A Novel Self-Configuration Mechanism for Heterogeneous P2P Networks

James Z. Wang Matti A. Vanninen
Department of Computer Science
Clemson University, Box 340974
Clemson, SC 29634-0974, USA
{jzwang, mvannin}@cs.clemson.edu

***Abstract*— In this paper, we propose a simple and efficient mechanism for self-configuring semi-structured overlay network for heterogeneous peer-to-peer systems with no global knowledge. The overlay network is constructed as a hierarchical graph based on peer power which is an aggregate measure of available resources. We use index based, random walk JOIN and LEAVE protocols to build the overlay network in which node degree and workload correspond to its peer power. We also propose an index based query scheme for peers to search documents in the network. Performance studies demonstrate that the proposed self-configuration protocol and search scheme are efficient and viable. The self-configured P2P system has significantly lower query cost than traditional Gnutella-like P2P systems. We further propose a caching assisted query mechanism where query results are cached by intermediate nodes along the search path, thereby exploiting the surplus storage power of the peers and further improving query efficiency. Caching assisted query permits a quicker response to subsequent queries for popular data without incurring the excessive overhead of globally replicated directory information. Our experimental study reveals that the caching assisted query provides an additional reduction of up to 40% in messages needed per query.**

***Index Terms*— P2P, Peer power, Self-configuration, Index based search, Caching assisted query, Overlay networks**

I. Introduction

Efficient self-configuration and query mechanisms are primary issues in heterogeneous peer-to-peer (P2P) systems. Most existing P2P systems rely on a structured network to route the data, generally using a variant of Plaxton's prefix based routing algorithms [1] or distributed hashing [2]. At the other extreme, pure unstructured systems, such as the original Gnutella [3], permit any peers to connect to each other, and randomly establish connections.

Recently, quite a few schemes [6], [7], [8], [9], [10] have been proposed to address the query efficiency issue in P2P system. Structured topologies such as HyperCuP and distributed hashing mechanisms attempt to evenly balance load on all nodes in the network. These approaches seem to be appealing by assuming all nodes are equal in capacity. However, analysis of real P2P networks [4] shows that the capacities of nodes in P2P systems vary widely. Hence weak nodes are bottlenecks in such structured topologies. Furthermore, the relative capacities of nodes may vary as nodes enter and exit the network. An ideal P2P network should be able to dynamically adapt to the changes and adjust the role of existing nodes.

To solve the problem, superpeer approaches in which superpeers act as gateways for weaker nodes in the system have been implemented in latest P2P systems such as Gnutella [11] and FastTrack [12]. These techniques improve the efficiency and scalability of the unstructured networks, but do not completely address the problem of varying node capabilities. The distinction between a normal and superpeer in these systems is discrete and binary, and does not match the actual distribution of node capacities. Another approach to address the network heterogeneity problem is proposed in Gia [13]. Gia is an extension of Gnutella. It incorporates several techniques to compensate for the heterogeneity of the underlying network and the poor scalability of classic Gnutella. Nonetheless, the cost in terms of network resources of this method is unclear and replying on random walks for searches increases latency and reduces hit count and may cause false negative responses.

In this paper, we propose a new protocol for self-configuring and searching P2P networks, based on our observation that it is possible to exploit the natural heterogeneity of P2P networks to optimize searches without requiring a globally consistent view. Our objective is to coerce a hierarchical network structure using simple protocol and heuristics. Degree distribution of the nodes in our proposed network tends to be non-uniform so that a small number of highly connected nodes constitute the "core" of the network and act as hubs through which the weaker nodes of the network are connected. Such a network can be searched more efficiently than a standard unstructured Gnutella network as demonstrated in [5]. We must note here that the proposed network is essentially unstructured, because it does not restrict the location of entering nodes. The workload of message handling on each node in the resulting overlay network is comparative to node power. Nodes join the network by first locating one node in a set of *index nodes*, whose capacity exceed that of non-index peers, and then following a random path away from that node, and finally creating a link with the first available node. Search is also facilitated by the index nodes, which maintain a complete directory view of the system. Queries reach an index node from a source via a path of increasingly powerful nodes, and responses are propagated along a reverse path. To further improve the query efficiency, we propose a *caching assisted query* mechanism in which the intermediate nodes cache query responses, and are hence able to immediately respond to future queries. The *caching assisted query* scheme avoids requiring for all queries to reach all nodes without introducing false negatives.

