HPC5: An Efficient Topology Generation Mechanism for Gnutella Networks

Santosh Kumar Shaw, Joydeep Chandra, and Niloy Ganguly

Indian Institute of Technology, Kharagpur, India

Abstract. In this paper, we propose a completely distributed topology generation mechanism named HPC5 for Gnutella network. A Gnutella topology will be efficient and scalable if it generates less number of redundant queries and hence consists of lesser short length cycles. However, eliminating cycles totally, reduces the coverage of the peers in the network. Thus in the tradeoff between the cycle length and network coverage we have found that a minimum cycle length of 5 provides the minumum query redundancy with maximum network coverage. Thus our protocol directs each peer to select neighbors in such a way that any cyclic path present in the overlay network will have a minimum length of 5. We show that our approach can be deployed into the existing Gnutella network without disturbing any of its parameters. Simulation results signify that HPC5 is very effective for Gnutella's dynamic query search over limited flooding.

1 Introduction

Peer-to-peer (P2P) network is an overlay network, useful for many purposes like file-sharing, distributed computation, etc. Depending upon the topology formation, P2P networks are broadly classified as structured and unstructured. An unstructured P2P network is formed when the overlay links are established arbitrarily. Decentralized (fully distributed control), unstructured P2P networks (Gnutella, FastTrack etc) are the most popular file-sharing overlay networks. The absence of a structure and central control makes such systems much more robust and highly self-healing compared to structured systems [8,12]. But the main problem of these kinds of networks is scalability due to generation of large number of redundant messages during query search. Consequently as these networks are becoming more popular the quality of service is degrading rapidly [5,9].

To make the network scalable, *Gnutella* [1,2,3] is continuously upgrading it's features and introducing new concepts. All these improvements can be categorized into two broad areas: improvements of search techniques and modification of the topological structure of the overlay network to enhance search efficiency. In enhanced search techniques, several improvements like *Time-To-Live* (TTL), *Dynamic query*, *Query-caching* and *Query Routing Protocol* (QRP) have been introduced. One of the most significant topological modifications in unstructured network was done by inducing the concept of *super-peer* (ultra-peer) with a two-tier network topology.

The basic search mechanism adhered by Gnutella is *limited flooding* [1,2,3]. In flooding, a peer that searches for a file, issues a query and sends it to all of its neighbor peers. The peer that receives the query forwards it to all its neighbors except the neighbor from

V. Garg, R. Wattenhofer, and K. Kothapalli (Eds.): ICDCN 2009, LNCS 5408, pp. 114–126, 2009.

which it is received. By this way, a query is propagated up to a predefined number of hops (*TTL*) from the source peer. The TTL followed by Gnutella is generally 1 or 2 for popular search. However, the query with $TTL(3)$ (numeric value inside parenthesis represents the number of hops to search with) is initiated for rare searches.

The main goal of this paper is to improve the scalability of the Gnutella network by reducing redundant messages. One of the ways to achieve this is to modify the overlay network, so that small size loops get eliminated from the overlay topology. The rationale behind the proposition is explained through Fig. 1. In this figure, both networks have the same number of connections. With a $TTL(2)$ flooding, the network in Fig. 1(a) discovers 4 peers at the expense of 7 messages, whereas the network in Fig. 1(b) discovers 6 peers without any redundant messages. This happens due to the absence of any 3-length cycle in the network of Fig. 1(b). On generalizing, we can say that for a $TTL(r)$ flooding, networks devoid of cycles of length less than $(2r + 1)$ do not generate any redundant messages. In this paper, we propose a handshake protocol that generates a cycle-5 network topology (a network which does not have any cycles up to length $(r - 1)$ is referred as cycle-r network). However, some redundant messages are produced for rare searches ($TTL(3)$). Since this (rare search) is performed rarely, in this case we are more interested in having higher coverage than eliminating generation of redundant messages. The strength of our proposed mechanism is its simplicity and the ease of deployment over existing Gnutella networks along with its power to generate topologies having high efficiency in terms of message complexity and network coverage.

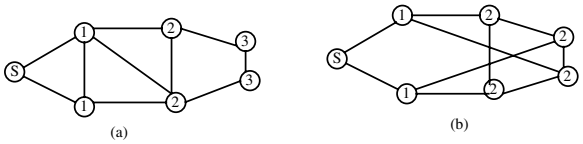


Fig. 1. Effect of topology structure on limited flood based search. The number inside the circle represents the TTL value required to reach that node from start node S.

