2009 Fourth International Conference on Internet and Web Applications and Services

Impact of Self-Organization in P2P Overlays on Underlay Utilization

Konstantin Pussep*, Simon Oechsner†, Osama Abboud*, Miroslaw Kantor‡ and Burkhard Stiller§
* Multimedia Communications Lab, Technische Universität Darmstadt
Email: {pussep,abboud}@kom.tu-darmstadt.de
† Institute of Computer Science, University of Würzburg
Email: oechsner@informatik.uni-wuerzburg.de
‡Akademia Gorniczo-Hutnicza im. Stanislawa Staszica W Krakowie
Email: kantor@kt.agh.edu.pl
§CSF@IFI, University of Zürich
Email: stiller@ifi.uzh.ch

***Abstract*—Peer-to-Peer (P2P) systems gained popularity and are responsible for a large share of today's Internet traffic. Nevertheless, their dynamic nature and the intended lack of control through central instances make their behavior unpredictable and, therefore, it is difficult to achieve a high level of Quality-of-Service for P2P traffic. Thus, peers are themselves responsible for dealing with these issues by applying so-called self-organization mechanisms to deal with their heterogeneity, unpredictable behavior, and asymmetric resources. This paper discusses and classifies relevant self-organizing aspects of P2P systems, including metrics and mechanisms. Hereby, the key focus is in better understanding on how such self-organizing mechanisms - originally designed to improve the performance of P2P overlays - affect the underlying Internet infrastructure.**

I. Introduction

The peer-to-peer (P2P) paradigm has proven its importance through many popular applications and is responsible for a large part of Internet traffic [1]. The unpredictable environment and the intended lack of central control force P2P systems to adapt themselves to the dynamics of user participation (acting both as consumers and providers), variable load, and unstable resources. P2P systems try to overcome these limitations through a high level of self-organization, especially regarding the creation of overlay structures and content exchange mechanisms. This enables them to achieve better QoS for users, better utilization of peer resources, and better adaptability to a changing environment.

Typical P2P overlay applications apply self-optimization metrics in order to improve their performance for just the users, or in case of hybrid applications (such as Skype or Zattoo) for users and the overlay provider. This results often in a negative impact on underlying Internet resources (such as bandwidth, traffic, and traffic shapes) that are used inefficiently, if their utilization is not considered by self-organization mechanisms. Therefore, several initiatives [2], [3] try to alleviate the negative impact of overlay-based self-organization on the Internet by optimizing the usage of network links. This requires a thorough understanding of relevant Self-organization Mechanisms (SOMs) and their interplay with Internet resources.

As a general observation, adaptation mechanisms vary a lot among different applications, mainly depending on their scope and goal. However, common reoccurring requirements and mechanisms can be identified in different systems, which deal, e.g., with an overlay performance, the lack of global information about the system wide state, and an insufficient knowledge about the underlying networks. Since such self-organization mechanisms are the key property of P2P applications, better understanding them can help to achieve higher QoS for users, better utilization of underlay resources, and better adaptability to changing environment.

This paper focuses on the classification of Self-organization Mechanisms for P2P overlays, especially on those metrics used to perform self-organization and on those mechanisms that use these metrics. We analyze how these mechanisms residing at the overlay level impact the Internet infrastructure and which measures can improve the interaction with the underlay.

In Section II, main properties of self-organization in the context of P2P systems are identified. Accordingly, self-organization in P2P networks is classified into two main categories, (a) self-organization of the overlay structure (Section III) and (b) self-organization of resources (Section IV). Finally, Section V draws final conclusions.

II. Classification of Self-organization

While Self-organization (SO) is known and defined in different ways in physics, chemistry, biology and a variety of other fields, a common denominator in all these definitions is that self-organization in a system increases its complexity without any external influence. In the context of P2P networks, this means that a network structure evolves by local decisions made by the nodes participating in the network, without any higher authority directly intervening.

