Peer-to-Peer Keyword Search: A Retrospective

Patrick Reynolds[1] and Amin Vahdat[2]

[1] GitHub, Inc.
[2] University of California, San Diego and Google, Inc.

Abstract. Peer-to-peer systems have been an exciting area of research. Challenges in building them have included scalability, reliability, security, and—of particular interest to these authors—search functionality. This paper surveys some of the history of the field, looks at the lasting impacts of peer-to-peer research, and provides at least one view of where we go from here.

1 Introduction

2001 was an exciting time for research on peer-to-peer systems. Napster [38] had recently been shut down for abetting widespread copyright violation [18]. Gnutella [41] and Freenet [9,10] survived but used completely unstructured overlays that compromised performance and search completeness. Other peer-to-peer systems reused Napster's ideas or Gnutella's protocol, and these too were eventually shut down [25,34,35,55]. Peer-to-peer systems needed both better protocol design and applications other than file sharing.

Chord [46], CAN [39], Pastry [43], Tapestry [57], and Kademlia [36] collectively introduced the idea of structured, decentralized overlay networks. The abstraction they implemented was a distributed hash table, or DHT, which mapped fixed-size, opaque keys to arbitrary values. All of the DHTs were efficient, requiring just $O(\lg n)$ operations to look up a key in an n-node system.

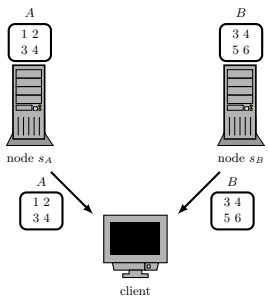
Soon after, distributed file systems like the Cooperative File System (CFS) [14] and PAST [15] were built on top of DHTs. At least initially, however, neither the DHTs nor the distributed file systems provided any search functionality.

In 2002, we set out to design a complete, efficient keyword search service [40] for applications based on DHTs.

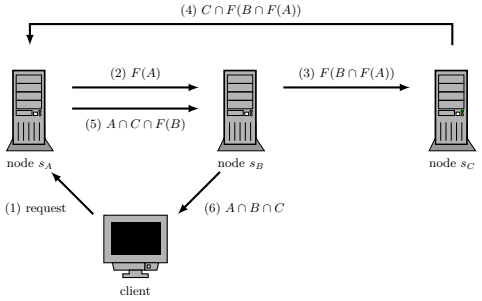
This paper revisits our original paper, surveys other interesting research on peer-to-peer keyword searching, examines the state of peer-to-peer technologies today, and identifies some of the lasting impacts that peer-to-peer networks have had.

1.1 A definition

In 2001 as well as today, the definition of a peer-to-peer system is a fuzzy one. The most prominent distinguishing characteristic of a peer-to-peer system is that nodes owned by individuals make up the bulk of both the consumers (clients)



(a) A simple approach to "and" queries. Each node stores a list of document IDs corresponding to one keyword.



(b) Bloom filters help reduce the bandwidth requirement of "and" queries. $F(A)$ is a Bloom filter representing the set A, and $B \cap F(A)$ is the set of all elements from B that matched the Bloom filter $F(A)$.

Fig. 1. Adding a Bloom filter to DHT search

and providers (servers) of the service. A peer-to-peer service generally uses bandwidth, storage, and/or CPU time provided by users of the service. Further, a robust peer-to-peer service should be decentralized enough that no administrator can disable the system.

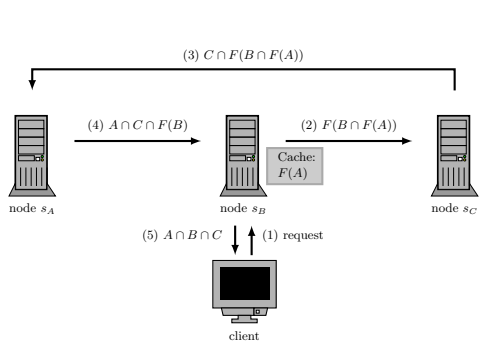
Some prominent peer-to-peer services, including Napster, Skype prior to 2012, and all BitTorrent search pages, rely on centralized components that can or could be administratively disabled. We still consider them peer-to-peer services, albeit ones with room for improvement.

