# LOCALITY-AWARE AND LOW MAINTENANCE OVERHEAD P2P SYSTEM

Chi-Jen Wu, De-Kai Liu and Ren-Hung Hwang
Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan
{cjwu,dkliu,rhhwang}@exodus.cs.ccu.edu.tw

**ABSTRACT**
This work describes a novel locality-aware structured peer-to-peer (P2P) overlay network, referred to as LAPTOP. One important aspect of constructing a structured P2P network is how to reduce the maintenance overheads while maintaining efficient routing and network locality. LAPTOP organizes nodes into a tree-based overlay network in a self-organizing manner and builds the routing table by taking network locality into account. The tree-based overlay network enables LAPTOP to utilize the parental relation to reduce the overheads of overlay maintenance. Moreover, LAPTOP adopts proximity neighbor selection (PNS) in the routing table construction to achieve low routing latency. Mathematical analysis and simulations are conducted to evaluate the efficiency and scalability of LAPTOP. The mathematical analysis performed here demonstrates that the routing table size of a node is bounded by $\log_d N \times (d-1)$, the routing path length is bounded by $\log_d N$, and the joining and leaving overhead is bounded by $d \log_d N$, where N denotes the number of nodes in the system and d represents the maximum degree of each node on the overlay tree. Our simulation results show that the average path stretch and routing path length are just 1.48 and 3.8, respectively, in a system of 100 000 nodes with a maximum degree of 16 . Compared to previous research, the proposed LAPTOP approach reduces overlay maintenance overheads while still providing efficient routing.

**KEY WORDS**
Peer-to-Peer, Overlay Network

## 1. Introduction

The demand for sharing the computing power of billions of personal computers on the Internet is increasing rapidly as the computing capability of personal computers becomes more and more powerful, making the peer-to-peer (P2P) network model preferable to the traditional client/server model for Internet applications. P2P networks have been used in various distributed applications.

The literature roughly divides P2P networks into three categories: centralized, decentralized and unstructured, and decentralized but structured, based on the underlying technology [1]. For example, the Napster file sharing system [2] adopts a centralized approach that enables nodes to search for interesting files by issuing queries to the central directory server. Meanwhile, the Gnutella file sharing system [3] applies a decentralized and unstructured approach to let nodes identify the location of a file of interest by flooding queries to all the nodes in the system. In contrast, the Chord [4] system uses a decentralized but structured approach to provide a lookup service framework for P2P networks. The whole Chord system can be viewed as a distributed hash table. Data items and nodes in Chord are associated with identifiers, and identifiers for data items and nodes are so-called keys and node ids, respectively. Each node is responsible for storing a range of keys and also for being aware of some other nodes.

This study believes that the decentralized but structured P2P network is most suitable for large-scale distributed applications, since it searches for the node that keeps interesting data in a scalable and efficient manner and can be widely deployed as an application level routing service for various P2P applications. However, two critical issues must be overcomed when designing decentralized but structured P2P networks. The first issue is the high complexity of locality-aware overlay construction. To improve the routing efficiency of a decentralized but structured P2P network, several approaches [5-7] have been proposed to build overlay networks in a locality-aware manner. Although the routing performance of these locality-aware overlay networks is competitive with that of underlying IP routing, these approaches require either pre-chosen landmarks or complete BGP routing table information, resulting in high complexity overlay construction. The second issue is high routing table maintenance complexity. In a P2P network, the network topology changes frequently since nodes can join and leave the overlay network arbitrarily. To avoid routing performance degradation because of topology changes, P2P networks in the literature always enable each node to periodically invoke a routing table maintenance procedure to maintain the accuracy of all of their routing table entries. Nevertheless, to maintain accurate routing table entries, the number of nodes that a node needs to contact during the routing table maintenance procedure may increase rapidly as the system size grows, resulting in high maintenance cost.

