Peer to Peer: peering into the future

Jon Crowcroft & Ian Pratt,
University of Cambridge Computer Laboratory,
J J Thomson Avenue,
Cambridge CB3 0FD

Jon.Crowcroft@cl.cam.ac.uk, Ian.Pratt@cl.cam.ac.uk

May 23, 2002

Abstract

In this paper, we survey recent work on peer-to-peer systems, and venture some opinions about future requirements for research.

The paper is a survey to support the tutorial at the Networks 2002 Conference and is therefore neither complete, nor likely to be up-to-date by the time you are reading this, since the topic was extremely fast-evolving at the time of writing.

Instead, we try to bring some historical perspective and structure to the area, and to shed light on where the novel contributions, and where it is likely that there are research questions to answer.

1 Introduction

The topic of peer-to-peer networking has divided research circles in two: on the one hand there is the traditional distributed computing community, who tend to view the plethora of young technologies as *upstarts with little regard for, or memory of the past* – we will see that there is evidence to support this view in some cases. On the other hand, there is an emergent community of people who regard the interest as a opportunity to revisit the results from the past, with the chance of gaining widespread practical experience with very large scale distributed algorithms.

The term *peer-to-peer* (P2P) came to the fore very publicly with the rise and fall of Napster[1]. Although there are prior systems in this evolutionary phase of distributed computing (e.g. Eternity[45]), we choose to limit the scope of this survey to the period from "Napster 'til Now"[1].

1.1 Background

Peer-to-peer networking has come to refer to a family of technologies and techniques for organising distributed applications, that has emerged over the past four-to-five years.

The consensus seems to be that a peer-to-peer system can be contrasted with the traditional twenty-five or more year old *client-server* systems: client-server systems are asymmetric; the server is distinguished as running over some longer period of time and looking after storage and computational resources for some number of clients; the end system is therefore a single bottleneck for performance and reliability. Server sites may make use of a number of techniques to increase reliability and performance, such as replication, load balancing and request routing; at some point along the evolution of this thinking, it is a natural step to include the client's resources in the system – the mutual benefits in a large system, especially given the spare resources of today's clients, are clear.

Thus peer-to-peer systems emerge out of client-server systems by removing the asymmetry in rôles: a client is also a server, and allows access to its resources by other systems.

A claim sometimes made about peer-to-peer systems is that they no longer have *any* distinguished node, and thus are highly fault tolerant and have very good performance and scaling properties. We will see that this claim has some truth to it, although there are plenty of peer-to-peer systems that have some level of distinguished nodes, and also plenty of peer-to-peer systems that have performance limitations. In fact, the fault tolerance claims are hardly born out at all in the early instantiations of the peer-to-peer movement. Initial availability figures in Napster[1], Gnutella[2] and Freenet[3] do not compare favourably with even the most humble of web sites!

However, second and later generation systems do indeed provide the claimed functionality and performance gains, and we will see in Pastry[40], Chord[11] and CAN[12] very promising results, and even more recent work

[1] i.e.1998-2002

building applications and services over these systems shows great potential gains[16].

At the same time as peer-to-peer, a number of network researchers have been frustrated in their attempts to deliver new network services within the context of traditional telecommunications or Internet networks[2]. Instead, researchers have built experimental infrastructures by constructing *overlay* systems. An overlay may be a simple as a collection of static IP in IP tunnels, or as complex as a full dynamic VPN ("virtual private network"). Some of these systems are in use in the Active Networks research community. Others are more straightforward in their motivation, such as the GRID communities' requirements for more robust Internet connectivity!

The classical distributed systems community would claim that many of these ideas were present in the early work on fault tolerant systems in the 1970s. For example the Xerox Network System's name service, *Grapevine*[43] included many of the same traits[44] as systems mentioned above and discussed below. Other systems that could easily be construed as architecturally true peer-to-peer systems include Net News (NNTP is certainly not client-server) and the Web's Inter-cache protocol, ICP. The Domain Name System also includes Zone Transfers and other mechanisms which are not part of its normal client-server resolver behaviour.

Also, IP itself is built out of a set of autonomous routers running a peer-to-peer protocol (for example, the routing protocols OSPF and BGP are peer-to-peer, and certainly not client-server, and are designed that way for exactly the reasons given above for peer-to-peer); not only that, but IP was originally an overlay service , implemented above other layered communications system: the PSTN, ARPANET and X.25 circuit switched networks. Indeed, this overlay model keeps re-emerging as network operators deploy faster switched infrastructures such as Frame Relay, ATM and WDM and PONS (Pure Optical Networked Systems) core networks. Most of the peer-to-peer work that we discuss in this document is positioned at the Application Layer, as illustrated in Figure 1.

One can look at differences and similarities between classical client-server and modern peer-to-peer systems on another axis: statefulness. Despite successes with stateless servers, many Web servers use cookies and other mech-

[2] New Internet network level services such as IP QoS in the form of integrated services, and differentiated services, as well as novel service models such as multicast and mobility have proved notoriously hard to build and deploy in their native forms.