2 Efficient peer-to-peer searching

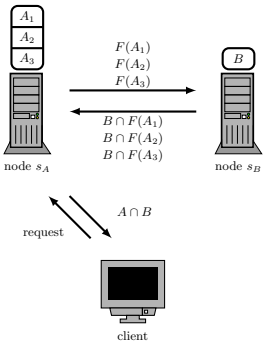
A good peer-to-peer search feature needs to be decentralized, efficient, and complete. Early peer-to-peer systems were not. Napster's search feature was centralized, which made it unscalable and easy to shut down. Gnutella's search feature was both inefficient and incomplete: it flooded queries throughout the network up to a fixed number of hops away from the requester, limited by a time-to-live (TTL) value. It could not locate any resources beyond that number of hops. Early DHTs did not provide search at all.

2.1 Our contribution

The simplest implementation of keyword search on a DHT uses an *inverted index*, as shown in Figure 1(a). The DHT maps each keyword to a list of document IDs, corresponding to the documents that contain the keyword. A client performing a search for one keyword retrieves the list of document IDs associated with that keyword. To perform a conjunctive ("and") query of multiple keywords, the client



(a) $F(A)$ is already in the cache at node s_B.



(b) Nodes send their data one chunk at a time until the desired intersection size is reached.

Fig. 2. Caching and incremental results

retrieves the list of document IDs for each keyword and locally calculates the intersection. This approach is clearly decentralized and complete, but it is not especially efficient. If the user searches for keywords that individually appear in many documents but that rarely appear together, then downloading the entire list of document IDs for each word is wasteful.

Our paper proposed three optimizations to this simple approach: Bloom filters, caching, and incremental results. For these to work, we changed the protocol from Figure 1(a) so that intersections are calculated within the DHT, as shown in Figure 1(b). In this revised protocol, the client sends the entire query—e.g., "efficient AND network AND protocols"—to the node hosting the first keyword. That node sends the remaining words in the query, along with a list of document IDs for the first keyword, to the node responsible for the second keyword. This second node calculates the intersection between the second keyword's set of document IDs and the set of document IDs it received from the first node. Forwarding continues in this fashion until all keywords have been considered (all sets of document IDs have been intersected), at which point the last node sends the final list—IDs of the documents containing all the keywords—back to the client.

Bloom filters [5, 19, 37] are a compact but lossy way to represent membership in a set. They answer the question "Is element x in the set," occasionally returning false positives—a value of "true" even when x is not in the set. In our search system, we used them as a compact way to represent the set of document IDs for documents that contain a given keyword. Instead of transferring entire lists of document IDs, the system transferred Bloom filters representing those lists.

Caching reduces both network traffic and latency by avoiding the transfer of information that has been used recently. Our system cached Bloom-encoded lists of document IDs corresponding to a given keyword. Figure 2(a) shows an example where node s_B already has the Bloom filter $F(A)$; the client sends the query directly to node s_B, eliminating one hop and the cost of transferring $F(A)$. Each cache hit eliminates one hop and the associated transfer cost. Caching allowed us to use larger Bloom filters, with a correspondingly lower false-positive rate.

Incremental results take advantage of the fact that users often only want a few results—say, the best ten—even when many documents match their query. At each hop, our system transferred only a few Bloom filter-encoded document IDs at a time, rather than the whole list. This optimization tied the cost of answering a query to the size of the answer the user wanted, rather than to the total number of results available. Figure 2(b) shows an example in which three chunks of the document list A are sent, and then the requested result size is reached.

