Research Issues in Peer-to-Peer Data Management

Özgür Ulusoy
Department of Computer Engineering
Bilkent University, Ankara, Turkey
oulusoy@cs.bilkent.edu.tr

***Abstract*-Data management in Peer-to-Peer (P2P) systems is a complicated and challenging issue due to the scale of the network and highly transient population of peers. In this paper, we identify important research problems in P2P data management, and describe briefly some methods that have appeared in the literature addressing those problems. We also discuss some open research issues and directions regarding data management in P2P systems.**

Keywords: **Peer-to-peer systems, data management, overlay network, indexing, data integration, query processing, data replication, clustering, free riding, incentive mechanism.**

I. INTRODUCTION

Peer-to-Peer (P2P) is a distributed computing paradigm that enables a collection of nodes (peers) to share computer resources in a decentralized manner. Beside decentralization and extensive resource sharing capabilities, P2P systems are also characterized by their other desired features such as, anonymity of peers, increased autonomy, improved scalability, and self-organization through automatic adaptation to dynamic nature of peers which may join, leave or fail at any time.

P2P computing can be considered as an alternative to the traditional centralized and client-server models of computing, where central servers are required for the coordination of sharing and computing activities among client computers. In a client-server model, a server computer is dedicated to provide a particular kind of service to client computers. In a P2P system, the peers can act as both clients and servers. Leading examples of P2P systems include Gnutella [1], Napster [2], Freenet [3], BitTorrent [4], Chord [5], and the Content Addressable Network (CAN) [6].

P2P paradigm gained visibility with its best-known application, which is *file sharing* (e.g., music files in Napster). However, P2P can be applied to many domains beyond data sharing, such as *distributed computation*, *communication and collaboration* between peer computers, and *Internet service support* [7]. As file-sharing P2P systems have evolved to support advanced applications which must deal with semantically rich data, a requirement has arisen to address various data management issues [8, 9]. However, highly transient population of autonomous peers and the large scale of the network make data management a challenging problem in P2P systems.

In this paper, we highlight a number of important research issues in P2P data management. We start with a brief description of different types of P2P overlay networks, and give an overview of a few representative P2P systems. Then, we discuss the research issues relevant to P2P data management, together with the challenges raised by the unique features of P2P systems. We point out open problems that can be subject to further research. We also discuss the free riding problem and incentive mechanisms used to encourage peer cooperation.

II. OVERLAY NETWORK STRUCTURE

Each peer in a P2P network maintains links with a selected subset of other peers, forming an overlay network [10]. A message between the peers is routed through the overlay network. The overlay network is built on top of a physical (typically IP) network, and the peers that are neighbors in the overlay network do not need to be adjacent in the physical network.

There exist three classes of P2P overlay networks with different degrees of centralization: *purely decentralized, partially centralized*, and *hybrid decentralized* [7]. In purely decentralized systems, all peers perform the same tasks, and there is no central coordination of their activities. In partially centralized systems, some of the peers, called *superpeers*, act as local central servers maintaining indices for the files shared by local peers. In hybrid decentralized systems, there exists a central server which maintains the metadata describing the shared files stored at peers. The central server processes the search requests, and identifies the peers which store the requested files. File exchange takes place directly between two peers.

Overlay networks can also be distinguished in terms of their structure. In *unstructured* class of networks, the placement of files on the peers does not follow specific rules. In other words, the overlay network is created in an ad hoc manner as peers and files are added to the system. In *structured* networks, on the other hand, files are placed at precisely specified peers. Mapping of files to peers is typically achieved by using Distributed Hash Tables (DHTs). In a structured network, queries submitted to locate files can be efficiently routed to the peers with the desired file.

Unstructured networks are appropriate for accommodating highly-transient peer populations, in which peers are joining and leaving at a high rate. However, the networks of this kind are faced with the scalability problem. While the scalability issue is handled smoothly in structured networks, the maintenance cost of structured overlays is high in the presence of transient peer population.

Typical examples of P2P systems with unstructured network overlays (Gnutella, Napster), and structured network overlays (CAN, Chord) are described briefly in the following sections.

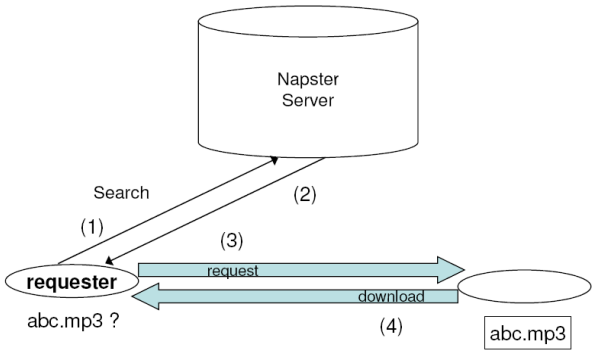


Fig. 1. Search mechanism of Napster. The file "abc.mp3" is searched through the central server. The numbers next to the arrows represent the order of message flow.

