2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing

Peer Interest-based Discovery for Decentralized Peer-to-Peer Systems

Andreea Visan[(1)], Mihai Istin[(1)], Florin Pop[(1)], Fatos Xhafa[(2)], Valentin Cristea[(1)]

[(1)]Faculty of Automatics and Computer Science, University POLITEHNICA of Bucharest, Romania
[(2)]Department of Languages and Informatics Systems, Technical University of Catalonia, Barcelona, Spain
{andreea.visan, mihai.istin}@cti.pub.ro, florin.pop@cs.pub.ro, fatos@lsi.upc.edu, valentin.cristea@cs.pub.ro

Abstract—The success of content distribution oriented peer-to- peer systems heavily depends on the resource discovery mechanism. In case of large-scale distributed systems, this mechanism must be scalable and robust. The paper proposes a structured solution for resource discovery in decentralized peer-to-peer systems, which is guided by peer interest in collaborating with other peers. The problem of discovering peers of interest has many applications in file sharing, in data-aware scheduling, and in optimizing the files and documents downloads. Moreover, if trust is added as another parameter to define peers of interest, the interest-based discovery is useful in trusted P2P applications. We focused on developing the overlay network to ensure a very small number of messages required to retrieve, insert or delete a file even in the case of a very large network containing millions of nodes. In the experimental validation we used OverSim, a simulation tool for P2P systems. The experimental results highlight the good performance obtained regarding message communication and system's scalability.

Keywords

peer-to-peer, decentralized networks, resource discovery, Oversim

I. INTRODUCTION

Peer-to-peer systems were designed for the sharing of resources such as content, storage, CPU cycles by direct exchange between participants, without the intermediation of a centralized system. The main goals of those systems refer to the ability to function, scale and self-organize even in the presence of a high degree of failures and transient populations of nodes, ensuring to maintain at the same time high connectivity and performance.

Such architectures have as inherent characteristics scalability, resistance to censorship and centralized control and increased access to resources. Administration, maintenance, responsibility for the operation, and ownership of peer-to-peer systems are distributed among the users. Finally, peer-to-peer architectures have the potential to accelerate communication processes and reduce collaboration costs through the ad-hoc administration of working groups.

The success of a content distribution oriented peer-to-peer system heavily depends on the peer interest-based discovery mechanisms involved, on its scalability and robustness. Interest may be defined in terms of similarity of resources provided, trust values associated, hardware resources available, etc. The design of a service and resource discovery is not a trivial task

because we have to deal with several issues such as bootstrapping, security and privacy, search result ranking, departure and failure handling and also efficient resource management, data representation and load balancing. For instance, large-scale decentralized systems suffer also from problems such as topology mismatching, pollution, collaboration issues, free riders and flash crowds.

Our work focuses on peer interest-based discovery in decentralized hierarchical peer-to-peer systems, having the main purpose to provide reduced number of messages sent to retrieve information about files or chunk files stored by nodes within the structured network, even for large scale networks, formed by millions of participants. In [15] some interesting work on clustering of peers for content search has been done.

The rest of this paper is organized in the following way: Section II presents the related work referring to peer interest-based discovery mechanisms used in different peer-to-peer systems. Section III briefly introduces the network's architecture we focus on in our research. Section IV describes the peer interest-based discovery mechanism proposed for the decentralized structured peer-to-peer system previously described. Section V presents the results obtained using OverSim as simulation environment. Section VI draws the conclusions.

II. RELATED WORK

This section briefly describes resource and service discovery mechanisms currently used in different peer-to-peer systems. The structure of a network refers to the way the overlay network is created and nodes and content are added [1], [10], [9]. Depending on their structure, networks can be cataloged in structured and unstructured.

In an unstructured network [14], the placement of content is unrelated to the overlay network and thus, it needs to be located. The searching mechanism ranges from brute force (for instance, network flooding with queries until the desired content is located) to random walks and routing indices. Obviously, this type of search has implications over availability, scalability and persistence. Unstructured systems are suitable for accommodating highly-transient node populations. Representative examples of unstructured systems are Napster [7], Gnutella [13], Kazza, Publius, FreeHaven, etc.