In addition, our system incorporated the idea of virtual hosts [14]. Peer-to-peer systems are often heterogeneous in their capabilities: nodes differ in their available CPU power and network capacity. Assigning more-capable nodes a larger number of virtual hosts allowed us to take advantage of their additional capacity.

We measured the effectiveness of our optimizations using a corpus of 105,593 HTML documents and a trace of 95,409 web searches. For each query, we calculated the number of bytes transferred and the total time for the system to satisfy the query. Taken together, our optimizations reduced the time to answer a query by about an order of magnitude.

2.2 Similar work

Other research projects tackled the problem of peer-to-peer keyword search differently. This section explores three of those systems.

PIER. PIER [26] implements relational queries on top of a DHT, scalable to at least thousands of nodes. Relations composed of tuples are stored in the DHT; each tuple is stored according to its namespace and primary key. Joins are performed by retrieving the relevant relations with multicast queries, then storing them back in the DHT in temporary relations keyed (in the DHT) by the appropriate column for the join. Joins based on Bloom filters are provided, as well, to reduce the number of tuples that must be transferred to temporary tables.

PIER provides expressiveness well beyond keyword-index systems, and it can easily be used to implement keyword queries. However, we believe that the number of nodes involved in each query and the number of bytes transferred among those nodes will be much higher than in a purpose-built keyword-search system.

PlanetP. PlanetP [13] is a file sharing system built with an emphasis on searchability. Each node in the system hosts the documents its user wishes to share, as opposed to a DHT where those documents would be copied onto unrelated nodes. Each node builds an inverted index of the keywords in the documents it shares, then computes a Bloom filter to represent the set of keywords found in at least one document on that node. All nodes in the system flood their Bloom filters to all other nodes, then gossip updates as the Bloom filters change. When a user wants to perform a query, PlanetP uses the Bloom filters to figure out which remote nodes to contact. The false-positive aspect of Bloom filters adds an essentially harmless probability that some nodes will be contacted and return zero results.

OverCite. OverCite [47] is a distributed version of CiteSeer, which is a library and search engine for scientific research papers. OverCite distributes responsibility for storing, indexing, and searching papers among all participating nodes using a DHT. The responsibility for indexing those documents is divided among k partitions; the nodes responsible for a partition in the DHT index the documents contained in the partition. The rationale for using partitions is to avoid making every search query a broadcast. Each partition is $1/k$ the size of the full index and is replicated n/k times, so each node must store $1/k$ of the total index and will receive $1/n$ of the query load. Two additional optimizations are proposed: replicating author and title metadata to all nodes, and replicating common search terms on all nodes. Both optimizations allow certain queries to be answered by a single node, rather than sending each query to k nodes to cover all partitions.

3 Where we are now

Fifteen years have passed since Napster was released, and ten years have passed since the first peer-to-peer search papers were published. Where are we now?

Very few prominent Internet services or businesses use peer-to-peer systems. The most successful peer-to-peer system by far is file sharing, accounting for 10% to 20% of traffic during peak hours, on fixed (not mobile) networks [52]. The only other peer-to-peer system that is a household name in the U.S.—clearly a subjective distinction on our part—is Spotify. Skype used to rely on a peer-to-peer overlay but no longer does. Peer-to-peer networks are also popular for live and on-demand video streaming services in China.

Spotify's network is a hybrid of client-server and peer-to-peer protocols [29]. Each client keeps a persistent connection open to a Spotify server, through which it can browse available content, learn about other clients currently online, and receive the first fifteen seconds of any song where low latency is required. Desktop (not mobile) clients retrieve full songs directly from other clients whenever possible. Clients form an unstructured overlay network and use flooding queries with a TTL of two to search for songs.

Spotify's central servers make the peer-to-peer protocol simpler by providing lists of online clients and by providing a backstop data source for content not present within two overlay hops. The peer-to-peer network offloads the majority of song download traffic.

