ADVANCES IN PEER-TO-PEER CONTENT SEARCH

Deepa Kundur[1]*, Zhu Liu*[2]*, Madjid Merabti*[3]*, and Heather Yu*[4]

[1]Electrical & Computer Engineering Department, Texas A&M University, College Station, TX, USA
[2]AT&T Labs – Research, Middletown, NJ, USA
[3]School of Computing & Mathematical Sciences, Liverpool John Moores University, UK
[4]Huawei Technologies (USA), Plano, TX, USA

ABSTRACT

This paper provides a timely review of influential work in the area of peer-to-peer (P2P) content search. We begin with a survey of text-based P2P search mechanisms and continue with an exposition of content-based approaches followed by a discussion of future directions.

1. INTRODUCTION

The power to access relevant and salient information in a timely and cost-effective manner is critical in today's information age. Models for information sharing between ad hoc groups of users are currently being investigated for collaborative and cooperative multimedia sharing applications. The most common model to date is based on the peer-to-peer (P2P) communication network model. Ideally, a P2P network consists of equal *peer* nodes that can take on the roles of both client and server to other peer network nodes. Such network operation makes use of the data acquisition capabilities, computing power and bandwidth of the network nodes rather than exploiting an existing network infrastructure for operations such as routing and information retrieval.

Naturally, processes such as content search must take into account this inherent network model as well as the characteristics of the information being acquired, stored and communicated. For multimedia data, such P2P search and retrieval processes are especially challenging to design; inherent compromises among search time, retrieval accuracy, and query message bandwidth, must be accounted for. This paper addresses recent advances in P2P content search by providing an overview of influential research in the area. We begin by giving an overview of text based P2P file search and then content-based search mechanisms are reviewed followed by a discussion of future directions.

2. TEXT BASED P2P SEARCH

One of the earliest P2P implementations that brought P2P computing into the mainstream and which sparked a large amount of media attention was Napster [1]. Napster was created purely for the distribution of MP3 audio files, and as such it was swamped with negative press because people were downloading digital content illegally and consequently ignoring content copyright. Each Napster node downloads and installs the client software used to connect the peer to the centralized Napster server. Once connected, peers share MP3 files stored locally on their hard drives, with text-based information about them being indexed and stored by the Napster server. Clients submit text-based queries to the Napster servers for a particular audio file. This results in a list of files that match, along with the connection information, username, IP and port address the querying client must use to connect to the peer hosting the file. Once the querying peer has this information it attempts to connect to the peer and transfer the target content in a P2P fashion. At this point the Napster server is no longer required [2].

Although Napster proved successful and is said to be the grandfather of modern P2P computing models it suffered from a number of limitations. The major limitation was the fact that it could only share MP3 content. In addition, its hybrid model was reliant on client-server technology - if the server became unavailable the discovery mechanism used to find content was lost.

Another hybrid protocol, similar to Napster called iMesh [3] uses a centralized server, to which clients connect to in order to search for content. The iMesh model differs somewhat to Napster in two respects. First, it allows any content to be shared including MP3 audio files. Second - the reason why iMesh has not been subjected to the same legal problems as Napster - it has a mechanism to remove copyrighted files from the network.

Computational expense and scalability issues associated with the above mentioned models are well documented, which has resulted in new P2P networks devoid of any centralization. The most popular being the Gnutella protocol [4]. Like iMesh it provides a generic file sharing mechanism that allows any digital media content to be shared. However it differs from iMesh and Napster because the Gnutella protocol uses a purely decentralized model, which is not reliant on any centralized authority.

The search mechanism used by Gnutella adopts a different approach to Napster in that it does not require any centralized server to manage the location of content within the network. Search packets containing text queries are used with predefined TTL values, the default value being 7, which corresponds to the number of hops the message can take. The packet is passed to all the immediate peers' the querying peer is connected to, which in turn is passed to all the peers the peer is connected to. The Horizon as defined by Kan [1], given a TTL of 7 encompasses about ten thousand nodes. If a node is found with a file name matching the query, the information is routed back to the querying peer. The file can then be downloaded directly from the target node. This is commonly referred to as blind search.

