Self-Configuration Protocols for P2P Networks

James Z. Wang Matti Vanninen
Department of Computer Science
Clemson University, Box 340974
Clemson, SC 29634-0974, USA
+1-864-656-7678
{jzwang, mvann}@cs.clemson.edu

Abstract

This paper proposes and evaluates different protocols for individual peers to self-configure the P2P overlay networks in various network environments. For small-scale homogeneous networks, an optimizer peer selection protocol is proposed and found to be superior to other peer selection protocols, including random, egoist, altruist and greedy protocols, in terms of network diameter and node eccentricity. For large-scale heterogeneous P2P systems, peer-power based peer selection protocols are proposed to self-configure semi-structured overlay networks for heterogeneous P2P systems. An index-based query mechanism is also proposed for peers to search documents in the P2P network. Performance studies demonstrate that the proposed self-configuration protocols and search scheme are efficient and viable, and the self-configured P2P network has a significantly lower query cost than traditional Gnutella-like P2P systems. In addition, a caching-assisted query mechanism is proposed to exploit the surplus storage power of the peers, improving further the 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.

Keywords: P2P, Self-Configuration, Peer-power, Index-based query, Cache-assisted query

1. Introduction

Recently, P2P systems and applications have become wide-spread due to the success of P2P file sharing systems [1, 2, 3]. The two approaches for implementing such systems are the centralized directory approach and the distributed P2P network approach. The centralized directory solutions, such as Napster [3], not only create hot spots on directory servers but also raise legal issues due to their centralized architectures. To address these problems, many P2P applications use decentralized approaches in which there is no centralized directory for searching the network. These decentralized P2P networks can be further categorized into two groups, structured and unstructured, based on their network topologies and content search methods [4]. The structured networks, such as Chord [5], CAN [6], Pastry [7] and Tapestry [8], rely on structured topologies to route the data, generally using a variant of Plaxton's [9] prefix-based routing algorithms or distributed hashing [10] schemes. A second structured P2P network approach, HyperCup [11], requires specific connections between peers to maintain the HyperCube topology while query messages route only along the edges of the Hypercube. However, mandating a specific network topology or assuming total control over the location of data is not desirable in P2P societies. Furthermore, partial matching queries are not efficiently supported in these approaches.

Unstructured P2P networks can be categorized into two classes based on their network scales and topologies. One class is the small-scale homogenous P2P network in which all peers have similar computing powers and network bandwidth residing within a corporate network. Such networks have existed for quite some time in the form of work groups created with operating systems from Microsoft and Apple. In a corporate network, a P2P network is usually used to share the storage capacities or computing powers of individual workstations. As a result, false negative responses significantly affect the usability and reliability of these small-scale P2P systems, meaning flooding-based search becomes a necessity in such P2P systems without index servers. A second category of P2P networks is the large-scale heterogeneous network in which peers have different computing powers, storage capacities and network bandwidths. In such P2P networks, it is possible to take advantage of the network heterogeneity to improve the search efficiency.

When a network node joins an unstructured P2P network, it virtually connects to a subset of peers in the network to form an overlay network. Data search in such

an unstructured P2P network is essentially a peer probing process, with the query messages among the peers being propagated through this overlay network. As a result, this blind flooding search, at times, generates excessive query messages in the P2P network and may take a long time to find the target data if the diameter of the overlay network is large. Although the random-walk approach [12] reduces the number of query messages by having the query forwarded only to a randomly chosen neighbor instead of all of them, this approach often generates false negative responses.

Since in unstructured P2P networks the efficiency of data search relies on the quality of the overlay network, it is necessary to study the self-configuration protocols for P2P networks so that efficient overlay networks can be formed when individual peers enter and exit the networks. In this paper, we investigate the intrinsic properties of different P2P networks, proposing various protocols for individual peers to self-configure the overlay networks in various P2P environments.

The rest of the paper is organized as follows. Section 2 proposes different P2P self-configuration protocols for small-scale P2P networks, discussing the advantages and disadvantages of each, with Section 3 evaluating these protocols using simulation. Section 4 addresses the self-configuration protocols and query mechanisms for large-scale heterogeneous P2P networks, while Section 5 evaluates their feasibility and performance also through simulation. Finally, concluding remarks and future work are discussed in Section 6.

2. Self-configuration protocols for small-scale P2P networks

In an unstructured P2P network, peers are restricted only to exchanging messages with their neighbors in the P2P overlay network. Normally, the number of virtual connections to a node should be limited so that its corresponding message routing workload is limited. However, each node has to maintain at least a certain number of connections so that the overlay network remains connected even if a few nodes quit the P2P network. The diameter of the overlay network, i.e., the longest shortest network distance path between any pair of nodes, affects the efficiency of the flooding-based P2P search since disseminating the query message to every node in the P2P network is mandatory to avoid false negative responses. This work proposes and evaluates five peer selection protocols used by individual peers to self-configure the P2P overlay network. The goal here is to find the best protocol leading to a small network diameter and efficient P2P search.

Building a P2P overlay network must take into account two constraints. First, the network bandwidth of a P2P node constrains the number of connections it can form with other peers in the P2P network. Too many connections to one node may result in overloading that node during query message routing. Second, a good P2P overlay network must tolerate the simultaneous failure or disconnection of several nodes, even in a small-scale corporate network environment. Thus, a P2P node must maintain at least a certain number of links with its peers. The objective of this study is to build a good P2P overlay network given these constraints.

2.1 P2P self-configuration procedures

The P2P overlay network is self-configured by nodes forming virtual connections with existing peers as they join the P2P network. During their lifetime in the P2P network, nodes exchange query messages directly with their connected neighbors or probe their neighbors to maintain a stable overlay network. Given the minimum number of connections (M) and the maximum number of connections (N) that a node can form, each P2P node must connect to at least M but not more than N neighbors to ensure a stable P2P overlay network. A node in this network may be in one of the three states: JOIN, UNSTABLE, or STABLE.

Initially a node enters the P2P network the JOIN state, calling the procedure **node_join** depicted in Figure 1 to gather information about the current state of the P2P overlay network.

```
Procedure node_join
begin
   send INIT_REQ message to any existing node;
   wait for INIT_ACK message;
   if no response received, return failure; end if
   if INIT_ACK message is received
      send PING to each node in the network;
      wait for responses until 2*T time passes;
      record the arrival times of all PONGs;
      set state UNSTABLE;
   end if
end
```

Figure 1: Procedure node_join.