Skype originally used peer-to-peer technology to provide a user directory and NAT traversal [45]. Audio and video streams and chat messages go directly from one user to another whenever possible. Each *supernode* maintains a list of logged-in users and a current IP address for each one. Supernodes also assist in routing calls and chat messages to clients whose device is suspended or behind a firewall. In 2012, Skype moved all supernode functionality off of end-user PCs and onto dedicated *mega-supernodes* in data centers run by Microsoft [23, 24].

Two large, live-video streaming services in China, PPS.tv (PPStream) and Funshion, use peer-to-peer technology. PPStream uses the DONet protocol, which is an unstructured, mesh-style multicast [8, 56]. Funshion is based on BitTorrent [20].

3.1 Disadvantages of peer-to-peer systems

From the perspective of running an Internet service, peer-to-peer systems couple some desirable properties with some serious challenges. The most appealing property of a peer-to-peer system is that it lets a business use customers' bandwidth and computing resources without paying for them. However, in most cases, the challenges overwhelm this potential cost savings:

- Most last-mile connections have asynchronous bandwidth, heavily favoring downloads over uploads. A bandwidth-limited peer-to-peer service like file sharing or video streaming must therefore find many uploaders for each downloader.
- End-user network connections, especially when geographically distant from each other, have roughly 1,000 times lower bandwidth and 1,000 times higher latency than connections within a data center.
- Using customers' computers in unexpected ways can lead to bad publicity.
- Peer-to-peer services require users to download and install software, which providers must write and maintain for each target platform. An unmodified web browser can access centralized services, but it cannot access peer-to-peer services.
- Customers turn off their computers more often than data centers do.
- Mobile devices, which account for a rapidly increasing fraction of Internet traffic [52], magnify all of these issues: they are battery- and CPU-constrained, their bandwidth is usually slower and often metered, and they cannot run the same client software as desktop PCs.
- Customer-owned nodes are not trustworthy.
- Services with data retention or wiretap requirements will likely find it easier to comply with the law if infrastructure is centralized [21].

Some of the limitations of customer-owned computing resources, particularly security and reliability, can be overcome with software, at a cost of additional

redundancy and complexity. Others, including poor network connectivity and customers' aversion to installing additional software, are more stubborn. Overall, the risks and costs of harnessing customer-owned resources are almost always higher than the risks and costs of running services in professionally managed data centers.

Further, hosted computing, storage, and bandwidth resources have gotten dramatically cheaper per unit in the last decade [54]. Utility computing, both infrastructure as a service (IaaS) and platform as a service (PaaS), allows new Internet services to start out cheaply with just a fraction of a single server, without relying on peer-to-peer systems. Both Amazon and Google offer a resource-limited tier of hosting services for free. At this point, even cash-strapped startups favor starting new Internet services in data centers.

Web search in particular is a service that favors data centers over peer-to-peer systems. Our test corpus contained 105,593 documents, while Google's contains around fifty billion. A fully featured web search service does things like spelling suggestions, autocomplete, instant search, location awareness, and personalization that are not amenable to caching or representation in Bloom filters. Our and others' search protocols mask wide-area bandwidth and latency constraints well enough for the demands of a file-sharing service, perhaps, but not well enough to be competitive with a modern search engine built with dedicated computing resources.

3.2 Advantages of peer-to-peer systems

In spite of the challenges, the most popular file-sharing services still use peer-to-peer systems [2, 11, 30, 52]. We believe that this fact is due almost entirely to censorship resistance. No matter how widely replicated an infrastructure-based service is, it is still vulnerable to a well placed letter, phone call, domain seizure, or DMCA takedown notice. Peer-to-peer systems have proven far more resilient. In 1993, John Gilmore said, "The Net interprets censorship as damage and routes around it" [17]. Peer-to-peer systems codify that ideal in software.