In P2P systems self-organization is defined at the level of overlay networks, i.e. logical networks above the physical network. By implication, the physical network is called the underlay. All SOMs aim at improving some characteristic of the supported application to obtain faster response times or a higher degree of availability. Therefore, the term *self-*

978-0-7695-3613-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICIW.2009.20

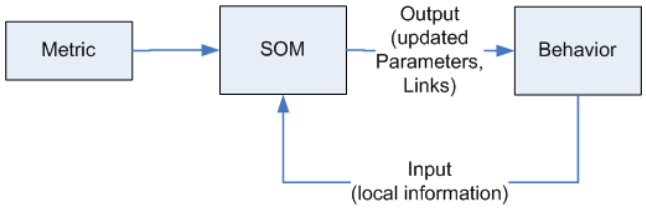

Fig. 1. SOM overview.

organization is tightly related with *self-optimization* in most of the mechanisms. However, for the sake of consistency, we will use the term self-organization throughout the rest of this paper. Based on the classification of self-organization given in [4] we identify the most relevant SO criteria for P2P systems:

- Mutability: The system's ability to change its structure by rearranging connections among peers.
- Organization: The system is able to create an appropriate structure for improved performance.
- Metrics: The system is able to detect any environmental perturbations.
- Adaptivity: The system can react on changing metric values accordingly.
- Feedback: The adaptive nature of the system leads to changes in the system's performance. The system collects this information (locally and/or from other peers) and provides this feedback to the adaptation activity.
- Randomness: This feature allows to create and update complex structures without global knowledge, e.g., unstructured overlays use random connections to achieve high diversity of neighbors.
- Emergence: The whole system exposes properties that are not present at single peers. E.g. a P2P file system system can make a large amount of content available in a long term that is not possible to single peers due to limited storage capacity and possible failures.

A SOM is a concrete algorithm implemented at each peer forming the overlay. It makes the local decisions that, in interplay with the decisions made at other peers, achieve self-organization. In order to influence the overlay network, SOMs make some kind of choice, e.g., between peers used as overlay neighbors. This choice is based on locally available data that is the input to the algorithm (see Figure 1). This data may be provided by other peers or by the underlay. The choice is made by applying a metric to this input. This metric provides semantics to the choice process by defining what makes one alternative *better* than the other.

III. Self-organization of Overlay Structure

A self-organization of the overlay structure changes the logical links interconnecting peers according to quality criteria and system state. This is generally done at the peer level, where each peer makes local decisions, on the peers it selects as neighbors, which peers to upload to, etc. These decisions are made based on locally available information about the candidate peers. The outcome of all these local changes in their sum affects the complete overlay structure. A selection among two or more candidates is made based on a ranking of these peers according to a metric. We will describe some possible metrics first before listing several SOMs that utilize them. For the SOMs, we discern between neighbor selection and resource exchange mechanisms. The former govern which peers are directly known by a local peer, i.e., to which peers an overlay connection is maintained. The latter are applied for utilizing these connections, especially in the context of content distribution. Specifically, they govern with which neighbors data is exchanged.

A. Metrics for Peer Evaluation

Typical overlay-based applications have some degree of freedom in choosing their neighbors. For example, in the Kademlia Distributed Hash Table (DHT) [5] there are often several alternative peers which can be put into a certain bucket. Similarly, in a content distribution overlay such as BitTorrent, a peer can decide which requesting peer is to be served. Many of the SOMs presented here are based on influencing this choice. Peers are selected with a certain goal in mind, which makes one peer a better choice than another. This ranking of peers for a given purpose, e.g., uploading a chunk, is made by applying a metric. Table I gives an exemplary collection of different metrics.

The simplest 'metric' that can be applied is a random choice between the candidates. This is a first step to distribute load and may be successful if the peer population is homogeneous. To make a more informed decision, different attributes can be considered. They may correspond to the underlying network, where a number of technical metrics are possible, semantic or social factors, such as fairness, similarity or social association between peers. Another very important metric is the underlying network's cost, which is not considered by most overlay applications.

