# Towards Location-aware Topology in both Unstructured and Structured P2P Systems

Tongqing Qiu, Guihai Chen, Mao Ye
State Key Lab of Novel Software
Nanjing University

Edward Chan
Department of Computer Science
City University of Hong Kong

Ben Y. Zhao
Department of Computer Science
UC Santa Barbara

## Abstract

*A self-organizing peer-to-peer system is built upon an application level overlay, whose topology is independent of underlying physical network. A well-routed message path in such systems may result in a long delay and excessive traffic due to the mismatch between logical and physical networks. In order to solve this problem, we present a family of Peer-exchange Routing Optimization Protocols (PROP) to reconstruct the overlay. It includes two policies: PROP-G for generic condition and PROP-O for optimized one. Both theoretical analysis and simulation experiments show that these two protocols greatly reduce the average latency of the overlay and achieve a location-aware topology with low overhead. Their overall performance can be further improved if combined with other recent approaches. Specifically, PROP-G can be easily applied to both structured and unstructured systems without the loss of their primary characteristics, such as efficient routing and anonymity. PROP-O, on the other hand, is more efficient, especially in a heterogeneous environment where nodes have different processing capabilities.*

## 1 Introduction

Peer-to-Peer (P2P) systems are massively distributed computing systems in which peers (nodes) communicate directly with one another to distribute tasks, exchange information, or share resources. There are currently several P2P systems in operation and many more are under development. Gnutella [1] and Kazaa [2], which are often referred to as the first generation P2P file sharing systems, construct the unstructured overlay without rigid constraints for search and placement of files. They use a decentralized file lookup scheme. Requests for files are flooded with a certain scope. However, there is no guarantee of finding an existing file within a bounded number of hops. Chord [20], Pastry [19] and Tapestry [23] are examples of the second generation of peer-to-peer systems. These systems can be viewed as providing a scalable, fault-tolerant distributed hash table (DHT). Any data item based on a unique identification can be located within a bounded number of hops using a small per-node routing table. Unstructured P2P systems are widely used due to their simplicity; but structured systems can be more efficient. Consequently, these two models coexist and in some sense complement each other [3].

All P2P systems are built upon application-level overlays, the topology of which is independent of the underlying physical network. In unstructured systems, a new node randomly chooses some existing nodes of the systems as its logical neighbors; while in structured ones, a new node will get an identification by certain hash function and construct connections with other nodes based on specific rules of the DHT. As a result, the neighborhood of two nodes on the top of overlay does not inherently reflect proximity in the physical network, due to an arbitrary organization or the hash-based property. A well-routed message path in an overlay network with a small number of logical hops may lead to a long delay. The mismatch problem between the overlay and physical network is a major obstacle in building an effective large-scale overlay network.

In this paper, we propose a family of **P**eer-exchange **R**outing **O**ptimizing **P**rotocols (PROP) to handle the mismatch in peer-to-peer systems. It includes two relevant policies: PROP-G (generic) and PROP-O (optimized). They both adaptively adjust the connections of the overlay, and efficiently reduce the average logical link latency of the whole system. Combining them with other recent mechanisms will further improve their performance. Moreover, they are adaptive to dynamic changes in the system. PROP-

G, to the best of our knowledge, is the first scheme that can be deployed effortlessly on both unstructured and structured P2P systems, while preserving the logical topology of overlay at the same time. PROP-O, on the other hand, is more efficient, especially in a heterogeneous environment where nodes have different processing capabilities.

The rest of paper is organized as follows. In Section 2, we review related work. Section 3 describes the design of PROP. In Section 4, we evaluate the effectiveness of our design analytically. The methodology and results of simulation experiments are presented in Section 5. Finally, we conclude the paper in Section 6.

## 2 Related Work

The issue of mismatch between physical and logical networks in P2P systems has been the focus of intensive research in recent years. A location-aware topology matching (LTM) technique [10] is proposed for unstructured P2P systems. In LTM, each peer issues a detector in a small region so that the peers receiving the detector can record the relevant delay information. Based on the information, a receiver can detect and cut most of the inefficient and redundant logical links and add closer nodes as its direct neighbors. LTM is a typical method which is only applicable for Gnutella-like overlay networks where each peer can freely cut and add connections. Moreover, free modification of connections, to some extent, impairs the natural feature of self-organizing overlay where powerful, reliable nodes always provide more services and inherently have more connections [18]. Other methods for unstructured systems like [11] and [12] share similar features with LTM and will not be discussed in detail due to space limitation.