Much of the censorship exercised against web sites and peer-to-peer systems is copyright related. Publishers of movies, songs, books, and software, among others, hope to preserve their ability to charge for each copy made of their copyrighted works. However, censorship happens for other reasons, too. For example:

- **Political** - In 2008, both the McCain [53] and Obama [16] campaigns had political messages removed due to DMCA takedown notices. Also in 2008, during the South Ossetia War, the nation of Georgia blocked all websites with addresses ending in `.ru` [7].
- **Moral** - Russia has instituted an unpublished blacklist of sites relating to drugs, suicide, or pornography [3]. The United Kingdom is creating its own list, currently optional but enabled by default, to restrict access to pornography; web forums; information about violence, terrorism, and eating disorders; and "esoteric material" [27].

- **Security** - Volkswagen successfully sued to censor research about vulnerabilities in its keyless entry systems [49]. Life science researchers have instituted policies for censoring themselves when research seems to pose more risk than social good [44].
- **Competitive** - Universal Music Group briefly had Megaupload's "Mega Song" removed from YouTube, despite not having any copyright claim against it [33].
- **Suppressing criticism** - KTVU used DMCA takedown requests to remove copies of a newscast in which its anchor read obviously fake and offensive names for the pilots of Asiana flight 214 [28].
- **Accidental** - The U.S. Immigration and Customs Enforcement (ICE) accidentally took down the hip-hop music blog `dajaz1.com` for a year before returning it without explanation [32]. YouTube's Content ID system automatically flags videos for possible removal, but it often flags videos that do not contain copyrighted content or that might qualify as fair use [12]. While attempting to prevent unauthorized distribution of Windows 8 and Office, Microsoft has accidentally requested that Google remove links to the BBC, Wikipedia, and OpenOffice, among others [50, 51].

In each case, censorship was possible because a small number of administrative entities—ISPs, domain registrars, YouTube, research conference organizers, etc.—could be compelled to block or remove the content. Peer-to-peer systems provide an alternative distribution channel for content when censors get too heavy handed.

Formal peer-to-peer systems are not the only way that the Internet routes around censorship. Content that is small and not obviously offensive or copyright-infringing will often end up widely distributed and widely mirrored if someone tries to censor it. Far more people know what Barbara Streisand's house looks like, how badly the Suburban Express bus company treats its customers, and how to 3D print a gun than would have if interested parties had not attempted to censor that information [4, 22, 42]. In a sense, technology news sites, parodies, memes, and web forums act like an informal peer-to-peer network when they mirror content in this fashion.

3.3 Impacts

Several ideas from peer-to-peer systems have found their way into systems that are not strictly peer-to-peer. Most large-scale Internet services are geographically distributed, self-organizing, and resilient. Some systems, including the Cassandra database and the Tahoe-LAFS file system, explicitly incorporate DHTs [1, 48].

BOINC—the computing platform that runs SETI@home [6]—runs primarily on end users' computers, much like a peer-to-peer system. Unlike nodes in a peer-to-peer system, nodes in BOINC are only servers and do not consume the service that other nodes provide. Also, nodes are centrally managed rather than self-organizing. However, like a peer-to-peer system, BOINC successfully deals with malicious participants and harnesses spare CPU cycles.

4 Where we go from here

We believe that peer-to-peer systems continue to have both technical and social value. They may be a good way for modestly funded research groups to bootstrap an Internet service. They provide a stress test for new protocols, because protocols and techniques that work within the resource and security constraints of a peer-to-peer system will often work even better in a centralized system. Finally, of course, they provide outstanding resistance to censorship, in a way that commercial and centrally managed services cannot.

To that end, we believe the most important focus areas in peer-to-peer research are:

- **Security** - Services that become popular, or that host content that someone wishes to censor, become the target of attacks. Attackers may disrupt routing or searching, or they may intercept or modify content. Peer-to-peer systems have to deal with the possibility that nodes are Byzantine faulty [31].
- **Anonymity** - If the main use case for peer-to-peer systems is distributing censored content, then participants might need to remain anonymous when publishing or retrieving that content. Providing anonymity in a robust, efficient way could provide immense social value.
- **Searchability** - Simply put, we cannot read what we cannot find. BitTorrent users currently rely on centralized, commercial web sites to map keywords to document identifiers. Eventually, whether through legal changes or technical attacks, these sites will probably get shut down. Existing and new peer-to-peer search technologies should be applied to ensure that users can find content.