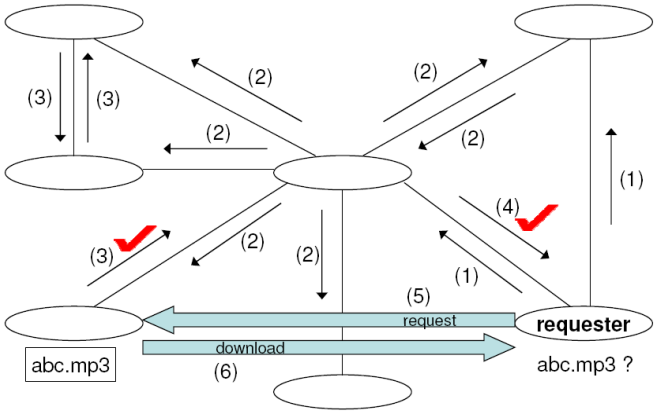


Fig. 2. Search mechanism of Gnutella. The requester peer issues a query for the file "abc.mp3". The numbers next to the arrows represent the order of message flow. An arrow with symbol ✓ represents a success message.

III. Representative P2P Systems

A. Napster

Napster is the first P2P file sharing system which was originally developed to enable the sharing of music files over the Internet [2]. A central Napster server maintains a list of music files shared by the peers currently connected to the network. The shared file list is automatically updated as the peers connect to or disconnect from the network. Napster supports keyword searches for the music files. The underlying network has a hybrid decentralized architecture: search requests are handled in a centralized manner; however the shared files are distributed, and file transfers take place directly between the peers.

The search mechanism of Napster is presented in Fig. 1. A peer submits a search request for a particular file to the centralized server. After getting the response to its query, the requester establishes a connection with the peer which possesses the requested file. The download of the file is done between two peers, bypassing the Napster server.

B. Gnutella

Gnutella is a file sharing P2P protocol with a purely decentralized and unstructured network overlay [1]. There is no central coordination of the sharing activities in the network and file downloads are done directly between two peers. A *flooding* mechanism is used for the distribution of query messages: each message received by a peer is forwarded to all of the neighboring peers.

A peer joining the network first announces itself by sending a special message (*Ping*) to neighboring peers it is connected to. The peers send back a message (*Pong*) identifying themselves, and also propagate the *Ping* message to their neighbors. In order to locate a file, a peer issues a *Query* message to the neighboring peers. The Query message is propagated from peer to peer using the flooding method. When the requester peer receives a *QueryHit* message, indicating that the target file has been identified at a certain peer, it establishes a direct connection to the destination peer, and initiates file download. An example of the Gnutella search mechanism is illustrated in Fig. 2.

A Time To Live (TTL) value, which is initiated to a small integer, is associated with each Gnutella message to prevent it from circling the network forever. This value is decremented at each hop, and when it reaches 0, the message is dropped from the network.

The notion of *superpeers* has been used in recent versions of the Gnutella protocol. Once a node with sufficient bandwidth and processing capabilities joins the network, it becomes a *superpeer* and establishes connections with other superpeers. The partially centralized Gnutella network is organized into an interconnection of superpeers and client peers. A superpeer indexes the files shared by the client peers connected to it, and queries are initially directed to superpeers.

C. Content Addressable Network (CAN)

Content Addressable Network (CAN) is an Internet-scale DHT that maps file names (keys) to their location in the P2P overlay network [6]. It uses a virtual d-dimensional Cartesian coordinate space to implement the distributed location and routing table. The coordinate space is partitioned into hyper-rectangles, called zones. Each individual peer in CAN is responsible for a zone, and a peer is identified by the boundaries of its zone. A key is deterministically mapped onto a point P in the coordinate space, and then stored at the peer that is responsible for the zone which contains P. Each peer maintains a routing table (IP addresses) of its neighbors in the coordinate space. A 2-dimensional CAN is presented in Fig. 3.

A query submitted to search for a file is forwarded along a path in the coordinate space from the requester to the peer storing the key. Each peer along the path forwards the query to the neighbor closest in the coordinate space to the peer storing the key (see Fig. 4).

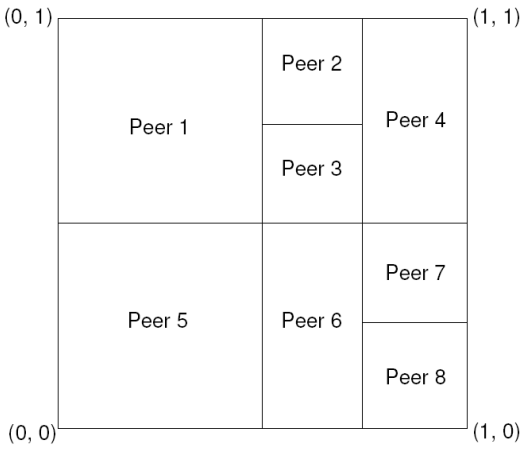


Fig. 3. A 2-dimensional CAN with 8 peers.