In this procedure, it is assumed that each node in the P2P overlay network maintains a view of the network, even though it may not be an up-to-date view. An INIT_REQ message is used by the joining node to query any node in the P2P network initially. An INIT_ACK message containing the responding node's view of the network and the overlay network parameters (M, N, T) will then be sent to the joining node, where M and N are the minimum and maximum number of connections for a node respectively, and T is a timeout value which is usually the longest round-trip physical network latency between any pair of nodes in the P2P network. It is not necessary that the responding node have an up-to-date global view of the P2P overlay

network, although the views in every nodes are constantly updated through message exchange. Because of the dynamic nature of the P2P network, temporarily inconsistent views of the overlay network among peers are acceptable and do not have a significant impact on the joining node's peer selection since such inconsistency is usually trivial.

After receiving the INIT_ACK message, the joining node pings all other nodes in the network to collect the physical network latencies. It then enters the UNSTABLE state in which it will use the procedure **node_connect** in Figure 2 to select at least M peers as its neighbors in the P2P overlay network.

```
Procedure node_connect
begin
  while # of links to this node is less than M
    select the best suitable node in the network;
    send a CONN_REQ message to selected node;
    wait for response until 2*T time passes;
    if CONN_ACK message is received
      establish virtual connection;
    end if
    if CONN_REJ message is received or timeout
      mark the responding node as NO_CONN;
    end if
  end while
  broadcast UPDATE message to neighbors;
  set state STABLE;
end
```

Figure 2: Procedure node_connect.

In this procedure, an UNSTABLE node tries to establish at least M virtual links with other peers in the P2P network by selecting the best suitable nodes not marked as NO_CONN based on the following information:

- The distance in hops to these nodes
- The latency to the nodes through message routing in the P2P overlay network (logical distance)
- The latency to the nodes through the underlying physical network (physical distance)
- The number of connections to the nodes
- The capacity of the nodes for further connections

When a node is selected, a CONN_REQ message will be sent to it requesting the establishment of a virtual link. A CONN_ACK will be sent back to the requesting peer if the selected peer can accept more connections. The payload of the CONN_ACK message also contains the latest view of the overlay network of the selected node. If this peer cannot accept new connection, it will respond the CONN_REQ message with a CONN_REJ message also containing the latest view of overlay network by the selected node. If a node does not respond, it may have already quit the P2P network. A CONN_REJ response usually means that a node has reached its connection capacity. The joining node will mark these nodes as NO_CONN so that no further link attempt will be made to these nodes. Once a node establishes enough links in the P2P overlay network, it broadcasts an UPDATE message to its neighbors and enters the STABLE state. This UPDATE message, which contains information about any nodes or connections which the source node knows having been added or removed from the P2P overlay network, will be disseminated across all nodes in the P2P network through message routing.

When a node is in the STABLE state, it constantly communicates with its neighbors so that the overlay network is always maintained properly. Usually neighbors are aware of one another through normal query message exchanges. However, if a node does not participate in message routing with its neighbors, it should periodically send KEEPALIVE messages to its neighbors. If a node has not received any message from one of its neighbors during time d*T (d > 2), it can assume that neighbor is disconnected. The node then should remove this disconnected neighbor from its virtual link list. If the number of its virtual links is less than M, the node enters the UNSTABLE state in which the procedure **node_connect** is used to reconfigure the overlay network.

2.2 Peer selection protocols

The peer selection protocol determines the topology of the P2P overlay network. To build a small diameter overlay network so that query messages can reach all peers quickly, the following peer selection protocols are studied in this paper:

- ***Random:*** The simplest peer selection protocol is random selection. This protocol was used by the original Gnutella P2P network [2]. If the number of virtual connections to a peer is less than M, it randomly sends a CONN_REQ message to a peer in the P2P overlay network. However, this simple approach may not build a small diameter overlay network.
- ***Egoist:*** This peer selection approach first finds the set of nodes with the largest number of connections, and then it sends a CONN_REQ message to one of the nodes with the highest capacity for further connections. The rational behind this approach is that nodes with a higher number of connections are normally at the centroid of the overlay network. Connecting nodes to the centroid of the overlay network may result in a small diameter network as measured by hops. However, the quality of the overlay network is ultimately evaluated by the latency of the message routing. This approach ignores the network latencies between the peers, and, hence, may not build the best overlay network in terms of network latencies.

Furthermore, it may cause workload imbalance among the peers during message routing.

- ***Altruist:*** This approach selects the nodes with the lowest number of virtual connections to try to balance the number of virtual links among the peers. By doing so, it hopes to bring the nodes on the edge of the overlay network into its centroid, there by balancing the message routing workload among the peers. An additional advantage of this approach is that the joining node can quickly establish the necessary virtual connections because it has fewer chances of receiving a CONN_REJ message. However it may not result in a small diameter network in terms of network latency. Furthermore, due to the dynamic nature of the P2P network, the entire P2P overlay network may be easily partitioned into disjoint subnets with this approach.
- ***Greedy:*** The greedy approach always tries to connect to nodes resulting in the lowest physical network latencies. However, the local optimum may not always result in the global optimum in terms of message routing latency. Based on current network technologies, the latency among nodes in the same segment of the network is normally lower than the inter-segment latency. Thus, the greedy protocol tends to create very few links between segments, potentially leading to network splits and high variance in transmission time through the P2P overlay network.
- ***Optimizer:*** This approach considers not only the physical network latencies but also the logical network latency between peers. Given two nodes *i* and *j*, the physical network latency from node *i* to node *j* (LP_{ij}) is the latency time for node *i* to send a ping message directly to node *j* through the underlying physical network; the logical network latency from node *i* to node *j* (LL_{ij}) is the latency time for a message to travel from node *i* to node *j* by being routed through the P2P overlay network. The optimizer approach tries to minimize the average difference between the logical network latency and the physical network latency for all pairs of nodes in the P2P network. To reach this goal, an intuitive solution suggests that a node has to analyze the entire overlay network in order to select its ideal neighbors, an expensive process in terms of communication and computation costs. In addition, since the P2P overlay network is dynamically reconfigured as nodes join and quit, it is difficult to obtain an up-to-date snapshot of the consistent view of the overlay network. For any joining node, peer selection is based only on the simple heuristic information collected by it through querying the other peers. When a new node joins the P2P network, the physical latencies between peers can be sent to this node in an INIT_ACK message if all peers are required to keep latency records. The latency information is updated through JOIN_ACK and UPDATE messages. In the optimizer approach, a node takes advantage of this available latency information to select its neighbors. Given three nodes *i, j* and *k* in the network, $O_{ijk} = \left| LP_{ij} + LP_{jk} - LP_{ik} \right|$, where LP_{ij}, LP_{ik}, and LP_{jk} are physical network latencies between them. It is reasonable that $LP_{ij} + LP_{jk}$ is viewed as the logical network distance between nodes *i* and *k* if a P2P overlay network *i-j-k* is built (i.e., *j* is the node that connects both *i* and *k*). Thus, O_{ijk} can be viewed as the latency overhead of routing messages using the overlay network *i-j-k*. If the cost of any node *i* connecting to a node *j* is defined as $C_{ij} = \sum O_{ijk}$, where *k* is any node in the network other than *i* and *j*, then this optimizer protocol uses connection cost C_{ij} as the metric for peer selection, i.e., a node selects the peers that result in the least connection cost as its neighbors.