Regarding structured P2P systems, most solutions fall into three broad categories [8] [15]:Proximity Neighbor Selection (PNS) [4],Proximity Route Selection (PRS) and Proximity Identifier Selection (PIS)[13].

However, all of these approaches have a common limitation: *protocol-dependence*. For example, the entries in routing table are deterministic in systems like Chord or CAN, where the PNS scheme cannot be applied directly. Similarly, PRS also has the requirement that there must be more than one choice for the next hop. Topologically-aware CAN, which ensures that nodes which are close in the network topology are close in the node ID space, is only suitable for systems like CAN [21], where the similarity of node IDs means less hops in routing. In short, recent methods based on DHT cannot be applied to other variants of the DHT protocols, not to mention other unstructured P2P systems.

Recently, some researchers focus on the configuration of AS or ISP level [9] [7] . Although this kind of central or cluster-like management can improve the efficiency of the system, it is more related to the deployment of different nodes instead of the deployment of end systems. Moreover, such control is impractical in loosely organized peer-to-peer systems.

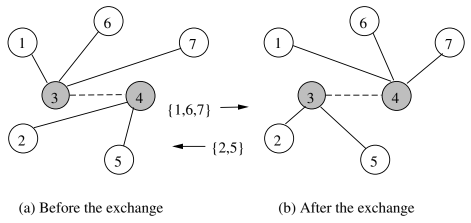


**Figure 1. PROP-G, exchange all neighbors**

## 3 Design Description

### 3.1 PROP-G and PROP-O

In order to satisfy the above requirements, we use "peer-exchange" as the basic operation of our scheme. Generally speaking, peer-exchange means a series of exchanges of some neighbors between two peers, and one exchange can be viewed as a pair of cut-add operations.

A simple and direct way of peer-exchange, called PROP-G, is to exchange *all* neighbors of the two nodes. Figure 1 shows an example of PROP-G, where node 3 and 4 exchange their neighbor sets ($\{1, 6, 7\}$ and $\{2, 5\}$). This can be viewed as exchanging their "position" in the overlay network. Intuitively, the topology of overlay is not affected by the PROP-G operation. That is why we call it a *generic* method, which will be proved in Section 4.

Another way for peer-exchange, called PROP-O, is to *selectively* choose neighbors for exchange. Figure 2 illustrates the process: nodes 3 and 4 exchange equal number ($m = 2$) of neighbors. Note that exchanged neighbors should never lie on the path of nodes 3 and 4, which ensures that nodes 3 and 4 will still be connected after the exchange. The primary reason that we exchange equal number of connections instead of an arbitrary number is to ensure the degree of each node remains the same after the exchange, so that the topology can maintain its essential features. The effectiveness and characteristics of both PROP-G and PROP-O will be illustrated by theoretical analysis in Section 4 and validated by simulations in Section 5.

A traditional way to accomplish topology optimization is to let each source node select one nearest node in the candidate list and establish the connection with it. This "selfish" method, in our opinion, is beneficial to the source node itself but is not always beneficial to (or in some case may actually detracts from) system-wide optimization. Our approach is to utilize the collaboration of two peers, say $u$ and $v$, to

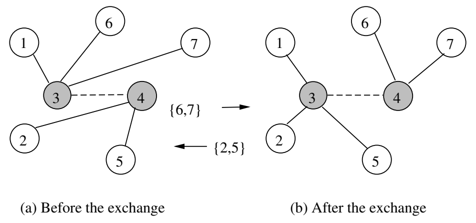


**Figure 2. PROP-O, exchange $m$ neighbors, where $m = 2$**

discover potential opportunities to optimize their neighborhood environments, and then perform the exchange operation. In this way, reconfiguration of the overlay will improve overall system performance and avoid many, if not all, of the potential conflicts and pitfalls in "peer competition".

## 3.2 Description of basic method

Assuming that there is a potential exchange between nodes $u$ and $v$, node $u$ is the *counterpart* of $v$, and vice versa. $d(u, v)$ means the delay (latency) between nodes $u$ and $v$. $t_0$ represents the time before an exchange, while $t_1$ represents the hypothetical time when the potential exchange really occurs. The neighbor set of node $u$ is formally defined as follows:

