Literature Review: Purely Decentralized P2P File Sharing Systems and Usability

Nicky Reid

June 3, 2015

1 Introduction

Peer-to-Peer (P2P) is said to be the next era of the internet, with its decentralized and highly scalable nature allowing for the growth of powerful distributed networking applications. Online file sharing is an activity carried out by millions of users daily, and while file transfer from one person to another is a one-to-one mapping and is therefore an inherently scalable process, much research has been done in the areas of network configuration and file location and searching. There are numerous methods, such as Distributed Hash Tables (DHT) and hierarchical systems, that exploit the efficiency, scalability and dynamically self-organizing nature of a purely decentralized P2P system.

We look at the features of a P2P file sharing system from the end user's perspective, and discuss how intuitiveness, familiarity, usability, security and privacy affect design, development and user experience.

2 Peer-To-Peer

P2P is a set of protocols, an IT architecture and a design philosophy with an emphasis on decentralization (Schoder and Fischbach, 2003). Each node in a P2P network installs a single package which incorporates both client- and server-type code, so that each peer is able to issue queries and serve requests (Vu, Lupu, and Ooi, 2009). P2P is rapidly changing networking as we know it and is said to be the next era of the internet, as decentralization and resource pooling are increasingly being exploited to promote efficiency and scalability in networking applications. In 2003, P2P file sharing applications accounted for 50% of the most downloaded applications on Download.com (Lee, 2003). Karagiannis, Broido, Brownlee, Claffy, and Faloutsos (2003) estimated the fraction of P2P traffic within the internet to be at least 20%, but since most P2P applications now allow for the use of random ports and are thus not detectable as P2P traffic, the actual figure is likely to be much higher.

One of the most significant advantages of a P2P system is its abstraction and encapsulation of client, server and router roles. P2P creates an overlay, or virtual network comprised of logical links, built on top of but not structurally related to the underlying physical network. As functionality for client, server and router are contained in each peer node's implementation, P2P removes the need for a dedicated server. P2P overlay networks are able to form and change dynamically and spontaneously.

Grid computing allows for geographically dispersed computers to contribute their unused processing and memory resources to others, addressing and making use of heterogeneous resource capacities in various machines. It is a form of parallel computing but differs from cluster computing in that in that each node contributes resources on demand to achieve a certain goal, while clusters are usually geographically conecntrated nodes acting together as a single entity (Kaur and Rai, 2014). Ganguly, Agrawal, Boykin, and Figueiredo (2006) propose enabling self-configuring virtual IP networks - similar to the the internet - built on P2P overlays, allowing seamless access to Grid resources spanning multiple domains which are aggregated into a virtual IP network, isolated from the underlying physical network. Their technique, IPOP (IP over P2P,) utilizes P2P's self-configuring, scalable and fault-tolerant nature to achieve overlay routing without centralized administrative control and with acceptable latency overhead.

3 Uses of P2P Networks

A P2P file sharing collaboration tool can help facilitate group work and resource sharing among knowledge workers. A distributed file system aims for consistency and synchronization between files. A collaboration tool synchronizes files in a distributed file system to provides spatially isolated users the ability to collaborate on the same data (Vu et al., 2009).

Eisenstadt, Komzak, and Dzbor (2003) discuss how the real-time online presence of peer-group members in a long distance learning environment could improve students' emotional well-being and academic performance. Their P2P application, BuddySpace, uses online presence, status and geolocation information to facilitate conversations and improved learning among student peers.

Iamnitchi, Ripeanu, and Foster (2004) define "small-world" file sharing communities as spatially concentrated clusters of peers within P2P networks who share similar interests, like scientific communities and peers with interest in similar files. They propose exploiting this phenomenon in user behaviour to address issues like freeloading (see section 7) in order to create more efficient P2P systems.

4 File searching and Decentralization

The two core functions of a P2P file sharing system are the lookup mechanism and the actual file download. The decentralized nature of P2P means that the file transfer process is inherently scalable, while the lookup function is somewhat more complex and problematic (Ratnasamy, Francis, Handley, Karp, and Shenker, 2001). A file searching mechanism should be efficient and effective, returning results accurately and quickly with minimal communication. Locality-aware searching within a large-area P2P network greatly improves response time and reduces network bandwidth usage (Kwon and Ryu, 2003).

One problem with the BitTorrent protocol was the requirement to download file location information (.torrent) files from a server (Pouwelse, Garbacki, Epema, and Sips, 2004). To address this centralized method of file location, magnet links were introduced, reducing the problem of availability and file searching. Magnet links are URIs containing the hash code and thus information location of the required file.