This work presents LAPTOP, a novel lightweight, locality-aware P2P overlay network with efficient, scalable, robust lookup and routing schemes. LAPTOP organizes nodes into a tree-based overlay network in a self-organizing way and uses a heuristic proximity neighbor selection (PNS) scheme to take network proximity into consideration. Furthermore, LAPTOP reduces routing overhead by having a node periodically

update the routing information of its parent node. This approach enables LAPTOP to route efficiently with low routing table maintenance overhead as compared to previous approaches. The maintenance overhead of the overlay network are also relatively low since a LAPTOP node only checks its connectivity to its parent node periodically, while most other P2P systems need to check the connectivity with a set of nodes. With these novel features, the routing path length in a LAPTOP overlay network is bounded by $O(\log_d N)$ hops and the routing table size of the node is bounded by $\log_d N \times (d\text{-}1)$, where $N$ denotes the number of nodes and $d$ represents the maximum degree of each node.

The rest of this paper is organized as follows. Section 2 then presents the overview of LAPTOP and describes how LAPTOP solves the two challenges mentioned above. The mathematical analysis and simulation results for evaluating the performance of LAPTOP then are presented in sections 3 and 4. Finally, section 5 offers conclusions and future research directions.

## 2. Design of LAPTOP

LAPTOP utilizes the basic principles of the PNS approach to construct a hierarchical overlay network. However, the overlay network in LAPTOP is formed in a self-organizing manner. Specifically, LAPTOP enables a newcomer to randomly select a LAPTOP node as its parent node to obtain a logic address and to form a tree-based overlay network. LAPTOP also employs a hierarchical addressing scheme to facilitate the PNS mechanism process. Thus, unlike other approaches that need to periodically exchange a large number of messages among nodes, a LAPTOP node only exchanges information with its parent node periodically.

### 2.1 LAPTOP Operations

The overlay topology formed by LAPTOP is a tree, but each node builds a routing table to aid routing process and strengthen system robustness. Each node in LAPTOP has a node address which is used to indicate the logical location of the node in the overlay, and is represented by a sequence of dotted decimal numbers, resembling the IP address format but with a variable length. The address of LAPTOP node is formally defined as follows:

**Definition 1 (Node address):** Each node has a unique address represented by a dotted decimal number, in which each decimal number ranges from 1 to ***d***, where ***d*** denotes the maximum degree of a node in LAPTOP. The LAPTOP overlay network assumes that a root node is a preparation node that is initially assigned the address, "1" and should never fail.

*1) Host joining:* When a new node joins the LAPTOP overlay network it will invoke the **Join procedure** to locate its parent location in the overlay network and obtain its node address. As in mOverlay [14], the approach described here assumes the existence of a well-known bootstrap node, which can be a single machine or a set of machines. The bootstrap node maintains a global node information cache and will guide a new node to find its parent node. Therefore, the first step of the join procedure is to contact the bootstrap node to obtain information. In LAPTOP, the start node, which refers to the potential parent node of the new node, is randomly selected from the existing nodes. The new node selects the start node as its parent node if the degree of the start node is not full; otherwise, the start node will randomly select one of its child nodes as the new start node for the new node. The procedure is repeated until a parent node is found. The parent node then creates a logic address by appending a unique number to its own address for the new node. The advantage of this join procedure is that the constructed overlay tree is likely to be a balanced tree due to the random selection of the start node.

Figure 1 shows an example of the join procedure. The maximum degree of a node is 3. The new node first contacts with the bootstrap node and gets a randomly selected start node whose address is 1.1. The new node then contacts with node 1.1, but is redirected to node 1.1.1 since the degree of node 1.1 is full. Thus, the new node becomes the child node of node 1.1.1 and is assigned an address of 1.1.1.1.

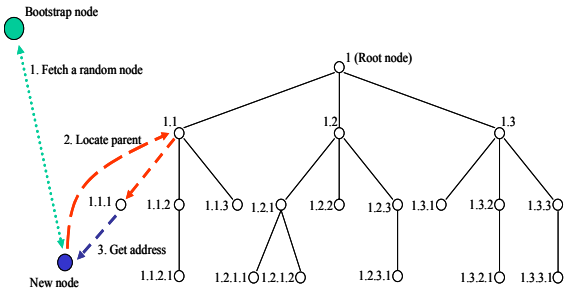


Fig. 1: Example of the Join procedure

**Node address 1.2.3.1**

| default routing set | |
|---|---|
| 1.2 | 140.123.105.14 |
| 1.2.3 | 10.123.195.220 |

| Routing table | | |
|---|---|---|
| 1.1.x | 1.1.3 | 120.190.10.3 |
| 1.3.x | 1.3.2.1 | 120.140.190.1 |
| 1.2.1.x | 1.2.1.1 | 192.83.193.39 |
| 1.2.2.x | 1.2.2 | 110.19.100.17 |
| 1.2.3.2.x | | |
| 1.2.3.3.x | | |