$$N(u) = \{i | i \in V \wedge (ui \in E \vee iu \in E)\} \tag{1}$$

There are two separate procedures here: the warm-up and the maintenance phases. Having joined the system based on a random or DHT based assignment, a new node $u$ will start the warm-up procedure. It begins probing its neighbors and collecting the initial latency information $\sum_{i \in N_{t_0}(u)} d(u, i)$. Then it will periodically contact a random node $v$ which is $nhops$ hops away at each time interval of $timer$. A priority queue $neighborQ$ is used to choose nodes $s$ for the first hop of random walk. The use of priority is to ensure that "active" nodes will be probed first, which is useful in the maintenance procedure. In the beginning, it is initialized with a random sequence of node $u$'s neighbors, so each neighbor has an equal probability to be probed. A message containing the source IP address, the source timestamp and a small TTL value $nhops$ is used to realize the random contact. Any node that receives this message will add an identifier like the IP address into the message [1], decrement the TTL field by 1 and forward it. The target node $v$ is located when TTL value becomes zero. Then nodes $u$ and $v$ selectively exchange their address lists and initial latency information, with arbitrary $m$ neighbors for PROP-O and all neighbors for PROP-G. So the value of $m$ is no more than the minimum degree of overlay $\delta(G)$. We choose $m = \delta(G)$ by default. After collecting the new latency information $\sum_{i \in N_{t_1}(u)} d(u, i)$ and $\sum_{i \in N_{t_1}(v)} d(v, i)$ by probing new neighbors (hypothetical neighbors when the potential exchange occurs), they exchange information and calculate the variable $Var$ independently, as in the following equation:

$$\begin{aligned} Var = \quad & \textstyle\sum_{i \in N_{t_0}(u)} d(u, i) + \sum_{i \in N_{t_0}(v)} d(v, i) \\ & - \textstyle\sum_{i \in N_{t_1}(u)} d(u, i) - \sum_{i \in N_{t_1}(v)} d(v, i) \end{aligned} \tag{2}$$

If $Var \leqslant$ MIN_VAR, it means that the exchange cannot gain any benefit, and therefore, no subsequent operation will be performed. Otherwise, nodes $u$ and $v$ will do the peer-exchange operation as follows: they rewrite corresponding routing entries and even exchange node identifiers (for PROP-G in DHT systems) respectively. Both of them cache the address of their counterparts so that the lookups in progress during peer-exchange can be forwarded correctly. Moreover, both of them will notify their neighbors to change the routing tables and recalculate the initialized sums.

If the routing tables are extended to record both successor nodes and predecessor ones (bidirectional connections in other words), the notification can be realized directly. We have at least two reasons for this simplification. First, most structured systems selectively record several predecessor nodes in order to improve fault resilience. The size of the extended routing table is at most twice as large as the size of the original one. There is even no increase in some symmetrical systems like Gnutella or CAN. More importantly, even if there is no such extension, notifications can still be implemented by using the underlying mechanisms just as what happens when peers arrive or depart, although it leads to more complicated reconstruction operations. The warm up procedure will last for MAX_INIT_TRIAL times; simulations in a later section shows this number to be less than ten.

Next node $u$ will enter the maintenance phase, which differs from the initialization procedure in two ways. First, the selection of node $s$ will depend on the result of peer-exchange trials. If an exchange occurs, which implies that selection of main "direction" is successful, node $s$ will merely decrease the priority number by a small number like 1 so that it could be chosen in near future. Otherwise it will be replaced at the tail of $neighborQ$, waiting for the next probing cycle. Another difference lies in the modification of $timer$ based on a Markov chain model [16]: $Timer$ will be doubled after a failed peer-exchange attempt, and reset to INIT_TIMER after a successful one; if $Timer \geq$

[1]To avoid repetitive forwarding and exchange neighbors which stand on the random walk path.

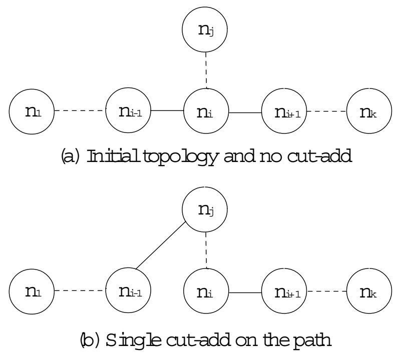

(a) Initial topology and no cut-add
(b) Single cut-add on the path

**Figure 3. A generic path where node $n_j$ is off the path**

MAX_TIMER, it will also be set as INIT_TIMER. Here MAX_TIMER $= 2^5 \times$INIT_TIMER, so there are at most five times of suspending (half of MAX_INIT_TRIAL). Similarly, in order to handle departures and sudden failures gracefully, the value of $timer$ will be reset to INIT_TIMER and the new neighbors will be added into the front of $neighborQ$ with a maximum priority value, so that these peers can be probed earlier during the maintenance procedure.

## 4 Theoretical Analysis

### 4.1 Characteristics of Peer-exchange

The basic requirement of overlay reconstruction, as mentioned above, is that the change of connections should never lead to a graph partition.