In order to overcome the scalability issues that unstructured systems were faced with, structured systems were developed
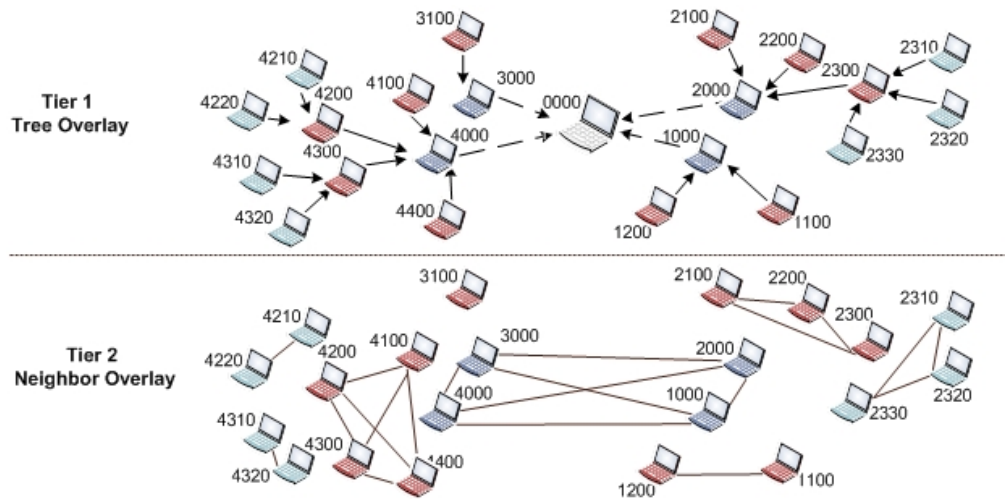
IEEE Computer society

Fig. 1. The SOPSys architecture. Tier 1 is represented by a multi-way well-balanced tree having a virtual root. On tier 2, each node maintains the list of its neighbors, its parent and also one neighbor of its parent, for redundancy.

[6]. The overlay topology is tightly controlled and files or pointers to them are placed at specific locations. These systems provide a mapping between content and location in the form of a distributed routing table. Based on it, queries can be efficiently routed to the node that stores the desired content. Structured systems represent a very scalable solution for exact-match queries. A drawback of this type of peer-to-peer system is that, in a very transient node population in which nodes are joining and leaving frequently, it is hard to maintain the structure required for efficient routing. Examples of structured systems are Chord, CAN, Tapestry, [11] etc.

Between the class of structured and unstructured systems we can distinguish another category represented by the loosely structured systems, in which, although the content location is not completely specified, it is affected by routing hints. Freenet is the representative example of loosely structured system.

We are also interested in the way resource registration is done, considering the following possibilities [8]: registration at a local server, registration at a unique server (Jini, Napster [7]), resources cached at node with closest identifier (Chord, CAN, Pastry) [11] or cached at nodes with a closer identifier (Freenet [5]).

The query routing method has a great importance and can by represented by a flood query (Gnutella [13], Jini), backtracking (DNS), a query forwarding (Freenet [5]), a query to a central server (UPnP, Napster [7]) or route to node with closest identifier (Chord, Tapestry, CAN) [11].

Resource naming and queries can be realized in several ways [8], [12]: using a hash identifier (Chord [11], Tapestry, CAN, Freenet [5]), string naming and queries (Gnutella [13], Napster), using directories or attributes (Jini, UPnP). Resource supported by the peer-to-peer system may be fixed (DNS), replicated (almost all of the presented P2P systems), mobile (Jini, UPnP) or dynamic resources (UPnP).