The applicability of the metrics depends on the overlay type and goal, e.g., search overlays are interested in other performance indicators (such as RTT) than content distribution overlays (often throughput, in case of VoD or live streaming also RTT). Combinations of metrics for a SOM are possible. For example, the tit-for-tat strategy used in BitTorrent conceptually combines an underlay metric (the available bandwidth of a peer for upload) with a social metric (how freely this bandwidth is provided to other peers).

Additionally, there are metrics for content replication that evaluate relevant parameters for keeping content available in an efficient way. They are used to decide about the generation and placement of new copies in an overlay.

B. Neighbor Selection Mechanisms

An overlay can be seen as a graph with overlay nodes being vertices of this graph and the connections between nodes being edges of the graph. In this Section, we define neighbors of a local peer as the remote peers it stores in the form of their

TABLE I
CLASSIFICATION OF METRICS.

Type of metric	Examples	Used in
Load Distribution	Random	Gnutella v0.4
	Round Robin	BitTorrent (seeding)
Underlay	RTT	Pastry
	Bandwidth	BitTorrent, Tribler, Skype (Supernodes)
	Reachability (e.g., NAT)	Skype (Supernodes)
	Provider information	P4P
Social	User interests	Tribler
	Reputation	eMule, BitTorrent, Tribler
Replication	Availability	DHTs
	Popularity	Web caching, CDNs
	Overlay Size	
	Node Capacity	

network addresses, i.e., the peers it can contact directly without an additional lookup.

In this context we must again distinguish between search overlays (e.g., DHTs) and content distribution overlays (e.g., BitTorrent or eDonkey). In the latter, neighbors as defined above may or may not be used, depending on the content they store and, e.g., on the upload capacity they provide to the local peer. Here, another peer selection process takes place that operates on the known neighbors. Thus, it is usually enough to know a sufficient number of other nodes in order to be able to apply this second selection, which is described in more detail in the next subsection. In contrast, neighbor selection is typically very important in a search overlay in order to assure short respond times and high query success rates. Each neighbor may be used for a specific query route at any time, and neighbors normally have to fulfill certain criteria to be eligible, e.g., their IDs have to fall into a certain range of the overlay ID space. Thus, the neighbor relationships are very pronounced here and normally only change when existing peers go offline or new peers join. As a consequence, the mechanisms described in the following have a much higher relevance for search overlays than for distribution overlays.

1) Proximity-based Routing: Proximity neighbor selection and proximity routing [6] are mechanisms that route overlay messages while favoring peers that are close to the local peer in terms of a metric. Here, proximity neighbor selection tries to fill the routing tables of peers mostly with peers that are close according to the metric while proximity routing selects among several next hop candidates the closest one. The selection of the next hop during a message forwarding can also take into account short term fluctuations in network conditions. However, this is paid for by the larger routing table and a potentially higher routing complexity. Also, the optimization potential depends on the choices available. As shown in [7] different proximity metrics result in different impact on the underlay. Optimization for the Internet domain of peers results in lower inter-domain traffic and optimization for the RTTs in lower end-to-end delays.

2) Geographic layout: Another approach similar to proximity routing and proximity neighbor selection is the geographic layout [6]. Here, the aim is to structure the overlay in a way that preserves neighbor relationships from the physical network, i.e., peers that are close in the physical network according to a metric are ideally also neighbors in the overlay. This can be done by adapting the ID generating process of the peers, so that locality information pertaining to the metric is included. This creates a mapping between overlay and underlay, trying to limit the overhead introduced by single overlay hops that span several long physical links. This mechanism is mostly applicable to DHTs, since they create relations based on node IDs.

3) Supernode selection strategies: The term supernode denotes peers that play a special role in the network, somewhat violating the pure P2P approach. Examples are the ultrapeers or hubs from Gnutella, or the supernodes in the Skype network. They are normally selected from the complete set of peers. Supernodes are employed to reduce load on 'normal' peers. In the example of Gnutella, they handle most of the query traffic, while in Skype they are also used to establish indirect voice connections between two peers that can not connect directly to each other, e.g., due to firewalls or NATs. As a consequence, not every peer is useful to be selected as a supernode (or to promote itself to that status). Normally, peers are useful as supernodes if they can provide enough resources, e.g., bandwidth and processing power, and if they are easily reachable by all peers. Under these conditions, a peer might become a supernode if not enough supernodes exist in the network.