**Theorem 1 (connectivity persistence)** *Let G be an undirected connected graph, and let $G'$ be the graph that is derived from G by applying an exchange operation in PROP-G or PROP-O. G' is an undirected connected graph.*

**Theorem 1** *Both PROP-G and PROP-O consist of an exchange of several neighbor nodes. This exchange can be performed by a series of* cut-add *operations of two nodes: cut one connection of one's neighbor and add another one for its counterpart. Hence, we can restrict our attention to a single cut-add operation. It follows from induction that if the graph remains connected after a single cut-add operation, it remains connected after exchange. Let $P =< n_1, ..., n_k >$ be an arbitrary sequence of nodes that forms a path in G. Node $n_i$ will remove a connection between one of its neighbor, $n_s$, and itself. Then a connection between $n_j$ and $n_s$ is established.*

- *Case 1: if $n_i$ lies off the path. This means that while there may be nodes on the path whose edges change, the changed edges connect to the nodes implementing cut-add. Hence, no edges that form the path are changed, so the path remains after the cut-add is complete.*
- *Case 2: $n_i$ lies on, and $n_j$ lies off the path, as in figure 3(a). Since nodes $n_i$ and $n_j$ are connected both before and after a cut-add (As mentioned in Section 3, exchanged neighbors should never lie on the probing path between nodes $n_i$ and $n_j$, which ensures that two nodes will be still connected after the exchange), two possible scenarios occur: $n_i$ cut no edges or one edge on the path. As can be seen in Figure 3(b), a path between $n_1$ and $n_k$ remains after the cut-add where $n_s = n_{i-1}$.*
- *Case 3: Both nodes $n_i$ and $n_j$ lie on the path, as in figure 4(a). There are two similar scenarios as in case 2: $n_i$ cut no edges or one edge on the path. As can be seen in Figure 4(b), a path between $n_1$ and $n_k$ remains after the cut-add where $n_s = n_{i-1}$.*

*So the graph is still connected after a single cut-add operation. We can further conclude that $G'$ is connected after a series of cut-add operations for both PROP-G and PROP-O.*

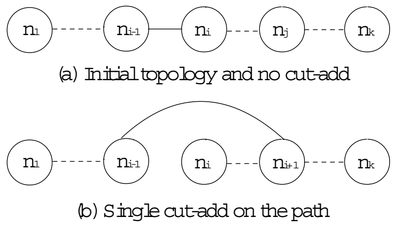

(a) Initial topology and no cut-add
(b) Single cut-add on the path

**Figure 4. A generic path where node $n_j$ is on the path**

The above theorem ensures that there is no graph partition after a peer-exchange operation. Moreover, it is trivial to proof that PROP-O preserves the original degrees of each node, so it never breaks the natural Power-law-like characteristic (i.e. powerful nodes own more connections) of unstructured P2P systems.

**Theorem 2 (isomorphic characteristic)** *Let graph $G(V, E)$ denote the network overlay, and let $G'(V', E')$ be the graph that is derived from G by applying an arbitrary sequence of PROP-G exchange operations. $G'$ is isomorphic to graph G, i.e. $G \cong G'$.*

**Theorem 2** *It follows from induction that if the derived graph $G'' \cong G$ after a single exchange operation of PROP-G, then $G' \cong G$ based on the transitivity of isomorphism. Without loss of generality, we assume nodes $u$ and $v$ do a single exchange during the period $t_0$ to $t_1$. We try to find a bijection $\varphi : V \to V'$ with $xy \in E \Leftrightarrow \varphi(x)\varphi(y) \in E'$ for all $x, y \in V$. $V_1$ is used to denote the set of un-exchanged nodes and $V_2$ presents the set of exchanged ones. A mapping $\varphi$ between $E$ and $E'$ is constructed as follows:*

- *For $\forall x, y \in V_1$, $xy \in E \Leftrightarrow xy \in E'$*
- *For $\forall x \in V_1$, $y = u$, $xy \in E \Leftrightarrow xv \in E'$, $yx \in E \Leftrightarrow vx \in E'$. Similarly, for $\forall x \in V_1$, $y = v$, $xy \in E \Leftrightarrow xu \in E'$, $yx \in E \Leftrightarrow ux \in E'$.*
- *For $\forall x \in V_2$, $y \in V_1$, the proof is similar to the above one.*
- *For $\forall x, y \in V_2$, $xy \in E \Leftrightarrow yx \in E'$*

*Observing the constructed mapping, it is easy to conclude that G is isomorphic to $G'$.*

Theorem 2 illustrates that PROP-G not only keeps the connectivity of logical network but also maintains the overlay topology. Therefore, as an auxiliary method, it is suitable for different topologies: ring, hypercube, tree, and so on. Moreover, the change of positions using PROP-G is not arbitrary. As an example in DHT systems, instead of regenerating its identifier, each node is only allowed to get old identifiers of other nodes. It preserves anonymity provides a certain measure of security.

