/sd.html

8) Knowles, Jeffrey, "The Debate over Sony-Betamax and Peer-to-Peer File Sharing", The Computer and Internet Lawyer, Vol. 22, No. 3, pg.1, March 2005

9) Ngan, Wallach and Druschel, "Enforcing Fair Sharing of Peer-to-Peer Resources"
http://iptps03.cs.berkeley.edu/final-papers/fair_sharing.pdf