The optimizer peer selection protocol should result in a favorable P2P overlay network in terms of reducing the overall message routing latencies. For any two nodes *i* and *j* in a P2P overlay network *N*, it is assumed that $O_{ij} = \left| LL_{ij} - LP_{ij} \right|$ is the latency overhead of routing messages between nodes *i* and *j*, where LL_{ij} and LP_{ij} are the logical network latency and the physical network latency respectively. If the overall latency overhead of using the overlay network N is defined as $O_N = \sum_{i,j \in N} O_{ij}$, it is not hard to prove that the optimizer protocol results in minimal O_N.

Since the optimizer approach considers both the physical and logical network latencies of this node to all other nodes, it is more expensive than the greedy protocol in terms of communication and computation costs. The communication cost is due to disseminating the physical network latencies between any pair of nodes to all nodes in the network. However, in a small-scale corporate network environment, peers enter and quit the P2P network more infrequently than in large-scale internet environments. Since the majority of the information is sent to a node in the INIT_ACK message as the node first joins the P2P network, the JOIN_ACK and UPDATE messages have much smaller payloads. As node joining is relatively infrequent, it is worth using the optimizer approach if the margin of reducing the overlay network diameter is very large.

2.3 Ping sampling

In both the greedy and optimizer approaches, a node pings other nodes in the network to obtain the physical network latencies. However, since this type of network latency testing is error prone due to transient network events, a series of probes may be necessary to find the

network latency with a degree of certainty. This peer probing in the network, which is quite noisy, is likely to affect the network adversely during periods of flux. As a result, an alternative method for determining physical network latencies is randomly probing a limited number of sample nodes and then extrapolating the probable physical network latencies to other nodes based on these sampled data. If the physical network latency between node *i* and node *j* is known to be LP_{ij}, and the physical network latency between node *j* and node *k* is known to be LP_{jk}, it is highly possible that the physical network latency between node *i* and node *k* is between $\left|LP_{ij} - LP_{jk}\right|$ and $\left|LP_{ij} + LP_{jk}\right|$.

2.4 Message format

Three classes of messages used in maintaining a P2P overlay network are KEEPALIVE and PING messages, which are point-to-point message used for handshaking or physical network latency probing, broadcast messages (query messages and UPDATE messages), which are forwarded by a node to all neighbors except the incoming node, and responding messages, which are only forwarded along the query path. A common message format is depicted in Table 1.

Table 1: Message format

ID	Unique ID of the message
Source	IP address of the source node
Destination	IP address of the destination node
Type	Type of the message
Payload	Payload of the message (optional)

As shown in this table, all messages have a unique ID so that duplicates can be identified and discarded at each node. Peers then process the messages based on type, which also determines the format of the payload. The source and destination IP addresses in the message are used to facilitate the message responses.

3. Evaluating peer selection protocols for small-scale P2P networks

The objective of this performance study is to evaluate the quality of P2P overlay networks configured using different peer selection protocols. Although using the optimizer neighbor selection protocol is expected to create the best P2P overlay network, it is important to evaluate quantitatively how good the overlay networks configured using different neighbor selection protocols can handle the message routing. Since overlay networks are used to facilitate searches for P2P networks, a low latency of query responses is desirable. The quality of these networks can be evaluated using several metrics. One such metric is the diameter of the P2P overlay network, which is measured as the maximum latency time of message routing between any two nodes through the shortest distance path. The second is the network eccentricity, which is measured as the maximum logical network distance to the centroid of the overlay network from any node in the network. Ideally, the network diameter should be small so that any query message can reach all nodes quickly. Second, the average eccentricity of the nodes should be low for individual peers so that their queries can be responded promptly. Because these two metrics are easier to measure than latency itself, they are used to compare the P2P overlay network quality.

3.1 Simulation model

Specifically, a discrete event simulation will be used to evaluate the P2P overlay networks self-configured using different peer selection protocols. Using an evolving P2P network similar to that in [13], with nodes joining and exiting the network at the exponentially distributed intervals, the designed simulation will model the node join and exit activities in the P2P network as a standard M/M/∞ queue, where an arrival in this queue represents a node joining the P2P network, the node lifetime is presented as the time spent in the queue, and the number of nodes in P2P network is assumed to be the queue length. The arrival of nodes follows the distribution $p(arrival < t) = 1 - e^{-\frac{t}{\lambda_1}}$, and the lifetime of nodes has distribution $p(lifetime < t) = 1 - e^{-\frac{t}{\lambda_2}}$, where λ_1 and λ_2 are the mean arrival time and the lifetime of the P2P nodes. This memoryless distribution is a reasonable model for a P2P network since nodes presumably are acting independently. Based on Little's Law, the expected number of nodes in the P2P network is determined by $N = \frac{\lambda_2}{\lambda_1}$. This model is sufficient to describe the general behavior of the P2P network in an abstract sense although it may not completely accurate (e.g., computers in a corporate network may be rebooted on a fixed schedule or may be influenced by other global events).

In addition, it is assumed that the minimum number of connections for a node is 3, and the maximum number is 7. The physical network latency between a pair of nodes within a segment follows an exponential distribution having a mean value of 3,000 ns, while the physical network latency between a pair of nodes across different segments follows an exponential distribution with the mean value of 30,000 ns. These latency values were obtained through an experimental study by pinging network nodes in the Clemson University campus network. The entire network is assumed to be divided into 5 segments. The simulation was run for 1,000,000 ticks with an expected number of nodes $N = \frac{\lambda_2}{\lambda_1}$. During each tick, nodes arrive based on the arrival distribution, or depart if their time-to-live reaches zero. Without loss of generality, the mean interval of node

arrivals is assumed to be 100 ticks. The expected number of nodes ranges from 100 to 3500, typical sizes for cooperate networks. With the simulation running for 1,000,000 ticks, the P2P network is guaranteed to reach the expected number of nodes and maintain the number for considerably long period of time for the experimental study. The simulation was repeated for five times, and the network diameter and average node eccentricity were measured under various numbers of network nodes each time. The average results of the network diameter and node eccentricity for five runs are reported here.