The rest of the paper is organized as follows: In section 2,

we proposed and discuss the self-configuration protocols and query mechanisms. In section 3, we use simulation to evaluate the feasibility of our proposed protocols, and to demonstrate the performance advantages of using these protocols in P2P networks. We give our concluding remarks and discuss future studies in section 4.

II. Self-configuration Protocols and Query Schemes

Establishing random connections as that in the Gnutella protocol is not an effective way to guarantee low diameter and connectedness in an unstructured P2P network [14]. Our approach is a natural extension of the superpeer mechanism, whereby the nodes with higher capacities serve as redundant indices to store global data. The network is constructed so that it tends to form a hierarchy, with the most powerful nodes near the centroid. This structure permits efficient searches and avoids the hot-spot problem, as the workload carried by a node corresponds to its capacity.

A. Index Nodes

An essential feature of our proposed P2P network is the notion of index nodes. The set of index nodes is $I = \{n|\forall x \in N(n), pp(x) < pp(n)\}$, where $N(n)$ represents the neighborhood of n, and $pp(n)$ stands for peer power of node n. Each index node maintains secondary connections with all other index nodes, which are used for message routing but is not used in the index node selection. This architecture guarantees the graph is connected, since any node either is an index node or can reach an index node via a parent. In this network, broadcasting can be facilitated by sending a message upstream to an index node, disseminating to all index nodes, and then sending downstream to all other nodes.

B. Peer Power

The goal of our self-configuration scheme is to guarantee that the number of messages handled by a node must correspond to its capacity. A reasonable approach is requiring that the degree of a node in the network is comparative to its capacity, and that high capacity nodes exist near the center of the network. To facilitate this approach, we define *peer power* as the metric for measuring node capacity. Peer power is a unitless aggregate measure of the resources in a peer node. It is used to determine the node's capacity to accept links and process messages. Our approach transcends the traditional superpeer concept by categorizing peers with much finer granularity. Peer power is a node's own estimate of its capacity to host connections. Although it is impossible to quantify the relevant performance of computers absolutely, it is possible to define consistent heuristics so that similar machines advertise similar values and the relationship among values corresponds to true performance difference. The metric should reflect three facts:

1) *Diminishing Returns*: The connection capacity increase tends to be sublinear with respect to an increase in available resources, such as bandwidth, cpu power, or memory.
2) *Weakest Component Dominates*: The capacity of a system is limited by the weakest component of the system.
3) *Uptime*: The only available estimate of a node's stability is its uptime. The stability does not directly impact the capacity of the node, but rewarding nodes with a high uptime is effective in practice.

With these considerations, we propose the following formula to compute the peer power of a node n:

$$\begin{aligned} pp(n) \quad = \quad & MIN(\alpha \cdot lg(BW/BW_{min}), \\ & \beta \cdot lg(CPU/CPU_{min}), \\ & \gamma \cdot lg(RAM/RAM_{min})) \\ & + MAX(0, \delta \cdot lg(TIME/TIME_{min})) \end{aligned} \quad (1)$$

This formula captures the available parameters into a single unitless value which can be used to differentiate among the capabilities of nodes on the network. Peer power directly corresponds to the number of connections a node can support. The coefficients $\alpha, \beta, \gamma, \delta$ are used as weighting factors. Their values can be identified through experimental studies.

C. Network Configuration

JOIN and LEAVE are the two fundamental operations in the network self-configuration process. A node uses the JOIN operation to enter the P2P network, and the LEAVE operation to exit. The JOIN protocol uses a random walk originating from one of the index nodes toward the edge of the overlay network to locate a peer for an incoming node. As used before, $N(n)$ represents the neighborhood of n in our algorithms.

JOIN: Node n is joining network G.