5 P2P Architectures

Ge, Figueiredo, Jaiswal, Kurose, and Towsley (2003) divide P2P architectures into 3 categories. The first, Centralized Indexing Architecture (CIA) depends on a central server to coordinate node participation and maintain the index of available files, and is typified by the Napster network. In the second, Distributed Indexing with Flooding Architecture (DIFA), each peer is responsible for the indices of only its own shared files. A query is flooded through the network, but their limited scope means that some queries may never be satisfied even if the requested files are available in the system. This type of system is typified by the Gnutella network. Finally, the Distributed Indexing with Hashing Architecture (DIHA) is typified by such systems as described in section 5.1 and 5.2 below. Ge et al. (2003) use these three categories to create a mathematical model of P2P file sharing systems. Application of the model led them to conclude that the DIHA architecture scales best as it is not constrained by a central server and there are no query failures such as occur in DIFA.

5.1 Asymmetric Hierarchical System

Kwon and Ryu (2003) propose an Asymmetric Hierarchical system as opposed to a Distributed Hash Table, which is partially centralized with assigned Supernodes and leafnodes. The P2P Asymmetric file Sharing System (PASS) accounts for the varying resource capacities of different machines on the network. In this system, nodes with high available computing power are assigned as Supernodes which take responsibility for all routing within the network. The file location information file is only replicated over three nodes, meaning that a file search can be completed within a maximum, constant number of hops which is independent

of the number of participating nodes. The maximum number of search hops of their current implementation is four, and their system fully takes advantage of locality by first searching in the area which the searching node is in, branching out to other areas if the local lookup fails. PASS requires a substantial amount of memory (estimated to be at least 20mb) at each Supernode to store routing tables, but imposes far less computational overhead than DHT implementations where the routing table is distributed over all nodes. Storing the directory files in central Supernodes diverges somewhat from the serverless nature of P2P, but the advantage of this architecture is that discovery time is greatly reduced while there is still no single point of failure. When a Supernode goes down, the network self-configures and creates another one. Kazaa and Morpheus are two proprietary systems that use this partially decentralized method of assigning Supernodes.

5.2 Distributed Hash Tables

In DHT systems, files are associated with a key (which may be produced, for example, by hashing the file name) and each node stores a certain range of keys. The operation `lookup(key)` returns the identity of the node that currently stores the requested file. Routing algorithms vary among implementations and the scalability of a DHT is dependent on the efficiency of its routing algorithm (Ratnasamy et al., 2001).

The CHORD protocol, given a key, maps it on to a node using consistent hashing, which balances load and involves relatively little reorganization when nodes leave or join the network. Its lookup protocol differs from many others in its simplicity and provable efficiency. CHORD's pure decentralization and load balancing make it robust and appropriate for large-scale P2P networks even in the face of concurrent node arrivals and departures (Stoica et al., 2001).

Pastry performs efficient routing of messages through locality-aware forwarding in wide-area P2P systems (Rowstron and Druschel, 2001). Tapestry improves on Pastry as it provides well-defined probabilistic bounds on routing distances and guarantees that an existing object is reachable (Zhao et al., 2001).

Content-Addressable Network (CAN) uses a hash table to provide a purely decentralized, scalable and fault tolerant indexing system. CAN uses a virtual multidimensional Cartesian coordinate space, dynamically partitioned among participating nodes in order to perform optimized routing (Ratnasamy et al., 2001).

Kademlia minimizes the number of configuration messages routed between nodes, using parallel asynchronous queries to automatically spread configuration messages as a side effect of key lookups, thus avoiding time-out delays from failed nodes (Maymounkov and Mazieres, 2002). TomP2P, the DHT upon which our application is built, uses the same routing algorithm as Kademlia.

The dynamic joining and leaving of peers in a network is referred to as churn. CHORD, Pastry, Tapestry, CAN and Kademlia are said to comprise the next generation in P2P protocols and have demonstrated that in networks with moderate and even significant churn, DHT is highly scalable and is even capable of achieving one- or two- hop lookups with a modest sacrifice of bandwidth (Feldman and Chuang, 2005).

The efficiency of these protocols demonstrates that DHT can be implemented in such a way that is scalable, robust, purely decentralized, load-balanced, dynamically self-organizing, fault tolerant and efficient in routing and file lookups with the ability to exploit locality.