However, it does not mean that PROP-G can be used in *all* P2P systems. In fact, there are several classes of P2P applications where neighbor relationships cannot be set arbitrarily. For instance, in some systems where each node has a certificate which binds its identifier to a public key for security reasons, it seems that PROP-G which exchanges node ID may not be feasible.

## 4.2 Effectiveness of the Peer-exchange Mechanism

We use the following definitions to explain the effectiveness of the peer-exchange mechanism. *Stretch* is defined as the ratio of the average logical link latency over the average physical link latency. It is a common parameter to quantify the degree to which the physical and logical topology matches. *Average latency (AL)* is a basic parameter to quantify the property of a network. If there are $n$ nodes in a network, then[2]

$$AL = (\sum_{i \in V} \sum_{j \in V} d(i,j))/n^2 \tag{3}$$

[2]We assume the latency between a node and itself is zero.

Given that the physical network is usually static, only the average logical link latency affects stretch. Furthermore, supposing that the number of nodes is constant during the period $t_0$ to $t_1$, the *accumulated latency* ($L_{t_i}$) can be analyzed as follows. The next two equations show the accumulated latency at $t_0$ and $t_1$:

$$L_{t_0} = C + \sum_{i \in N_{t_0}(u)} \alpha_i d(u,i) + \sum_{i \in N_{t_0}(v)} \beta_i d(v,i) \tag{4}$$

$$L_{t_1} = C + \sum_{i \in N_{t_1}(u)} \gamma_i d(u,i) + \sum_{i \in N_{t_1}(v)} \delta_i d(v,i) \tag{5}$$

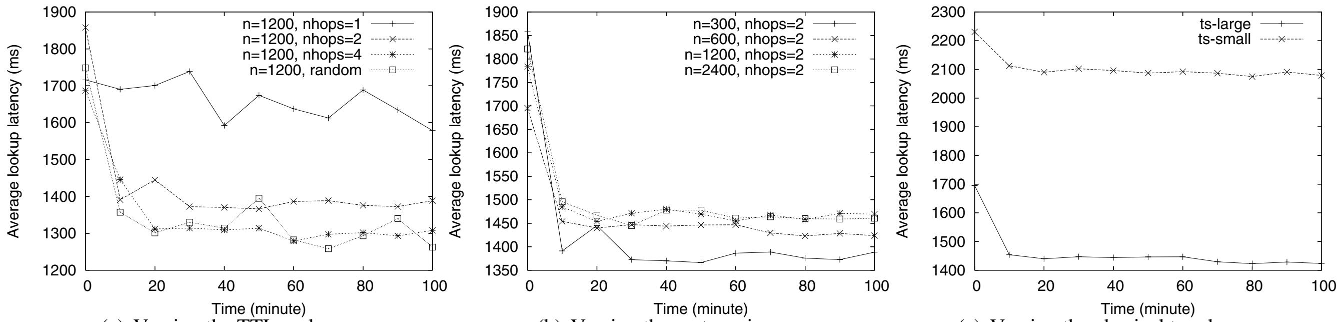
In equations 4 and 5, $C$ represents the invariable part during the period $t_0$ to $t_1$. The coefficients of the summations $\alpha_i, \beta_i, \gamma_i, \delta_i$ represent the visited times of each neighbor-link when calculating $AL$. The equation $\alpha_i \approx \gamma_i \approx \beta_i \approx \delta_i$ is valid by assuming that each link has the same probability to be visited. To calculate the variation by $(4) - (5)$, it is easy to find that if $Var > 0$ then $L_{t_0} > L_{t_1}$, which implies that a peer-exchange reduces stretch. So in our simulation part, we will set $\mathsf{MIN\_VAR} = 0$.

We notice that the latency for other peers to reach a certain object on exchanged nodes might have been increased. For example, assume peer $u$ and $v$ exchange their identifiers and keep pointers to each other. Peer $i$ was originally a neighbor of $v$, but is now a neighbor of $u$. If it tries to retrieve an object stored at $v$, it takes it two hops instead of one now. However, according to the above analysis, the average latency of all queries issued from all nodes to $u$ and $v$ will be decreased.