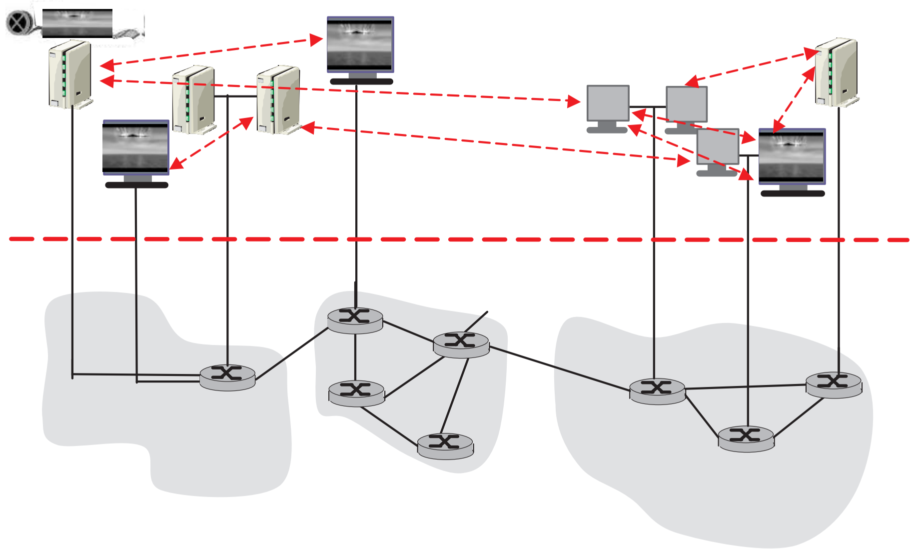

Figure 1: Peer-to-peer architectural situation

anisms (Web Services) to keep state over various transactions with a client. Peer-to-peer systems may keep track of each other although the implications of the privacy/anonymity models we discuss below mitigate this.

Yet another viewpoint from which one can dissect these systems is that of the use of intermediaries. In the Web (and client-server file systems such as NFS and AFS) we use caches to improve average latency and to reduce networking load. Peer-to-peer systems (particularly Freenet[3]) offer intermediary services very naturally, although again, anonymity (and locality) may not necessarily be preserved.

As an aside, we would say that since this topic seems to recur every few years, there appears to be something compelling and unavoidable about peer-to-peer and overlays.

In the next section we take a further look at both this recent, and longer term history in a little more detail.

1.2 History

While we could talk about the 1970s and 1980s work on autonomous distributed systems, or the use of ideas from those in other areas such as IP routing, here we choose mainly to look at the recent peer-to-peer work. [3] Lest readers think *we* are not aware of the longer history of peer-to-peer protocols, we cite the work of 1215[46].

Peer-to-peer "year zero" can effectively be set to Napster[1]. Napster was used heavily to share music content. This is a political hot-potato – the music industry was (and still is at the time of writing) very slow to make content available through the Internet, perhaps due to lack of perception of the market, perhaps due to lack of efficient rights management, and content tracking and protection mechanisms. This seems commercially naïve, given that a large part of the customer base were early network adopters with low tolerance to profiteering. Indeed, the media industry as a whole has preferred to pursue legal mechanisms to protect its legacy hard copy business than to pursue a lower cost (but not necessarily lower margin or profit) soft copy distribution business – the fear of copying would be plausible if it were not for the fact that this risk already exists, so the lack of availability of online music services until fairly recently is all the more remarkable[4].

Napster was not just a system architecture. It also became a company which provided a central directory. This directory is where file listings are uploaded by peers, and where lookups go to. The true behaviour is that of alternating client-server rôles. This is illustrated in Figures 2, 3, 4 and 5.

The central directory server exhibits problems which are not just reliability and performance bottlenecks, but are also a single point of security, political, legal and economic attacks.

To work around all these problems, there has been an evolutionary sequence of research and development of next generation peer-to-peer systems, largely improving on the lookup mechanism, but then moving on to better and better content availability.

The first of these was Gnutella, which dispensed with the central directory and replaced it with a flood based search [2] [35]. While this averts the prob-

[3]It has been remarked, we believe by Roger Needham, that the World Wide Web has created a form of amnesia amongst researchers who find it almost too easy to find recent work online, and forget to check the large body of older literature offline in traditional print in libraries. Luckily, this is being rectified through concerted programs at various publishers, notably the ACM and IEEE to scan and make available online entire back catalogs. However, one can see several instances of the lack of historical perspective today in papers which cite nothing before WWW year zero, submitted to recent conferences in which the authors have been involved.

[4]One might observe that the best way to ensure that kids do something is to ban it!