6 NAT

A Network Address Translator (NAT) is very useful in small office and home communities, and is a means by which all users on a subnet share one globally routable IP address (Hu, 2005). To address the shortage of IPv4 addresses for each internet-capable device in the world, many routers are NAT routers which provide users with a unique private IP address. This poses some problems for certain protocols, specifically P2P, where multiple users may need to connect accross NATs. TomP2P, the DHT upon which our application is built, automatically sets up NAT traversal. If a peer is not reachable by its external IP address, it will attempt to set up port forwarding using Universal Plug And Play (UPNP) and NAT Port Mapping Protocol (NATPMP.) Failing this, TomP2P allows for the setting up of distributed relaying.

UPNP and NATPMP allow for the application to configure the NAT router or gateway in order to create port mappings which let the router know which packet is intended for which user in the network (Cheshire and Krochmal, 2013). Relaying allows for peers behind NATs to communicate with each other via another "relay" peer who is not constrained by a NAT.

7 Freeloading: Social Challenges to P2P Uptake

An analysis of the Gnutella network showed that 70% of system users were "freeloaders" who downloaded from the P2P resource pool but did not upload or share any original content (Feldman and Chuang, 2005). Freeloading (or "leeching") is a significant challenge facing P2P systems. A study of end user perspectives on file sharing system features (see section 8) suggested that anti-freeloading mechanisms, such as upload requirements or point-based reward systems, would be detrimental to the system. An anti-freeloading mechanism may limit the user base, which would greatly diminish the success of the system. The results of the study (see Table 1) illustrate that methods to overcome

freeloading ranked at lowest importance. In keeping with the *laissez-faire* nature of P2P networking - that is without user fees, priority user groups or restrictions on use - P2P remains a powerful, accessible and democratic infrastructure. The requirement of a minimum number of shared files, coupled with the need for a central server, are reasons why well-known file sharing applications such as DirectConnect (DC) have relatively low popularity among similar systems (Pouwelse et al., 2004).

Freeloaders provide common-service capacity while not contributing to file-serving capacity (Ge et al., 2003). The load-balancing nature of DHT makes use of the resources of even those users not contributing to the file base. Each node in the P2P network still assists in the maintenance of the hash table and the routing of messages (Feldman and Chuang, 2005).

A file sharing system may depend on popular content being available in order to sustain its user base. Without such content, it would tend to lose some of its user base; this, in turn, may cause the volume of popular files to continue to decrease, causing a downward spiral in the number of users as users lose interest and pull out.

Ge et al. (2003) adjust their mathematical model of P2P file sharing systems to account for freeloaders, concluding that in DIFA, a high ratio of freeloaders significantly degrades performance, as the success probability of a query decreases. They note, however, that CIA and DIHA can support high ratios of freeloaders, taking advantage of spare capacity.

8 End User Perspectives

One challenge facing the user interface design of a file sharing system is that users typically want different levels of sharing modality for different types of files and peers. For example, users may want certain files to be publicly available for any peer to download, or they may perhaps have a set list of friends with whom they would like to share their photos, or a single peer with which they want to share a particular file. Voida, Edwards, Newman, Grinter, and Ducheneaut (2006) address this by introducing a user interface component called a sharing palette, which incorporates aspects of push-orientated and pull-orientated sharing. Push-orientated sharing requires effort on the part of the sender, while pull-orientated sharing requires the receiver to search for and request the file. The sharing palette uses different colours to identify different semantic groups with which files are shared, allowing the user to quickly and easily specify visibility and permissions for files without the need to maintain an access control list.

User Interface designers should be aware of the demographics of their intended users, such as their level of experience, environment, task characteristics and

Table 1: User Rated Features Of A File Sharing System

	Features	Average Importance Score (1-7 increasing scale)
1	Charges no fee	6.50
2	Is fast	6.38
3	Is stable	6.34
4	Is reliable	6.33
5	Can resume loading	6.05
6	Has large file selection	5.78
7	Can exit nicely	5.75
8	Has large user base	5.56
9	Has good search features	5.53
10	Gives error messages	5.32
11	Can organize file as library	5.28
12	Can control spam	5.16
13	Provides server information	5.04
14	Can turn off ad	5.01
15	Has good security features	4.93
16	Supports passive search	4.36
17	Supports direct messaging	4.14
18	Can filter content	4.10
19	Support buddy list	3.99
20	Is open source	3.83
21	Can credit contributors	3.82
22	Has colourful interface	3.57
23	Supports chat	3.46
24	Has points for uploading	3.44
25	Supports only legal files	3.14
26	Has voice connection	2.93

abilities. A multi-modal interface adapts to different users and different contexts Reeves, Lai, Larson, Oviatt, Balaji, Buisine, and Colling (2004).