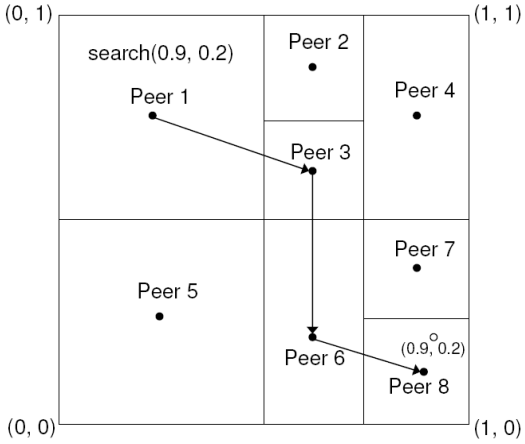


Fig. 4. Implementation of a search operation issued by peer 1 for the data file whose key is mapped to point (0.9,0.2) in the coordinate space.

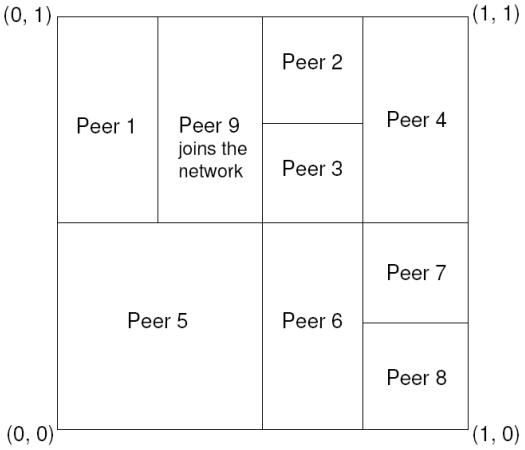


Fig.5. Before peer 9 joins the network: Neighbors of peer 1are peers 2, 3, and 5. After peer 9 joins: Neighbors of peer 1 are peers 5, 9; Neighbors of peer 9 are peers 1, 2, 3, 5.

To join the network, a new peer randomly chooses a point P in the coordinate space and sends the join request to the peer whose zone covers P. The zone is then split, and half of it is assigned to the new peer, as illustrated in the Fig. 5. When a peer leaves the network voluntarily, it hands over its zone to one of its neighbors. Merging the two zones of the neighbor may result in a larger valid zone. Peer failures can be traced through periodic messages the peers send to their neighbors. When a peer fails, one of the neighbors of the failed peer will take over.

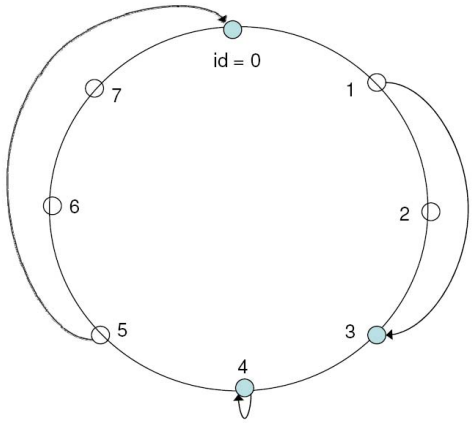
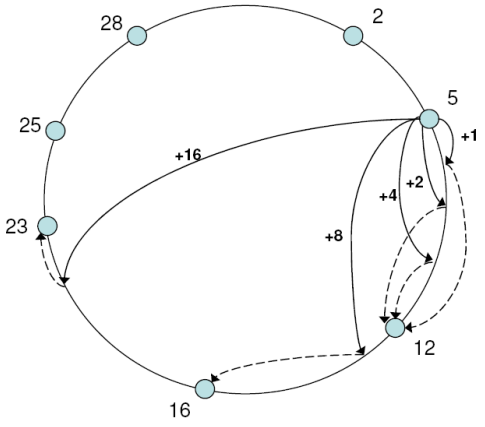


Fig. 6. A Chord circle with three peers: 0, 3 and 4. Key 1 is located at peer 3 (*successor*(1) = 3), key 4 at peer 4, and key 5 at peer 0.



Finger Table of Peer 5	
5+1	12
5+2	12
5+4	12
5+8	16
5+16	23

Fig. 7. The Chord ring has 7 peers. The finger table of peer 5 is shown. The first finger of peer 5 points to peer 12, as peer 12 is the first peer that succeeds $(5+2^0) \bmod 2^5 = 6$. Similarly, the last finger of peer 5 points to peer 23, as peer 23 is the first peer that succeeds $(5+2^4) \bmod 2^5 = 21$.