Algorithm 1 Link_Creation

n transmit link request to any existing node n'
while n' is not an index node **do**
  n' selects random parent p
  n' passes link request to p
  let $n' = p$
end while
n' selects random index node i
let $n' = i$
while $degree(n') >= capacity(n')$ **do**
  if n' is an edge node **then**
    link creation fails; return
  else
    n' selects random child c
    n' passes link request to c
    let $n' = c$
  end if
end while
n' creates link to n

Algorithm 2 Node_Join

n calls **Link_Creation** MIN times
if $\forall n' \in N(n), capacity(n) > capacity(n')$ **then**
  locate index i using path of random parents from a random node in the network
  if number of indices less than $ln|V_g|$ **then**
    n becomes index
  else
    n creates link to i
    if $degree(i) > capacity(i)$ and i has a child c with two or more parents **then**
      i severs link with c
    end if
  end if
end if

When a new node n wants to join the P2P network, it contacts any existing node $n' \in V_g$. n' transmits a message containing the address of n to a random index node i, which creates a link with n if possible. If i is already at full capacity, the message is transmitted to another index at random, and then downward using the random walk method, and the first node which is able to do so creates a link with n. The links created using this mechanism are as close to the core nodes as possible. This approach can minimize network diameter, and reduce load on the weaker edge nodes. n will request up to MIN connections from the P2P network using the same mechanism, where MIN is the minimum number of neighbors required for a node. If n has no parents after creating MIN links, it must become an index or create a link with one. The naive approach of making an index node out of each node which does not have a parent is infeasible as the number of such nodes grows linearly with the size of the network. Rather, the number of indices is latched to a maximum value, $log|V_g|$. If a new node would cause this limit to be exceeded, the node locates and creates a link with an existing index. Since these additional links may overload the indices, an overloaded index node may drop an existing link to a node that has another parent.

A simple network with two index nodes is shown in Figure 1. The number in a node represents its peer power. The solid lines demonstrate parent/child peer relationships, where the node with a higher peer power is the parent node. The secondary connection between indices is represented by the dashed line.

LEAVE: Node n leaves network G

When a node $n' \in N(n)$ detects failure of node n due to connection timeout, **Node_Exit** is called. The failure of a node is detected by its neighbors as a dropped connection. Upon detecting the failure of n, its strongest non-overloaded child c will drop its existing connections and create connections with all of the neighbors of n, thus preserving the local structure of the network. c in turn must be replaced by one of its children,

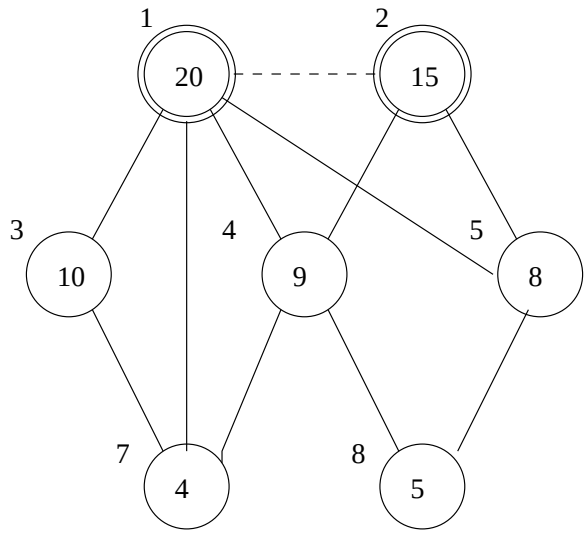

Fig. 1. A basic network.

Algorithm 3 Node_Exit

n' transmits exit notification to an index node via path of random parents. indices transmit cache invalidate messages as necessary.
if $n' \in children(n)$ and $\forall c \in children(n), capacity(n') >= capacity(c)$ **then**
  n' exits network, and creates links with all peers of n
end if
if $degree(n') < MIN$ **then**
  use **Link_Creation** on n' to reassert invariant
end if

and the process is propagated toward the edge of the network. Since c is a child of n, its capacity is likely to be lower than that of n, and so the operation may overload c. If n was an index node, c now becomes an index node and forms secondary connections to all other indices. If n was an edge node and hence has no children, any of its neighbors whose link count falls below MIN must create new links following the JOIN protocol. Thus each node is required to be aware of their neighbors as well as their parents. An announcement of the failure of n must be transmitted to the indices via a random upstream path to invalidate any cached information of its files. If $|N(n')| < MIN$, where n' was a neighbor of n, n' follows the JOIN protocol to establish new links

The topology of the network in Figure 1 following the deletion of node 4 is demonstrated in Figure 2. The stronger child, node 8, has assumed the links formerly held by node 4.