Unlike Napster, it is difficult to disrupt the network because no one single node is responsible for creating it. If any given node is lost it does not affect the overall search mechanism of the Gnutella network. The worst case is that you only lose the content provided by that node. Consequently Gnutella provides mechanisms to counteract some of the limitations associated with Napster. As such many Gnutella clients have been developed since the protocol was first released in 2000, including Bearshare [5] and Shareaza [6].

The FastTrack protocol claims to be better than Gnutella and its variants. A number of popular applications such as Kazaa [7], Morpheus [8] and Grokster [9], use the FastTrack protocol which divides users into two groups. The first group contains supernodes and the second contains ordinary nodes. Supernodes are defined as computers with significant computation, network and bandwidth capabilities. All supernodes are connected together to create an overlay network that acts as a hub processing all data requests received from ordinary nodes within the network, which are inherently less capable nodes.

When a node wants to share or search for a file a request is submitted to the supernode, which in turn submits it to all other supernodes, which in turn propagate the request to the ordinary nodes they are servicing. Like Gnutella, messages are configured with a TTL value of 7, ensuring that message propagation is terminated once seven hops have been reached.

Once the content has been found it is transferred directly from the target node to the querying node using the HTTP protocol, without using the supernode. There is a subtle distinction between the FastTrack model and that of Napster in that the Napster server managed an index of audio file information thereby breaching copyright laws. The FastTrack protocol avoids this problem because it only manages a list of supernodes and not information regarding the content itself. Supernodes are ad hoc in nature and are free to join and leave the network at any time. So information about supernodes held by the FastTrack servers continually changes. This abstraction detaches the FastTrack protocol, including the applications that use the protocol, from media content and thus some believe that FastTrack-based applications do not aid copyright infringement.

The difficulty with protocols such as Gnutella and FastTrack is that they rely on flooding or random walking for content search, with messages propagated to every peer. This results in increased costs and network traffic. Wang *et al.* [10] aim to alleviate these limitations using their proposed Differential Search (DiffSearch) algorithm. They claim DiffSearch improves search efficiency of unstructured P2P networks by giving higher querying priority to peers with high query/reply capabilities, known as ultrapeers. Ultrapeers form an overlay and serve visiting peers known as leaf nodes. The indices of leaf nodes are uploaded to ultrapeers allowing all shared content to be searched within what they call the first round. Based on test using Gnutella, Wang argues that 1% of peers answer the main portion of queries. Consequently by routing queries to these peers it is possible to save up to 90% of query traffic. Using counters to track which files answer queries, which they call *effective files,* a matrix is created allowing ultrapeers to be self-aware by counting the number of shared files which have been visited. If the number of shared files exceeds a threshold, a peer can promote itself to ultrapeer status. This results in an overlay where members have higher priority depending on where they reside in the hierarchy. To further decrease traffic, DiffSearch hitchhikes query/response messages to perform network management task. For example, allowing ultrapeers to advertise themselves to leaf nodes and vice versa.

The Foreseer P2P system aims to address various limitations using distributed indices [11]. Cai *et al.* claim their approach improves efficiency in decentralized unstructured P2P systems using two orthogonal overlays, which they term neighbor and friend overlays. To use their example, everyone has neighbors and friends which form part of an individual's social network. People tend to get to know their neighbors over time as they become more settled within their environment and make friends through social interactions. Implementing this scenario in Foreseer, friend nodes can serve future requests with a high probability (temporal locality), whilst neighbor nodes can offer quality of service, such as fast responses and low resource consumption if they are able to carry out the request (geographical locality). Locality is also discussed in Datta *et al.* [12], where it is considered important for scalability in data mining – the same principles apply to Foreseer.

Extending this scenario content is searched for using a collection of business cards provided by all neighbor and friend nodes. Finding content is as simple as sending/receiving a request. Using each business card, the node checks to see if a peer exists capable of service the request. If a suitable node is found the node in question is contacted. However, if the request cannot be serviced the request is passed to all its neighbors and friends. In Foreseer, business cards refer to a peer's content filter, which is derived using the Bloom filter on all the content it shares. It is therefore only able to address text-based content.