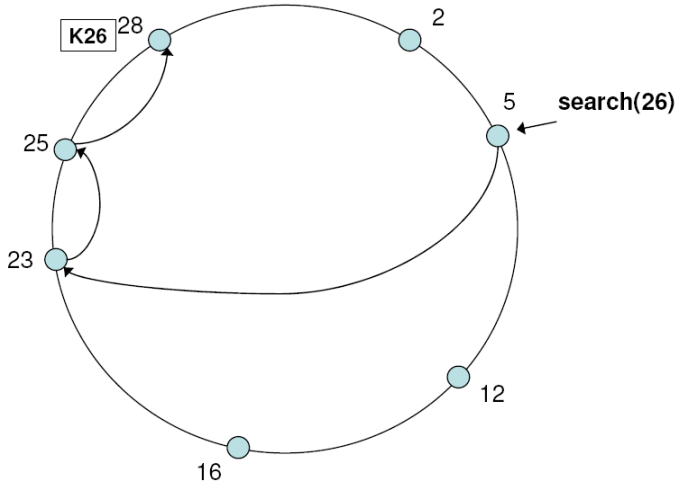


Fig.8. Search mechanism of Chord. The file with key 26 is searched by peer 5.

D. Chord

Another example of P2P systems with a structured network overlay is Chord [5]. Peers in Chord form a ring, called *identifier circle* or *Chord circle*. Peers and file names (keys) are mapped to *m*-bit identifiers using consistent hashing which tends to balance load as each peer is assigned roughly the same number of keys. Peer identifiers are ordered on the ring based on the modulo of the key with 2^m. A data file with key *k* is assigned to the first peer whose identifier is equal to or follows *k,* and this peer is called the *successor* peer of key *k*. An example of a Chord circle is provided in Fig. 6.

Chord maintains a skiplist-like data structure. Each peer in Chord maintains a finger table to keep track of a small number of other peers which are searched to find the desired key. The number of peer entries in the finger table of a peer is *log(N)* in an *N*-peer system. Each entry *i* in the finger table of peer *p* points to the successor of peer $(p + 2^i) \bmod 2^m$ for *m*-bit identifier space. In Fig. 7, a Chord ring with $m = 5$ is illustrated.

In order to perform a file search operation, a peer uses the information stored in its finger table. However, a peer's finger table would probably not contain enough information to directly determine the successor of a key. In that case, the peer forwards a query for key *k* to the peer in its finger table with the highest ID not exceeding *k.* As an example, for the Chord circle in the Fig. 8, suppose that peer 5 wants to find the successor of key 26. The largest finger of peer 5 that precedes 26 is peer 23; therefore, peer 5 will ask for the help of peer 23 for the query. Then, peer 23 will find out the largest finger in its finger table that precedes 26, i.e. peer 25. Peer 25 will determine that its own successor, peer 28, stores the key 26. The query visits every peer on the path between peer 5 and peer 28. The response is returned along the reverse of the path.

When a new peer arrives, some of the keys previously assigned to its successor are assigned to the new peer. Finger tables of the peers are arranged appropriately. When a peer leaves the network, all the keys it stores are assigned to its successor.

IV. INDEXING

Traditional distributed systems use centralized or distributed indices to keep track of the location of data. Indices are consulted for processing the queries at the appropriate nodes where the required data reside. With P2P systems, additional requirements arise in maintaining indices. Indexing structures must be designed to handle frequent updates due to the existence of highly-transient peer populations. Scalability of the system must also be supported, since massive number of participating peers is not unusual in P2P systems.

There are three basic types of P2P indices: *local, centralized*, and *distributed* [11]. In *local indexing*, peers index only their own content. In P2P systems with a purely local data index, typically flooding method is used in searching for data. The classic example of such P2P systems is the early versions of the Gnutella protocol (see Section III.*B*). A large volume of query message traffic is generated with flooding, which leads to scalability problems. Various attempts have been made to reduce the query load and improve the scalability in P2P systems with local index (e.g., expanding rings [12], random k-walkers [13], probabilistic flooding [14]).

In the case of a *centralized index,* a single server maintains the location information about the data stored on all the peers in the system. Napster is a well-known example of P2P systems with centralized index (see Section III.*A*). A data search request of a peer is directed to the central index server. Such systems are also called 'hybrid' since the index is centralized but the data is distributed [11]. The centralized index approach has the reliability problem as the central server is a single point of failure.

In partially centralized P2P systems, each *superpeer* acts as a central server maintaining indices for the data stored at a number of regular peers connected to it. A peer sends a query to its superpeer which then consults its index to propagate the query to its relevant peers or to the other superpeers.

In P2P systems with a *distributed index*, the distribution of the index depends on whether the underlying overlay network is structured or unstructured. One representative example of distributed indexing in unstructured P2P systems is *routing indices* [15]. The routing indices maintained in a peer capture some information about the data stored at other reachable peers. This information is used to direct the queries towards the peers that hold the required data. The concept of *horizons* is used to limit the number of peers for which each peer maintains indexing information. The peers reachable from a particular peer are placed at a maximum distance *h,* which is called the *radius* of the horizon.