Resource discovery in a peer-to-peer system is influenced by several factors [4]. First of all, we have to mention the network topology that can have great influence on the search mechanism performance or protocol. Also, the overlay topology construction and topology mismatching have a great importance. For instance, in theory most of the DHT-based systems [3] can guarantee that any object in the network will be located in $O(\log(n))$ steps, where n represents the number of available nodes in the network. Designing the resource discovery also has to deal with several issues such as bootstrapping, security and privacy, search result ranking, departure and failure handling and also efficient resource management, data representation and load balancing.

Our research focuses on peer interest-based discovery for decentralized hierarchical peer-to-peer systems and its aim was to provide a very scalable mechanism, able to ensure a very small number of messages necessary to retrieve information referring to files stored by participants. Our approach considers the trust to influence the decisions regarding the peer's position within the network.

III. Peer Discovery Mechanisms Assumptions

This paper proposes a structured solution for resource discovery in decentralized hierarchical peer-to-peer systems. The architecture we focused on in our research is the one proposed in the SOPSYS project in the case of content distribution networks. This section briefly introduces its main characteristics and design considerations.

Figure 1 presents the multi-tier architecture for this type of self-organizing peer-to-peer system, architecture organized according to the trust value associated with participants. The first tier is organized as a well-balanced multi-way tree and is used for the message routing and resources discovery. In our example, the overlay network branching factor is considered equal to 4. The root of the network represents in fact a virtual root (named 0000 in Figure 1). The naming mechanism is based on the topology of this tier.
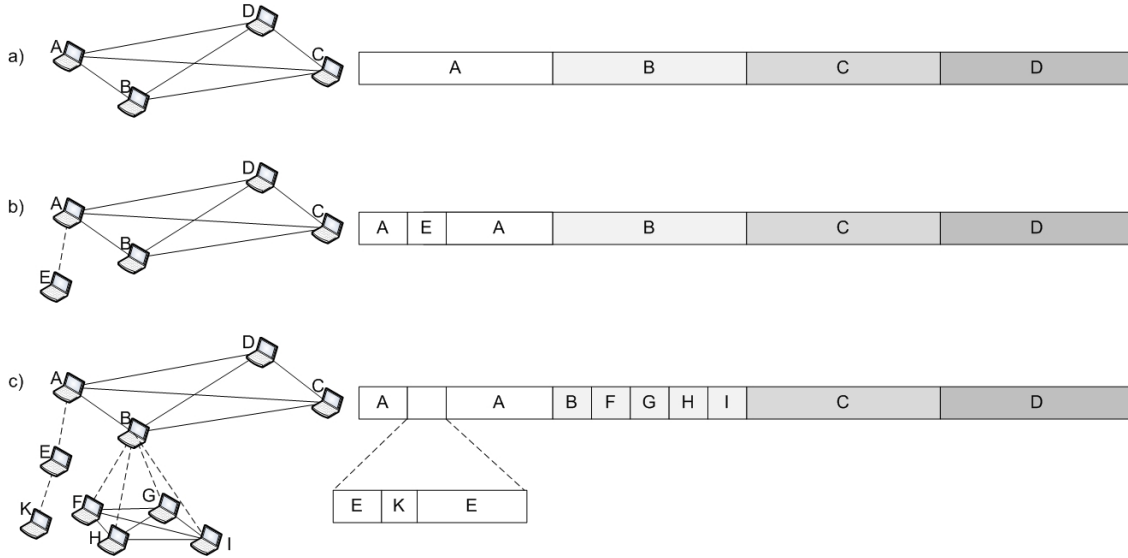
Fig. 2. Peer Discovery Mechanism in Structured P2P Networks, considering that the branching factor is equal to 4. a) For the first level in the overlay network, the domains are delegated similar to Chord, uniformly distributed. b) When E joins the network, he will be delegated by his parent A to be responsible for a chunk of his parent previous domain. c) B completes his level of children, thus, his previous domain is uniformly distributed between him and his children. K also receives from his parent E a file sub-domain.
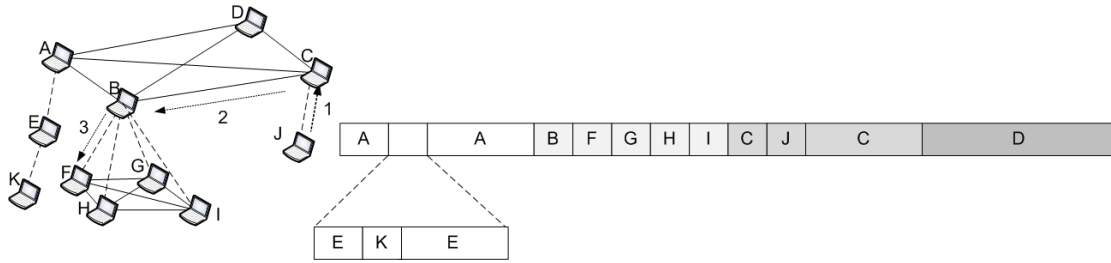