Quantitative usability requirements such as those that UI design pose are critical but challenging. A study of end user perspectives of P2P file sharing system requirements conducted by Lee (2003) surveyed what end users felt were necessary or desirable features in a file sharing system (see Table 1.) The result was a list of features ranked numerically from most to least important. The results indicated that the most important components in a P2P file sharing system are that it is free to use, has a large selection of files, and has a large user base. The importance of certain features differed significantly between experienced and less-experienced users, with features like chat support, buddy list support, passive search and a colourful interface proving more important to less experienced users. This suggests that these features are useful when learning and remembering how to use the system.

9 Human Computer Interaction

Human Computer Interaction (HCI) is described as the intersection of Computer Science with behavioural sciences, design and media studies among many other fields (Carlisle, 1976). It is involved the in planning, designing and studying of interfaces between computers and end users.

Rogers, Sharp, and Preece (2011) identify 6 main goals for a good application interface:

1. Effectiveness
2. Efficiency
3. Safety
4. Utility
5. Learnability
6. Memorability

The effectiveness and efficiency refer to the ability and degree to which the system aides the user in achieving their goals. A safe interface is so laid out that unsafe activities can't accidentally be carried out (see section 10.) Utility refers to overall satisfaction gained by the user from using the system. Learning and memorability are explored as a component of usability.

Jef Raskin, Macintosh founder and HCI expert, opposes the use of the word "intuitive" with regards to HCI design, saying that something is only truly intuitive if it can be done without any training or rational thought, and is identical to something that the user has experience with. Raskin proposes that the word

"intuitive" be replaced with "familiar" within HCI literature, and that intuitiveness might be the worst quality one could ask for in an interface. Interface designers are required to produce iteratively "better" interfaces; however with the expectation of increased intuitiveness and the restriction of an interface that is familiar to what the user is used to, potential improvements and modifications are limited and sometimes outright rejected. Raskin argues that the word "familiar" in place of "intuitive" will illustrate the boundaries that this requirement places on interface design and improvements, and give reasons for decision makers to be more flexible and open to new ideas (Raskin, 1994). Familiarity of a new interface may lead to a minimal initial learning time, but foregoing this and allowing for an initial learning phase for a product that is not entirely "intuitive" may lead to greater productivity and improved software after that time.

Brooke (1996) suggests that the "usability" of an system refers to its *appropriateness to use*. We couple this with the assumption that a usable interface lends itself to ease in learning and remembering the system.

The usability of a system is apparent to and evaluated by the user long before they have committed to it, and so this property should be evident at the outset (Lee, 2003). Evaluating appropriateness for use requires us to examine the intended use of the system, while evaluating ease of learning requires the examination of the intended user base.

10 Secure User Inteface

Good and Krekelberg (2003) conducted a study on the usability and privacy of the KaZaA network, revealing that the ambiguity of the user interface commonly leads to users carrying out unsecure, undesirable actions such as sharing personal data. Their study shows that many users of KaZaA inadvertently share personal files like their entire email inbox, credit card information and tax reports. It also revealed that other users, whether malicious or misinformed, take advantage of these mistakes, as dummy files with such names as "creditcards.xls" and "inbox.dbx" received download requests. Good and Krekelberg (2003) outline four properties of a safe and usable P2P file sharing system:

1. users are clearly made aware of what files are being offered for others to download.

2. users are able to determine how to successfully share and stop sharing files.

3. users do not make dangerous errors that can lead to unintentionally sharing private files, and

4. users are sufficiently comfortable with what is being shared with others and confident that the system is handling this correctly.

Only two out of a surveyed 12 users could accurately identify which files on their system were being shared. Good and Krekelberg (2003) conclude that the KaZaA interface makes too many assumptions regarding the users' knowledge of file sharing, and violates all four above-mentioned guidelines. They suggest that usability and security be of top priority when designing a file sharing application.

11 Anonymity

An IP address can identify users and their locations. One disadvantage of a purely serverless P2P system is the requirement that peers share their IP addresses to facilitate connection. Significant amount of research has been done in the area of *initiator* anonymity in P2P networks. This involves the spoofing of the IP address of the peer sending a request, by routing the request through various other participating peers so that the originating IP address is hidden. Scarlata, Levine, and Shields (2001) propose further measures to provide *responder* anonymity, building on existing initiator anonymity protocols using multi-cast routing and solving the problem of anonymity degradation over time.