As mentioned in Section II, index information in structured P2P systems is maintained in the form of Distributed Hash Tables (DHTs). Each peer maintains a DHT for the data assigned to it by the hash function. In evaluating a query, the hash value of the data is computed and the query is forwarded

towards the peer that is responsible for maintaining index information for the requested data. Greedy routing and robustness are two important features of DHTs.

V. DATA INTEGRATION

Data sharing is one of the primary design goals of P2P systems. When the shared data maintained in different peers are related, some semantic issues are introduced [9]. *Schema mappings* are required if the peers use different names or formalisms to represent the same data. Heterogeneity of data sources should be hidden to provide uniform querying environment to the peers. A common data sharing approach proposed for traditional distributed systems is to provide a global mediated schema [16]. Queries are specified in terms of the global schema, and then reformulated into subqueries to be executed at the local schemas. Translation between the global schema and local schemas is provided as needed.

Traditional data sharing approaches are not directly applicable to P2P environments, given the peer autonomy, volatility and scalability aspects of P2P systems. It may not be possible to define a unique global mediated schema. The peers may not be willing to share their schema information. Volatile nature of peers also makes the use of a unique global schema impractical. The global schema would need to be continuously modified as peers join and leave the system at a high rate. The scalability issue also makes the maintenance of a unique global schema quite difficult, because it is possible to have a massive number of peers each with its own local schema.

Data integration approaches in P2P systems can be categorized into three groups: *pair mappings, peer-mediated mappings* and *super-peer mediated mappings* [9]. Pair mappings are defined between pairs of peers, and this kind of schema mappings are maintained at any peer which would like to access the data stored at other peers. A generalized form of pair mappings is called peer-mediated mappings which can be defined between the schemas of more than two peers. Examples of P2P systems that implement peer-mediated mappings include Piazza [17] and PeerDB [18]. If peer-mediated mappings among peers are defined at the level of the superpeer connecting regular peers, the resulting mappings are called super-peer mediated mappings. A P2P system that follows the super-peer mediated mappings approach for data integration is Edutella [19].

VI. QUERY PROCESSING

The research in P2P query processing has mainly focused on *key lookups* in structured networks and *keyword queries* in unstructured networks. In order to support various types of applications the query language used for a system must be expressive enough to be able to describe the required data in sufficient detail. If we would like to perform search over text documents, key lookups would not be expressive enough. Similarly, for efficient searching of structured data (such as relational tables) we require more sophisticated querying approaches than simple keyword queries.

Queries submitted in a P2P system need to be routed to the peers which are responsible for maintaining the location information of the requested data. Routing schemes used for that purpose can be generalized into two main categories: *blind search* and *informed search* [9]. With a blind search method, no information is stored regarding data placement. Queries are routed arbitrarily to particular peers without guaranteeing to find the results if they exist. An example of this simple routing strategy is flooding. As it is discussed earlier in this paper, flooding can overload the network quickly, and various approaches have been proposed to reduce the message traffic of flooding.

With informed search methods, some form of data placement information is maintained at each peer. Queries are routed to peers that have some information about the location of requested data. Therefore, routing is performed more effectively compared to blind searching, and the number of messages is reduced in locating data. One example of informed search method is Query Routing Protocol (QRP) which makes use of routing tables containing keywords which describe the file contents that a peer offers [20]. The contents of routing tables are exchanged with the neighbors. Every peer merges its routing table with the routing tables of its neighbors, and propagation of routing tables is restricted to a fixed number of hops. When a peer receives a query, it forwards it to the neighbor whose routing table contains the keywords specified in the query. If none of the neighbors has the keywords in its routing table, this method degenerates to blind search. The other examples of informed search methods include routing indices (see Section IV) and FreeNet's routing scheme [21].

Query processing in traditional distributed systems is implemented by decomposing a query into subqueries, and executing each subquery at the appropriate site which is the source of the data required by that subquery. This approach requires a centralized control providing a global schema of the data distributed over different sites. However, decentralized query processing approach is more suitable for P2P systems, as frequent changes in schemas or in data availability are very common in P2P environments [9]. Therefore, a collection of peers, probably with different data models, are involved in formulation and execution of queries.

The variety of application types supported by recent P2P systems has led to the requirement for supporting advanced query types. As an example of the systems that support complex queries, Multi-Attribute Addressable Network (MAAN) was built on Chord to service both *multi-attribute queries* and *range queries* in a structured P2P system [22]. In MAAN, a locality preserving hash function is used to map attribute values to peers. When a range query is submitted to search for data (or any kind of resources, in general) with an attribute value in a range (l, u), where l and u being the lower and upper bound respectively, the query is forwarded to peer pl where l has been hashed to. The peer pl searches its local data entries and adds the data that satisfy the range query to the result. Then it checks whether the value u has also been hashed to itself. If true, it sends back the search response to the query originator peer. Otherwise, the query is forwarded to the

immediate successor of peer *pl* in the ring which also searches its local data entries, appends matched data to the result and forwards the query to its immediate successor, and so on, until the query reaches peer *pu* where *u* has been hashed to.