A by-product of the neighbor and friend overlays is that it provides an efficient search direction, where random walking or query flooding is unnecessary. Query requests contain one or more terms and it is these terms that are compared to the content filters for the neighbors and friends the node is aware of. In local matching a node computes the query filter by mapping the query terms and comparing it with the content filter of each node it has routing information about. If a match is found it indicates that a node may contain all the key words with high probability. If the local matching fails the query is selectively forwarded based on the results obtained from the first approach. The query in this instance is forwarded along the neighbor and/or friend links where local matching is performed.

Taking an opposing view, efficiency in P2P network has been addressed using different techniques, where processes are more finely controlled so that structure can emerge. Many approaches use Distributed Hash Tables to achieve this. Chord [13][14] is one such protocol that adopts these principles where order emerges using their DHT routing algorithm. Its basic structure forms a ring topology, whereby each node only has to establish one connection. A consistent hashing function, such as SHA-1, is used to generate node and object identifiers known as keys. The node identifier is created using the IP address and port. The object identifier, which can be any kind of shared content, is created using the data to be shared within the ring. Node identifiers are arranged in a circle modulo 2^m, where m is the length of the hash value. Every key k is assigned to the node whose identifier n is larger than or equal to the hash value of k. The node the key belongs to is called the successor. In Chord, node identifiers increase clockwise and keys are assigned to the first nodes that reside closest to them clockwise. Chord uses a hashing function designed to distribute keys evenly throughout the ring topology, whereby all nodes roughly receive the same number of keys.

Every node is aware of their successor and as such queries are passed from successor to successor. When a node is reached that has a hash value greater than or equal to the hash value of the key, this node can map the query to the key. In order to overcome the need to traverse every node, a node can attempt to find the predecessor of some key k using a finger table. Node n achieves this by searching its finger table for some node x that immediately precedes some key k. If it finds node x then it queries it to determine which node is closet to k. By repeating this process n moves the query closer and closer to k. In Chord this is called iterative routing.

CAN [15] is similar to Chord and also uses the DHT concept to self-organize, share content and route queries. CAN forms a P2P overlay network that stores chunks of a distributed hash table, known as zones. The protocol is based on a virtual d-dimensional Cartesian coordinate space. This space is dynamically partitioned among all the nodes in the system, so that every node owns its own zone within the global coordinate space. This space stores key-value pairs where k_1 is mapped onto a point p in the space using a uniform hashing function. The key-value pairs are stored on the node that owns the zone in which p resides. To discover the values of some key k_1 any node can use the hash function to map k_1 onto p and retrieve the contents from p. This may be the content or a pointer to the content. If the p is not owned by the querying node or its neighbor, then the request is routed towards the node where p resides.

Pastry [16] is not too dissimilar to Chord and CAN in that it also uses a DHT-based protocol to form a self-organized overlay network. Pastry nodes are identified in the network space using a 128 bit identifier, known as the *nodeId*. The *nodeId* indicates a node's position in the circular *nodeId* space. The *nodeId*s themselves are assigned randomly when the node first connects to the Pastry network. Several mechanisms can be used to derive the *nodeId*, however typical implementations use the node's public key or IP address to create a hash. In Pastry *nodeId*s are thought of as a sequence of digits in base 2^b. Nodes within Pastry maintain their own routing table, which is organised into $128/2^b$ columns. As well as the routing table, each node also maintains a neighborhood set *M* containing the *nodeId*s and IP addresses of the *M* nodes closest to the local node. The set is not used for routing, but rather for maintaining locality properties [17]. Nodes also maintain a leaf set *L* containing the set of nodes with numerically closest but larger *nodeId*s and numerically smaller *nodeId*s, relative to the present nodes *nodeId*. The leaf set is used when messages are routed. When a node receives a message it first checks to see if the key falls within the range of *nodeId*s covered by its leaf set. If it is, the message is forwarded directly to the destination node. If the key is not covered by the leaf set, the routing table is used and a message is forwarded to the node that shares a common prefix with the key by at least one more digit.