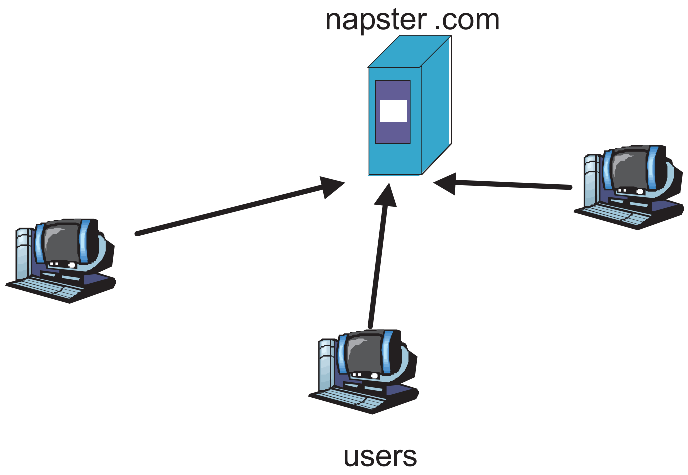

Figure 2: Napster Central Directory Use, Directory Upload

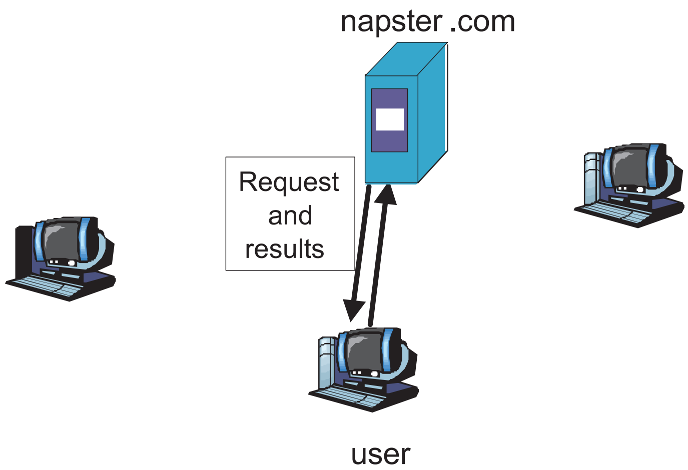

Figure 3: Napster Central Directory Use, Item Lookup

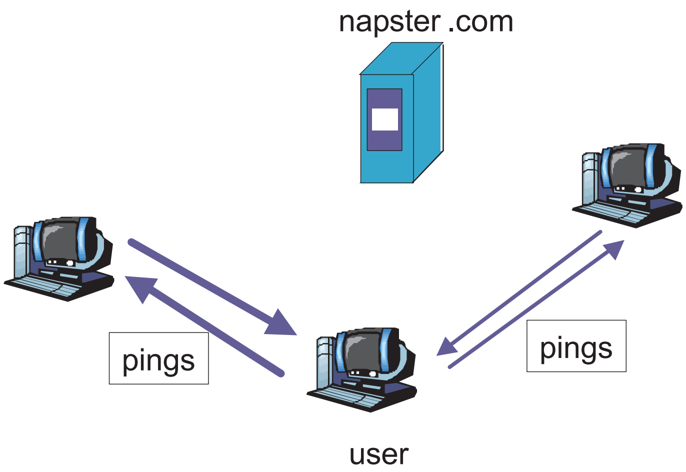

Figure 4: Napster Peer-to-Peer phase, Peer Selection

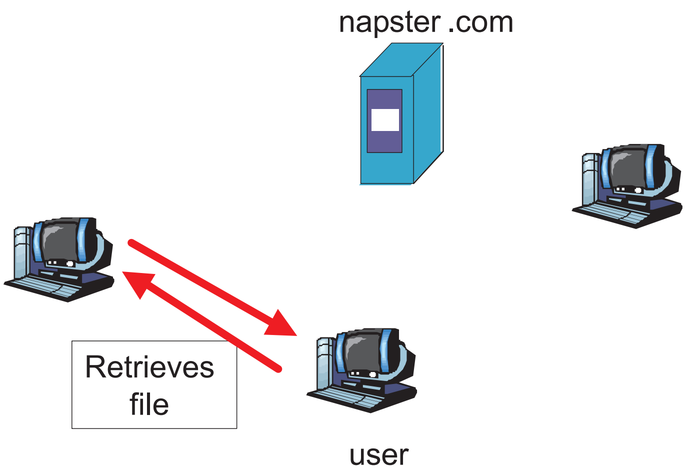

Figure 5: Napster Peer-to-Peer phase, Content Retrieval

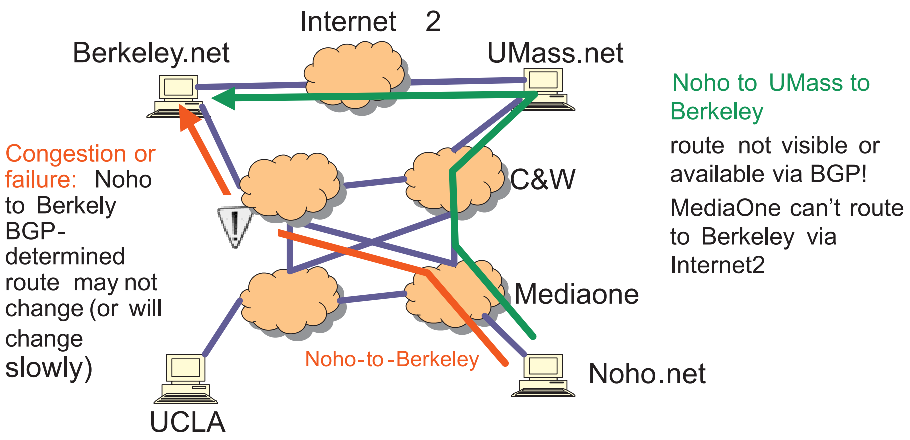

Figure 6: RON

lems above, it has rather pathological network load effects. The next steps were Freenet[3] and Morpheus[5] which add mechanisms to *route* requests to an appropriate node where the content is *likely* to be. The techniques involve two separate mechanisms: the first makes use of hashing on content keys to derive a node number. This is routed to nodes which are where content keys are stored; the second part of the mechanism is to carry out demand caching. The last phase of Freenet that is particularly interesting in that content is encrypted so that servers (oops, sorry, *peers*) do not even *know* what content they store. This is similar to the Eternity[45] file system which predates most peer-to-peer systems by a few years.

A web based service that has many of the same characteristics as Eternity is the Publius system[24], which we mention for interest.

The commonality of the use of Java in some research projects, combined with the observation that there are enough common components in peer-to-peer development activities, has led to some common toolkits for future work – one such is JXTA[4] from Sun.

1.2.1 Overlays

As the ease of development and deployment of peer-to-peer became clearer to the network research community, the use of what one might call "multi-hop" applications, or more normally *overlays*, has started to take off in a big way.

Balakrishnan *et. al.* at MIT have developed the Resilient Overlay Network system[6, 33]. This idea is illustrated in Figure 6, where a number of sites collaborate to find a longer path at the IP "level", which has better properties (such as throughput or loss) at the application level, by dynamically routing via a set of dynamically created tunnels. In parallel, Turner *et. al.* at Washington developed a similar approach for multicast[30].

The difficulties in deployment of native IP multicast have led several groups, most notably the CMU group have developed an "End System only Multicast" system which constructs distribution trees for applications by running a routing algorithm for a degree-constrained multicast tree without including any intermediate IP routers, but with fairly limited negative impact on the efficiency of the distribution tree [7, 8]. Further work used this to implement multimedia conferencing systems and demonstrated the workability and usability of the scheme [9]. A group at Stanford also used this approach for streaming media [28].