Many algorithms exist in the literature that modify the topology in unstructured P2P networks to solve the excessive traffic problem. The structural mismatch between the overlay and underlying network topology is alleviated by using location aware topology matching algorithms [6,7]. A class of overlay topology based on distance between a node and its neighbors in the physical network structure is presented in [10]. Papadakis *et* al. presented an algorithm to monitor the ratio of duplicated message through each network connection. Consequently a node does not forward any query through that connection whose ratio exceeds certain threshold [11]. Zhu *et* al. very recently presented a distributed algorithm in [15] to improve the scalability of Gnutella like networks by reducing redundant messages. They have pointed the same concept of elimination of 3 and 4-length cycles. However this is demand driven and involves a lot of control overhead. Also it is not clear how the algorithm will perform in the face of heavy traffic and inconsistency. The algorithm also does not take care in preserving the Gnutella parameters (like degree distribution, average peer distance, diameter, etc), hence robustness

of the evolved network is not maintained. In our work we take into considerations all the above aspects and propose a holistic and simple approach to topology formation. The algorithm initiates as soon as a peer enters in the network rather than having it demand driven. Thus the algorithm works during the bootstrap phase when the network is forming so that less overhead in involved afterwards.

A list of main notations that will be used throughout the paper is summarized in table 1 for ready reference.

Table 1. Notations

$TTL(r)$	Query search with $TTL = r$	d_{uu}	Avg. no. of ultra-neighbors of an ultra-peer
cycle-r	A cycle of minimum length r	d_{ul}	Avg. no. of leaf-neighbors of an ultra-peer
cycle-r network	A network which does not have any cycle up to length $(r-1)$	d_{lu}	Avg. no. of ultra-neighbors of a leaf-peer
cycle-3 network	Gnutella network	H_k	Hit ratio to select k^{th} ultra-neighbor
N	Total number of peers in the network	$\langle H \rangle$	Average hit ratio of a peer
U	Total number of ultra-peers in the network	$\langle H_{ev} \rangle$	Average evolved hit ratio of a peer
L	Total number of leaf-peers in the network	r^{th} neighbor	A peer at a distance of r hops. All immediate neighbors are 1^{st} neighbors, all neighbors of 1^{st} neighbors are 2^{nd} neighbors and so on.

2 Basic System Model of Gnutella 0.6

In order to carry on experiments, a basic version of *Gnutella 0.6* [1,2,3] has been implemented. The basic Gnutella consists of a large collection of nodes that are assigned unique identifiers and which communicate through message exchanges.

Topology: Gnutella 0.6 is a two-tier overlay network, consisting of two types of nodes: *ultra-peer* and *leaf-peer* (the term peer represents both ultra and leaf peer). An ultra-peer is connected with a limited number of other ultra-peers and leaf-peers. A leaf-peer is connected with some ultra-peers. However, there is no direct connection between any two leaf-peers in the overlay network. Yet another type of peer is called legacy-peers, which are present in ultra-peer level and do not accept any leaves. In our model we are not considering legacy-peers.

Basic Search Technique: The network follows limited flood based query search. A query of an ultra-peer is forwarded to its leaf-peers with $TTL(0)$ and to all its ultra-neighbors with one less TTL only when ($TTL > 0$). A leaf-peer does not forward query received from an ultra-peer. On the other hand ultra-peers perform query searching on behalf of their leaf peers. The query of a leaf-peer is initially sent to its connected ultra-peers. All the connected ultra-peers simultaneously forward the query to their neighbor ultra-peers up to a limited number of hops. Since multiple ultra-peers are initiating flooding, a leaf-peer's query will produce more redundant messages if the distance between any two ultra-neighbors is not enough. Gnutella 0.6 incorporates *dynamic querying* over limited flooding as query search technique. In dynamic querying, an ultra-peer incrementally forwards a query in 3 steps ($TTL(1), TTL(2), TTL(3)$ respectively) through each connection while measuring the responsiveness to that query.

The ultra-peer can stop forwarding query at any step if it gets sufficient number of query hits. Consequently dynamic querying uses $TTL(3)$ only for rare searches. Modern Gnutella protocol uses *QRP* technique over dynamic querying in which a leaf-peer creates a hash table of all the files it is sharing and sends that table to all the immediate ultra-neighbors. As a result, when a query reaches an ultra-peer it is forwarded to only those connected leaf-peers which would have query hits [1,2].