DHT-based P2P protocols are said to provide considerable benefits over previous generations, providing emergent behaviors that support order and increased performance. However, they are expensive to maintain because the network topology is continually changing. Consequently managing a consistent DHT requires considerable effort. In an attempt to leverage the benefits of DHT, but also minimize some of its inherent limitations the JXTA [17] specification have tried to create a balance by creating a hybrid system that uses a loosely consistent DHT [18]. Whilst protocols such as Chord rely on more costly mechanisms to keep the network view consistent, JXTA uses a less costly mechanism that ensures the network view is only loosely-consistent. The advantage with this approach is that it is less expensive to maintain, however the disadvantage is that it may be temporarily or permanently inconsistent.

Yang *et al.* [19] enhances the DHT-based P2P networking approaches discussed above by proposing a keyword search scheme called Proof that utilizes advances in information retrieval research. The Proof protocol reduces network traffic, decreases search latency and provides high quality search results. The Proof system comprises a crawler, a database, an index generator, and a distributed P2P system. The crawler collects web pages and extracts hyperlink information for computing page rank values, whilst the index generator produces new index structures and publishes them to the P2P system. The P2P system itself assumes *N* peers are contained and uses a consistent hash function to assign an identifier to each peer. The system contains documents and a vocabulary, which contains all of the keywords in the documents. Each document is composed of several keywords, which are a subset of the vocabulary. Given a query containing several keywords, a subset of the vocabulary and a user-specified result threshold, the search problem is defined in terms of finding the top relevant documents that contain all the query keywords.

While many interesting solutions have been proposed in both structured and unstructured P2P networks, Ahmed *et al.* [20] believe that flexibility and efficiency remain unsolved problems. To address these challenges Ahmed proposes their Distributed Pattern Matching system (DPMS). DPMS is based on Bloom filter based pattern matching distributed throughout the P2P network. Given a search pattern *Q*, DPMS tries to find peers containing some pattern *P* that matches *Q*, i.e. the 1-bits of *Q* for a subset of the 1-bits found in *P*. DPMS peers can act as either a leaf peer or an indexing peer where the former resides at the bottom level of the indexing hierarchy. This type of node advertises its indices representing the content the peer wishes to share. Whilst, indexing peers store indices received from other peers - these peers may be leaf peers or indexing peers. Peers join different levels within the hierarchy and can act both as a leaf peer and an indexing peer. Within this hierarchy indexing peers disseminate index information using repeated aggregation and replication. Replication is used for disseminating patterns from leaf peers to a large number of indexing peers. To overcome increased traffic load, DPMS combines replication with lousy-aggregation. Advertisements provided by different peers are aggregated and propagated to peers in the next level along the aggregation tree. Based on repeated lousy aggregation, information content of the aggregates is reduced as you move towards the top of the indexing hierarchy. This helps balance the system and improve fault tolerance. Furthermore, peers can route queries towards a target without having any global knowledge of the overlay topology. It also helps minimize query forwarding traffic.

The query life-cycle can be divided into three phases: ascending phase, blind search phase and descending phase. Using the ascending phase, an initial (or intermediate) peer checks its local information for the existence of a match. If a match is found the query is forwarded to the matching child, otherwise it is forwarded to any of its parents. This process continues until a query hits a peer with a match or reaches the highest level peer. Blind search is executed by a highest level peer receiving a query (from a child) that does not match any aggregate in its aggregate lists. This peer floods the query to all other peers in its group. If no peer in a group at the highest level contains a match then the search fails. When a peer hits a peer containing the matching aggregate, it enters the descending phase. The query is forwarded to the child peer advertising the matching aggregate. This process continues until the query reaches a leaf peer.

Today, more and more P2P systems are seeking to support more powerful queries. Joost, a broadcast quality Internet TV service, for example supports phrase queries, wildcard queries, proximity queries, range queries and more. An algorithm that supports range queries over DHT is proposed in [21]. It implements range queries over DHT via a trie-based scheme in which every vertex corresponds to a distinct prefix of the data domain being indexed. A range query can be performed via longest common prefix search followed by a parallel traversal in the P2P network to retrieve all the desired items.