Other researchers have used the approach for anycast [14], and server selection [15].

1.2.2 Next Generation

Dissatisfied with the poor resilience and network load caused by super-nodes and flooding, respectively, a number of researchers have been working on distributing the directory for peer-to-peer storage systems.

The main approach is to implement a distributed hash table. There are then a number of design choices as to how the keys are distributed amongst nodes, and how requests are routed to the appropriate node(s).

The principle schemes are Pastry[40], Chord[11], CAN[12] and Tapestry[10].

Chord uses a clever *fingertable* to reduce the size and failure probability of the directory. Pastry uses a similar scheme but with network locality hints. CAN uses several hashes, to map into a multi-dimensional node id space. Tapestry creates an inverted index of nodes.

The success of Pastry has led to its use in Scribe for multicast above the level of a peer-to-peer service (overlay on P2P!)[22]. In turn, the creators of CAN have also investigated its use for application layer multicast[13].

1.2.3 Next Generation Applications

As the scalability of peer-to-peer became apparent, and the resources that could be made available on the large number of systems on Intranets and the public Internet, novel versions of applications on P2P have been developed.

These include object/file stores, such as ASF[16], Oceanstore[18], PAST[29], and databases[31], and even novel VOIP (Voice Over IP) routing systems such as Bazooka[32].

1.2.4 Measurement and Security

There is an interesting tension in measurement work in peer-to-peer systems. Due to their open nature, one can introduce instrumented nodes very easily. Due to the anonymity of some of the systems, it can be hard to ascertain the real behaviour of the whole system at a higher level[19].

Some of the more interesting work has centred on the combined behaviour of users and the peer-to-peer system itself. The seminal paper "Free riding on Gnutella"[20] shows that even mutual benefits do not stop people behaving selfishly.

Other aspects of peer-to-peer that have garnered study are security and collective behaviour [23, 25].

1.2.5 Modelling Work

Alongside measurement, we usually find modelling, and peer-to-peer networking is no different from telecommunications or the Internet in this regard.

Examples of interesting models include papers on the power law of organisation of the system and the small world model of user, content and peer connectivity [21, 26, 27, 34, 36].

1.2.6 Industry and other interest

Peer-to-peer has attracted a lot of industry interest [37, 38, 39].

Possibly the most interesting work to date is the Microsoft Farsite project[42]. Building a massive, highly available, distributed file system out of (massively) unreliable components obviously has its attractions for this organisation.

2 P2P Taxonomy

In this section, we form a rough taxonomy of peer-to-peer systems. Following that, we take a slightly more detailed look at each aspect of peer-to-peer through some examples.

Examples of peer-to-peer system claimed properties include:

Server-less As stated above, *true* peer-to-peer systems auto-configure with a minimum of initial state, and then run without any distinguished node. Having said that, many systems build a structure which may include tuning the peering relationships to improve locality [41] and possibly adding hierarchy to improve the performance of directory lookup (size and robustness) and searching. However, this structure is usually maintained through soft-state, and is largely in place in most peer-to-peer systems to improve performance rather than to add specific functionality.

Arbitrary Scaling Peer-to-peer systems often add resources as they add customers. Thus they should scale (at least at the computing and storage resource level, if not networking) linearly, or better, with number of peers. Of course, many networks are designed assuming client-server traffic, and so it is entirely possible that this performance scaling properties many not be achieved transparently. There are some claims that the "small world" models of human behaviour and interaction, combined with the preferential way in which people link to known locations, lead to power law structures in connectivity. This in turn leads potentially to even better than linear scaling in the system in terms of potential locality (in Freenet for example, however this claim is disputed by some researchers). This means that the study of peer-to-peer systems is partly a sociological, partly an ecological, and partly an economic one.

Symmetry Software for peer-to-peer systems is symmetric. Such software systems are harder to write than client-server systems at the moment as there are few toolkits (although JXTA[4] is one such system). The peer-to-peer programmer needs to build event driven systems, which in general have proved harder to write correctly than client/server. One risk in such systems is that one has to pay attention to synchronisation effects. Another problem is that the programmer must cope with the type of erroneous requests that only server (rather than client) application programmers have to deal with. This makes peer-to-peer programming currently an expert systems programmer task, rather than the separation of concerns that client-server architecture achieves.

File Sharing To a large degree, peer-to-peer is used to share content. Napster and its descendants, have been used for media distribution. Gnutella is also being used by the Genome project for distributing the database.

Processor Cycle Sharing There are few peer-to-peer CPU sharing systems. SETI@Home is one of the few, although (like Napster) it has a super-node. Xenoservers[17] are a system attempting to fill this gap in the *true* peer-to-peer portfolio. SETI@Home is also in use for other applications such as genome database searching and cryptographic cracking.

Anonymity and Resistance to Censorship Some peer-to-peer systems offer privacy at the level that users' identities are masked. Some go further and mask content so that neither peer exchanging data knows who delivered or stores which content. True anonymity is typically two layer, requiring some form of anonymized IP routing system (e.g. onion routing) as well as application layer mechanisms to protect privacy.

High Availability Eternity provides a high degree of availability by striping and replicating objects using FEC like codes over multiple nodes. This provides some level of assurance that single (or some level of multiple) node outages, do not lead to object non-availability.

Directories and Super-nodes and Searching A number of peer-to-peer systems are not strictly peering, but depend on some designated node or server (or cluster of nodes and servers), to head-up the system. Some systems volunteer peers to perform this function. This appears to be hard to avoid if one wants to provide a scalable search or browse capability.