In 2013, as in 2001, peer-to-peer networking remains an exciting, fruitful area of research.

References

1. The Apache Cassandra Project. `http://cassandra.apache.org/`. Accessed: 2013-08-16.
2. Ares. `http://aresgalaxy.sourceforge.net/`. Accessed: 2013-08-16.
3. Russia internet blacklist law takes effect. *BBC News*, October 2012.
4. US government orders removal of Defcad 3D-gun designs. *BBC News*, May 2013.
5. Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
6. BOINC. `http://boinc.berkeley.edu/`. Accessed: 2013-08-16.
7. Reporters Without Borders. War still having serious impact on freedom of expression. `http://bit.ly/14mTqDm`, October 2010. Accessed: 2013-08-16.
8. Shuangjia Chen, Longshe Huo, Qiang Fu, Rui Guo, and Wen Gao. FBSA: a self-adjustable multi-source data scheduling algorithm for P2P media streaming. In *Proceedings of the International Workshop on Multimedia Content Analysis and Mining (MCAM)*, pages 325–333. Springer, 2007.
9. Ian Clarke. A distributed decentralised information storage and retrieval system. Master's thesis, University of Edinburgh, 1999.

10. Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000.
11. Bram Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, volume 6, pages 68–72, 2003.
12. Content ID disputes - YouTube. `https://www.youtube.com/yt/copyright/content-id-disputes.html`. Accessed: 2013-08-16.
13. Francisco Matias Cuenca-Acuna, Christopher Peery, Richard P. Martin, and Thu D. Nguyen. PlanetP: Using gossiping to build content addressable peer-to-peer information sharing communities. In *Proceedings of the International Symposium on High Performance Distributed Computing (HPDC)*, pages 236–246, 2003.
14. Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, October 2001.
15. Peter Druschel and Antony Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *Proceedings of Hot Topics in Operating Systems (HotOS)*, pages 75–80. IEEE, 2001.
16. John Eggerton. NBC, Obama campaign spar over YouTube video. `http://bit.ly/13GPgWW`, October 2008. Accessed: 2013-08-16.
17. Philip Elmer-Dewitt. First nation in cyberspace. *TIME International*, December 1993.
18. Benny Evangelista. Napster files for bankruptcy. *San Francisco Chronicle*, June 2004.
19. Li Fan, Pei Cao, Jussara Almeida, and Andrei Broder. Summary cache: A scalable wide-area web cache sharing protocol. In *Proceedings of ACM SIGCOMM*, pages 254–265, 1998.
20. Funshion online - about us. `http://www.funshion.com/english/about_us.html`. Accessed: 2013-08-16.
21. Ryan Gallagher. Skype won't say whether it can eavesdrop on your conversations. *Slate*, July 2012.
22. Sean Gallagher. Express to Internet hate: Bus company threatens redditor with lawsuit. *Ars Technica*, April 2013.
23. Mark Gillett. What does Skype's architecture do? `http://blogs.skype.com/2012/07/26/what-does-skypes-architecture-do/`, July 2012. Accessed: 2013-08-16.
24. Dan Goodin. Skype replaces P2P supernodes with Linux boxes hosted by Microsoft. *Ars Technica*, May 2012.
25. Jon Healey. StreamCast's undoing. `http://opinion.latimes.com/bitplayer/2008/05/streamcasts-und.html`, May 2008. Accessed: 2013-08-16.
26. Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, and Ion Stoica. Querying the Internet with PIER. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 321–332, 2003.
27. Jim Killock. Sleepwalking into censorship. `https://www.openrightsgroup.org/blog/2013/sleepwalking-into-censorship`, July 2013. Accessed: 2013-08-16.
28. David Kravets. Local newscast uses DMCA to erase air crash reporting blunder. `http://www.wired.com/threatlevel/2013/07/youtube-newscast-asiana/`, July 2013. Accessed: 2013-08-16.
29. Gunnar Kreitz and Fredrik Niemelä. Spotify—large scale, low latency, P2P music-on-demand streaming. In *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10. IEEE, 2010.