Multi-attribute range queries are also supported by MAAN. When a peer submits a multi-attribute query, the query is split into single attribute subqueries. Following the execution of subqueries at appropriate peers, the results are intersected at the query originator to have the final answer. One serious limitation of the MAAN system is the assumption of a fixed schema, which is difficult to ensure in P2P environments.

VII. DATA REPLICATION

Data replication is used for improving availability and enhancing system performance. Classical issues of replication, such as which data to replicate, the granularity of replicas, where to place them, and how to manage replica updates, are also addressed by P2P systems. In order to maintain consistency of replicated data, there are some P2P specific challenges to overcome which include high rates of peer joins and failures, lack of global knowledge on shared data, unknown peer capacities, and so on.

Structured network overlays like DHTs are more appropriate to overcome data consistency challenges of P2P systems [9]. Data consistency based applications commonly make use of DHTs to provide the necessary replication. As an example, in P2P system CFS, *k* replicas are placed on the *k* successors in a Chord ring of the peer storing the data [23]. If the primary replica fails, the successor immediately takes over.

A replica update strategy based on a push/pull spreading algorithm is presented for P2P systems in [24]. Each new update is pushed to a number of peers which are known to store the replicated copies. A peer enters the pull phase when it comes back online, after being offline for a while. It contacts the other peers storing the replicas and retrieves the most up to date copy. The update strategy is fully decentralized and robust, however it offers only probabilistic guarantees rather than ensuring strict consistency.

Different replication strategies, with regards to the number of replicas, were evaluated for unstructured P2P systems [25]. Two very common replication approaches, called uniform and proportional, were shown to yield worse performance than any replication strategy which lies between them. In *uniform replication,* the same number of replicas is created for all data items, while in *proportional replication* more replicas are created for more popular data. Uniform replication wastes system resources in replicating data that rarely appear in queries. With proportional replication, on the other hand, although queries on popular data are processed efficiently, unpopular data search may take a long time degrading the overall system performance. A replication approach between these two extremes, called *square-root replication,* was shown to perform better in terms of the search size on successful queries. With this approach, replicated copies are created in such a way that for any two data items the ratio of replication is the square root of the ratio of their query rates.

Two main strategies can be used with regards to where to place replicas in unstructured P2P systems [13]. With the first strategy, which is called *owner replication*, when a query on a data item is successful, a replica of the item is created at the requester peer. The other strategy is called *path replication,* where when a query succeeds, copies of the requested data are stored at all peers along the path from the requestor peer to the provider peer. Path replication offers higher levels of availability compared to owner replication; however ensuring consistency in the presence of updates will be more difficult with large numbers of copies.

VIII. CLUSTERING

In a distributed system, data items with common attributes or properties can be grouped together forming *data clusters*. Clustering aims to reduce the communication cost in query processing by placing related data in nearby locations. In structured P2P systems it is possible to cluster data, i.e., store similar data at the same or neighboring peers, by using an order-preserving hash function. If the inputs of an order-preserving hash function are similar, then the outputs produced by the function will be close in the identifier space. Content-based clustering of data files can be achieved if a semantic vector describing the contents of the files is used as the input of the hash function [26].

In P2P systems, besides clustering data, it is also possible to group the peers according to their interests or data into *peer clusters*. Similar to data clustering, peer clustering also aims to improve querying performance and produce better quality of results. A query is first routed to the appropriate cluster, and then it can reach all the peers with relevant data or interest within that cluster. *Semantic Overlay Networks* (SONs) is one approach introduced for peer clustering where peers with semantically similar content are logically linked to form overlay networks based on a classification hierarchy of their data [27]. Interest-based *peer communities* is another approach proposed for peer clustering in which membership depends on the relationship between peers that share common interests [28]. Clustering is implemented by using sets of attributes that the peers can choose from, and communities are formed between peers that share similar attributes.

Some characteristics of P2P systems make clustering a challenging task. Peers in a P2P system are autonomous, however autonomy is violated by data clustering since peers are enforced to store some specific data. Another concern for the application of clustering is the very dynamic nature of P2P environments. Clusters formed in a P2P system needs to dynamically adapt to the frequent changes in peer populations and their data. The lack of global knowledge of data and peer interests also causes a serious difficulty in forming clusters in P2P systems.

IX. INCENTIVE MECHANISMS

A. Free Riding

A *free rider* is a peer that exploits P2P network resources but does not contribute to the network at an acceptable level [29]. In a free riding environment where most of the peers are free riders, only a small number of peers serve a large population. This means that the benefits of the P2P architecture are not fully utilized. Therefore, if it is not dealt with appropriately, free riding poses a serious threat to the wide-spread use and efficient operation of P2P systems.