Fig. 3. Operation within a Structured Peer Discovery. When one peer want to search, insert or delete a file that doesn't belong to his parents' domains, the request should be forwarded by its ancient from the first overlay level to the corresponding node on the same level (message 2), and then, to the node on a higher overlay level delegated with the file hash (message 3).

The second tier contains two types of links for each node: links to all nodes that are placed within the same cluster and one link to a random sibling of its parent. In addition, each node will also hold a list of active links to some other nodes with which it has recently interacted.

Each node will have associated a trust value, computed based on feedbacks given by its partners, according to its availability, response time, bandwidth etc. The node chosen as root will be the most trustworthy. It will also be responsible for making the list of most trusted nodes publically available through a web site, but it will not be part of the routing process as it could cause a bottleneck.

The node naming maps to the multi-way tree architecture. For each node the corresponding identifier will be a fixed size sequence of slots, one for each tree level. For a node N, placed on the kth level in the topology, its identifier will have the same prefix as his parent for the first $k-1$ slots, a number representing the connection order of the child to its parent placed on slot k, and the rest of slots will be 0.

Each peer maintains a local cache containing information about its parent (from tier 1), its neighbors (from tier 2), a randomly chosen neighbor of its parent (to ensure fault tolerance) and also information about different peers to whom it had recently interacted.

Files in our content distribution peer-to-peer network will be retrieve according to their 160-bits hashes, computed using the SHA-160 algorithm. One or more nodes are responsible to store information about the existence in the network of a certain file.

IV. SEARCH MECHANISM FOR DECENTRALIZED STRUCTURED PEER-TO-PEER SYSTEMS

We consider that each peer connected to the network is responsible to respond to requests regarding files having certain hashes.

Each peer newly connected to the network receives from its parent a subset of keys to be responsible for. The responsibilities will be delegated in the following manner:

- For all nodes connected on the first level in the overlay network (nodes A, B, C, D in Figure 2a), keys are uniformly distributed to nodes, similar to the Chord network. Generally, these are pre-trusted nodes, belonging to the developers and dedicated to the network.
- For all nodes connected on a higher level, their domain represents a sub-domain of its parent domain, similar to Figure 2b,c. A parent uniformly splits its domain into $k + 1$ sub-domains, where k represents the branching factor of the overlay network. Each child will receive from its parent one of these domains, order according to their entrance into the network. For each sub-domain currently unassigned to another peer, the parent continues to be responsible for.

After its entrance within the network, a node will know its own domain, its neighbors', children' and direct parent domains. Nodes connected on the first level on the overlay network are able to forward messages between nodes responsible to different hashes domains.