4) Discussion: The mechanisms presented above can be modified to take into account the underlying topology and to reduce the inter-domain traffic as demonstrated for Kademlia. However, the impact of search overlays on the usage of Internet resources is lower than that of content distribution overlays, which are responsible for the actual exchange of large amounts of data. Overlays utilizing supernodes might be an exception here, e.g., if these nodes are used to relay user voice data.

C. Resource Exchange Mechanisms

In the context of resource exchange mechanisms, two major selection algorithms are used that can together be described as cooperation strategy. To explain these algorithms, We consider a peer X that wants to download content. Since most distribution overlays use multi-source download, it signals its download request to several other peers. The choice of peers that are requested for content is one part of the *peer selection* process: peer selection at the downloading peer. A peer Y receiving X's download request queues it locally with all the other download requests. When resources for a new upload at Y are available, then Y has to decide which peer it wants to upload data to. This is the peer selection at the uploading peer, which normally is the more critical peer selection. The mechanisms described below fall into this category.

1) Tit-for-tat: The uploading peer selection in BitTorrent is called choking/unchoking algorithm and uses a tit-for-tat principle described, e.g., in [8]. It tries to achieve pareto efficiency by assuring that peers reciprocate by uploading to peers that upload to them or - the other side of the coin - that peers refuse to upload to peers that do not upload to them. Disallowing downloading from peers is called choking, while allowing is called unchoking, e.g., peer p is choked at peer q if it is in the neighbor list of peer q, but not currently downloading anything. A number (typically three) peers with the best upload rates to the local peer in the last measurement interval are unchoked, while the rest is choked. An exception is the optimistic unchoking, which selects a random interested peer in order to discover new good data exchange connections. With this strategy, peers that utilize their upload bandwidth for the overlay are rewarded, which makes the so-called 'free-riding', i.e., the consumption of resources without reciprocating, less efficient. Also, it improves the generation rate of new copies of chunks in the whole overlay. On the other hand too strict compliance with this policy restricts the applicability of other metrics that could improve the download speed or usage of the underlay resources.

2) Credit-point systems: The credit point system used by the eMule client (a client used to connect to the eDonkey network) accumulates points at the peers that content is uploaded to. If peer A uploads some amount of data to peer B, B adds credit points for A locally. These credit points allow A to spend less time in the upload queue of B if A wants to download something from this peer. Since all credit points benefiting A are stored on remote peers, it is assumed that no counterfeiting of credit points is possible. An interesting difference between the tit-for-tat mechanism and this credit point system is the fact that credit points accumulated by uploading one or more files can be used to download other files quicker, whereas tit-for-tat tries to impose fairness in the sharing of the same file. Also, a credit point system allows a peer with low upload bandwidth to achieve a good standing with other peers, given enough time. There are also a number of reputation based systems in the same spirit which we will not describe due to space limitations.

3) Discussion: The presented mechanisms aim to provide incentives for user contribution at the application layer to improve application-level quality. Additionally, they govern between which peers data is exchanged, and therefore have an impact on the traffic pattern generated by an overlay. Therefore, in order to improve the usage of Internet resources such mechanisms are a prime candidate to be combined with "lower-layer" performance metrics (cf. Table I).

IV. Self-organization of Resources

Apart from the explicit requests of resources from other peers, different SOMs are employed in search and distribution overlays to ensure that data is efficiently distributed and does not get lost due to churn, and to facilitate an easier access to resources by placing them intelligently in the network. In the following, some of these mechanisms are discussed.

A. Chunk Selection Strategies

Distribution overlays work much more efficiently if they do not transmit files as a whole, but segment it into smaller parts, which can then be provided and downloaded separately and from different sources. This is called Multi-Source Download (MSD). Most popular P2P distribution overlays use MSD to exchange data. Peers are not limited to one connection at a time, but may receive data of the same file from several peers at the same time. Similarly, a peer may upload data to more than one peer at the same time.