D. Caching Assisted Query

A primary objective of the P2P network is to efficiently locate files with reliable queries. In practice this means any node must be able to obtain an accurate directory listing and file metadata with minimal latency. Piecing together a view of the namespace by querying every node on the network is

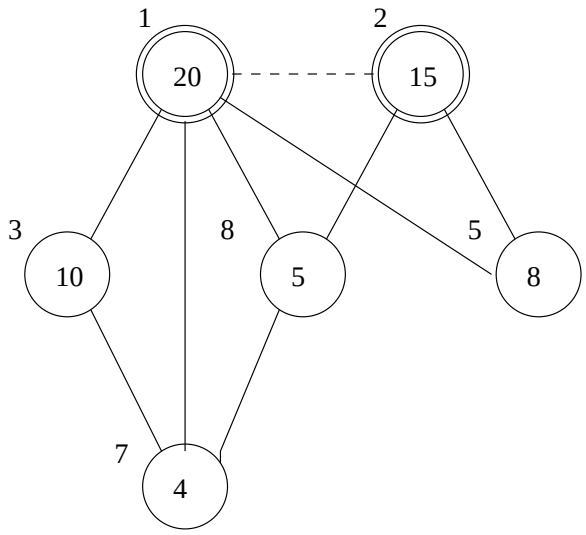

Fig. 2. Network following removal of node 4.

clearly infeasible; a complete view must be obtainable from some well known sources. A simple approach is to rely on the indices to respond to every directory query. To achieve query efficiency, the index nodes maintain a complete view of the file system. Most events in file sharing P2P systems are reads, and directory queries outweigh updates, and so caching directory data is advisable. Taking advantage of the hierarchical nature of the network, each node maintains a view representing the union of its descendants. A query is propagated upstream until an authoritative response can be made at an index node. The response is unicast along the reverse path back to the initial query node, and each intermediate node caches the file information. Future queries for the same data in the same region of the overlay network may henceforth be serviced without accessing the indices. This caching assisted query approach exploits the storage power of nodes to enhance network functionality.

The information cached in the intermediate nodes regarding a file must contain at least four fields as shown in Figure 3. The file name is a key by which queries are matched, and the owner identifies which node holds the requested file. A source field indicates which parent this cached information was obtained from, and a list of interested children indicates which children have requested information regarding the file. The latter two fields are necessary to ensure cached data can be effectively invalidated when the file is removed or a link is severed, as described in the following paragraph. We should note that the cache line is very small and will not take much disk space from the peer nodes. A comparable approach is used by web browsers, which rely on cookies to assist queries or session maintenance.

When a file is deleted, or the node that holds the file fails, the cached information regarding that file must be purged from the network. Since only part of the network is likely to have cached data regarding the file, broadcasting a global

filename	owner	source	interested children

Fig. 3. Cache line.

cache invalidate message is unnecessarily expensive. Hence it is necessary that each node retains a list of children who have requested information on a given file, and forward invalidate messages to those children only. Because links may be severed to avoid overloading indices and nodes may fail, the partial-broadcast invalidating messages may be prevented from reaching their destination. To avoid leaving stale cache lines in the network, each cache line must contain the source node from which the information was acquired, and in case the link is severed, that cache line can still be invalidated. The caching assisted query and invalidating mechanism work as follows:

SEARCH: Node n tries to locate file F.

Algorithm 4 Search

while n is not an index and n has no cache line for F **do**
  n selects random parent p
  let $n = p$
end while
if n has cache line for F **then**
  transmit affirmative query response along reverse path to source; create a cache line for F in each intermediate node.
else
  transmit negative query response along reverse path to source.
end if

when a query for file F is issued in node n, it passes the query via a random upstream path. Any ancestor which has a cache line for F responds with a unicast message, and appends the child from which the query was received. All nodes on the reverse path cache the response to benefit future queries. When creating a cache line, the node informs the parent from which the response was received. If the link to that parent is ever severed, or an invalidate message for the cache line is received, the node forwards an invalidate message to any children which have requested the cache line.

A network with two files, f2 at node 3 and f1 at node 8, is illustrated in Figure 4. The indices and the nodes which own the files maintain cache line at all times. The information cached by intermediate nodes following a query for f1 by Node 7 are shown in Figure 5. The message is transmitted along a random upstream path, via node 4, to an index node which provides a response along the reverse path. Nodes 7 and 4 create cache lines indicating node 8 is the owner of f1. Node 1 adds node 4 and node 4 adds node 7 as interested children.