Note that this is only an approximate analysis. In fact, when the positions of the nodes change, the visited times of each node varies accordingly. This explains why not all exchange operations can reduce the average latency, as shown in the simulation experiments.

## 4.3 Overhead Analysis

Our method improves the overlay topology in two types of cost: the information collection between two cooperative nodes, and the reconstruction of overlay. Both of them are determined by two factors: the number of nodes involving into one potential exchange operation and the number of probing times. For an overlay network with $n$ peers, we use $c$ to denote the average number of neighbors. For each peer, one step of adjustment will involve $(nhop + 2c)$ for PROP-G, and $(nhop + 2m)$ for PROP-O. The overhead of PROP-O is intuitively better than PROP-G especially when $c$ is much larger than $nhop$ and $m$. Our simulation will illustrate that. As for the second factor, we investigate the frequency of probing for each node, $f_p$. In the worst case, when each peer has to probe every time, the frequency will



(a) Varying the TTL scale (b) Varying the system size (c) Varying the physical topology

**Figure 5. Effectiveness of PROP-G in Gnutella-like environment**

be $f_p = 1/\mathsf{INIT\_TIMER}$. In fact, the topology will become stable after a warm-up procedure, and the frequency is very low after that, because we utilize a Markov chain model to exponentially postpone the time of probing. Even when churn occurs, the frequency of probing will reduce quickly after a short period of time. Our simulation experiments regarding performance in a dynamic environment will demonstrate this idea.

## 5 Performance Evaluation

### 5.1 Simulation Methodology

We use the GT-ITM topology generator [22] to generate two different transit-stub models of the physical network. The first topology, *ts-large* has 70 transit domains, 5 transit nodes per transit domain, 3 stub domains attached to each transit node and 2 nodes in each stub domain. The second one, *ts-small*, differs from ts-large in that it has only 11 transit domains, but there are 15 nodes in each sub domain. Intuitively, ts-large has a larger backbone and sparser edge network than ts-small. Except in the experiment of physical topology, we always choose ts-large to represent a situation where the overlay consists of nodes scattered in the entire Internet and only very few nodes from the same edge network join the overlay. We also assign latencies of 5, 20 and 100ms to stub-stub, stub-transit and transit-transit links respectively. Then a number of nodes (default set to 1200), are selected from the physical network as overlay nodes.

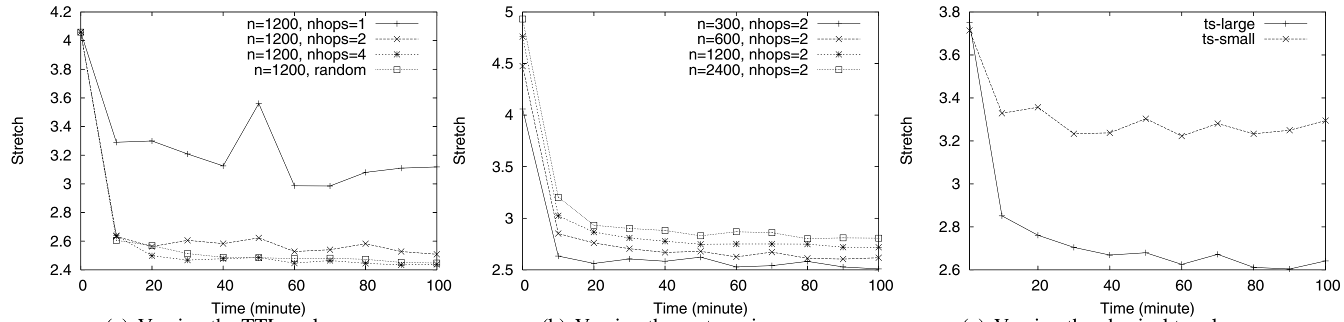
The simulation involves three P2P infrastructures, Chord, CAN and Gnutella; and different improving methods based on them like PNS, PIS and LTM. MIN_VAR is determined by the analysis in Section 4.2. The value of MAX_INIT_TRIAL is based on massive experiments. It is difficult to set the value of INIT_TIMER because it is related to the dynamics of the system. In our evaluation, we simply set it as 1 minute. The choices of other parameters will be discussed in the following subsections.

### 5.2 The Effectiveness of PROP-G

According to the analysis in Section 4, PROP-G is a generic mechanism, which can be used in both unstructured and structured systems. Figure 5 and 6 show its effectiveness in both Gnutella-like and Chord environments. *Stretch* is the metric used to characterize matching degree. As messages are sent by the flooding method in unstructured P2P systems, it is not practical to calculate the latency between each pair of nodes. Therefore, the average lookup latency derived from 10,000 lookup operations is chosen in Gnutella instead. Both stretch and average lookup latency are varied according to time.