Basic Handshake Protocol: Many softwares (clients) are used to access the Gnutella network (like *Limewire, Bearshare, Gtk-gnutella*). The most popular client software, Limewire's handshake protocol is used in our simulation as a base handshake protocol. Through handshaking, a peer establishes connection with any other ultra-peer. To start handshake protocol a peer first collects the address of an online ultra-peer from a pool of online ultra-peers. A peer can collect the list of online peers from *hardcoded* address/es and/or from *GwebCache* systems [4] and/or through *pong-caching* and/or from its own hard-disk which has obtain list of online ultra-peers in the previous run [5]. The handshake protocol is used to make new connections. A handshake consists of 3 groups of headers [1,2]. The steps of handshaking is elaborated next:

1. The program (peer) that initiates the connection sends the first group of headers, which tells the remote program about its features and the status to imply the type of neighbor (leaf or ultra) it wants to be.
2. The program that receives the connection responds with a second group of headers which essentially conveys the message whether it agrees to the initiator's proposal or not.
3. Finally, the initiator sends a third group of header to confirm and establish the connection.

This basic protocol is modified in this paper to overcome the problem of message overhead.

Simulated Gnutella: To generate existing Gnutella network, we have simulated a strip down version of Gnutella 0.6 protocols which follows parameters of Limewire [1]. Our simulated Gnutella network exhibits all features (like degree distribution, diameter, average path length between two peers, proportion of ultra-peers, etc.) exhibited by Gnutella network. These features are obtained from the snapshots collected by crawlers [3,13,14].

3 HPC5: Handshake Protocol for Cycle-5 Networks

Fig. 2 illustrates the proposed HPC5 graphically. In Fig. 2, peer-1 requests other online ultra-peers to be its neighbor, given that, peer-2 is already a neighbor of peer-1. In Fig. 2(a) and 2(b), the possibility of the formation of triangle and quadrilateral arises if a 1^{st} or 2^{nd} neighbor of peer-2 is selected. However, this possibility is discarded in Fig. 2(c) and a cycle of length 5 is formed.

Each peer maintains a list of its 1^{st} and 2^{nd} neighbors, which contains only ultra-peers (because a peer only sends request to an ultra-peer to make neighbor). The 2^{nd} ultra-neighbors of a leaf-peer represents the collection of 1^{st} ultra-neighbors of the

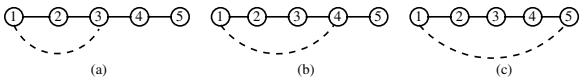


Fig. 2. Selection of neighbor by peer-1 after making peer-2 as a neighbor

connected ultra-peers. To keep updated knowledge, each ultra-peer exchanges its list of 1^{st} neighbors periodically with its neighbor ultra-peers and sends the list of 1^{st} neighbors to its leaf-peers. To do this with minimal overhead, piggyback technique can be used in which an ultra-peer can append its neighbor list to the messages passing through it.

The three steps of modified handshake protocol (HPC5) is described below.

1. The initiator peer first sends a request to a remote ultra-peer which is not in its 1^{st} or 2^{nd} neighbor set. The request header contains the type of the initiator peer. The presence of remote peer in 2^{nd} neighbor set implies the possibility of 3-length cycle. In Fig. 2, peer-1 cannot send request to peer 2 or 3, on the other hand peer 4 & 5 are eligible remote ultra-peers.
2. The recipient replies back with its list of 1^{st} neighbors and the neighbor-hood acceptance/rejection message. If the remote peer discards the connection in this step, the initiator closes the connection and keeps the record of neighbors of the remote peer for future handshaking process. On acceptance of the invitation by the remote-peer, the initiator peer performs the following tasks.
3. The initiator peer checks at least one common peer between its 2^{nd} neighbor set (say, A) and the 1^{st} neighbor set of the remote peer (say, B). A common ultra-peer between sets A and B indicates the possibility of 4-length cycle.

   If no common peer is present between sets A and B then the initiator sends *accept connection* to remote peer.

   Otherwise the initiator sends *reject connection* to remote peer.

HPC5 prevents the possibility of forming a cycle of length 3 or 4 and generates a cycle-5 network.

4 Hurdles in Implementing the Scheme

Before embedding HPC5 in Gnutella network, we need to consider certain issues to assess the viability of HPC5 : whether this scheme is compatible with the current populations of Gnutella network, the average number of trials required to get an ultra-neighbor, and whether there is any possibility of inconsistency. Each of the issues are discussed one by one.