3. CONTENT-BASED SEARCH

Most existing P2P systems provide very limited content search capabilities, for example, search based on document title, author names, keywords, or descriptive text. To retrieve the relevant content more effectively, we need an approach that provides richer searching features. Content-based search is essential for querying textual documents, and it is also desirable for querying multimedia data when text annotations are nonexistent or incomplete.

Multimedia content indexing and retrieval has been an active field for more than a decade. It draws tremendous research effort from the academia, the industrial, and the standard organizations. For example, MPEG-7 is a standard sponsored by the International Organization for Standardization for describing the multimedia content. It provides support to a broad range of applications, and it will make the web as searchable for multimedia content as it is searchable for text. The evolution of the World Wide Web, including the introduction of Rich Site Syndication (RSS), Web 2.0, and the semantic web, enables the web information be machine processable (rather than being only human oriented), thus permits browsers or other agents to find, share and combine information more easily. In this section we will briefly describe a few examples of content-based search in P2P systems.

Tang *et al.* [22] proposed PeerSearch, an efficient P2P information system that supports content and semantic search. PeerSearch extends existing information retrieval methods: the vector space model (VSM) and the latent semantic indexing (LSI) to work with the efficient routing mechanisms in a Content Addressable Network (CAN). Basically, LSI uses singular value decomposition to transform and truncate a matrix of document vectors computed from VSM to discover the semantics underlying terms and documents. The authors used the semantic vector of a document as the key to store the document index in CAN, such that the indices stored nearby in CAN are close in semantics. The same technology can be applied to audio or video data, where the semantic vectors have to be extracted from audio or video data.

Lu and Callan [23] explored content-based resource selection and document retrieval algorithms in hybrid P2P networks. In their approach, the leaf node determines the retrieval results for certain query using probabilistic information retrieval algorithm, and the directory node (supernode) builds a unified content model for all of its leaf nodes and a set of neighboring directory nodes. The content model is used for routing query messages.

Yang [24] described a content-based music retrieval system in P2P environment. Each audio document is converted into a stream of characteristic sequences, where each sequence is a vector representing a short segment of music data. All characteristic sequences are indexed using Locality-Sensitive Hashing scheme, such that similar (in term of human perception) vectors can be hashed into the same hash value with high probability. Given a query audio, the retrieval procedure is to find a list of matches on characteristic sequences with the tolerance of tempo changes. To improve the search efficiency in a P2P environment, a two-phase search protocol was proposed. In the presearch phase, the query peer broadcasts a small subset of query vectors to all potential peers. In the actual search phase, peers with a higher chance of a hit will conduct the more rigorous search.

In [25], Lee and Guan presented a content-based image retrieval system over a P2P network. Each peer in the system maintains two look-up tables, one for generic neighbors which are typically the neighbors with the least physical hop counts, and the community neighbors which share common interests at the image content. The system uses the historical retrieval results to identify the community neighbors, such that the subsequent retrieval within the community neighbors will result better retrieval precision. The adopted visual features include color, color moments, object shape, and texture.

4. WHAT'S NEXT

With the recent merging of content acquisition, communication networking and computation to form emerging multimedia sensor or surveillance systems, the need for advanced content mechanisms is paramount. Many of these systems are predicted to have P2P communication architectures [26] necessitating efficient P2P content-based search for higher-level content retrieval and understanding. Imagine a world in which multimedia sensor networks (interpreted as sensor databases containing time critical information) are employed to detect emerging natural disasters or terrorist activities. The utility of such systems is, in part, determined by the ability to effectively and efficiently retrieve information for a given application given often incomplete or vague information queries.

Furthermore, for such multimedia database systems security and privacy issues are paramount [27][28][29]. Research into security policy development, security architectures, authenticated querying, privacy protection, and access control for information retrieval are essential during system inception in order to guarantee the most seamless, cost-effective and robust solution.

REFERENCES

[1] A. Oram *et al.*, Peer-to-Peer: Harnessing the Power of Disruptive Technologies, 2001, O'Reilly.

[2] J. D. Gradecki, Mastering JXTA: Building Java Peer-to-Peer Applications. 2002, Wiley Publishing, Inc.

[3] http://www.imesh.com/.

[4] The Gnutella Protocol Specification v0.4, Gnutella, http:// www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[5] http://www.bearshare.com/.