Application Layer Routing Peer-to-peer routing is application layer. It can be used to overlay IP network layer routing to offer more services such as multi-path, multi-metric, multi-cast, mobile optimisation, and other techniques.

Interestingly, it is hard to find any successful decentralised P2P networks. These are the ones we would say have been most successful so far:

- Napster - centralised
- Gnutella - started out decentralised, but then evolved into a centralised system[37].
- KaZaA/Morpheus/FastTrack - centralised (only super-nodes route queries, central registration/authentication/index server)
- DirectConnect - centralised (pretty similar to FastTrack, except the more you share, the more nodes you get to share with)
- EDonkey - same as FastTrack and DirectConnect (why do people spend time doing the same things as everyone else?)
- Freenet - not successful yet, perhaps because it is decentralised
- MojoNation - also not successful yet, perhaps because it is decentralised

It is interesting to ask what is the problem? Is it lack of bandwidth? Or is it because people don't like to share? Later we look at failure to share[20].

Request and Content Routing Peer-to-peer systems still have actual *users* who make requests. These requests (for a resource) have to be routed to a node that can provide that resource (storage, object, CPU etc). Request routing has evolved from super-node indexes[1], and flooding[2], through to sophisticated content-key hash based routing[40, 12].

Overlays As we have observed, peer-to-peer systems carry out application layer routing. One parsimonious use of peer-to-peer techniques is *only* to provide an overlay for IP routing to make up for lack of an IP layer built-in service.

3 P2P Detailed Examples

3.1 File Sharing

File sharing was one of the first great successes of client-server distributed systems. Sun RPC and NFS were ubiquitous in the mid 1980s in research and education labs. Nowadays, we can add Samba and other systems. However, the brittleness of such systems has been remarked on, most famously by Lamport – the failure of a so-called stateless server causes systems to "hang" all around a network. Automounting mitigates against this somewhat but without completely eliminate the problem.

Solutions through server replication are typically expensive.

Peer-to-peer file sharing starts out as simply offering directories of files on *my* system to any and all other systems on a network. There are no guarantees that I will keep these files, or that I will serve them to anyone, or what resources I will give to serving a file to someone. Nor, typically, is there a way in early peer-to-peer file sharing applications, to upload a file, only a directory.

Later systems have rectified these shortcomings, but still struggle with notions of performance, availability and persistence.

3.2 Process Cycle Sharing

Processor cycle sharing is familiar through remote shell and earlier in RJE systems. Since Moore's "law" appears still to describe the growth in new CPU speed over time, we have several hundred million personal computing and workstation machines now on the public internet which have hundreds of MIPs going free all the time.

Attempts to exploit this vast processing resource have been limited since the unit of computation is still quite small, but more importantly, since the latency and bandwidth for code and data distribution to these nodes is typically very poor. This means that the class of computation for which it makes sense to use this huge distributed parallel, unreliable computer is rather restricted to jobs of the type that can be broken in to many relatively small pieces which have quite long individual processing time and perform little communication (i.e. the ratio of CPU step to job cycle step is large). Nevertheless problems in large scale signal processing (SETI@Home, Genome searching, Astrogrid, code cracking) have all been successfully tried, and continue to thrive[5].

[5]SETI is the Search for Extra Terrestrial Intelligence, which takes the signal from many radio telescopes around the world, and attempts to find low entropy strings in it which may or may not represent *artificial* signals from remote intelligent life. Anecdotally, when applied to RF signals sourced from planet Earth, the system failed to find anything remotely intelligent!

Recent moves towards finding locality in peer-to-peer systems might lead to the use of clustering, allowing computations of a less constrained style (e.g. in the Computational Fluid Dynamics area, for example in turbine design, atmospheric and oceanographic modelling).

3.3 Anonymity and Censor-proofness

One of the main novel characteristics of Eternity and subsequent peer-to-peer file systems has been the ability to withstand censorship. This is achieved by several means:

Partition By splitting a file into component parts we make sure that no single site carries the whole file, and a denial of service attack has to run over multiple sites. Later systems made clever use of techniques such as Rabin fingerprinting or other techniques for finding common elements of objects can also ensure that overlapping content between multiple files is exploited to reduce storage costs.

Replication By replicating blocks of a file over multiple sites, we also provide higher availability. Combined with locality information, this can be tuned to reduce latency and increase file sharing throughput, at some cost in terms of consistency in update.

Encryption By encrypting blocks of the file, we make sure that disclosure is unlikely, but also that a node can deny knowledge of the actual content it carries – again, P2P exploits mutual benefit: the argument is that "I might not approve of this content, but I approve of the ability to hide my content in a set of peers, so I will live with what I do not know here!". This is often termed "plausible deniability" and is used by Service Providers to align themselves with the "common carrier" defense against legal liability for content, as with telephony and postal providers can do.

Anonymization By masking the identity of sources and sinks of requests, we can also protect *users* from the potential *censor* or unsavoury *agency*. However we need to mask location information as well as identifiers, as otherwise a traffic analysis may effectively reveal identities, so some form of onion routing is also usually required.

3.4 Directories and Super-nodes

As has been mentioned, there is a trend for users to wish for searching facilities, including underspecified queries that return a mass of partial matches. To provide this there has been a move to create super-nodes which continually scan peers and construct indexes. Of course these distinguished nodes are now highly visible, and this rather reduces the idea of anonymity for themselves, and possibly partly for the set of peers too.

3.5 Overlays

Overlays are a good technique for implementing services that might otherwise belong in a lower layer. Certainly, at one extreme one could imagine that an idealised *anycast* service should be able to do almost all of the things that peer-to-peer storage systems do. However, the burden of putting this sort of information, even at the level of hints, into the network level addressing and routing infrastructure is clearly too high. Instead, an overlay is created. In some cases, the overlay may in fact serve the purpose of using additional information merely to route packets (i.e. the basic network service, but controlled using more complex application layer hints).