Compatibility with the Current Population of Gnutella

From ultra-peer point of view, the total number of ultra-leaf connections is $U \cdot d_{ul}$ and from leaf-peer point of view it is $L \cdot d_{lu}$. By equating both and considering $U + L = N$, we get

$$\mathrm{N} = U \cdot \frac{(d_{ul} + d_{lu})}{d_{lu}} \tag{1}$$

Fig. 3 represents a part of an ultra-peer layer where P has immediate neighbors at level Q. Suppose, P is already connected with $(d_{uu} - 1)$ number of ultra-peers at Q level and wants to get d_{uu}^{th} ultra-neighbor. According to HPC5, P should not connect to any ultra-peer from R or S level as its next neighbor. However, T can be a neighbor of P. Thus, we can say that if P wants to make a new ultra-neighbor then P has to exclude at most $(d_{uu} - 1)$, $(d_{uu} - 1)^2$ and $(d_{uu} - 1)^3$ number of ultra-peers from Q, R and S level respectively. So, total $[(d_{uu} - 1) + (d_{uu} - 1)^2 + (d_{uu} - 1)^3] \approx d_{uu}^3$ number of peers cannot be considered as next neighbor(s) of P. Therefore the number of ultra-peers in the network needs to be at least

$$U \approx d_{uu}^3 \tag{2}$$

From equations 1 and 2 we get

$$N \approx \frac{(d_{ul} + d_{lu}) \cdot d_{uu}^3}{d_{lu}} \tag{3}$$

Presently Gnutella network is having the population of almost 2000k of peers at any time [1]. From equation 3 it can be seen that for the present values of $d_{uu} = 26, d_{ul} = 22$ & $d_{lu} = 4$ found in present day Gnutella networks, 120-130k peers are sufficient to implement HPC5 protocol. However, to form cycle-6 networks (HPC6) the number of peers

$$N \approx \frac{(d_{ul} + d_{lu}) \cdot d_{uu}^4}{d_{lu}}$$

required is more than 2000k. Hence the current population will not be able to support any such attempts.

Hit Ratio: Hit ratio is defined as the inverse of the number of trials required to get a valid ultra-peer neighbor. As our protocol puts some constraints on neighbor selection, a contacted agreeing remote ultra-peer may not be selected as neighbor. Mathematically, on an average if a peer (say, P) is looking for its k^{th} ultra-neighbor and the m_k^{th} contacted ultra-peer satisfies the constraints and becomes k^{th} ultra-neighbor of P, then the hit ratio for k^{th} neighbor will be $H_k = \frac{1}{m_k}$. We first make a static analysis of hit ratio, then fine tune it considering that the network is evolving.

At the time of k^{th} ultra-neighbor selection in HPC5, a peer (say, P) does not consider its 1^{st} $((k-1)$ ultra-peers) and 2^{nd} $((k-1)(d_{uu} - 1)$ ultra-peers) ultra-neighbors

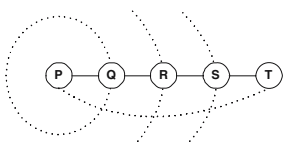


Fig. 3. A part of an Ultra-peer layer, where a node represents all nodes that are present in that level. Like, Q represents all 1^{st} neighbors of P.

as a potential neighbor and this exclusion is locally done by checking the 1^{st} & 2^{nd} neighbors lists of P. The number of ultra-peers excluded (U') is then $[(k-1)+(k-1)(d_{uu}-1)]$. According to step-3 of HPC5, P cannot make neighbor from any ultra-peer of level S (3^{rd} ultra-neighbors of P) of Fig. 3 as its neighbor which are $U'' = [(k-1)(d_{uu}-1)^2]$ in number. So, total $[U'+U'']$ number of ultra-peers are excluded. We assume that the probability of getting any ultra-peer is uniform. So hit ratio can be given as $H_k = \frac{U-(U'+U'')}{U-U'}$. Assuming $U' \ll U,\ \ U' \ll U''$ and $U'' \approx d_{uu}^2 \cdot (k-1)$, therefore H_k becomes

$$H_k \approx \frac{U - d_{uu}^2 \cdot (k-1)}{U} \tag{4}$$

The upper bound of k and consequently average ultra-degree differs in leaf-peer and ultra-peer. To generalize further calculations, let m be the average ultra-degree of a peer. So, average hit ratio, denoted as $\langle H \rangle = \frac{1}{m} \cdot \sum_{k=1}^{m} H_k$ is,