To enable MSD, files are usually segmented into smaller parts, which can be shared by a peer even if the complete file has not yet been downloaded. This greatly speeds up the spread of especially larger files, since peers can start to upload data much sooner. The file segments are commonly called chunks, which are in many cases partitioned again. Thus, when a peer X was selected for an upload slot at peer Y using a mechanism described in Section III-C, it has to be decided which chunk will be transferred. Of course, only chunks that Y has stored can be uploaded, and only chunks that X is missing are useful. This *chunk selection* takes place from the overlap of these chunk sets. We will describe several of such *chunk selection mechanisms* in the following.

1) Random: The easiest method to choose a chunk to upload is to select a random chunk available at the uploader that is missing at the downloader. This strategy however is prone to the problem of chunk starvation. Chunk starvation means that only a very small number of copies of a chunk are available in the overlay, while the other chunks are well spread. In the extreme case even no copy is left of the starved chunk. Peers that want to finish the download are normally just missing the starved chunk, which leads to a high competition at the few peers that can upload this chunk. As a consequence, the download times for that chunk increase. Moreover, if peers finishing their download do not act as seeders afterwards, the chunk remains starved.

2) Least Shared First: To prevent chunk starvation, overlays like, e.g., BitTorrent, employ a Least Shared First (LSF) chunk selection strategy, also called Rarest First [8]. This strategy chooses an eligible chunk that is spread least in the network. This works especially well in networks where an overview on the chunk population of a given file can be provided easily, as in the case for the tracker-based BitTorrent. If no entity which has global information about the chunk distribution exists, peers have to estimate this distribution.

3) CygPriM: The CygPriM strategy (cyclic priority masking) [9] tries to keep the chunk distribution balanced although only locally available information is used. It does not serve any chunk requests, but uploads only one defined chunk at a time. When this chunk is downloaded, the next available chunk from the file is offered, and so on. When the last chunk has been uploaded, the cycle begins anew. Chunks that are not requested by any of the peers are skipped. The strategy is robust against selfish peers and offers download times in the same order of magnitude as in a best-case scenario with benevolent peers.

TABLE II
REPLICATION STRATEGIES

Granularity	Distribution	Access type	Update mode	Replica placement	Location
file size	uniform	read-only	active	random	Central index
blocks	$\propto$ request rate	read-write	passive	availability-based	DHT based
erasure codes	$\propto \frac{requestrate}{filesize}$			capacity-based	super-peers

4) Closest Deadline: For the adaptation of chunk-based distribution overlays to streaming applications, not only the availability of chunks is important, but also their usefulness for the peers. A peer that already started viewing content while downloading might have passed the point in time where a chunk would have been played out. Therefore, this chunk is no longer of interest for that peer. On the other hand, missing chunks that are nearing their playout time have a higher priority to be downloaded. As a result, chunk selection includes the deadline of chunks in its strategy. An example named BiToS is presented in [10], where chunks are sorted in two categories, a high priority set and a low priority set. The high priority set contains chunks that are close to their playout time, which have a higher probability to be downloaded. For stability, the larger low priority set again follows the Least Shared First strategy. A similar strategy with three priorities is used in Tribler [11].

B. Data Replication Strategies

Replication is a strategy in which multiple copies of some data are stored at multiple sites [12]. This strategy is responsible for the maintenance of on-line copies of data and other resources. By storing the data at more than one site, if a data site fails, a system can operate using replicated data, thus increasing availability and fault tolerance. At the same time, as the data is stored at multiple sites, the request can find the data close to the site where the request originated, thus, increasing the performance of the system. On the other hand, replication introduces overhead due to creating, maintaining and updating the replicas, which may neutralize the advantages gained by replicating content.

Table II shows an overview of mechanisms used to optimize replication strategies. For example, replication strategies can consider the replica granularity, i.e., full file replication, block level replication, or erasure codes [13]. On the other hand, the replica distribution strategies compute the required number of replicas either uniform (same number of copies for each item), or based on the item popularity [14]. In the latter case, the desired number of replicas can be proportional to the logarithm of object's popularity to file size ratio [15]. The creation of replicas can be either done locally by the peers who requested the object, along the path through which the request is satisfied, or at random places.