3.2 Network diameter

In this simulation, qualities of overlay networks configured using different neighbor selection protocols in terms of network diameter are evaluated under various numbers of network nodes, with results being depicted in Figure 3. As expected, the overlay network built by using the random protocol has the largest network diameter. Both the egoist and greedy approaches outperform the random approach by large margins. The overlay network built using greedy peer selection protocol has a smaller network diameter than that configured using the egoist peer selection protocol, when the number of network nodes exceeds 1000. Conversely, when the number of network nodes is very small, the egoist protocol outperforms the greedy scheme slightly in terms of the network diameter, because the overlay network built by using egoist approach is very small in terms of hops. However, the greedy protocol does not guarantee a small diameter overlay network in terms of network hops, even though it tries to achieve the local optimum in physical network latency. When the number of network nodes is very small, the benefit of selecting the shortest physical latency link is not visible since only a few hops are needed for messages to reach all the nodes in the network. Thus, the diameter of the overlay network built using the greedy peer selection protocol is larger in terms of logical network latency since more hops might be needed for routing messages. As the number of network nodes increases, the diameter of the overlay network in terms of network hops increases because of the constraints on the number of links allowed for each node. The benefit of selecting shortest physical network latency link compensates for the weakness of the greedy protocol. Thus the greedy protocol outperforms the egoist approach as the number of network nodes increases. The results for the altruist approach are not reported because the altruist peer selection protocol can not consistently build a connected overlay network. The network diameter for the split overlay network is infinity.

As expected, the optimizer approach achieves the best performance in terms of network diameter. The diameter of the P2P overlay network configured using the optimizer peer selection protocol is less than half of that configured using the egoist or the greedy protocols. Even if the optimizer approach generates higher network traffic because more information is disseminated through the peers, the much smaller network diameter not only results in faster query responses but also reduces network traffic. In addition, the optimizer protocol creates virtual links along the shortest physical distance between a pair of nodes, ensuring that the diameter of the self-configured overlay network does not vary much as the number of network nodes changes as long as the underlying physical network is not changed.

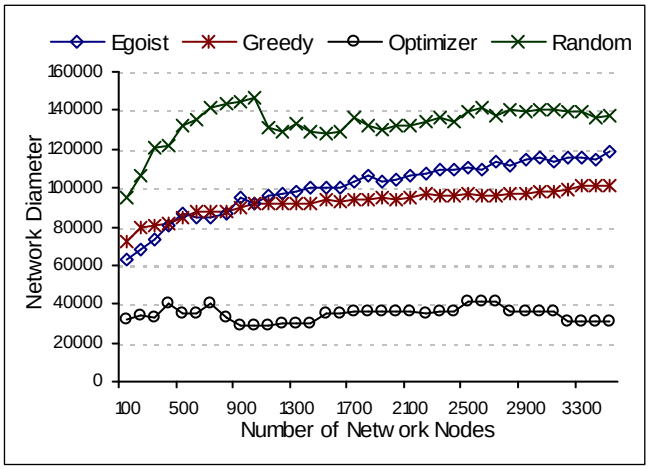

Figure 3: Network diameter under various numbers of network nodes.

3.3 Node eccentricity

In this simulation, the average node eccentricity was used as the evaluation metric. The eccentricity of a node in a P2P network is its logical network distance to the centroid of the overlay network. The centroid of the P2P overlay network is the node having the minimum variance of the shortest logical network distances to the other nodes. Assuming there are n nodes in the P2P network, the mean shortest logical network distance to other nodes for any node i in the overlay network is calculated as $\mu_i = \frac{1}{n-1} \bullet \sum_{j \neq i} d_{ij}$, where d_{ij} is the shortest logical network distance between nodes i and j. The variance of the shortest logical network distances for node i to other nodes is calculated as $s_i = \sum_{j \neq i} (d_{ij} - \mu_i)^2$. After identifying the centroid of the overlay network, the logical network distances from all other nodes to this center node are measured and the average is calculated. The simulation results are depicted in Figure 4.

Similar to results found for network diameters, the random approach generates the largest average node eccentricity. When the number of network nodes is small, the egoist protocol slightly outperforms the greedy

approach, while, conversely, the greedy peer selection protocol performs better than the egoist approach when the number of network nodes becomes larger. As expected, the overlay network configured using optimizer peer selection protocol has the lowest average node eccentricity. The average node eccentricity of the P2P overlay network built using this protocol is less than 1/3 of the average node eccentricity using either the egoist or greedy protocol. The results for the altruist approach are not reported because the overlay networks built using the altruist approach are not consistently connected, resulting in infinite network diameters and node eccentricities.

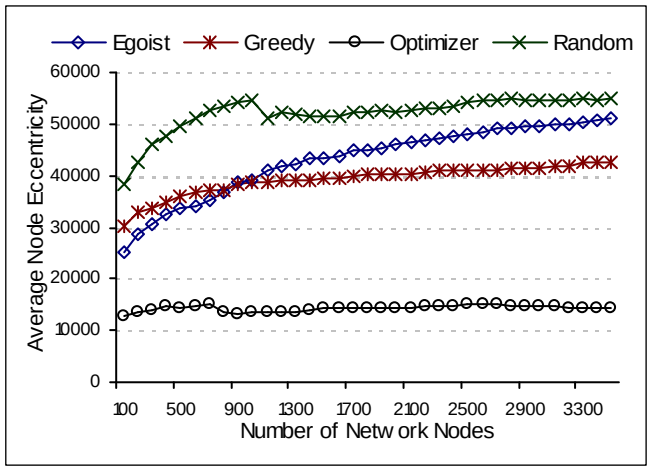

Figure 4: Average node eccentricity under various number of network nodes.

4. Self-configuration protocols for large-scale P2P networks

Many schemes have been proposed to address the self-configuration and query efficiency issues in large-scale P2P systems. Structured topologies such as HyperCuP [11] and distributed hashing mechanisms [5, 6, 7, 8] attempt to balance the loads evenly on all the nodes in the network. In dealing with unstructured networks, search algorithms such as directed breadth-first search [14], modified breadth-first search [15], adaptive probabilistic search [16], routing indices [17] and uniform random graphs [12] have been proposed to improve the efficiency of traditional flooding-based search mechanisms. These approaches are based on the assumption that all nodes have equal capacity throughout the P2P network. However, analysis of actual P2P networks [18] shows that the capacities of network nodes in these systems vary widely, meaning weak nodes are bottlenecks in these structured topologies and unstructured search schemes.

To solve this problem, some approaches, such as Ultrapeers [19] and FastTrack [1], use superpeers as gateways for weaker nodes in the system to improve the efficiency and scalability of unstructured networks. However, the distinction between a normal and a superpeer in these systems is discrete and binary, not matching the actual distribution of node capacities, thus not completely addressing the problem of varying node capabilities. The approach for addressing the network heterogeneity problem proposed in Gia [20] incorporates several techniques to compensate for the heterogeneity of the underlying network and the poor scalability of the classic Gnutella-like P2P network. Nonetheless, the cost of this method in terms of network resources is unclear, and its repliance on random walks for searches increases latency and may cause false negative responses.