Using the self-configuration protocols and search mechanisms, the semi-structured P2P network distributes the work-

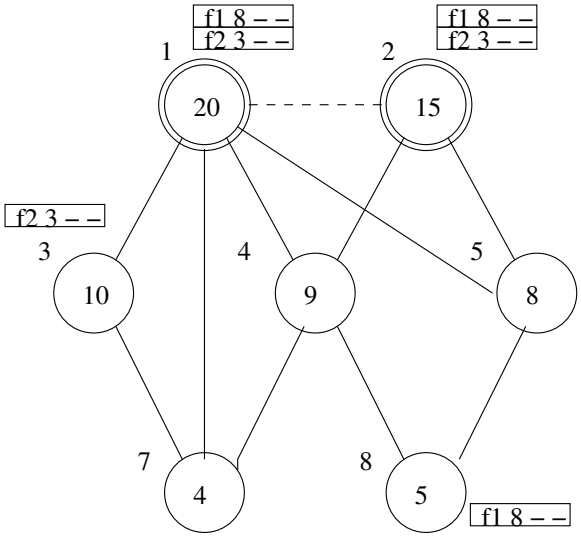

Fig. 4. Basic network with two files.

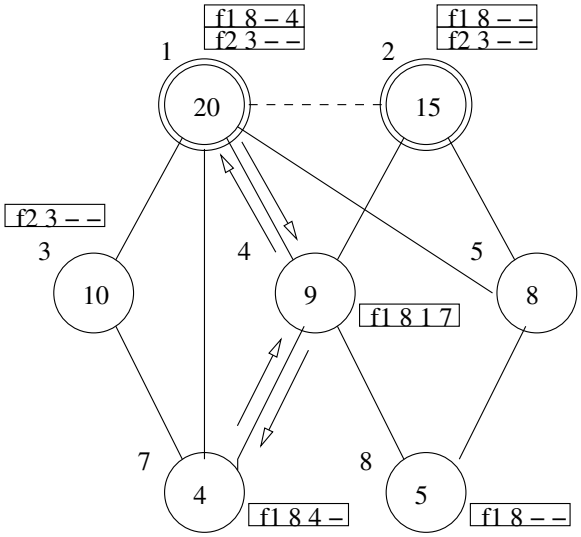

Fig. 5. Network following a query for f1.

load comparative to node capacity distribution. Since read operations are expected to outweigh updates, the caching assisted querying reduces load on the indices. Unlike the approach of globally replicating directory information, in our method only those nodes interested in a file cache its information. If a file is unpopular, its data will not be frequently cached, reducing the overhead of unnecessary cache invalidate messages.

III. Performance Study

To verify the feasibility of our protocols and algorithms, we use a discrete event simulation to evaluate our proposed P2P network. The effectiveness of such a P2P network can be measured in several different ways. In terms of the JOIN and LEAVE operations, we are most interested in confirming that the diameter of the network remains low, and that strong nodes migrate toward the graph centroid as anticipated. The second goal is to empirically examine that the workload in a node, as measured in messages processed, corresponds to its peer power. Third, we want to study the network performance enhancement achieved by our self-configuration protocol. Finally we evaluate the improvement of query efficiency contributed by caching.

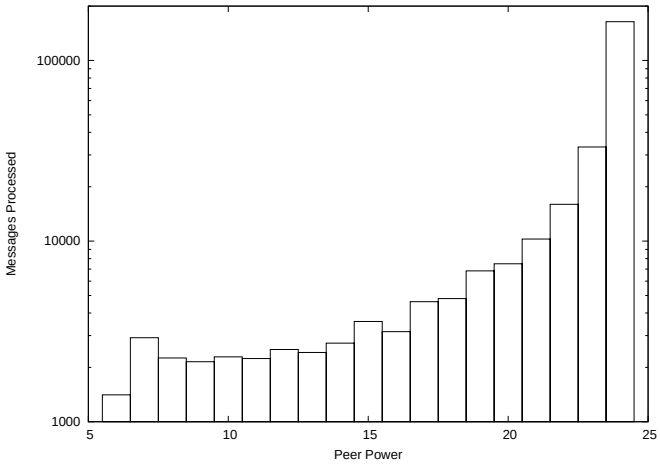

Fig. 6. Messages processed vs. peer power.