Fig. 2: Routing information at node "1.2.3.1"

*2) Message routing:* LAPTOP takes advantage of its hierarchical address to achieve efficient message routing. LAPTOP can route message in less than $\lceil \log_d N \rceil$ steps if the overlay tree is balanced. The longest prefix matching scheme is adopted as the default routing scheme for LAPTOP.

In LAPTOP, each node maintains a *default routing set* and a *routing table* for message forwarding. Each entry in these tables comprises the logic address of a node, as well

as its public IP address. The default routing set comprises the ancestor and children node information. The ancestor information is retrieved during the joining process, while the child node information is obtained when it accepts new child nodes.

In the routing table, each routing entry records the information of the relay node to a sub-tree, which is considered to be the "nearest" node to that sub-tree according to some proximity metric. Intuitively, if a message is to be routed from the current node to a node of that sub-tree, the node in the routing table of the corresponding entry is the best candidate for forwarding the message to. For a node that is at the $l$th level of the overlay, its routing table consists of $l * (d\text{-}1)$ entries. Restated, the routing table records the nearest node of each sub-tree from levels 1 to $l$. Notably, the overlay tree may not be not a complete tree, some of the routing entries may be empty. Figure 2 shows an example of these two routing tables of node 1.2.3.1 of Fig. 1. The default routing set which records the information of its parent and grandparent nodes. Since the maximum degree is 3 and node 1.2.3.1 is located at the 3rd level, the routing table contains six entries. Two of them are empty: entry 1.2.3.2.x and entry 1.2.3.3.x, where "x" represents a wildcard. Meanwhile, the routing entry of "1.1.x" indicates that node 1.1.3 is the nearest node to the sub-tree rooted at node 1.1, namely, all nodes that share the prefix "1.1" in their addresses. The routing table is built based on a heuristic PNS mechanism which is described later.

A new node can easily initialize its routing tables. After the new node contacts with its parent and obtains its node address, it can ask the parent to supply routing information from the routing tables of the parent as its initial routing information. Since the new node is one level below its parent node in the overlay, the entries of the lowest level initially are set to empty.

Laptop adopts the PNS mechanism to exploit locality. The major problem of PNS mechanism is its high maintenance overhead, which includes initial set up overhead and periodic update overhead. LAPTOP attempts to reduce these overhead by avoiding periodic update overhead. In LAPTOP, each node keeps the node address information, including logic address and public IP address, of all its children nodes as well as grandchildren nodes, i.e., two generations of descendents comprise a "descendant cache". This cache is a small cache with size $d^2$ records, and is updated only when the descendents are changed. When a node copies the routing table from its parent, it can either use it to update the entries in the routing table or perform a heuristic PNS mechanism. In the later case, for each routing entry, the new node contacts the node in the routing entry to measure the proximity as well as fetch node addresses in its descendant cache. The new node then randomly chooses a certain number of nodes, say $K$, to measure the proximity to these nodes according to some proximity metric. If the least proximity is less than that of the current node in the routing entry, the entry is then updated using the node that has the least proximity. Since a LAPTOP node is randomly placed on the overlay tree, this scheme resembles to randomly select $K$ nodes to perform PNS mechanism. A node can perform the procedure when it is not busy or to keep most up to date information for some frequently referred entries. A similar approach is adopted to fill in empty entries, but needs additional information, i.e., maintenance set, which will be described later.

*3) Overlay and routing table maintenance:* To maintain the network locality property of the routing table and reduce the overlay maintenance cost, each node periodically sends a Heartbeat message to its parent to probe the parent reliability. That is, a LAPTOP node only probes one node, namely, its parent node, periodically rather than probing each entry in its routing table. Consequently, the maintenance overhead is significantly reduced.