$$= 1 - \frac{d_{uu}^2 \cdot (m-1)}{2 \cdot U}. \tag{5}$$

The equation 5 shows the average hit ratio of peer joining the network when the population of ultra-peers in the network is U. It also reflects that $(1 - \langle H \rangle)$ is inversely proportional to the number of ultra-peers (U) in the complete network. Now as each node joins, the network grows. As a result the average hit ratio changes with the network growth. Therefore evolved hit ratio is the average value of all average hit ratios which are calculated at each growing stages of the network. Let U_0 and U_n be the number of ultra-peers in the initial and final networks. So, evolved hit ratio is

$$\begin{aligned} \langle H_{ev} \rangle &= \frac{1}{U_n - U_0} \cdot \sum_{U_i=U_0}^{U_n} \langle H \rangle \\ &= 1 - \frac{d_{uu}^2 \cdot (m-1)}{2 \cdot (U_n - U_0)} \cdot \sum_{U_i=U_0}^{U_n} \frac{1}{U_i} \qquad (6) \\ &\approx 1 - \frac{d_{uu}^2 \cdot (m-1)}{2 \cdot (U_n - U_0)} \cdot \log (U_n/U_0) \qquad (7) \end{aligned}$$

From equations 5 and 7 we get,

$$\langle H_{ev} \rangle \ \leq \ \langle H \rangle$$

As d and the maximum value of m are bounded, the value of $\langle H_{ev} \rangle$ increases with U. Again we have tested this phenomenon through our simulation and plotted the evolved hit-ratio against the network size of 200k-1000k in Fig 4 and observed the similarity between them. The similarity is not pronounced in the beginning as the approximations made to develop equations 4 and 7 play major role in smaller networks.

Consistency Problem: Periodically exchanging the list of neighbors facilitates the peers to get up-to-date information about their neighbors. In between two successive

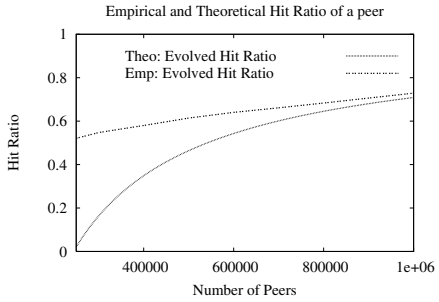


Fig. 4. Hit ratio of a peer against the number of peers in the network

updates, a peer may possibly have erroneous knowledge about it's neighbors. As a result, this inconsistency of the network leads to the presence of 3-length or 4-length cycles. Parallel update is possible when many peers enter simultaneously or there is a huge failure/attack in the network whereby many nodes have lost their neighbors and would now like to gain some.

1. *Parallel update:* In parallel update, due to inconsistency, smaller length cycles are formed as multiple peers from the same cycle handshake in parallel with a third common ultra-peer to become each other's neighbor. The parallel update situation is illustrated through an example (Fig. 5(a)) where peer-1 and peer-5 execute the following actions according to steps of HPC5 and form cycle-3.
   (a) Both peer-1 and peer-5 find that peer-P is a valid remote peer to contact and both send request to P.
   (b) Peer-P gets their request more-or-less at the same time and sends back the neighbor-hood status to them.
   (c) As peer-1 and peer-5 do not know each other's activity or updated status, they make P as their new neighbor, therefore a cycle-3 is formed due to this inconsistency.

   Similarly smaller cycles may be created when multiple peers contact each other as a directed cycle (as in Fig. 5(b)) within the period of two successive updates.

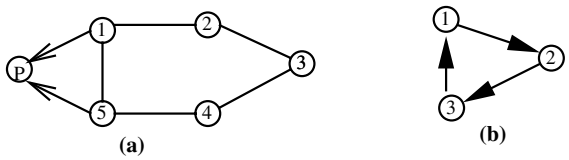


Fig. 5. A part of cycle-5 network, representing parallel update inconsistency

2. *Inconsistency arising in the face of failure/attack:*
   Here we discuss about the topological status of the network when x fraction (where $x \ll 1$) of nodes are left/removed from the network. We assume that the nodes have left uniformly from the different parts of the network. So each peer loses a fraction

of its neighbors and in effect the average degree of a peer in the network becomes less. To maintain the degree distribution of the network, each peer contacts other remote ultra-peers to fulfil neighbor deficiency. During this process, 3-length and 4-length cycles are created temporarily due to inconsistency between two successive updates. We calculate the effects of inconsistency due to peers removal from an ultra-peer point-of-view, so in this section the term peer will be used to represent an ultra-peer.