The quality of the self-configured network was measured by observing the eccentricity of the indices, as it is an indicator for expected search latency and a reasonable estimate of the graph diameter. We also measure the number of messages processed by each node due to node creation, node exit, or search event, and compute the number of queries and query responses with caching enabled or disabled.

A. Simulation Model

We assume an evolving network similar to that in [6]. The arrival of nodes follows the distribution $p(arrival < t) = \lambda_1 e^{-\lambda_1 t}$ and the lifetime of nodes follows distribution $p(lifetime < t) = \lambda_2 e^{-\lambda_2 t}$, where the number of nodes $N = \frac{\lambda_2}{\lambda_1}$. This memoryless distribution is a reasonable model for a P2P network since nodes presumably are acting independently. It is sufficient to describe the behavior of the network in an abstract sense although this may not actually be the case since, for instance, the computers in a corporate network may be rebooted on a fixed schedule or may be influenced by other global events.

We also assume that the peer power of each node, independent of the arrival time and duration, is uniformly distributed with a mean μ. A uniform distribution assumes that newer nodes are not more powerful simply by virtue of being newer. The peer power of a node is assigned when the node is created, and remains static through its lifetime. We further assume the minimum degree of nodes is MIN. The parameters used in the measurements were $\mu = 15$ and $MIN = 6$. All results are averaged over five trials.

1) Relationship of peer power to node workload: The ability of the networks to distributed workload in proportion to

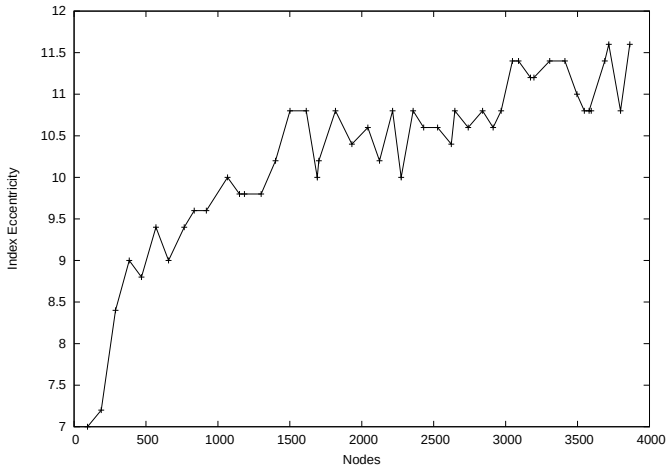

Fig. 7. Index eccentricity vs. number of nodes.

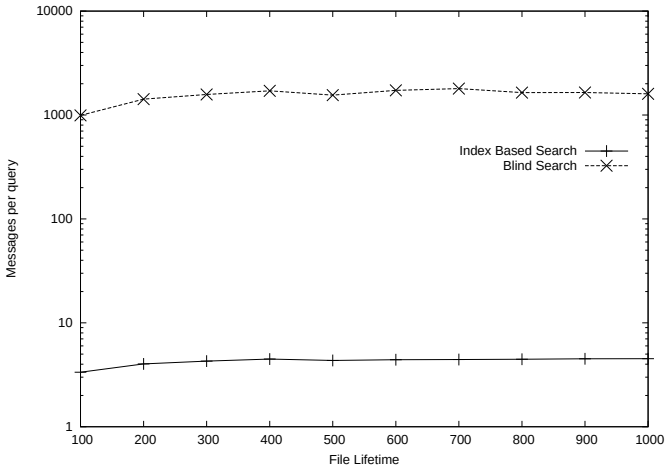

Fig. 9. Index based vs. Gnutella-like queries.