To achieve both good network locality property and low cost, this work proposes a heuristic PSN mechanism that assumes each node maintains a descendant cache and a "*maintenance set*". The maintenance set is similar to the routing table, but the node information recorded in each entry is a randomly selected node in the corresponding sub-tree. Accordingly, the size of the maintenance set for a node at the $l$th level of the overlay is $l * (d\text{-}1)$. The maintenance set fulfills two purposes. First, the information in the maintenance set can be used to fill empty entries in the routing table. Second, when the node of a routing entry fails, the node in the maintenance set can be used as a temporary substitute.

The maintenance set is maintained as follows. When a node sends the Heartbeat message to its parent, it will also include the address information of an active node randomly selected from its descendant cache. The parent node then sends back an ACK message to this child node with the active node information collected from all its children nodes, along with the maintenance set it received from its parent node. The difference between the routing table and the maintenance set is that information in the former table may be stale, but considers network proximity, while the later table is fresher but does not exploit network proximity. The advantage of having the maintenance set is that it frees a node from the need to update its routing table periodically which involves expensive proximity measurements but still can cope with the stale routing information problem adaptively.

With the maintenance set, a node can fill in the empty routing entries either by using the information in the maintenance set or by performing the PNS mechanism through contacting the corresponding node in the maintenance set. Similarly, if a node finds that the node in a routing entry has failed, it can also utilize the PNS mechanism to fix the routing entry.

*4) Host Failure/Leaving:* The **Recovery procedure** of LAPTOP relies on the Heartbeat message and the parental relationship. A node does not need to take any action when it leaves the overlay network, since each node needs to periodically send a Heartbeat message to its parent.

Therefore, a node will notice the leave/failure of its child or parent node. The failure of a child node can easily be handled by its parent node by marking the absence of a logic address in its child list. Meanwhile, the failure of a single parent node is handled by the grandparent node using a "take over" mechanism. Specifically, when a node does not receive the ACK of the Heartbeat message from its parent node before the expiry of its timer, it will send a *CONTENTION* message to its grandparent node. (Notably, the address information of ancestor nodes is in the default routing set.) By waiting for a certain period of time, the grandparent node will select one of the child nodes that have sent the *CONNECTION* message to take over the duty of the failed node. The parent node updates its routing information to reflect this change, and also forwards update information to all of the child nodes to which it has sent the *CONNECTION* message. The criteria for selecting the take over node are implementation dependent, and one possible choice is to select the node with the least number of child nodes. A node which is already a take over node should not send a *CONTENTION* message to the grandparent node, but should simply notify the grandparent node of its existence. For multiple simultaneous node failures, if the failed nodes do not have a parental relationship, the failure recovery process can be performed independently as aforementioned. In cases where failed nodes have ancestor-descendant relationship in the overlay, a child node of a failed node may be unable to contact its grandparent node, since the grandparent node is also failed. In this case, the child node will receive no response from the grandparent node. The child node will then select the closest ancestor and contact it again until it receives a response from some ancestor node. In the worst case, its ancestors had been failed, it can reach the root node which is assumed to always be up. The found ancestor then selects a takeover node for its failed child node. The process is repeated for each level such that a new selected takeover node selects a takeover node for the failed node located one level below.

## 3. Performance Analysis of LAPTOP

This section analyzes the complexity of several major algorithms of LAPTOP.

### 3.1 Node Joining Algorithm

LAPTOP adopts a random tree traversal scheme for a newcomer to find its parent node on the overlay network.

**Lemma 1**: **The Join procedure requires less than $O(log_d N)$ steps to locate the parent of each newcomer, where $N$ denotes the number of nodes in the system and $d$ represents the maximum degree of a node.**

**PROOF:**

A newcomer initially is assigned a randomly selected start node by the bootstrap node. In the worst case, the start node is a level 1 node, and the new node is redirected to a child node because the degree of the start node is full which repeated again and again, until a newly selected start node is a leaf of the current overlay. Since the start node is randomly selected, the overlay tends to be a balanced tree and the height of the overlay tree should be bounded by $\mathbf{O}(log_d \mathbf{N})$. Therefore, the complexity of the Join procedure is bounded by $\mathbf{O}(log_d \mathbf{N})$. □

### 3.2 Node Routing Algorithm

Now we will show that the length of a routing path in LAPTOP in bounded by $O(log_d \mathbf{N})$.