In this paper, a peer-power ranking based protocol [21] for self-configuring and searching large-scale P2P networks by exploiting the natural heterogeneity of P2P networks is investigated. The goal is to create a hierarchical network structure using simple protocol and heuristics so that the self-configured P2P network can be searched more efficiently than a standard unstructured Gnutella-like P2P network as demonstrated in [22].

To improve the query efficiency further, a caching-assisted query mechanism in which the intermediate nodes cache query responses and, hence, are able to respond immediately to future queries is designed. This scheme avoids the requirement that all queries have to reach all nodes to prevent introducing false negatives.

4.1 Index Nodes

An essential feature of this proposed self-configuration protocol for large-scale P2P networks is the concept of index nodes. A set of index nodes is defined as $I = \{n \mid \forall x \in N(n), pp(x) < pp(n)\}$, 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, used for message routing but not used in the index node selection. This architecture guarantees the self-configured overlay network is connected, since any node either is an index node or can reach an index node via a parent. In this overlay network, query broadcasting is facilitated by sending a message upstream to an index node, which, in turn, disseminates it to all index nodes. Subsequently all index nodes forward this message downstream to all other nodes.

4.2 Peer-power

The goal of this self-configuration protocol is to guarantee that the number of messages handled by a node matches its capacity. A reasonable approach is one requiring that the number of connections to a node in the network is comparable to its capacity and that high capacity nodes exist near the center of the network. To facilitate this approach, peer-power, a unitless aggregate measure of the resources in a peer node, is defined as the metric for measuring node capacity. The peer-power is used to

determine the node's capacity to accept links and process messages. The proposed approach refines the traditional superpeer concept by categorizing peers with much finer granularity.

Peer-power is measured as a node's 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 display similar values, with the relationship among values corresponding to true performance differences.

In essence, the metric should reflect three facts:

- ***Diminishing Returns***: The capacity increase tends to be sublinear with respect to the increase in available resources such as bandwidth, CPU power, or memory.
- ***Weakest Component Domination***: The capacity of a system is limited by its weakest component.
- ***Uptime***: The only available estimate of a node's stability is its uptime in the P2P network. The stability does not directly impact the capacity of the node, but rewarding nodes with a high uptime is practically effective in P2P networks.

With these considerations, the following formula is proposed for computing the peer-power of a node n:

$$pp(n) = MIN(\alpha \cdot \log(BW / BW_{\min}), \beta \cdot \log(CPU / CPU_{\min}), \gamma \cdot \log(RAM / RAM_{\min})) + MAX(0, \delta \cdot \log(UT / UT_{\min})) \quad (1)$$

In this equation, *BW*, *CPU*, *RAM*, and *UT* are the network bandwidth, the CPU power, the available memory and uptime for node *n* respectively, while $BW_{\min}$, $CPU_{\min}$, and $RAM_{\min}$ are the respective minimum values of the network bandwidth, the CPU power, and memory recorded on the index nodes in the P2P network. This formula captures the available parameters into a single unitless value which can be used to differentiate among the capabilities of the nodes on the P2P network. Peer-power directly corresponds to the number of connections a node can support. The coefficients α, β, γ, and δ are used as weighting factors, which are decided by the different focuses of the P2P networks and applications. Their values can also be identified through experimental studies.

4.3 Network Configuration

JOIN and LEAVE are the two fundamental operations in the self-configuration process for P2P networks. 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 before, *N(n)* represents the neighborhood of *n* in our algorithms below, while V_g is the set of nodes in the P2P network G.

JOIN: Node *n* is joining network *G*.

When a network node joins the P2P network, it calls **Node_Join**, which, in turn, calls **Link_Creation,** to self-configure into the P2P network.

Algorithm 1: Node_Join

```
n calls Link_Creation MIN times
if ∀n'∈ 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 log|V_g| then
      n becomes index
   else
      n creates link to i
      if deg ree(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
```

Algorithm 2: Link_Creation

```
n transmits 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
   n' = p
end while
n' selects random index node i
let n' = i
while deg ree(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
```

When a new node *n* wants to join the P2P network, it contacts any existing node $n' \in V_g$. Then node 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, with the first node not already at full capacity creating a link with n. The links created using this mechanism are as close to the core nodes as possible. As a result, this approach minimizes the network diameter, reducing workloads on the weaker edge nodes. Node 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. However, making an index node from each node which does not have a parent is not feasible as the number of such nodes grows linearly with the size of the network. Rather, the number of indices should be limited to a maximum value $\log|V_g|$. If a joining node may cause this limit to be exceeded, no new index will be created. Instead, this new node locates and creates a link with an existing index. Since these additional links may overload the indices, such an overloaded index may drop an existing link to a node that has another parent.

A simple network having two index nodes is shown in Figure 5, with the number in a node representing its peer-power. The solid lines indicate parent/child peer relationships, with the node having a higher peer-power being the parent node. The secondary connection between indices is represented by the dashed line.

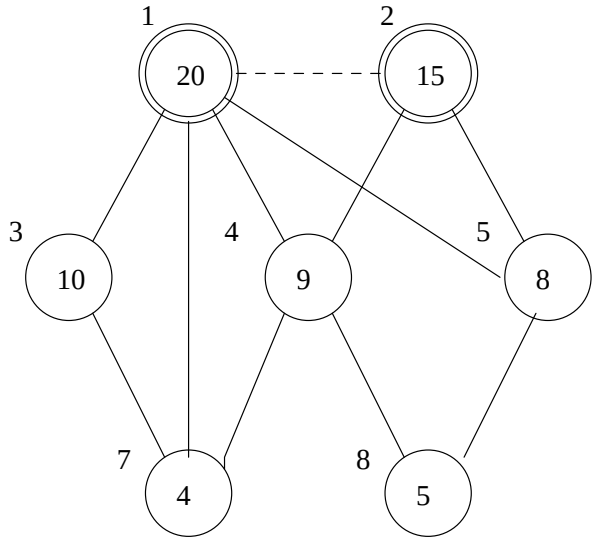

Figure 5: Basic Network.

LEAVE: Node *n* leaves network *G*

When a node $n' \in N(n)$ detects the failure of node *n* due to connection timeout, **Node_Exit** is invoked. Upon detecting this failure, its strongest non-overloaded child *c* will drop its existing connections, creating new ones with all neighbors of *n*, thus preserving the local structure of the network. In turn, node *c* must be replaced by one of its children. This process is propagated toward the edge of the network. Since node *c* is a child of node *n*, its capacity is lower than that of *n*; as a result, 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, hence, with no children, any of its neighbors whose link count falls below MIN must create new links following the JOIN protocol. Consequently, each node is required to be aware of its neighbors as well as its parents. An announcement of the failure of node *n* must be transmitted to the indices via a random upstream path to invalidate any cached information of its files. If $N(n') \geq MIN$, where n' was a neighbor of *n*, the network reconfiguration process completes. Otherwise, node n' must follow the JOIN protocol to establish new links.