Figures 5(a) and 6(a) show the impact of the $TTL$ on stretch in two different systems. There are four typical scenarios as far as probing is concerned. In the main scenario, instead of TTL packets, a random node is selected as the probing target. In other three conditions, TTL value $nhop$ is set to 1, 2, 4 respectively. Neighbors' exchange ($nhop = 1$) is not suitable because it cannot reduce the stretch significantly (which is consistent with our analysis of stability), while other three different ways have nearly the same impact on stretch reduction. Given that random probing is not practical in a distributed system, only when $nhop \geq 2$ can a good performance be attained in a P2P system. In order to minimize the cost, $nhop = 2$ may be a better choice, and it will be used in the following experiments. Figures 5(a) and 6(a) also illustrate that stretch is not reduced all the time, which is consistent with our approximate analysis.

Figures 5(b) and 6(b) demonstrate the impact of system size. The effectiveness of the schemes is slightly reduced as the system size becomes larger. There are at least two reasons. First, when the system has a larger size and PROP-G fixes $nhop$ as 2, the collected information is relatively limited. Second, as we choose the nodes from the same physical network, the overlay is getting closer to the physical topology when it is larger. Fortunately, PROP-G is still effective even when almost all physical nodes are chosen.



(a) Varying the TTL scale (b) Varying the system size (c) Varying the physical topology

**Figure 6. Effectiveness of PROP-G in Chord environment**

The impact of physical topology is presented in Figures 5(c) and 6(c). We have generated two different types of topologies: *ts-large* and *ts-small* by GT-ITM tools, both of which contain about 2400 nodes. It is obvious that the ts-large topology has much better performance. In the ts-large topology, only a few stub domains are attached to transit nodes. As a result the probability that two stub nodes belong to different transit domains is relatively high. In other words, two *far* nodes can execute the exchange operation with a high probability, and this kind of exchange will greatly improve the performance of the system. PROP-G is more efficient in ts-large topology, which is much like the Internet as we mentioned above. Finally, comparing structured and unstructured systems, the average lookup latency in Gnutella fluctuates more markedly. This is because Gnutella owns more random logical connections, and it is harder to find the better candidate nodes to exchange.

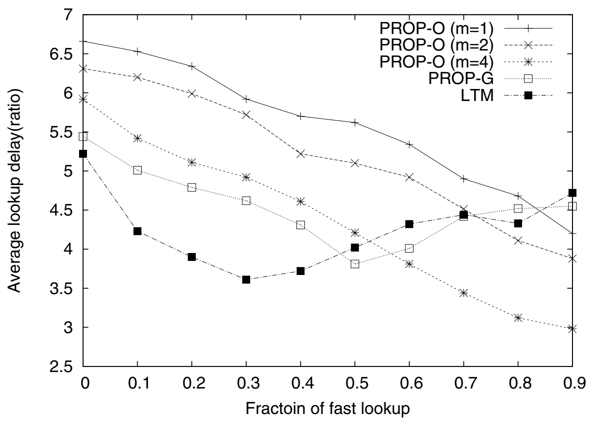


**Figure 7. Average lookup latency for bimodal processing delay distribution, when varying the fraction of fast node lookup**

## 5.3 The Effectiveness of PROP-O

In this section, we will compare PROP-O with PROP-G and LTM under a Gnutella-like environment. $m$ is allowed to vary from 1 to 4, where 4 is the minimum average degree in the system. In order to illustrate the features of PROP-O in a heterogeneous environment, we further introduce node heterogeneity in our simulations. There are many resource factors which result in node heterogeneity, including process speed, storage and bandwidth supported. We merely use processing delay to represent node heterogeneity because we are more interested in lookup latency in this paper. To simulate processing delay, the *bimodal* distribution is used. There are two kinds of nodes - fast and slow. The processing delay of the fast nodes is 10ms, while the delay of the slow ones is 100ms. The fraction of fast nodes is 5% of the total population: the overall setting is similar to that in [5]. Since the total delay is just the sum of the link delay plus processing delay of nodes, the resulting absolute delay will be much larger than the corresponding results for PROP-G. To avoid any confusion, we choose a normalized value instead of real lookup delay which is measured by millisecond.