Figure 2 represents an example of domain assignment, considering for the sake of simplicity the branching factor k equal to 4. Nodes A, B, C and D are those connected on the first level on the overlay network. The domain of the files hashes denoted by $\mathbb{D}$ is uniformly distributed between them. Generally, those nodes are pre-trusted nodes, dedicated nodes, which can belong to the network developers, etc.

$$\mathbb{D} = \mathbb{D}_A \bigcup \mathbb{D}_B \bigcup \mathbb{D}_C \bigcup \mathbb{D}_D, |\mathbb{D}_A| = |\mathbb{D}_B| = |\mathbb{D}_C| = |\mathbb{D}_D|$$

For all nodes connected to a higher level on the overlay network, the parent assigns to him a sub-domain. For instance, node E receives from its parent A, F, G from B are so on. Considering that the branching factor is equal to 4, $k \cdot |\mathbb{D}_E| = |\mathbb{D}_A|$ after E joins the network and considering that E is the only one child of A. When a parent completes a level of children, for instance, the situation of node B in Figure 2c), the following relation is valid:

$$|\mathbb{D}_B| = |\mathbb{D}_F| = |\mathbb{D}_G| = |\mathbb{D}_H| = |\mathbb{D}_I|$$

This schema of domains assignments assures the following property: *The smaller the level in the overlay network, the greater the number of files a node is responsible for.*

Thus, one node placed on a lower level in the overlay network, thus having a smaller trust value, will have associated a smaller number of files to be responsible for. A more trustworthy node will have associated a greater number of files than a less trustworthy node.

Each peer maintains a local information about the subset of files assigned to him, containing information structured in the following way: $< FileName, ListOfSources >$, where $FileName$ denotes the hash of the file and $ListOfSources$ the list of nodes that provide the resource. There is also the possibility to maintain not files locations but files themselves; this solution is discussed further in this paper.

Each operation that can be executed on a file (search, insert or delete) requires a number of $O(\log_k(N))$ messages to be sent from the requestor to the node responsible for the file's hash domain, where:

- N denotes the total number of nodes currently connected to the peer-to-peer network.
- k represents the branching factor.

This states because the joining algorithm (out of the scope of this paper) ensured that the overlay network is permanently kept as a well-balanced multi-way tree.

Algorithm 1 FILEOPERATION(FILE, OP, SOURCE)

1: {*myDomain* denotes the file hashes domain associated with the current node}
2: {*file* represents the hash of the file we want to search, insert or delete}
3: {*source* represents the overlay identifier of the request's source}
4: **if** file.isPartOf(myDomain) and !childResponsibleFor(file) **then**
5:   **if** op = INSERT **then**
6:     save locally the association between the file and its address (provider)
7:   **end if**
8:   **if** op = DELETE **then**
9:     delete the association between the file and the place earlier associated, if permitted
10:   **end if**
11:   **if** op = SEARCH **then**
12:     return to the requestor the list of nodes providing the resource
13:   **end if**
14: **else if** file.isPartOf(myChildrenDomains) **then**
15:   send the request to the corresponding child
16: **else if** file.isPartOf(myNeighborDomains) **then**
17:   send the request to the corresponding neighbor
18: **else**
19:   send the request to my parent
20: **end if**

A request trace example is presented in Figure 3 where node J wants to search for, insert or delete a file having the hash within the domain of B's subtree, namely in the F's domain. As node J has information only about its own domain and all domains for its children and neighbors (inexistent here) and its parent's domain and the file is not part of one of these domains, the request should be forwarded to the parent (message 1 in Figure 3). Node C, placed on the first level in the overlay network forwards the request received from its children J to node B because the requested file is placed within B's domain and C is aware about it. The request is further forwarded until it reaches the node responsible for the file's hash, node F in our example. The algorithm is detailed in Algorithm 1.

The search mechanism within a structured network guarantees that if a file exists in the network, it will be found by any requestor, considering that the node that stores the file address and also the node that stores the file itself are online.

TABLE I

Branching factor (k)	Number of completed levels	Total numbers of nodes (N)	Number of messages
16	4	69,904	7
16	5	1,118,480	9
32	4	1,082,400	7
32	5	34,636,832	9

TABLE II
SIMULATION SETTINGS

Network	Branching factor (k) of the overlay network	16
	Number of overlay levels	5
	Total number of nodes within the network	uniform random distribution over [40000, 69904]
Content distribution	Number of distinct files provided by one peer	uniform random distribution over [0,1000]
	Files hashes	uniform random distribution over the 160-bits domain
Simulation	Number of simulation steps	50

A. Data Replication

Our schema of files assignment ensures that a peer placed on a higher level within the overlay network, (thus having associated a higher trust value), will store information about a larger chunk of files than a peer placed on an inferior level, with a smaller trust value.