30. Yoram Kulbak and Danny Bickson. The eMule protocol specification. *eMule project, http://emule-project.net*, 2005.
31. Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.
32. Timothy B. Lee. ICE admits year-long seizure of music blog was a mistake. *Ars Technica*, December 2011.
33. Timothy B. Lee. UMG claims right to block or remove YouTube videos it doesn't own. *Ars Technica*, December 2011.
34. Jeff Leeds. Grokster calls it quits on sharing music files. *New York Times*, November 2005.
35. Jian Liang, Rakesh Kumar, and Keith W. Ross. The KaZaA overlay: A measurement study. In *Proceedings of the IEEE Annual Computer Communications Workshop*, pages 2–9. IEEE, 2004.
36. Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of the International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.
37. James Mullin. Optimal semijoins for distributed database systems. *IEEE Transactions on Software Engineering*, 16(5):558–560, May 1990.
38. Napster. `http://opennap.sourceforge.net/napster.txt`. Accessed: 2013-08-16.
39. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, 2001.
40. Patrick Reynolds and Amin Vahdat. Efficient peer-to-peer keyword searching. In *Proceedings of the ACM/IFIP/USENIX International Conference on Middleware*, pages 21–40. Springer-Verlag New York, Inc., 2003.
41. Matei Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of the International Conference on Peer-to-Peer Computing*, pages 99–100. IEEE, 2001.
42. Paul Rogers. Streisand's home becomes hit on Web. *Mercury News*, January 2003.
43. Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the ACM/IFIP/USENIX International Conference on Middleware*, pages 329–350. Springer, 2001.
44. Michael J. Selgelid. Governance of dual-use research: an ethical dilemma. *Bulletin of the World Health Organization*, June 2009.
45. Skype FAQ: What are P2P communications? `https://support.skype.com/en/faq/FA10983/what-are-p2p-communications`. Accessed: 2013-08-16.
46. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of ACM SIGCOMM*, 2001.
47. Jeremy Stribling. OverCite: A cooperative digital research library. Master's thesis, Massachusetts Institute of Technology, 2005.
48. Tahoe-LAFS. `https://tahoe-lafs.org`. Accessed: 2013-08-16.
49. Jason Torchinsky. VW demands British court censor scientific paper about car security. `http://bit.ly/1c75SPx`, July 2013. Accessed: 2013-08-16.
50. Microsoft DMCA notice mistakenly targets BBC, Techcrunch, Wikipedia, and U.S. govt. `http://bit.ly/QVArtf`, October 2012. Accessed: 2013-08-16.
51. Microsoft censors OpenOffice download links. `http://bit.ly/1a5Tu1J`, August 2013. Accessed: 2013-08-16.
52. Sandvine Inc. ULC. Global Internet phenomena report, 1H 2013.

53. Fred von Lohmann. McCain campaign feels DMCA sting. `https://www.eff.org/deeplinks/2008/10/mccain-campaign-feels-dmca-sting`, October 2008. Accessed: 2013-08-16.
54. Web hosting now vs 10 years ago. `http://royal.pingdom.com/2008/02/19/web-hosting-now-vs-10-years-ago/`, February 2008. Accessed: 2013-08-16.
55. Todd Woody. The race to kill Kazaa. *Wired*, February 2003.
56. Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum. CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. In *Proceedings of IEEE INFOCOM*, volume 3, pages 2102–2111. IEEE, 2005.
57. Ben Zhao, John Kubiatowicz, and Anthony Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division (EECS), University of California, Berkeley, 2001.