• *3-length cycle:* A 3-length cycle is created if two neighbor peers and third remote peer get involved in HPC5 as in Fig. 5. The initiation of handshake protocol in different combinations among three peers may create triangle. Here in calculation we are following the combination shown in Fig. 5(a) and peers are named as peer-1, peer-5 and P. After removal process $U_{rem} = (U - xU)$ number of ultra-peers remain in the network. According to HPC5, a peer cannot make any ultra-peers at level Q, R or S in Fig. 3 as its neighbor. We assume that the probability of getting any ultra-peer is uniform. So the probability of selecting an ultra-neighbor (here P) by peer-1 is

$$P_0 = \frac{U_{rem} - [d_{uu}(1-x)]^3}{U_{rem}}$$

The probability of choosing the same ultra-peer (P) as neighbor by any neighbor of peer-1 (here peer-5) is

$$P_1 \approx \frac{1}{U_{rem} - [d_{uu}(1-x)]^3}$$

So the probability of forming a 3-length cycle is

$$P_t = P_0 \cdot P_1 \approx \frac{1}{U_{rem}}$$

Therefore, the average number of 3-length cycles created around an ultra-peer is $[(d_{uu} + d_{ul})(1-x)/U_{rem}]$ and total number of 3-length cycles formed in the network is

$$\begin{aligned} L_3 &= \bigcirc\left(\frac{(d_{uu} + d_{ul})(1-x)}{U_{rem}} \cdot U_{rem}\right) \\ &= \bigcirc((d_{uu} + d_{ul})(1-x)) = \bigcirc(d_{uu}) \end{aligned}$$

($\bigcirc(f)$ represents big-oh(f)).

• *4-length cycle:* A 4-length cycle is created if P and one of its 2^{nd} ultra-neighbors or any two 1^{st} neighbors (leaf or ultra) of P contact T and become neighbors. Similar to 3-length cycles calculation, the average number of 4-length cycles created around an ultra-peer is

$$\frac{[d_{uu}(1-x)]^2 + d_{uu}d_{ul}(1-x)^2}{U_{rem}}$$

and total number of 4-length cycles formed in the network is

$$L_4 = \bigcirc([d_{uu}(1-x)]^2 + d_{uu}d_{ul}(1-x)^2) = \bigcirc({d_{uu}}^2)$$

So the total number of smaller length cycles created due to nodes removal is

$$L = L_3 + L_4 = \bigcirc({d_{uu}}^2) \tag{8}$$

which is very less compared to network size.

In the same way we can prove that the effect of inconsistency from leaf-peer point-of-view is $\bigcirc(d_{uu} \cdot d_{lu})$.

Although, the presence of a small percentage of smaller length cycles in the network is tolerable, as they do not affect much on the performance but we have developed an algorithm for detecting and removing small length cycles arising due to inconsistency. However the details are avoided due to lack of space.

5 Evaluation by Simulation

To validate our approach, we have performed numerous experiments. We have taken different sizes (up to 1000k nodes) of networks and through these experiments, we have shown that HPC5 performs better than the existing protocols.

5.1 Search Performance

In our simulations, we have used message complexity and network coverage as performance metrics to analyze the search efficiency. *Message complexity* is defined as the average number of messages required to discover a peer in the overlay network. *Network coverage* implies the number of unique peers explored during query propagation in limited flooding. We have plotted the network coverage and message complexity (y-axis) with $TTL(2)$ and $TTL(3)$ flooding against the size of the network (x-axis) for leaf as well as ultra peers. To get the overall performance of the network, we have chosen the number of ultra-peers and leaf-peers for query flooding in the same $\frac{U}{L}$ ratio. The performance of the network is greatly influenced by the value of TTL used in search and thus we have discussed the performance metrics based on $TTL(2)$ and $TTL(3)$ separately. The search performance (specially message complexity) also depends on the implementation of *QRP* technique [1]. Thus we have discussed the search performance without and with QRP.