We consider one node to be responsible for storing locations of files with hashes within its hashes interval, interval for which it has been delegated by its direct parent from the tier 1 network. There is also the possibility to store files themselves but this approach supposes a great amount of data to be sent over the overlay network and does not eliminate the problem raised by unforeseeable disconnections of participants.

A very important aspect that we have to take into account in our peer interest based discovery is to replicate the information about files that can be provided by peers. We considered the following approach: information stored by a peer is replicated at its direct parent, if applicable. An unannounced disconnection of one peer placed on level l for instance, will cause any loose of information because the same information is replicated on the higher level, $l-1$.

In the situation of wanted disconnection of a peer placed on the last level of the network, the node in discussion sends a notification to its parent to announce its intention and to trigger the process of information replication on the first higher level, namely to the parent of its parent within the overlay network. If the node who wants to disconnect is place on a higher level, its place will be taken by its most trustworthy child.

V. EXPERIMENTAL TESTING

Experimental testing was made using Oversim [2], an open-source overlay and peer-to-peer network simulation framework written in C++. It is a very flexible and modular simulation framework (its architecture is represented in Figure 4) and can simulate any number of hosts.

The overlay message handler provides an RPC interface to facilitate dealing with timeouts and packet retransmission due to packet losses. All of the underlying network models have a consistent UDP/IP interface to the overlay protocols. The graphical interface of OMNeT++ is used to display overlay and underlay topology and all network packets in detail.
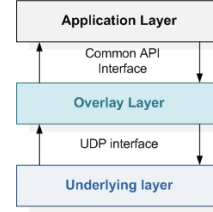

Fig. 4. Modular Architecture of Oversim (source: [2])

Table I presents the maximum number of messages required to retrieve information about one file within a structured network having the architecture presented in Section III. As previously mentioned, this number of messages depends on the branching factor of the network and on the total number of nodes currently connected in the network. This maximum number of messages (denoted by $\#messages$) is computed using the following formula:

$$\#messages = 2 \cdot \lceil \log_k(N) \rceil - 1$$

where k and N have the same meanings as previously mentioned. The maximum number of messages is encountered in the following situation (that is presented in Figure 3):

- The file requestor is placed on the last level of the overlay network.
- The node responsible to store information about the requested file is also placed on the last level of the overlay network.
- The ancients of these two nodes placed on the first level of the overlay network are different nodes.

Experimental testing has been done in the simulation settings presented in Table II, for a medium size network containing about 50,000 nodes. Each node provides a random number

(between 0 and 1000) of files. In each of the 50 simulation cycles, a randomly chosen node generates a request for a randomly generated file key. Figure 5 presents the number of messages sent in order to retrieve information about the file existence and in case of existence, its providers' overlay addresses. As expected, the number of sent messages in each simulation step is limited by $2 \cdot \lceil \log_k(N) \rceil - 1$, namely 9, considering the simulation settings described in Table II.
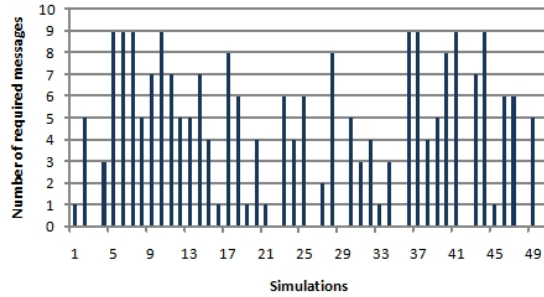