**Lemma 2: The length of a routing path is bounded by $\mathbf{O}(log_d \mathbf{N})$ hops, where N denotes the number of nodes in the system and d represents the maximum degree of a node.**

**PROOF:**

We shall assume that the routing table of each node is correct and up to date. Due to the longest prefix matching, each routing step takes the message to a node that has a longer common prefix with the destination node. Since the length of the prefix is bounded by the height of the overlay tree, the length of a routing path should be bounded by $O(log_d \mathbf{N})$ hops. □

If the routing table is inaccurate due to many simultaneous node failures, the length of a routing path should not increase too much since with the maintenance set, the message will be forwarded to a node of a sub-tree that is approaching the destination.

### 3.3 Node Recovery Algorithm

The tree structure and hierarchical self-addressing scheme enables LAPTOP to efficiently handle overlay network change due to departure of a node, either gracefully or abruptly.

**Lemma 3: The recovery procedure overhead is bounded by $\mathbf{O}(d log_d \mathbf{N})$ in terms of number of nodes to contact.**

**PROOF:**

For a single failure, the recovery procedure involves at most $d$ children nodes and the parent of the failed node. Therefore, the communication overhead is bounded by $\mathbf{O}(d)$ in terms of nodes involved or messages to send. For multiple node failures, a child node may need to contact multiple ancestors, but the number of contact node is bounded by the height of the overlay tree, which is $\mathbf{O}(log_d \mathbf{N})$ for a balanced tree. A take over node needs to be selected for each failed node, which requires a communication overhead of $\mathbf{O}(d)$. Therefore, in the worst case, the overall communication overhead is bounded by $\mathbf{O}(d log_d \mathbf{N})$. □

In summary, Table 1 compares the maintenance overhead of existing well-known P2P systems with LAPTOP in terms of the amount of control messages sent. Generally, hosts in a decentralized but structured P2P system must periodically exchange control messages to maintain the overlay structure as well as its routing table. As shown in Table 1, LAPTOP has a competitive network diameter compared to Pastry (by having d set comparable to b) and its network diameter is much smaller than Chord. On the other hand, the number of control messages sent by each host in LAPTOP is significantly less than that in Pastry and Chord. The major difference is that LAPTOP

does not attempt to maintain the routing table periodically, but merely maintain a maintenance set as mentioned previously.

Table 1: The Comparison of Current Major P2P System in Periodical Control Message

| | LAPTOP | Chord | Pastry |
|---|---|---|---|
| Periodic control message | O(1) | $O(\log_2 N)$ | $O(\log_b N \times (b-1))$ |
| Network Diameter | $O(\log_d N)$ | $O(\log_2 N)$ | $O(\log_b N)$ |

## 4. Performance Evaluation

This section describes the simulation results for evaluating the performance of LAPTOP. Three experiments are designed to evaluate the performance of LAPTOP from various aspects. The first experiment evaluates the routing performance of LAPTOP under various overlay network sizes. Meanwhile, the second experiment studies the locality-aware property of LAPTOP in terms of latency stretch. The latency stretch refers to the ratio of average inter-node latency on the overlay network to that on the underlying IP network. Finally, the final experiment evaluates the robustness of LAPTOP in the face of node failures.

The present simulations were run on the GA-Tech topology generated by Georgia Tech topology generator [9]. The topology generated consists of ten transit domains at the top level, each with an average of five routers. Each transit router then is attached to an average of ten stub domains, and each domain consists of an average of ten routers. Each router is attached with 100 end hosts. Thus, the topology consists of 5050 hierarchically routers. The delay between two routers is also generated by the topology generator and the delay between each end host and its local router is also set to 1ms.

### 4.1 Routing Performance

The first experiment examines the average length of routing paths between any two hosts in terms of hop count to evaluate the routing performance. Figure 3 shows the average path length under various numbers of LAPTOP nodes in the two network topologies, respectively. The results plotted are the average values of ten simulation runs. Within each run, 200,000 routing requests are generated, and each routing request sends a message from a source to a randomly selected destination node. The average routing path length is about 3.8 for a LAPTOP system with 100,000 nodes and a maximum degree of 16, as shown in Fig. 3. Note that, in a network of 100,000 nodes, the average routing path length of Chord and Pastry are approximately 8 and 4, respectively. Therefore, routing in LAPTOP is quite scalable.