In a recent study performed on the Gnutella network, it was observed that 85% of peers do not share any files at all [30]. Moreover, the top 1% of sharing peers provides 50% of all query hits, and the top 25% provides 98%. The experiments on Napster showed that about 20-40% of the Napster peers share little or no files [31]. It was also observed that many peers misreport their bandwidth. About 30% of the peers reported their bandwidth as 64 Kbps or less, however, it was found that they actually have a significantly higher bandwidth.

Most of the P2P systems in use lack effective mechanisms implemented against free riding and therefore suffer from free riding. To address this requirement, some approaches have been proposed to incorporate incentives in the existing protocols to stimulate cooperation among peers. These approaches can be categorized into three main groups: *micropayment-based, reciprocity-based,* and *reputation-based incentive mechanisms* [29].

B. Micropayment-Based Incentive Mechanisms

Micropayment approaches (e.g., [32], [33]) require peers to pay for the services they get or resources they consume. It is aimed to encourage peer cooperation within P2P systems by providing an efficient and secure pricing mechanism. Typically a central authority is employed to ensure honest transactions between peers. The central authority is responsible for keeping track of peer accounts, distributing virtual currency, and providing security. The problem with this approach is that the requirement for centralized authority conflicts with the nature of P2P paradigm which is highly distributed.

C. Reciprocity-Based Incentive Mechanisms

In reciprocity-based approaches, each peer decides how to react to another peer's service request based on the past behavior of that peer to its requests. As an example, BitTorrent is a P2P file-distribution system incorporating user incentives in its protocol for sharing network resources [34]. It employs a Tit-for-Tat mechanism to decide to which peer a file will be uploaded and at what bandwidth. A peer uploads to the peers that give it good downloading rate. The other peers are not allowed to download.

The incentive mechanism described in [35] is based on the local interactions of peers. Each peer assigns ratings to its neighbors depending on the reaction of the neighbors to its service requests, and the service quality offered to the neighbors is determined by those rating values.

D. Reputation-Based Incentive Mechanisms

A P2P reputation system is used to produce a reputation rating for the peers (e.g., [36], [37]). Contribution of the peers to the system is monitored to determine the reputation rating. The peers with high reputation in that rating are offered better services. Interaction between peers leads to the generation of local reputation information which is then spread through the network to produce global reputation for peers.

There are some important issues that are difficult to be ensured for the implementation of these methods. For instance, enabling the security and availability of reputation information in a P2P environment is not an easy task. Another major difficulty of implementing reputation-based (and also reciprocity-based) methods is peer identity management. Since peers can obtain network identities easily, they can change their online identities at any time. Any incentive mechanism depending on the strong identities of peers is in conflict with the peer anonymity goal of the P2P paradigm.

A recent protocol proposed in [38] adaptively modifies the topology of an unstructured P2P network in reaction to the contributions of peers. New connection types that can be dynamically established among peers are introduced with the aim to bring contributing peers closer to each other and to push the free riders away from the contributors. The protocol ensures that contributing peers are favored in getting service from the P2P network.

X. CONCLUDING REMARKS

We have outlined the key research issues relevant to data management in Peer-to-Peer (P2P) systems. We have discussed how the distinct features of P2P systems make the management of data a difficult problem. While describing various data management issues we have also addressed the associated challenges that need to be overcome for efficient operation of P2P systems. There are numerous open research problems and directions in this area as discussed throughout the paper.

ACKNOWLEDGMENT

I would like to thank Ismail Sengör Altıngövde and Rıfat Özcan for their help with drawing figures. This work is partially supported by The Scientific and Technical Research Council of Turkey (TUBITAK) with grant number EEEAG-105E065.

REFERENCES

[1] Gnutella. http://www.gnutella.com/.

[2] Napster. http://www.napster.com/.

[3] Freenet. http://freenetproject.org/.

[4] BitTorrent. http://www.bittorrent.com/.

[5] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp.149-160, 2001.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A Scalable Content-Addressable Network", *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp.161-172, 2001.

[7] S. Androutsellis-Theotokis, D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies", *ACM Computing Surveys*, vol.36, no.4, pp.335-371, 2004.

[8] P. Valduriez, E. Pacitti, "Data Management in Large-Scale P2P Systems", *High Performance Computing for Computational Science (VECPAR)*, pp.104-118, 2004.

[9] R. Blanco et al., *A Survey of Data Management in Peer-to-Peer Systems*, Technical Report CS-2006-18, David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2006.

[10] G. Koloniari, N. Kremmidas, K. Lillis, P. Skyvalidas, E. Pitoura, *Overlay Networks and Query Processing: A Survey,* Technical Report TR2006-08, Computer Science Department, University of Ioannina, 2006.