3.6 Request and Content Routing

The notion of a "middle box"[47] or *proxy* pre-dates peer-to-peer by a few years. Proxies use meta-data and state information to decide on routing queries to an appropriate one out of many possible (and hopefully, though not always) identical servers. Many of the query routing systems in peer-to-peer architectures take this idea to a logical extreme.

4 Shortcomings and Future Work

In this section, we look at seven areas where we believe additional research questions can be identified based on problems and shortcomings in peer-to-peer systems today. No doubt other researchers have other topics that they can add to this list, and may also feel that some items on this list are perhaps less important, but these are our current favourites:

Measuring and Modelling P2P Systems It seems that there is a great opportunity to build interesting systems for the end user with peer-to-peer technology. However, users then have the opportunity to offer traffic from multiple sources simultaneously. This might start to make normal user traffic look a lot like Distributed Denial of Service ("DDoS") attacks!

Update and Persistence in P2P Storage Systems Most of the classic peer-to-peer storage systems are actually ways to share files, and do not easily permit scalable update, and certainly do not offer Service Level Agreements on persistence.

Computation We really only have very early experience with sharing CPU resources. The economic motives are not the same: the mutual benefit arguments which work for file (music/film) sharing are not so obvious. Even with Digital Rights Management properly in place, file sharing makes efficient use of link bandwidth (upload while I download can take place on most networks since links are usually roughly symmetric in capacity provisioning).

The economic incentives are not so clear with CPU resource sharing, although the Xenoserver[17] project is one attempt to rectify this with accountability and fine-grained charging.

QoS and Accounting Peer-to-peer systems do not typically offer QoS (latency or throughput) guarantees. Even if they did, the accounting system could possibly undermine anonymity. Again, the Xenoserver work attempts to tackle this problem.

Locality versus Anonymity Peer-to-peer systems struggle with the apparently inherent contradiction between offering anonymous sharing of resources, and localisation of service offers. Without a substrate of anonymous packet level routing such as onion routing, it is hard to see how to reconcile this. Having said that, an approach such as recursive use of peer-to-peer, first to build an overlay that achieves anonymized request and response routing, and secondly to do content distribution might be worth exploring.

Evolution of P2P into Lower Layers As peer-to-peer approaches become better understood, we might see the techniques migrate into the infrastructure, as IP has migrated from overlay into native services.

P2P and Ad-hoc Wireless Network Duality Peer-to-peer systems incorporate many techniques for self organisation in an ad-hoc scenario without any pre-built infrastructure. It has been observed by quite a few people informally that these characteristics are shared with ad hoc wireless networks. There is perhaps some mileage in exploring this commonality.

4.1 Measuring and Modelling P2P Systems

Large scale Internet measurement has been recommended/proposed by the US National Academy of Science in their report on "looking over the fence at networking research"[48]. In fact they also recommended looking at overlay systems as a general approach to building research infrastructures.

4.2 Update and Persistence in P2P storage systems

The Eternity file-system incorporated file block striping and replication in a peer-to-peer system some time ago. This surely needs revisiting, especially in the light of recent advances in the design of redundancy coding algorithms such as those used in layered multicast schemes.

4.3 Computation

The critical difference between file objects and computations is that the former have a simple partition function. Computations are notoriously hard to distribute. There are some interesting cases which are well understood. The various world wide GRID research programmes have a number of such problems which entail very long timescale and large computations, which have some resemblance to the SETI@Home problem.

4.4 Congestion Control versus QoS and accounting

Recent peer-to-peer systems have a stage during the phase of setting up peering relationships when they use some simple tool such as the ubiquitous "ping" program, to select preferred peers by *proximity*.

However, such an approach is error prone, and more importantly, leads to a succession of local optimisations, rather than a solution optimised for global performance.

It is clear that regardless of whether we want an adaptive system that responds to congestion (e.g. via Explicit Congestion Notification or pricing) or an engineered solution based on some type of signaling, there is a long way to go in providing performance guarantees into P2P. Certainly, a market driven approach would fit the spirit of P2P best. However, as measurement projects produce more results to characterise the behaviour of peer-to-peer sources, it may be possible to develop traffic engineering, and even admission control and scheduling algorithms for the nodes.

4.5 Locality versus Anonymity

As mentioned before, there are a number of driving factors that are reducing the anonymity characteristic of peer-to-peer systems, at the very least, their immunity to traffic analysis is being lost.

However, developing an onion routing system as a peer-to-peer application, and overlaying this with any other peer-to-peer application seems like an elegant, but as yet untried solution to this tension.

4.6 Evolution of P2P into Lower Layers

We observed that successful overlay systems sometimes migrate over time into the infrastructure. However, many peer-to-peer systems are complex, and devising a minimal service enhancement in the lower levels that would support their algorithms is an interesting challenge. The IETF FORCES working group has been working on remote IP router control (separating out forwarding and routing). However, we would need more than this if very general P2P intermediary functions are to be carried out at wire or fibre (or lambda!) speed. Filtering and processing of content keys would be needed. Such a system is a long way from our current capabilities or understanding.

4.7 P2P and ad hoc wireless network duality

Peer to peer systems are a highly attractive way to build dis-intermediate content services. However, resilient mechanisms that have emerged in the P2P community from Freenet and Eternity, and via Content Addressable

Networks have weird effects on network utilisation and are prohibitive for wireless ad hoc networks. We propose to select some of these mechanisms, such as CAN, Chord, Pastry, and Mixnet, Publius, Xenoservers, and MojoNation and mitigate these effects. This could be achieved by:

1. making them "load aware";
2. distributing load information and implementing a distributed congestion control scheme;
3. adding a topographical metric based on location specification and using anycast.