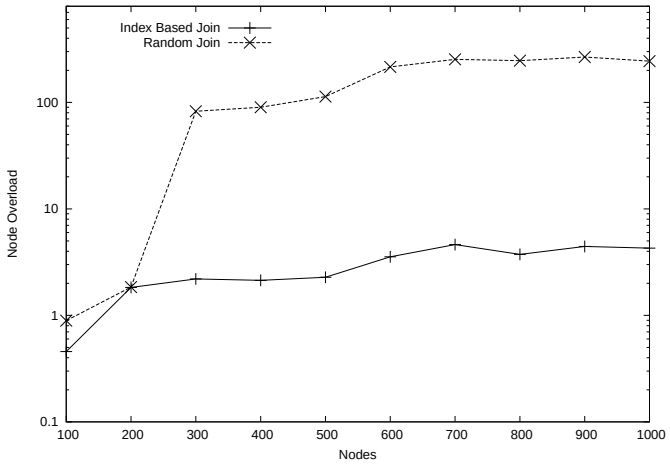

Fig. 8. Node overload.

peer power was studied by measuring the number of messages processed as nodes enter and exit the network. The simulation was run for 1000000 ticks with an expected number of nodes $N = \frac{\lambda_2}{\lambda_1} = 5000$. During each tick, nodes may arrive with a probability based on the arrival distribution, or depart if their time to live reaches zero. The relationship between the peer power and the number of messages processed is illustrated in Figure 6. Each bar of the chart shows the total number of messages received by all nodes with a particular peer power. Higher workloads in the network are located at those nodes which are most able to handle the load, indicating the network topology matches the variance in capacity of its nodes.

2) Effectiveness of index based join over random join protocol: Message handling latency and the number of involving nodes are dependent primarily on the distance of nodes from the indices. The effectiveness of our index based random walk JOIN/LEAVE protocols described in section II-C was evaluated by comparing to a Gnutella like approach whereby links are simply created to random existing nodes. As discussed in section II-C, the number of index nodes is bounded by $ln|V_g|$, and index nodes accepted overload connections if an excessive number of locally maximal nodes were created. As Figure 7 shows, the eccentricity of the index nodes is logarithmic with respect to network size when the index based JOIN and LEAVE protocols are used. Overload forces a workload on nodes whose capacity is exceeded and reduces performance. So we must minimize the degree of overloading a node.

Figure 8 illustrates the highest degree to which a node in the network was overloaded after 500000 ticks with $N = \frac{\lambda_2}{\lambda_1}$= 100 to 1000. When using our JOIN and LEAVE protocols, no node was forced to accept more than five links in excess of its capacity. On the other hand, using a policy of creating links with random peers forced in excess of a hundred overload links to some nodes. The simulation results show that our index based JOIN and LEAVE protocols are effective and necessary in order to minimize node overload.

3) Effectiveness of index based queries: Gnutella-like unstructured networks suffer from costly searches because queries are untargeted. If the possibility of false negatives is unacceptable, search distance cannot be limited by a maximum hop count or time to live of the query message. Rather, all nodes on the network are required to receive every query. Since nodes in an unstructured P2P network are not aware of network topology, they must forward all messages to all neighbors. As a result, every node receives $degree(n) - 1$ copies of query messages for every query, so search cost is linear in terms of network size, with a high coefficient. On the other hand, relying on a set of index nodes to respond to queries reduces query cost to the number of hops needed to reach an index node from a source node. To illustrate the advantages of our proposed self-configuration protocols and our index based searching mechanism, we perform a series of simulated searches in our self-configured P2P network. We also run the same searches on a Gnutella-like randomly built network with blind breadth first searches. A network with $N = 500$ nodes was generated, and "files" were inserted at

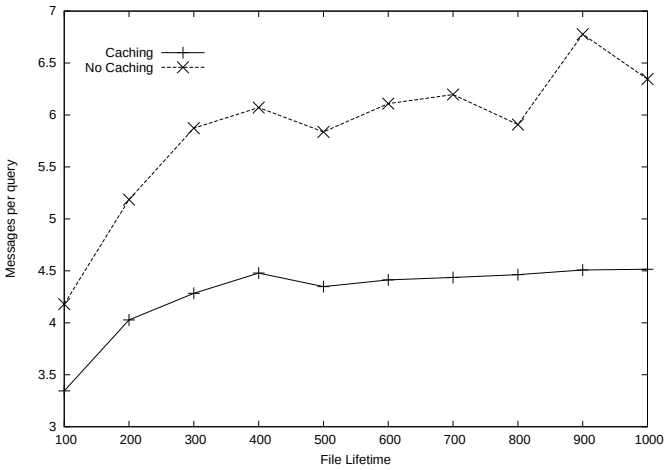

Fig. 10. Cache assisted queries.

random nodes at random intervals with a mean of 1000 ticks. A query for a random file is issued from a random node at each tick. Figure 9 shows the total number of query, response, and cache invalidating messages handled per query through 100000 ticks. The lifetimes for a file ranged from 100 to 1000 ticks following creation. When lifetime of a file expires, it was purged and an invalidate message was passed to any interested nodes. As demonstrated by Figure 9, the average number of messages needed by the classic Gnutella-like approach for each query is much greater than that using our index based approach.