References

John Brooke. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

James H Carlisle. Evaluating the impact of office automation on top management communication. In *Proceedings of the June 7-10, 1976, national computer conference and exposition*, pages 611–616. ACM, 1976.

S Cheshire and M Krochmal. NAT Port Mapping Protocol (NAT-PMP), 4 2013. RFC 6886.

Marc Eisenstadt, Jiri Komzak, and Martin Dzbor. Instant messaging + maps= powerful collaboration tools for distance learning. In *Proceedings of TelEduc03, Havana, Cuba*, 2003.

Michal Feldman and John Chuang. Overcoming Free-Riding Behavior in Peer-to-Peer Systems. Master's thesis, University of California, Berkeley, 2005.

Arijit Ganguly, Abhishek Agrawal, P Oscar Boykin, and Renato Figueiredo. IP over P2P: enabling self-configuring virtual IP networks for grid computing. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, page 10. IEEE, 2006.

Zihui Ge, Daniel R Figueiredo, Sharad Jaiswal, Jim Kurose, and Don Towsley. Modeling peer-peer file sharing systems. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 2188–2198. IEEE, 2003.

Nathaniel S Good and Aaron Krekelberg. Usability and privacy: a study of Kazaa P2P file-sharing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 137–144. ACM, 2003.

Zhou Hu. NAT traversal techniques and peer-to-peer applications. In *HUT T-110.551 Seminar on Internetworking*, pages 4–26. Citeseer, 2005.

Adriana Iamnitchi, Matei Ripeanu, and Ian Foster. Small-world file-sharing communities. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 952–963. IEEE, 2004.

Thomas Karagiannis, Andre Broido, Nevil Brownlee, Kimberly Claffy, and Michalis Faloutsos. File-sharing in the Internet: A characterization of P2P traffic in the backbone. *University of California, Riverside, USA, Tech. Rep*, 1(1):1, 2003.

Kiranjot Kaur and Anjandeep Kaur Rai. A Comparative Analysis: Grid, Cluster and Cloud Computing. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(3):1, 2014.

Gisik Kwon and Kyung Dong Ryu. An efficient peer-to-peer file sharing exploiting hierarchy and asymmetry. In *Symposium on Applications and the Internet Proceedings*, pages 226–233. IEEE, 2003.

Jintae Lee. An end-user perspective on file-sharing systems. *Communications of the ACM*, 46(2):49–53, 2003.

Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

Johan A Pouwelse, P Garbacki, D Epema, and HJ Sips. A measurement study of the BitTorrent peer-to-peer file-sharing system. Technical report, Technical Report PDS-2004-003, Delft University of Technology, The Netherlands, 2004.

J Raskin. Intuitive Equals Familiar. *Communications of the ACM*, 37(9):17, September 1994.

Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. *A scalable content-addressable network*, volume 31. ACM, 2001.

Leah M Reeves, Jennifer Lai, James A Larson, Sharon Oviatt, TS Balaji, Stephanie Buisine, and Colling. Guidelines for multimodal user interface design. *Communications of the ACM*, 47(1):57–59, 2004.

Yvonne Rogers, Helen Sharp, and Jenny Preece. *Interaction design: beyond human-computer interaction*. John Wiley & Sons, 2011.

Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001*, pages 329–350. Springer, 2001.

Vincent Scarlata, Brian Neil Levine, and Clay Shields. Responder anonymity and anonymous peer-to-peer file sharing. In *Ninth International Conference on Network Protocols*, pages 272–280. IEEE, 2001.

Detlef Schoder and Kai Fischbach. Peer-to-peer prospects. *Communications of the ACM*, 46(2):27–29, 2003.

Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.

Stephen Voida, W Keith Edwards, Mark W Newman, Rebecca E Grinter, and Nicolas Ducheneaut. Share and share alike: exploring the user interface affordances of file sharing. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 221–230. ACM, 2006.

Quang Hieu Vu, Mihai Lupu, and Beng Chin Ooi. *Peer-to-peer computing: Principles and applications*. Springer Science & Business Media, 2009.

Ben Yanbin Zhao, John Kubiatowicz, Anthony D Joseph, et al. *Tapestry: An infrastructure for fault-tolerant wide-area location and routing*. Citeseer, 2001.