Algorithm 3: Node_Exit

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

For example, the topology of the network following the departure of node 4 from the network depicted by Figure 5 is shown in Figure 6. The stronger child, 8, has assumed the links formerly held by 4.

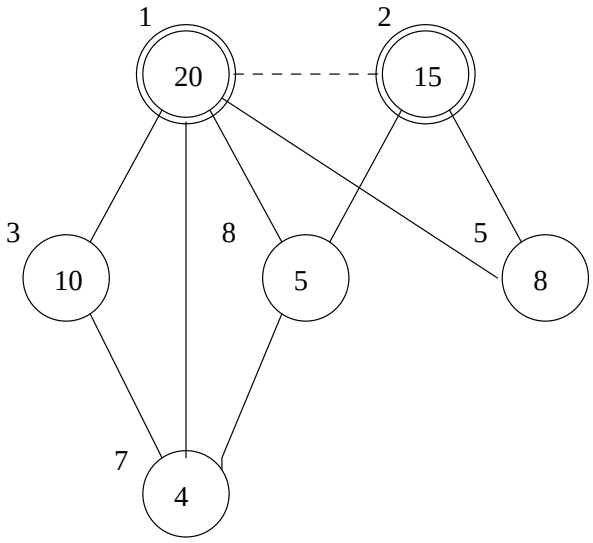

Figure 6: Network following removal of node 4.

4.4 Caching Assisted Query

A primary objective of the P2P network is to locate files with reliable queries efficiently, meaning that any node must be able to obtain an accurate directory listing and file metadata with minimal latency. Creating a view of the

namespace by querying every node on the network is clearly not feasible; rather, more practically, a complete view is obtained from well known sources. A simple approach is to rely on the indices responding to every directory query. To achieve query efficiency, the index nodes maintain a complete view of the P2P network. Since the majority events in file sharing P2P systems are reads, directory queries outweigh updates, making caching directory data advisable. Taking advantage of the hierarchical nature of the self-configured P2P overlay network proposed here, 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. This response is unicasted along the reverse path back to the initial query node, with each intermediate node caching the file information. Future queries for the same data in the same region of the P2P 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 7. The file name is a key by which queries are matched, with the owner identifying which node holds the requested file. A source field indicates the parent from which this cached information was obtained, and a list of interested children indicates which have requested information regarding the file. The latter two fields are necessary to ensure cached data can be invalidated effectively when the file is removed or a link severed. The cache line used in this approach is very small and will not take much disk space from the peer nodes. A comparable approach is currently used by web browsers, which rely on cookies to assist queries or session maintenance.

filename	owner	source	interested children

Figure 7: Cache line.

When a file is deleted or the node that holds the file fails, its cached information must be purged from the network. Since only part of the network probably has cached data regarding the file, broadcasting a global cache invalidate message is unnecessarily expensive. Hence, it is necessary that each node retain a list of children who have requested information about a given file, so that the invalidate messages are forwarded to those children only. Because links may be severed to avoid overloading indices and nodes may fail, these partially broadcasted invalidate messages may be prevented from reaching their destinations. To avoid leaving stale cache lines in the network, each cache line must contain the source node from which the information was acquired, so that cache line can still be invalidated if the link is severed. The caching-assisted query and invalidating mechanism are described below.

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 from node *n*, it is forwarded via a random upstream path. Any ancestor with a cache line for *F* responds with a unicast message, appending 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 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.

For instance, in the P2P network illustrated in Figure 8, two files f1 and f2 are located at nodes 8 and 3 respectively. The indices and the nodes which own the files maintain cache lines at all times. The information cached by the intermediate nodes following a query for f1 by Node 7 is shown in Figure 9. When a query for file f1 is issued from node 7, 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. Then, 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 workload to fit the node capacity distribution. Since read operations are expected to outweigh updates, the caching-assisted querying reduces workloads on the indices. Unlike in the approach of globally replicating directory information, in this 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.

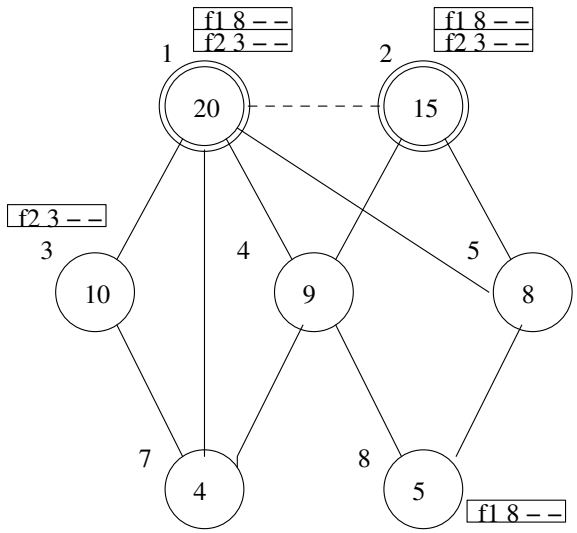

Figure 8: Basic network with two files.

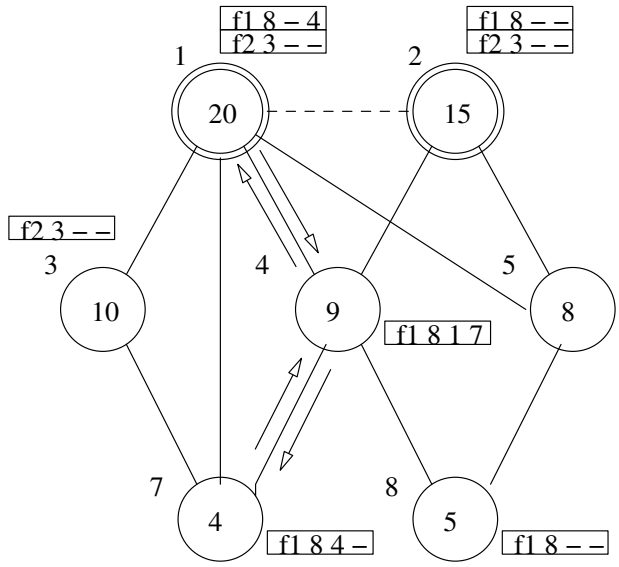

Figure 9: Network following a query for file f1.