In a P2P system the issues of object *availability* is especially critical due to the variable connectivity of nodes. In an environment with a wide variability, the system should not place replicas blindly: more replicas are required when placing on hosts with low availability, and fewer on highly available hosts. Moreover, the results of research done by [16] show that the number of hosts running Gnutella is well correlated with time of day. As a consequence, placing replicas in out-of-phase time zones may be a sound replication strategy.

1) Coding Methods: The goal of redundancy provisioning in overlay networks is related to fault-tolerance assurance. The availability to all resources is ensured only when a certain resource is not stored in one node, but is in some way stored in many sites simultaneously. The main difference of so-called redundancy methods for overlay networks with simple replica systems lays in the fact that for the latter systems we have a number of copies of protected resource (i.e. a file) distributed across the system. On the other hand, redundancy in this context means that each object is divided in some number of fragments (m), which are encoded into n ($n > m$) entities and stored at n entities across the overlay network [17]. The idea of erasure codes lies in allowing for item reconstruction on the basis of any m collected fragments. The advantage of this method above simple replication is that the storage requirements are lower across nodes. It also ensures the higher level of availability when the nodes have lower reliability parameters over replica systems

C. Caching Strategies

In contrast to dedicated caches offered by Internet Service Providers or Content Delivery Networks, we present here caching methods where peers themselves also provide the functionalities of caches. Overlay caching can be seen as a passive replication mechanism, since replicas are only created at peers that also consume files. Here, the main focus is put into cache replacement policies that should assure a high hit rate for object requests. Besides the policies known from other caching scenarios, such as LFU and LRU, there are policies customized for P2P caching overlays. Interestingly, due to the large amount of involved P2P caches and the resulting cache diversity simple policies such as FIFO perform similar to sophisticated ones [14].

For example, in [18] an efficient cooperative caching scheme for video-on-demand service over P2P networks is proposed. It applies a utility-based fine-grained policy that takes into account a peer's bandwidth and works at the level of segments instead of files. In this architecture, published videos are encoded into multi-layered source bit stream and split into many segments, which are distributed to overlay peers. The main design philosophy is that outbound bandwidth from multiple peers who have cached the same title can be aggregated to serve a single video streaming request.

D. Discussion

The mechanisms presented in this section can improve the overlay performance in different ways. From the user's point of view, they improve data availability and can result in better download success rates and higher download speed. If underlay topology is taken into account for replica placement and cache replacement strategies the utilization of underlay resources can be also improved.

We further observe that sophisticated resource management strategies require the knowledge about the current state of the system. This knowledge can be used to define metrics that are used to make resource management decisions, e.g., which objects to keep in nodes' caches, on which node to place a new replica, or how many replicas of a file should be created. We identify the following system characteristics relevant to make such decisions: object popularity measured by the absolute or relative request rate,

V. Summary and Conclusion

Self-organization Mechanisms (SOMs) determine an essential part of P2P overlay networks and facilitate easy and efficient deployment of P2P applications by end users. This paper classified the most important mechanisms used in current overlays and discussed their design goals. It has been shown that certain mechanisms are applied mainly to search networks and others are applied to content distribution overlays. Additionally, the types of metrics have been discussed, especially those, which are used to provide a decision criterion to SOMs. While there are many overlay protocols and implementations, this paper provides an overview on these algorithms with the most representative features that are in spirit shared by most of them.

Actually deployed overlays, which create most of the traffic in today's Internet, are content distribution networks, used for e.g., file-sharing and more and more for video distribution. As a consequence, main mechanisms affecting the behavior of these overlays, namely peer and chunk selection, currently have a higher significance with respect to P2P traffic. Additionally, other mechanisms for managing content, such as caching, are of importance.