### 4.2 Delay Stretch

The second experiment examined the latency stretch and distribution of each routing hop. As mentioned above, the latency stretch refers to the ratio of average inter-node latency on the overlay network to that on the underlying IP network. As in the first experiment, there are 100,000 hosts in LAPTOP. The average results of ten simulation runs are plotted, and 200,000 routing requests are randomly generated during each run.

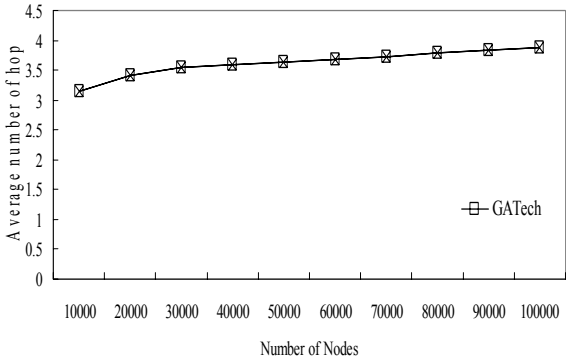


Fig. 3: Average routing path length under various network sizes. The LAPTOP system consists of 100,000 nodes, each with a maximum degree of 16

Three versions of LAPTOP were compared. The "No locality" version builds routing tables without taking network locality into account; the "PNS" version assumes global information is available to create perfect PNS based routing tables; the "PNS-256" is equivalent to LAPTOP that probes a maximum of 256 nodes in *maintenance set* when building each routing entry.

Figures 4 shows the latency stretch obtained in the GA-Tech network topologies. LAPTOP performs quite well, and yields a result very close to that of the perfect PNS approach. As shown in the figures, the latency stretch for a system of 100,000 nodes is only about 1.5. Since the perfect PNS approach needs to probe all nodes, it is infeasible. However, LAPTOP probes a maximum of 256 nodes and can yield a very competitive stretch. Therefore, the maintenance set concept proposed in LAPTOP is both scalable and effective. Compare to Chord and Pastry in the GA-Tech network topology, the average delay stretch of Chord and Pastry are around 8 and 1.6, respectively. Chord suffers the markedly delay stretch since it does not consider network locality. The delay stretch of Pastry is competitive to that of LAPTOP, but Pastry needs to exchange more node information than LAPTOP as shown in Table 1.

### 4.3 Node Failures

The third experiment explores the robustness of LAPTOP in the presence of node failures. Node failures are simulated on a LAPTOP system with 100,000 nodes, each with a maximum degree of 16. In this experiment, after 100,000 nodes have joined the LAPTOP, a certain fraction, from 0% to 80%, of randomly selected hosts becomes and remains failed during the rest of simulation time. These hosts are then considered simultaneous failed nodes. When selected nodes become failed, 200,000

routing requests are generated with source and destination nodes randomly selected from living nodes. Each result plotted is also the average of ten simulation runs.

Figure 5 shows the percentages of successful and failed routing under various percentages of failed nodes. As we can see the LAPTOP system is very robust for a large fraction of simultaneous node failures. Figure 6 shows the impact of node failures on route quality. The first bar shows the average route length (hop-count) in the case of no failure, while the rest of the bars show the average route length when 10%, 20%, and 30% of nodes failed, respectively.

Notably, in this experiment, the failed nodes are never repaired. Compared to other P2P system, such as Pastry and Chord, the failed nodes not only need a period to stabilize their routing tables but also need to maintain the leaf set or ancestor list. In LAPTOP, failed routing entries can be tackled more efficiently due to the tree structured overlay and the concept of maintenance set. As a result, LAPTOP yields better routing performance and lower maintenance cost than other P2P systems.

## 5. Conclusion and Future works

This work presented LAPTOP, which provides a simple, efficient, scalable and robust peer-to-peer overlay routing service. The performance of LAPTOP was analyzed mathematically and via simulations. The length of a routing path in LAPTOP was shown to be bounded by the height of the overlay tree, which is $O(\log_d N)$. Moreover, the complexity of the join and leave procedure is bounded by $O(d\log_d N)$. Simulation results showed that LAPTOP performs very well for a system with 100,000 nodes. Locality was explored and taken into consideration in building the LAPTOP overlay structure.