TTL(2) without QRP

It is clear from figures 7(a) and 7(b) that with $TTL(2)$, cycle-5 networks are better than cycle-3 networks in both message complexity and network coverage. In cycle-5 networks, the network coverage is approximately doubled and message complexity is almost 20% less than that of cycle-3 networks. With $TTL(2)$, a search query covers a significant portion (in our simulation it is more than 30%) of the cycle-5 network with lesser number of redundant messages. From the results we see that the message complexity is not close to 1 as expected. This is because the message complexity of the leaf-peer generated query is particularly high (Fig. 7(b)). In cycle-5 networks, a leaf-peer can be connected with two ultra-peers which are themselves 3^{rd} or 4^{th} neighbors of each-other and becomes a part of cycle-5 or cycle-6 (Fig. 6). From Fig. 6 we see, a leaf-peer search is initiated by its ultra-peers; both ultra-peers 1 & 4 (in Fig. 6(a))

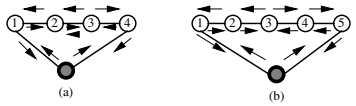
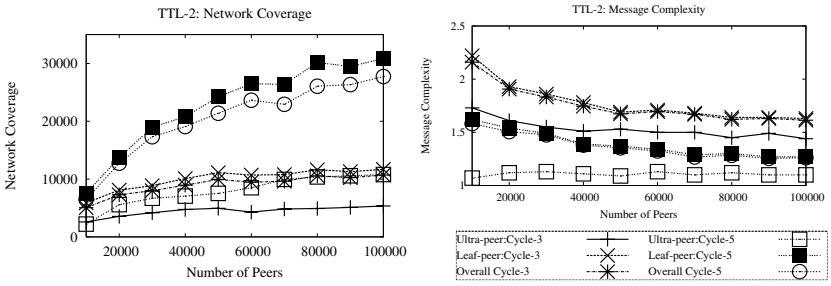


Fig. 6. Effect of leaf-peer layer with TTL(2) search. The arrows inside and outside of polygons indicate the directions of search by a leaf-peer and an ultra-peer respectively.

$\{1$ & 5 (in Fig. 6(b))$\}$ start a $TTL(2)$ flooding. Consequently redundant messages are produced at ultra-peers 2 & 3 $\{3\}$. However, hardly any redundancy is generated in ultra-peer initiated query ((Fig. 7(b)).



(a) Network Coverage (b) Message Complexity

Fig. 7. Network Coverage and message complexity with TTL 2 for cycle-3 and cycle-5 networks

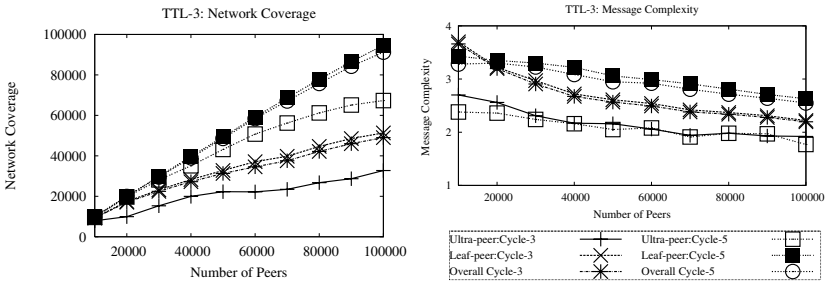
As a result, cycle-5 networks generate a large number of redundant messages which gets reflected in Fig. 7(b).

TTL(3) without QRP

Fig. 8(a) shows that in our simulation the entire cycle-5 networks are covered with $TTL(3)$ search which is almost double of the coverage attained in cycle-3 networks. If any pair of ultra-neighbors of a particular leaf-peer are not more than 6 hops apart from each other (in case of TTL(2) it was 4 hops), then query generates redundant messages. In cycle-5 networks the probability of forming cycle-5 and cycle-6 is very high. As a result, the message complexity of cycle-5 networks becomes higher than cycle-3 networks (Fig. 8(b)). As mentioned earlier, Gnutella (Limewire etc.) uses $TTL(3)$ in dynamic querying only for rare searches [1,2]. Therefore larger network coverage in this case, which increases the query hit probability, is more essential than slight increase of message complexity.