[11] J. Risson, T. Moors, "Survey of Research towards Robust Peer-to-Peer Networks: Search Methods", *Computer Networks*, vol.50, no.17, pp.3485-3521, 2006.

[12] B. Yang, H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks", *International Conference On Distributed Computing Systems (ICDCS)*, pp.5-14, 2002.

[13] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks", *International Conference on Supercomputing*, pp.84-95, 2002.

[14] F. Banaei-Kashani, C. Shahabi, "Criticality-Based Analysis and Design of Unstructured Peer-to-Peer Networks as Complex Systems", *IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp.351-358, 2003.

[15] A. Crespo, H. Garcia-Molina, "Routing Indices for Peer-to-Peer Systems", *International Conference On Distributed Computing Systems (ICDCS)*, pp.23-32, 2002.

[16] G. Wiederhold, "Mediators in the Architecture of Future Information Systems", *IEEE Computer,* vol.25, no.3, pp.38–49, 1992.

[17] I. Tatarinov et al., "The Piazza Peer Data Management Project", *ACM SIGMOD Record*, vol.32, no.3, pp.47-52, 2003.

[18] W. S. Ng, B. C. Ooi, K.-L. Tan, A. Zhou, "Peerdb: A P2P-Based System for Distributed Data Sharing", *International Conference on Data Engineering (ICDE),* pp.633-644, 2003.

[19] W. Nejdl et al., "EDUTELLA: A P2P Networking Infrastructure Based on RDF", *International Conference on World Wide Web (WWW),* pp.604-615, 2002.

[20] C. Rohrs, "Query Routing for the Gnutella Network", Unpublished Work. Available at http://rfc-gnutella.sourceforge.net/src/qrp.html, 2002.

[21] I. Clarke, S. Miller, T. W. Hong, O. Sandberg, B. Wiley, "Protecting Free Expression Online with Freenet", *IEEE Internet Computing,* vol.6, no.1, pp.40–49, 2002.

[22] M. Cai, M. Frank, J. Chen, P. Szekely, "Maan: A Multi-Attribute Addressable Network for Grid Information Services", *IEEE International Workshop on Grid Computing (GRID)*, pp.184-191, 2003.

[23] F. Dabek et al., "Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service", *IEEE Workshop on Hot Topics in Operating Systems (HoTOS)*, pp.81-86, 2001.

[24] A. Datta, M. Hauswirth, K. Aberer, "Updates in Highly Unreliable, Replicated Peer-to-Peer Systems", *International Conference on Distributed Computing Systems (ICDCS)*, pp.76-85, 2003.

[25] E. Cohen, S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks", *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp.177-190, 2002.

[26] G. Koloniari, E. Pitoura, "Peer-to-Peer Management of XML Data: Issues and Research Challenges", *ACM SIGMOD Record,* vol.34, no.2, pp.6-17, 2005.

[27] A. Crespo, H. Garcia-Molina, *Semantic Overlay Networks for P2P Systems*, Technical report, Computer Science Department, Stanford University, 2002.

[28] M. Khambatti, K. Dong Ryu, P. Dasgupta, "Structuring Peer-to-Peer Networks Using Interest-Based Communities". *International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, pp.48-63, 2003.

[29] M. Karakaya, I. Korpeoglu, Ö. Ulusoy, "Free Riding Problem in Peer-to-Peer Networks", submitted manuscript, 2007.

[30] D. Hughes, G. Coulson, J. Walkerdine, "Free Riding on Gnutella Revisited: the Bell Tolls?", *IEEE Distributed Systems Online*, vol.6, no.6, 2005.

[31] S. Saroiu, P. Krishna Gummadi, S. D. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts", *Multimedia Systems*, vol.9, no.2, pp.170-184, 2003.

[32] V. Vishnumurthy, S. Chandrakumar, E. Gun Sirer, "KARMA: A Secure Economic Framework for P2P Resource Sharing", *Workshop on the Economics of Peer-to-Peer Systems,* 2003.

[33] M. Ham, G. Agha, "ARA: A Robust Audit to Prevent Free-Riding in P2P Networks", *IEEE International Conference on Peer-to-Peer Computing*, pp.125-132, 2005.

[34] B. Cohen, "Incentives Build Robustness in Bittorrent", *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[35] M. Karakaya, I. Korpeoglu, Ö. Ulusoy, "A Distributed and Measurement-Based Framework Against Free Riding in Peer-to-Peer Networks", *IEEE International Conference on Peer-to-Peer Computing*, pp. 276-277, 2004.

[36] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks", *ACM Conference on Computer and Communications Security*, pp.207-216, 2002.

[37] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks", *International Conference on World Wide Web (WWW),* pp.640-651, 2003.

[38] M. Karakaya, I. Korpeoglu, Ö. Ulusoy, "A Connection Management Protocol for Promoting Cooperation in Peer-to-Peer Networks", *Computer Communications*, to appear, 2007.