[6] http://www.shareaza.com/.

[7] http://www.zeropaid.com/kazaalite/.

[8] http://morpheus.com/.

[9] http://www.grokster.com/.

[10] W. Wang and L. Xiao, "An Effective P2P Search Scheme to Exploit File Sharing Heterogeneity," *IEEE Transactions on Parallel and Distributed Systems*, 2007. 18(2), pp. 145 - 157.

[11] H. Cai and J. Wang, "Exploiting Geographical and Temporal Locality to Boost Search Efficiency in Peer-to-Peer Systems," *IEEE Transactions on Parallel and Distributed Systems*, 2006. 17(10): pp. 1189 - 1203.

[12] S. Datta, K. Bhaduri, C. Giannella, H. Kargupta, and R. Wolff, "Distributed Data Mining in Peer-to-Peer Networks," *IEEE Internet Computing*, 2006. 10(4): pp. 18-26.

[13] J. Eberspacher, R. Schollmeier, S. Zols, and G. Kunzmann, "Structured P2P Networks in Mobile and Fixed Environments," *HET-NETs '04*, 2004, West Yourshire, UK.

[14] F. Dabek, E. Brunskill, F. Kaashoek, and D. Karger, "Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service," *HotOS-VIII*, 2001, Germany, pp. 81-86.

[15] S. Ratnasamy, P. Fancis, M. Handley, and R. Karp, "A Scalable Content-Addressable Network," *ACM SIGCOMM 2001*, San Diego, California, USA: ACM Press, pp. 161-172.

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

[17] L. Gong, "JXTA: A Network Programming Environment," *IEEE Internet Computing, 2001*. 5(3): pp. 88-95.

[18] B. Traversat, M. Abdelaziz, and E. Pouyoul, "Project JXTA: A Loosely-Consistent DHT Rendezvous Walker," URL: http://www.jxta.org/docs/jxta-dht.pdf.

[19] K. Yang and J. Ho, "Proof: A DHT-Based Peer-to-Peer Search Engine," *WI* 2006, Hong Kong, pp. 702 - 708.

[20] R. Ahmed and R. Boutaba, "Distributed Pattern Matching: A Key to Flexible and Efficient P2P Search," *IEEE Journal on Selected Areas in Communications*, 2007. 25(1): pp. 73-83.

[21] S Ratnasamy, J Hellerstein and S Shenker, "Range queries over DHTs," IRB-TR-03-009, June, 2003,available at http:// berkeley.intel-research.net/sylvia/range.pdf

[22] C. Tang and Z. Xu and M. Mahalingam, "PeerSearch: Efficient Information retrieval in Peer-Peer Networks," 2002, Hewlett-Packard Labs: Palo Alto.

[23] J. Lu and J. Callan, "Content-Based Retrieval in Hybrid Peer-to-Peer Networks," *Proceedings of ACM CIKM'03*, New Orleans, LA, Nov. 2003.

[24] C. Yang, "Peer-to-peer Architecture for Content-based Music Retrieval on Acoustic Data," *WWW 2003*, Budapest, Hungary, May 20-24, 2003.

[25] I. Lee and L. Guan, "Content-based Image Retrieval with Automated Relevance Feedback Over Distributed Peer-to-peer Network," *ISCAS 2004*, Vancouver, Canada, May 23-26, 2004.

[26] D. Kundur and W. Luh, *Encyclopedia of Multimedia.* Springer 2006, ch. Multimedia Sensor Networks.

[27] B. Thuraisingham, "Security and Privacy for Sensor Databases," *Sensor Letters*, vol. 2, no. 1, pp. 37-47, March 2004.

[28] W. Luh and D. Kundur, "Distributed Privacy for Visual Sensor Networks via Markov Shares," *Proc. 2nd DSSNS*, Columbia, MD, April 2006.

[29] W. Luh, D. Kundur and T. Zourntos, "A Novel Distributed Privacy Paradigm for Visual Sensor Networks Based on Sharing Dynamical Systems," *EURASIP Journal on Applied Signal Processing Special Issue on Visual Sensor Networks*, vol. 2007.