Most current second generation peer-to-peer overlay systems use content request routing schemes that relay on arbitrary hash functions to map from some attribute (content) to node identifier – in the case of CAN there are as many hash functions as there are dimensions on the hyperspace.

None of these takes account of actual location. These functions are largely aimed at providing robust and anonymous content location, and a high degree of cooperative, distributed indexing. There have been some attempts to modify schemes to take account of actual node proximity to the clients/requesting nodes. Freenet and other systems migrate copies in a manner similar to demand caching, as often used in the World Wide Web.

It is likely that next generation systems will use actual location and scope information to influence routing functions so that content is initially placed and requests are routed to copies that have proximity on a number of QoS axes – these would be delay, as well as potentially throughput and loss-based, but could also include battery budget considerations for wireless users. Thus the distribution of replicas in the service infrastructure would evolve to meet the user demand distribution, optimising use of the scarce wireless resources to better match user concerns.

Note that some of the resulting algorithms and heuristics serve a dual purpose: they can be used for the actual packet routing decisions too! The duality between self-organisation in P2P content location and routing, and ad-hoc wireless routing has been commented on in the past, but not directly exploited to our knowledge.

5 Acknowledgements

The authors gratefully acknowledge discussion with Jim Kurose and Don Towsley, and the use of materials generated by their class on peer-to-peer from last year in the Computer Science department at the University of Massachusetts at Amherst. Thanks are also due to Ian Brown, Ken Carlberg, Tim Deegan, Austin Donnelly and Richard Mortier for extensive proof reading.

References

[1] Napster. “The Technology Behind Napster”, copy of this is cached at: http://gaia.cs.umass.edu/cs591/hwpa/napster.htm

[2] The Gnutella Protocol, version 0.4, http://www.clip2.com/GnutellaProtocol04.pdf

[3] “Freenet: A Distributed Anonymous Information Storage and Retrieval System”, I. Clarke, B. Wiley, O. Sanberg, T. Hong, in Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, Springer-Verlag LNCS 2009, ed. by H. Federrat, Springer: New York (2001). http://freenet.sourceforge.net/index.php?page=icsi-revised

[4] “Sun’s Project JXTA: A Technology Overview”, L. Gong, http://www.jxta.org/project/www/docs/TechOverview.pdf

[5] “Morpheus out of the Underworld”, K. Truelove, A. Chasin, http://www.openp2p.com/pub/a/p2p/2001/07/02/morpheus.html?page=1

[6] “The Case for Reslient Overlay Networks”, D. Anderson, H. Balakrishnan, F. Kaashoek, R. Morris, Proc. HotOS VIII, May 2001, http://nms.lcs.mit.edu/papers/ron-hotos2001.html

[7] “A Case For End System Multicast”, Y. Chu, S. Rao, H. Zhang, Proceedings of ACM SIGMETRICS , Santa Clara,CA, June 2000, pp 1-12. http://www.cs.cmu.edu/afs/cs/project/cmcl-yhchu/www/Sigmetrics2000/sigmetrics-2000.ps.gz

[8] “Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture”, Y. Chu, S. Rao, S. Seshan, H. Zhang, Proc. ACM Sigcomm 2001, http://www.acm.org/sigs/sigcomm/sigcomm2001/p5-chu.pdf

[9] "Overcast: Reliable Multicasting with an Overlay Network", J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr., Proceedings of OSDI'00. http://gaia.cs.umass.edu/cs791n/Jannotti00.pdf

[10] "Tapestry: A Fault Tolerant Wide Area Network Infrastructure", B. Zhou, D. A. Joseph, J. Kubiatowicz, Sigcomm 2001 poster and UC Berkeley Tech. Report UCB/CSD-01-1141. http://www.cs.berkeley.edu/ ravenben/publications/CSD-01-1141.pdf

[11] "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications", I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan, ACM Sigcomm 2001, http://www.acm.org/sigcomm/sigcomm2001/p12.html

[12] "A Scalable Content-Addressable Network", S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, ACM Sigcomm 2001, http://www.acm.org/sigcomm/sigcomm2001/p13.html

[13] "Application-level Multicast using Content-Addressable Networks", Sylvia Ratnasamy, Mark Handley, Richard Karp, Scott Shenker In Proceedings of NGC 2001 http://www.icir.org/sylvia/can-mcast.ps

[14] "Application-Level Anycasting: a Server Selection Architecture and Use in a Replicated Web Service", E. Zegura, M. Ammar, Z. Fei, and S. Bhattacharjee. IEEE/ACM Transactions on Networking, Aug. 2000. ftp://ftp.cs.umd.edu/pub/bobby/publications/anycast-ToN-2000.ps.gz

[15] "Evaluation of a Novel Two-Step Server Selection Metric", K. M. Hanna, N. Natarajan, and B.N. Levine, in IEEE ICNP 2001. http://www.cs.umass.edu/h̃anna/papers/icnp01.ps

[16] "Fault Tolerant Replication Management in Large Scale Distributed Storage Systems", R. Golding, E. Borowski, 1999 Symposuium on Reliable Distributed Systems, http://gaia.cs.umass.edu/cs791n/peters_paper.pdf

[17] "Xenoservers: Accounted Execution of Untrusted Code", D. Reed, I. Pratt, P. Menage, S. Early, N. Stratford, Proceedings, 7th Workshop on Hot Topics in Operating Systems HotOS 1999 http://www.cl.cam.ac.uk/Research/SRG/netos/xeno/

[18] “OceanStore: An Architecture for Global-Scale Persistent Storage”, J. Kubiatowicz, et al., in Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), November 2000. http://oceanstore.cs.berkeley.edu/publications/papers/pdf/asplos00.pdf

[19] “A Measurement Study of Napster and Gnutella as Examples of Peer-Peer File Sharing Systems”, P. Gummagi, S Sariou, S. Gribble, ACM Sigcomm 2001 poster.