5. Evaluating the peer-power based self-configuration protocols

As in the study of small-scale P2P networks, Simulation is used to evaluate the feasibility of the protocols and algorithms for large-scale P2P networks. Using the same simulation model, the join and exit activities of nodes in the P2P network are modeled as a standard M/M/∞ queue. The effectiveness of a large-scale P2P network can be measured in several ways. In terms of the JOIN and LEAVE operations, it is important to confirm that the diameter of the network remains small and that strong nodes migrate toward the graph centroid as anticipated. In addition, the network performance enhancement achieved by the peer-power based self-configuration protocol requires evaluation, as does the improvement of query efficiency contributed from caching.

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. The number of messages processed by each node due to node creation, node exit, query requests and query responses was measured with caching enabled or disabled.

It is assumed 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, which is assigned when the node is created, remains static throughout its lifetime. It is further assumed that the minimum number of connections for any node is *MIN*. The parameters used in these performance studies were $\mu = 15$ and $MIN = 6$. The simulation results are averaged over five trials.

5.1 Relationship of peer power to node workload

The ability of the P2P network to distribute workloads in proportion to node capacities was studied by measuring the number of messages processed as nodes enter and exit the network. This simulation was run for 1,000,000 ticks with the expected number of nodes to be 5,000. During each tick, nodes may arrive at a probability based on the arrival distribution, or depart if their time to live reaches zero. The relationship between the peer-power of a node and the number of messages it processes is illustrated in Figure 10. Each bar of the chart shows the total number of messages received by all nodes having a particular peer-power. Higher workloads are located at those nodes most able to handle the workloads, indicating that the network topology matches the variance of its node capacities.

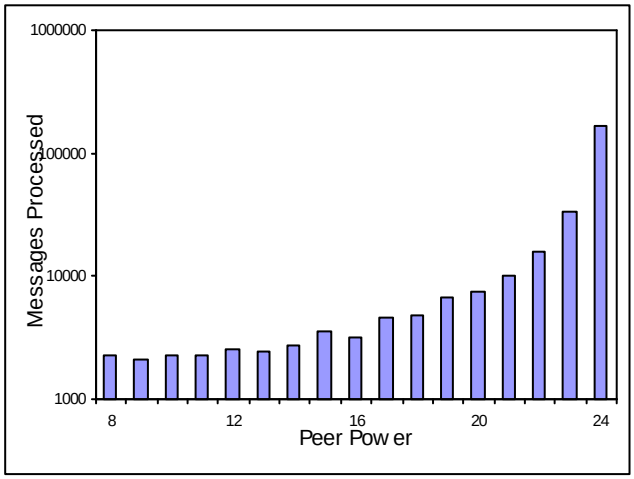

Figure 10: Message processed vs. peer power.

5.2 Efficiency of the peer-power based self-configuration protocols

Message handling latencies and the number of nodes involved in message routing are primarily dependent on the distances of the nodes from the indices. The effectiveness of the peer-power based self-configuration protocols studied here is evaluated by comparing it to a Gnutella-like approach whereby random links to existing nodes are created.

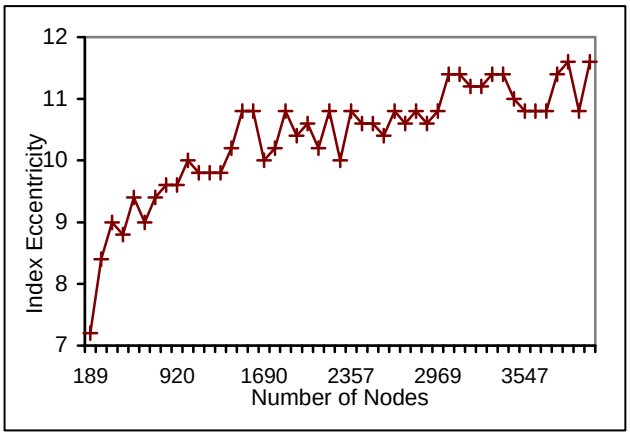

Figure 11: Index Eccentricity.

As discussed previously, the number of index nodes is bounded by $\log|V_g|$, and index nodes accept overload connections if an excessive number of locally maximal nodes are created. Bounding the number of index nodes can maintain a small diameter for the network, as indicated by the simulation results depicted by Figure 11, in which the eccentricity of the index nodes is logarithmic with respect to the size of the P2P network self-configured by the proposed JOIN/LEAVE protocols.

An overload forces some nodes to exceed their workload capacities, thereby degrading system performance. The degree of overloading using these proposed self-configuration protocols is compared with that of creating links with random peers.

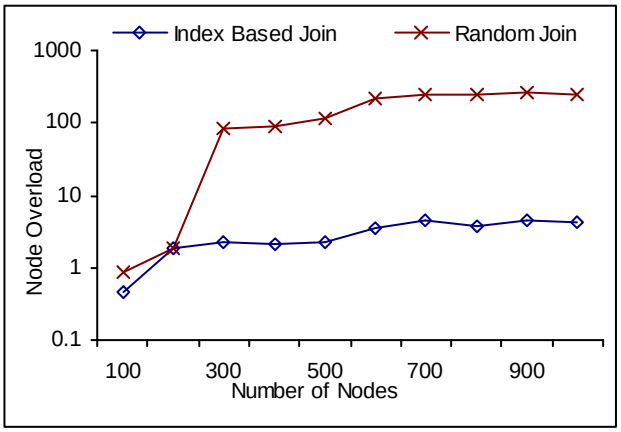

Figure 12: Node Overload.

Figure 12 illustrates the highest degree to which a node in the network is overloaded after 500,000 ticks with the expected number of nodes varying from 100 to 1,000. Using the peer-power based self-configuration protocols, no node is forced to accept more than five links in excess of its capacity. However, using a policy of creating links with random peers forces in excess of one hundred overload links to some nodes. The simulation results show that the proposed self-configuration protocols are effective in terms of minimizing node overload.

5.3 Effectiveness of index based queries

The searches in Gnutella-like unstructured networks are costly 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 all query messages. 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. On the other hand, using the index-based query mechanism proposed here reduces the query cost to the number of hops needed to reach an index node from a source node.

To illustrate the advantages of this index-based searching mechanism, a series of simulated searches were conducted in the self-configured P2P network. The same searches were also run 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 random nodes at random intervals, with a mean of 1,000 ticks. A query for a random file was issued from a random node at each tick.