Figure 7 shows an important feature of PROP-O. In real-life P2P systems, powerful nodes provide much more services than poor ones. Accordingly, the destination of lookup operations will be concentrated on the powerful nodes. We simulate this phenomenon by increasing the fraction of lookups whose destination is a fast node in the *bimodal distribution* environment. When all queries are directed to slow nodes, LTM shows best routing performance. However, when more queries are directed to fast nodes, the delay of both PROP-G and LTM increase. On the other hand, the delay for PROP-O keeps decreasing. We will explain it from the following two perspectives. On the one hand, given that the physical network we construct is "Internet-like", only a few nodes from the same edge network will join the overlay. So the query with largest latency is usually from a slow node to another slow one in different areas. As a result, it is likely that the latency of a query to slow node tends to be larger than the one to fast node. However, if the two slow end-nodes are connected by a number of fast intermediate nodes, maintaining the positions of these fast nodes will have a more significant impact on the performance of the system. Because fast nodes have more connections, it is more likely that they will be located in better positions after a peer-exchange, and PROP-O is able to maintain the connection number of each node so that the fast nodes can keep this kind of priority.

## 6 Conclusion

This paper proposes a family of peer-exchange methods called PROP to solve the mismatching problem in P2P systems. PROP is adaptive scheme which can be easily embedded into most P2P systems without affecting the characteristics of the original systems. Simulation experiments show that PROP is an efficient way to match the physical network. By combining it with other recent methods, the overall performance can be further improved. It is also adaptive to dynamic change of peers. Unlike most previous studies that try to discover a better structure for P2P overlay, we accept the fact that many different structures coexist and our approach is another choice to make P2P systems more efficient.

## Acknowledgements

The work is partly supported by China NSF grants (60573131, 60673154), Jiangsu Provincial NSF grant (BK2005208), and China 973 project (2006CB303004). The Conference Participation is supported by Nokia Bridging the World Program.

## References

[1] http://rfc-gnutella.sourceforge.net.
[2] http://www.kazaa.com.
[3] M. Castro, M. Costa, and A. Rowstr. Peer-to-peer overlays: structured, unstructured, or both? Technical report, MSR-TR-2004-73, 2004.
[4] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting network proximity in peer-to-peer overlay networks. In *MSR-TR-2002-82*, 2002.
[5] S.-G. Chun, B. Y. Zhao, and J. D. Kubiatowicz. Impact of neighbor selection on performance and resilience of structured p2p networks. In *International workshop on P2P Systems(IPTPS'05)*, 2005.
[6] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris. Designing a DHT for low latency and high throughput. In *USENIX Symposium on Networked Systems Design and Implementation*, 2004.
[7] J. Fan and M. H. Ammar. Dynamic topology configuration in service overlay networks: a study of reconfiguration policies. In *Proceedings of INFOCOM 2006*, 2006.
[8] K. Gummadi, R. Gummadiy, S. Gribblez, S. Ratnasamyx, S. Shenker, and I. Stoica. The impact of DHT routing geometry on resilience and proximity. In *Proceedings of ACM SIGCOMM*, 2003.
[9] J. Han, D. Watson, and F. Jahanian. Topology aware overlay networks. In *Proceedings of INFOCOM 2005*, 2005.
[10] Y. Liu, L. Xiao, X. Liu, L. M. Ni, and X. Zhang. Location awareness in unstructured peer-to-peer systems. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 16(2):163–174, Febuary 2005.
[11] Y. Liu, L. Xiao, and L. Ni. Building a scalable bipartite P2P overlay network. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS04)*, 2004.
[12] Y. Liu, Z. Zhuang, L. Xiao, and L. Ni. Distributed approach to solving overlay mismatching problem. In *Proceedings of the 24th International Conference on Distributed Computing Systems(ICDCS'04)*, 2004.
[13] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of INFOCOM 2002*, 2002.
[14] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of IEEE INFOCOM*, 2002.
[15] S. Ratnasamy, S. Shenker, and I. Stoica. Routing algorithms for DHTs: Some open questions. In *1st International workshop on P2P Systems(IPTPS'2)*, 2002.
[16] S. Ren, L. Guo, S. Jiang, and X. Zhang. SAT-Match: A self-adaptive topology matching method to achieve low lookup latency in structured P2P overlay networks. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04)*, 2004.
[17] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In *Proceedings of the USENIX Annual Technical Conference*, 2004.
[18] M. Ripeanu, A. Iamnitchi, and I. Foster. Mapping the gnutella network. *IEEE Internet Computing*, 6(1):50–57, Febuary 2002.
[19] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.
[20] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM*, 2001.
[21] M. Waldvogel and R. Rinaldi. Efficient topology-aware overlay network. In *Proceedings of HotNets-I*, 2002.
[22] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of INFOCOM*, 1996.
[23] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, Jan 2004.