4) Enhancing searches with caching assisted querying: Caching query results by intermediate nodes can further reduce the cost of queries, particularly when files are accessed frequently before they are destroyed, because the overhead of invalidate messages is relatively low. To investigate the benefit of the caching assisted query, we repeat the search simulation with caching enabled and disabled. The efficiency of the query mechanism was measured by the number of messages needed per query as depicted in Figure 10. As expected, disabling the caching of queries by intermediate nodes increased the message needed per query by 25-40%. Despite needing additional invalidate messages when caching by intermediate nodes is enabled, the number of messages required per query is substantially lower with the caching assisted query mechanism.

IV. Concluding Remarks and Future Work

In this paper, we propose a simple and efficient mechanism for self-configuring semi-structured overlay network for heterogeneous peer-to-peer systems with no global knowledge. The overlay network is constructed as a hierarchical graph based on peer power which is an aggregate measure of available resources. We use index based, random walk JOIN and LEAVE protocols to build the overlay network where node degree and workload correspond to its peer power. We also propose an index based query scheme for peers to search documents in the network. Performance studies demonstrate that the proposed self-configuration protocol and search scheme are efficient and viable. The self-configured P2P system has significantly lower query cost than traditional Gnutella-like P2P systems. Furthermore, we propose a caching assisted query mechanism where query results are cached by intermediate nodes along the search path, thereby exploiting the surplus storage power of the peers and further improving query efficiency. Caching assisted query permits a quicker response to subsequent queries for popular data without incurring the excessive overhead of globally replicated directory information. Our simulation results show that the caching assisted query provides an additional reduction of up to 40% in messages needed per query.

Currently we are studying the viability of our proposed measure of peer power by measuring the actual performance of systems in comparison to their predicted capacity based on the peer power estimate. We will also investigate extending the caching assisted query to consider file replicas, and including file metadata in the cache line to avoid unnecessary contact to the file owner. Finally, recognizing that some files are required by virtually all nodes but rarely changed (for instance, a directory listing for /), a global replication mechanism, which distributes updates of such files to all nodes in the network, will be considered.

References

[1] C. Plaxton, R. Rajaram, and A. Richa. Accessing nearby copies of replicated objects in a distributed environment. In Proceedings of the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures, pages 311–320, June 1997.

[2] J. Cates Efficient Data Management for a Distributed Hash Table. Masters Thesis, MIT, 2003.

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

[4] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts. Multimedia Systems, volume 9, issue 2, pages 170–184, August, 2003.

[5] Lada Adamic, et. al., Search in Power-law Networks. Physical Review E64, 2001.

[6] Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Building low-diameter p2p networks. In STOC 2001, Crete, Greece, pages 492–499 2001.

[7] Mario Schlosser, et. al., HyperCuP - Hypercubes, Ontologies and Efficient Search on P2P Networks. First Workshop on Agents and P2P Computing, Bologna IT, July 2002.

[8] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In Proc. ACM SIGCOMM 2001, pages 161–172, August 2001

[9] Ben Zhao, John Kubiatowicz, and Anthony Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, CS Division, U. C. Berkeley, April 2001.

[10] Ion Stoica, et. al., Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, Technical Report TR-819, MIT, March 2001.

[11] A Singla and C. Rohrs. Ultrapeers: Another Step Towards Gnutella Scalability. Lime Wire LLC, December 2001. http://rfc-gnutella.sourceforge.net/Proposals/Ultrapeer/Ultrapeers.htm

[12] The FastTrack Protocol. 2003. http://developer.berlios.de/projects/gift-fasttrack

[13] Yatin Chawathe, et. al., Making Gnutella-like systems Scalable. In proc ACM SIGCOMM 2003, pages 407–418 August 2003.

[14] Gopal Pandurangan, et. al., Building P2P networks with good topological properties, 2002.