[20] “Free Riding on Gnutella”, E. Adar, and B. Huberman, First Monday, Vol. 5, No. 10, http://www.firstmonday.dk/issues/issue5_10/adar/

[21] “Search in Power-Law Networks”, L. Adamic, A. Lukose, R. Lukose, and B. Huberman, http://www.hpl.hp.com/shl/papers/plsearch/

[22] “SCRIBE: The Design of a Large-scale Event Notification Infrastructure”, A. Rowstron, A-M. Kermarrec, P. Druschel and M. Castro, Submitted June 2001. http://www.research.microsoft.com/ãntr/PAST/scribe.pdf

[23] “Security Aspects of Napster and Gnutella”, S. Bellovin, Distinguished lecture at Polytechnic U., Real-Audio presentation (1 hour). http://media.poly.edu/RealMedia/electrical/eesem10_26.ram. Slides available at http://www.research.att.com/s̃mb/talks/NapsterGnutella.

[24] “Publius: A Robust, Tamper-evident, Censorship-resistant, Web Publishing System”, Marc Waldman, Aviel D. Rubin and Lorrie Faith Cranor, Proc. 9th USENIX Security Symposium, pp 59-72, August 2000. http://publius.cdt.org/publius.pdf

[25] “Crowds: Anonymity for Web Transactions”, M. Reiter and A. Rubin, ACM Transactions on Information and System Security, November 1998

[26] “Responder Anonymity and Anonymous Peer-to-Peer file sharing”, V. Scarlata, B.N. Levine, and C. Sheilds, ICNP 2001. http://signl.cs.umass.edu/pubs/scarlata.apfs.ps.gz

[27] “The Small-World Phenomenon: An Algorithmic Perspective”, Jon Kleinberg, http://www.cs.cornell.edu/home/kleinber/swn.ps

[28] "Streaming Live Media over a Peer-to-Peer Network", H. Deshpande, M. Bawa, H. Garcia-Molina, http://dbpubs.stanford.edu:8090/pub/2001-31

[29] "Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility", A. Rowstron, P. Druschel, SOSP 2001. http://www-cse.ucsd.edu/sosp01/papers/rowstron.pdf

[30] "Routing in Overlay Multicast Networks", S. Shi and J. Turner, Technical Report, TR 01-19, Washington University, http://www.arl.wustl.edu/arl/Publications/2000-04/wucs0119.pdf

[31] "What Can Peer-to-Peer Do for Databases, and Vice Versa?", S. Gribble, A. Halevy, Z. Ives, M. Rodrig, D. Suciu, in the Fourth International Workshop on the Web and Databases (WebDB '2001). http://www.cs.washington.edu/homes/gribble/papers/p2p.ps.gz

[32] "A peer-to-peer VOIP network", http://phonebazooka.com/

[33] "Resilient Overlay Networks", D. Anderson, H. Balakrishnan, F. Kaashoek, R. Morris, Proc. 18th ACM SOSP, Banff, Canada, October 2001. http://nms.lcs.mit.edu/papers/ron-sosp2001.html

[34] "Using the Small World Model to Improve Freenet Performance", H. Zhang, A. Goel, R. Givindran, Proceedings of IEEE Infocom, June 2002.

[35] "Peer-to-Peer Architecture Case Study: Gnutella", M. Ripeanu, in proceedings of 2001 International conference on P2P computing http://people.cs.uchicago.edu/m̃atei/PAPERS/P2P2001.pdf

[36] "Determining Characteristics of the Gnutella Network", B. Gedik, http://www.cs.gatech.edu/people/home/bgedik/mini-projects/mini-project1/FinalReport.htmq

[37] Limewire tech papers, http://www.limewire.com/index.jsp/tech_papers

[38] Industry viewpoint: http://www.openp2p.com/ and http://conferences.oreilly.com/p2p/

[39] " Peer-to-peer, Harnessing the Power of Disruptive Technologies", Edited by Andy Oram, pub. O'Reilly, Mar 2001, ISBN 0-596-00110

[40] "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems" A. Rowstron and P. Druschel, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.

[41] "Exploiting Network Proximity in Peer-to-Peer Overlay Networks", M. Castro, P. Druschel, Y. C. Hu, A. Rowstron, http://www.research.microsoft.com/ãntr/Pastry/pubs.htm, submitted for publication.

[42] "Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs", William J. Bolosky, John R. Douceur, David Ely, and Marvin Theimer http://research.microsoft.com/sn/farsite/ ACM SIGMETRICS 2000.

[43] "Grapevine: An Exercise in Distributed Computing", Andrew D. Birrell, Roy Levin, Roger M. Needham and Michael D. Schroeder, Communications ACM, vol. 25, no. 4, pp. 260–274, Apr. 1982.

[44] "Experience with Grapevine: The Growth of a Distributed System", Michael D. Schroeder, Andrew D. Birrell and Roger M. Needham, ACM Transactions on Computer Systems, vol. 2, no. 1, pp. 3–23, Feb. 1984.

[45] "The Eternity Service", R. J. Anderson. In Proceedings of Pragocrypt 96, 1996,

[46] "The Magna Carta" http://www.nara.gov/exhall/charters/magnacarta/magmain.html

[47] "Middle Box Taxonomy" Brian Carpenter, and Scott Brim, April 2001. Work in progress.

[48] "Looking Over the Fence at Networks: A Neighbor's View of Networking Research" Committee on Research Horizons in Networking, Computer Science and Telecommunications Board, National Research Council, USA http://www.nap.edu/books/0309076137/html/