TTL(2) and TTL(3) with QRP

With QRP technique, searching is performed only at the ultra-peer layer, since ultra-peers contain the indices of their children [1,2]. So, the measurement of message complexity at the ultra-peer layer is more appropriate to compare results with Gnutella networks. The ultra-peer layer message complexity is shown in the Fig. 9. Simulation reflects that the message complexity in $TTL(2)$ of cycle-3 networks is almost 2-2.5



(a) Network Coverage (b) Message Complexity

Fig. 8. Network Coverage and message complexity with TTL 3 for cycle-3 and cycle-5 networks

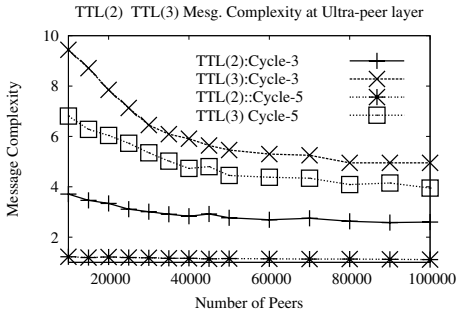


Fig. 9. Message Complexity with TTL(2) and TTL(3) at the Ultra-peer layer

times than that of cycle-5 networks. Even in $TTL(3)$ search cycle-3 networks generate 25% more messages than that of cycle-5 networks. So HPC5 protocol will be more effective in the Gnutella network in the presence of QRP protocol.

6 Conclusion and Future Work

In this paper, we have presented a handshake protocol which is compatible with Gnutella like unstructured two-tier overlay topology. We have shown that the protocol is far more efficient than existing protocols. A relation among TTL, minimum cycle length in the topology and network performance has been observed and proposed. A major fraction of internet bandwidth is occupied by Gnutella-like unstructured popular networks. P2P implementation of the 2^{nd} generation web applications requires a huge internet bandwidth which initiates the optimum utilization of bandwidth. In this regard our protocol can be instrumental in improving the scalability of P2P networks.

In our future work we want to increase query hits through index table replication for cycle-5 networks. We are also planning to work on some design issues related to hit ratio, performance and bandwidth of the cycle-5 networks.

References

1. Gnutella and limewire, www.limewire.org
2. Gnutella protocol specification 0.6, http://rfc-gnutella.sourceforge.net
3. Gnutella, www.gnutellaforums.com
4. Gwebcache system, www.gnucleus.com
5. Karbhari, P., Ammar, M.H., Dhamdhere, A., Raj, H., Riley, G.F., Zegura, E.W.: Bootstrapping in gnutella: A measurement study. In: Barakat, C., Pratt, I. (eds.) PAM 2004. LNCS, vol. 3015, pp. 22–32. Springer, Heidelberg (2004)
6. Liu, Liu, Xiao, Ni, Zhang: Location-aware topology matching in P2P systems. In: INFOCOM: The Conference on Computer Communications, joint conference of the IEEE Computer and Communications Societies (2004)
7. Liu, Xiao, Liu, Ni, Zhang: Location awareness in unstructured peer-to-peer systems. IEEETPDS: IEEE Transactions on Parallel and Distributed Systems 16 (2005)
8. Lua, K., Crowcroft, J., Pias, M., Sharma, R., Lim, S.: A survey and comparison of peer-to-peer overlay network schemes. Communications Surveys & Tutorials, 72–93 (2005)
9. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proceedings of the 2002 International Conference on Supercomputing (16th ICS 2002), pp. 84–95. ACM, New York (2002)
10. Merugu, S., Srinivasan, S., Zegura, E.W.: Adding structure to unstructured peer-to-peer networks: The role of overlay topology. In: Stiller, B., Carle, G., Karsten, M., Reichl, P. (eds.) NGC 2003 and ICQT 2003. LNCS, vol. 2816, pp. 83–94. Springer, Heidelberg (2003)
11. Papadakis, C., Fragopoulou, P., Athanasopoulos, E., Dikaiakos, M.D., Labrinidis, A., Markatos, E.: A feedback-based approach to reduce duplicate messages in unstructured peer-to-peer networks. In: Integrated Research in GRID Computing (February 2007)
12. Saroiu, S., Gummadi, P.K., Gribble, S.D.: A measurement study of peer-to-peer file sharing systems. Technical report, July 23 (2002)
13. Stutzbach, D., Rejaie, R.: Capturing accurate snapshots of the gnutella network. IEEE INFOCOM, 2825–2830 (2005)
14. Stutzbach, D., Rejaie, R., Sen, S.: Characterizing unstructured overlay topologies in modern p2p file-sharing systems. In: Internet Measurment Conference, pp. 49–62. USENIX Association (2005)
15. Zhenzhou, Z., Panos, K., Spiridon, B.: Dcmp: A distributed cycle minimization protocol for peer-to-peer networks. IEEE Transactions on Parallel and Distributed Systems 19, 363–377 (2008)