The authors believe strongly that a better understanding of different kinds of self-organization will allow for an improvement of the performance of P2P overlays by applying mechanisms for an optimized utilization of resources. This must include underlay resources, since P2P applications mostly rely on users contributing *free* or unused resources. However, resources that are free for users (such as bandwidth for flat-rate based Internet access) can be very expensive in the underlay. Self-organization Mechanisms that take into account underlay resources and, therefore, can reduce Internet traffic will probably only be applied if right incentives from the underlay are in place.

Appropriate Self-organization Mechanisms can reduce Internet traffic but will probably only be applied if right incentives from the underlay are in place. The current research interest in overlays supporting locality promotion leads us to the conclusion that Self-Organization Mechanisms able to efficiently include locality will be in the focus in the near future. Current approaches in this field include biased neighbor selection and biased unchoking, which both take locality into account but differ in the way they influence the overlay with it. While biased neighbor selection tries to manage the known contacts in the overlay according to locality, biased unchoking does so only for the actual data transfer connections. One of the first interesting topics is a comparison of these two approaches.

Acknowledgments

This work has been performed in the framework of the EU ICT Project SmoothIT (FP7-2007-ICT-216259). The authors would like to thank all SmoothIT partners for useful discussions on the subject of the paper.

References

[1] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is P2P dying or just hiding," in *IEEE Globecom*, 2004.

[2] "SmoothIT - simple economic management approaches of overlay traffic in heterogeneous internet topologies," http://www.smoothit.org/.

[3] H. Xie, A. Krishnamurthy, A. Silberschatz, and Y. Yang, "P4p: Explicit communications for cooperative control between p2p and network providers," *SIGCOMM*, 2008.

[4] R. Steinmetz and K. Wehrle, *Peer-to-Peer Systems and Applications*. Springer, 2004.

[5] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peeiptr information system based on the XOR metric," in *International Workshop on Peer-to-Peer Systems, (IPTPS)*, 2002.

[6] M. Castro, P. Druschel, Y. C. Hu, and A. I. T. Rowstron, "Topology-aware routing in structured peer-to-peer overlay networks," in *12th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS)*, 2003.

[7] S. Kaune, T. Lauinger, A. Kovacevic, and K. Pussep, "Embracing the peer next door: Proximity in kademlia," in *8th International Conference on Peer-to-Peer Computing (IEEE P2P '08)*, 2008.

[8] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," in *Conference on Internet measurement (IMC)*, 2006.

[9] D. Schlosser, T. Hoßfeld, and K. Tutschku, "Comparison of robust cooperation strategies for p2p content distribution networks with multiple source download," in *6th International Conference on Peer-to-Peer Computing (IEEE P2P '06)*, 2006.

[10] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "Bitos: enhancing bittorrent for supporting streaming applications," in *In IEEE Global Internet*, 2006.

[11] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. V. Steen, and H. J. Sips, "Tribler: A social-based peer-to-peer system," in *5th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.

[12] N. G. P. Bernstein, V. Hadzilacos, *Concurrency Control and Recovery in Database Systems*. Addison-Wesley Publishers, 1987.

[13] R. B. S. Goel, "Data replication strategies in wide area distributed systems," in *Enterprise Service Computing: From Concept to Deployment*, 2006.

[14] S. Tewari and L. Kleinrock, "Proportional replication in peer-to-peer networks," in *Proceedings of IEEE INFOCOM*, 2006.

[15] M. Zhong, K. Shen, and J. Seiferas, "Replication degree customization for high availability," *SIGOPS Operating Systems Review*, vol. 42, 2008.

[16] R. Bhagwan, D. Moore, S. Savage, and G. Voelker, "Replication Strategies for Highly Available Peer-to-Peer Storage," *Future directions in Distributed Computing*, 2002.

[17] A. Duminuco and E. Biersack, "Hierarchical codes: How to make erasure codes attractive for peer-to-peer storage systems," in *8th International Conference on Peer-to-Peer Computing (IEEE P2P '08)*, 2008.

[18] H. Guo, K.-T. Lo, and J. Li, "An efficient caching scheme for on-demand streaming service on overlay networks," *Consumer Communications and Networking Conference*, pp. 322–326, 2007.