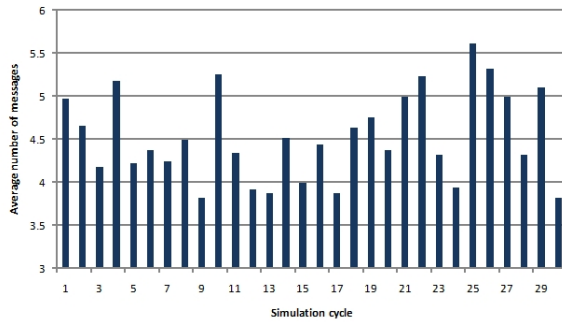Fig. 5. Experimental results for networks set according to Table II

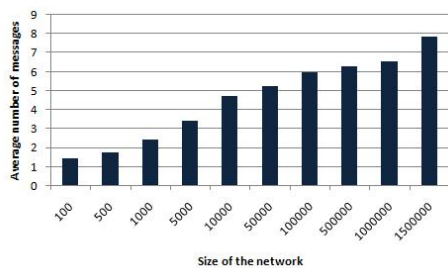Fig. 6. Average number of messages for fixed networks

Fig. 7. Average number of messages for increasing networks

Figure 6 presents the average number of messages required to retrieve information about the searched files. The average number for all experiments is equal to 4.48 messages, a reduced number of messages, taking into account the network size. One simulation cycle measures the average over 50

simulation steps. Figure 7 presents the average number of messages for different sizes of the network, exponentially increasing, considering $k = 32$.

VI. CONCLUSIONS

This paper proposed a peer interest-based discovery mechanism for decentralized structured peer-to-peer systems, focusing on network having the overlay organized as a multi-way well-balanced tree with virtual root. We ensure a very good performance and scalability, our schema requiring a reduced number of messages sent in order to retrieve information about stored files, even in large scale networks, taking maximum advantage on the hierarchical structured, organized according to the trust values associated with participants.

ACKNOWLEDGMENTS

REFERENCES

[1] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.

[2] Ingmar Baumgart, Bernhard Heep, and Stephan Krause. OverSim: A Flexible Overlay Network Simulation Framework. In *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, pages 79–84, May 2007.

[3] Andrew Brampton, Andrew MacQuire, Idris A. Rai, Nicholas J. P. Race, and Laurent Mathy. Stealth distributed hash table: a robust and flexible super-peered dht. In *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, pages 1–12, New York, NY, USA, 2006. ACM.

[4] Adeep S. Cheema, Moosa Muhammad, and Indranil Gupta. Peer-to-peer discovery of computational resources for grid applications. In *GRID*, pages 179–185, 2005.

[5] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46, 2001.

[6] Sameh El-Ansary and Seif Haridi. An overview of structured overlay networks. In *Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks*. CRC Press, 2005.

[7] Nic Garnett. Digital rights management, copyright, and napster. *SIGecom Exch.*, 2(2):1–5, 2001.

[8] Elena Meshkova, Janne Riihijärvi, Marina Petrova, and Petri Mähönen. A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. *Comput. Netw.*, 52(11):2097–2128, 2008.

[9] Rozlina Mohamed and Siti Zanariah Satari. Resource discovery mechanisms for peer-to-peer systems. *Computer and Electrical Engineering, International Conference on*, 2:100–104, 2009.

[10] John Risson, Tim Moors, John Risson, and Tim Moors. Survey of research towards robust peer-to-peer networks: Search methods. Technical report, Computer Networks, 2004.

[11] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.

[12] Girish Suryanarayana, Richard Taylor, and Girish Suryanarayana. A survey of trust management and resource discovery technologies in peer-to-peer applications, 2004.

[13] The Gnutella Protocol Specification v0.4. 2003.

[14] S Chandra W Acosta. Unstructured peer-to-peer networks - next generation of performance and reliability. 2005.

[15] Fatos Xhafa, Leonard Barolli, Enric Jan Villoldo, and Santi Caball. Evaluation of p2p multimedia clustering techniques in jxta-overlay. *Multimedia Syst.*, 15(5):283–293, 2009.