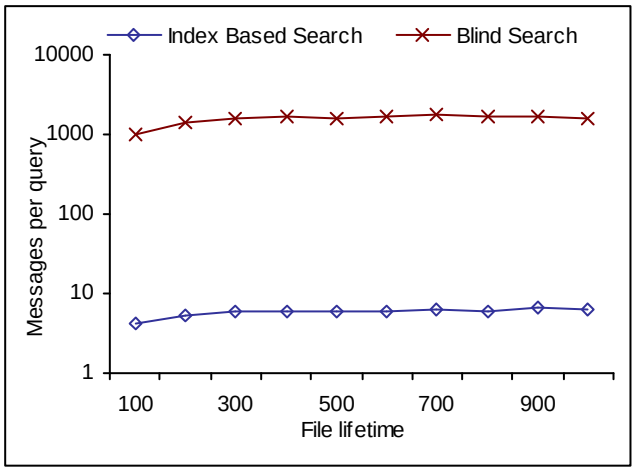

Figure 13: Query cost comparison.

Figure 13 shows the number of messages handled per query for query, response, and cache invalidating operations through 100,000 ticks. The lifetimes for a file

ranged from 100 to 1,000 ticks following creation. When the lifetime of a file expired, it was purged, and an invalidate message was passed to any interested nodes. As demonstrated by Figure 13, the average number of messages needed for each query using the classic Gnutella-like approach was significantly greater than that using the index-based search approach.

5.4 Benefits of caching assisted querying

Caching query results at intermediate nodes can further reduce the cost of queries, particularly when files are accessed frequently before they are destroyed, since the overhead of invalidate messages is relatively low. To investigate the benefit of caching query results, the search simulation was performed with both caching enabled and disabled. The efficiency of the query mechanism was measured by the number of messages needed per query as depicted in Figure 14. As expected, search without caching queries at intermediate nodes required up to 40% more messages even if the extra invalidate messages were included in the caching-assisted query mechanism.

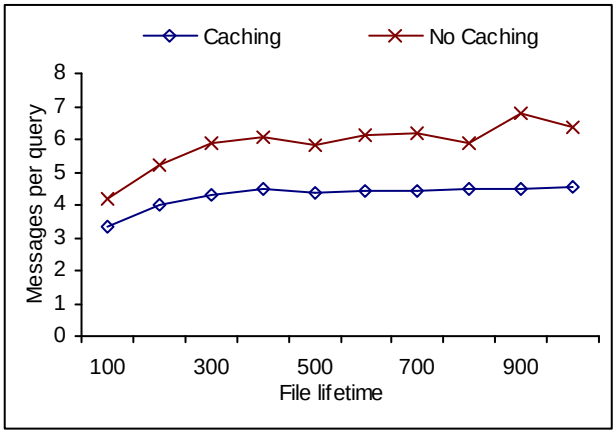

Figure 14: Benefit of cache assisted queries.

6. Conclusion and future studies

This paper proposes and evaluates different protocols for individual peers to self-configure the P2P overlay networks in various network environments. For small-scale homogeneous networks, an optimizer peer selection protocol was proposed and found to be superior to other peer selection protocols, including random, egoist, altruist and greedy protocols, in terms of P2P overlay network diameter and node eccentricity. The optimizer protocol considers both the physical and the logical network latency in neighbor selection, attempting to minimize the difference between the physical and the logical network distances between two nodes by creating virtual links along the shortest physical network distance path in the overlay network configuration. Although the peers need to exchange more information with other peers, using optimizer neighbor selection protocol is still worthwhile since the overlay network diameter and average node eccentricity are much smaller.

For large-scale heterogeneous P2P systems, a peer-power ranking based mechanism is proposed to self-configure semi-structured overlay networks for heterogeneous P2P systems. An index-based query scheme is also proposed for peers to search documents in the P2P network. Performance studies have demonstrated that the proposed self-configuration protocols and search scheme are efficient and viable. The self-configured P2P network has significantly lower query cost than traditional Gnutella-like P2P systems. In addition, a caching-assisted query mechanism is proposed to exploit the surplus storage power of the peers, further improving the 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.

Currently, for small-scale P2P networks, the cost of the proposed peer selection protocols in terms of network traffic overhead is being investigated. A possible solution to reducing this cost during the P2P overlay network configuration is to require peers to cache only information about a small set of network nodes. For large-scale P2P networks, studies are currently conducted to assess the viability of the proposed peer-power estimate by measuring the actual connection capacities of nodes in comparison to their estimated capacities. In addition, considering file replication and including file metadata in cache lines may be used in cache-assisted query mechanism to avoid unnecessary contact to the file owners.

7. Reference

[1] The FastTrack Protocol, *http://developer.berlios.de/projects/gift-fasttrack.*

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

[3] Napster protocol specifications. *http://opennap.sourceforge.net/napster.txt*, 2000

[4] E.Cohen, A.Fiat and H.Kaplan: Associative Search in Peer-to-Peer Networks: Harnessing Latent Semantics. *In Proc. IEEE Infocom*, Apr.2003

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

[6] S. Ratnasamy, P.Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. *In Proc. ACM SIGCOMM 2001*, pages 161-172, August 2001

[7] A. Rowstron and P. Druschel, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *IFIP/ACM International*

Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.

[8] B. Zhao, J. Kubiatowicz, and A. 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.

[9] 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.

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

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

[12] Q.Lv, P.Cao, E.Cohen, K.Li, and S.Shenker, Search and replication in unstructured peer-to-peer networks. *In Proc. of the 16th Annual ACM International Conference on supercomputing*, 2002.

[13] Gopal Pandurangan, Prabhakar Raghavan and Eli Upfal, Building low-diameter p2p networks, *Proceedings of STOC 2001*, Crete, Greece, Pages 492 - 499, 2001

[14] Beverly Yang and Hector Garcia-Molina, Improving Search in Peer-to-Peer Networks, *in Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, pages 5 -14, Vienna, Australia, July, 2002.

[15] Vana Kalogeraki and Dimitrios Gunopulos and D. Zeinalipour-Yazti, A local search mechanism for peer-to-peer networks, *in Proceedings of the eleventh international conference on Information and knowledge management*, pages 300 - 307, Virginia, 2002.

[16] D. Tsoumakos and N. Roussopoulos, Adaptive Probabilistic Search (APS) for Peer-to-Peer Networks, *CS-TR-4451*, University of Maryland, 2003.

[17] Arturo Crespo and Hector Garcia-Molina, Routing Indices For Peer-to-Peer Systems, *in Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, pages 23 - 34, Vienna, Australia, July, 2002.

[18] Stefan Saroiu, P. Krishna Gummadi and Steven D. Gribble, Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts, *Multimedia Systems*, Volume 9, Number = 2, pages 170 - 184, August, 2003.

[19] 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*.

[20] Yatin Chawathe, et al., Making Gnutella-like systems Scalable, *ACM SIGCOMM 2003*, Pages 407-418, August, 2003

[21] James Z. Wang and Matti Vanninen. A Novel Self-Configuration Mechanism for Heterogeneous P2P Networks, *In Proceedings of IEEE/WIC/ACM Conference on Intelligent Agent Technology (IAT'04)*, Beijing, September 2004.

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