LAPTOP differs from other P2P systems by its self-organizing locality-aware hierarchical overlay structure and self-addressing, longest-prefix matching routing scheme. Specifically, the contribution of LAPTOP is to demonstrate a feasible overlay network framework that adopts the locality-aware concept with low maintenance overhead. We believe that the concept of LAPTOP is also applicable to large-scale distributed computing systems, P2P applications in ad hoc networks, and more.

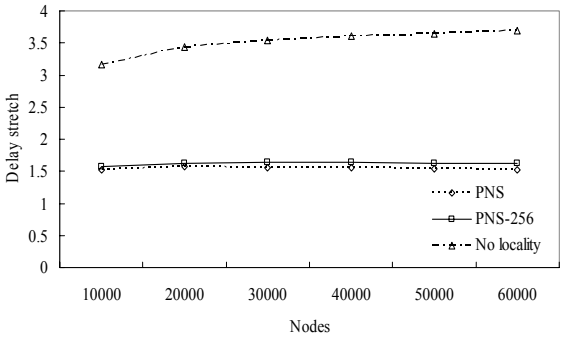


Fig. 4: Latency stretch under various network sizes within node degree of 16

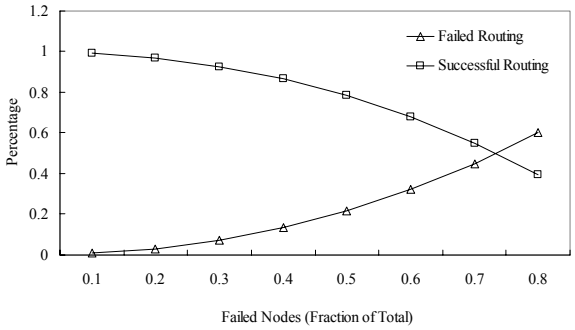


Fig. 5: Percentage of succeed and failed routing request for varying percentage of node failures

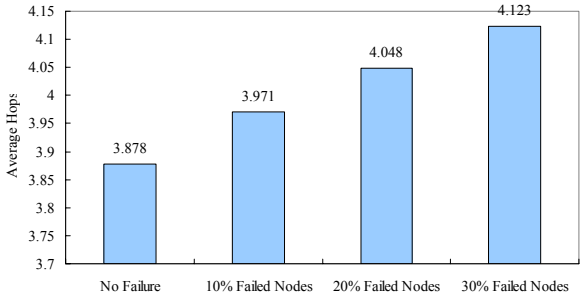


Fig. 6: Average routing path under various percentages of nodes failed

## References:


[1] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," in *Proceedings of ACM Supercomputing*, 2002.

[2] The Napster home page, http://www.napster.com/

[3] The Gnutella home page, http://gnutella.wego.com/

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," *in Proceedings of SIGCOMM*, 2001.

[5] S. Ratnasamy, M. Handley, R. Karp, S. Shenker, "Topologically-Aware Overlay Construction and Server Selection," In *Proceedings of IEEE INFOCOM* , 2002.

[6] B. Zhao, Y. Duan, L. Huang, A. Joseph, J. Kubiatowicz, "Brocade: Landmark routing on overlay networks," in *Proceedings of IPTPS*, 2002.

[7] Z. Xu, M. Mahalingam, and M. Karlsson, "Turing Heterogeneity into and Advantage in Overlay Routing," In *Proceedings of IEEE INFOCOM*, 2003

[8] B. Y. Zhao, J. Kubiatowicz, A. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," *IEEE JSAC*, Vol. 22, No. 1, January 2004.

[9] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of IEEE INFOCOM*, 1996

[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A Scalable Content-Addressable Network," in *Proceedings of SIGCOMM*, 2001.

[11] A. Rowstron, P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proceedings of IFIP/ACM ICDS*, 2001.

[12] X. Y. Chang, Q. Zhang, Z. Zhang, G. Song, and W. Zhu, "A Construction of Locality-Aware Overlay Network: mOverlay and Its Performance," *IEEE JSAC*, Vol. 22, No. 1, January 2004, pp. 18-28.