INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2007; **20**:83–102
Published online 16 May 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/dac.815

A location-aware peer-to-peer overlay network

Chi-Jen Wu[1], De-Kai Liu[2,*,†] and Ren-Hung Hwang[2]

[1] *Communication Engineering, National Chung Cheng University, Taiwan*
[2] *Computer Science and Information Engineering, National Chung Cheng University, Taiwan*

SUMMARY

This work describes a novel location-aware, self-organizing, fault-tolerant peer-to-peer (P2P) overlay network, referred to as Laptop. Network locality-aware considerations are a very important metric for designing a P2P overlay network. Several network proximity schemes have been proposed to enhance the routing efficiency of existing DHT-based overlay networks. However, these schemes have some drawbacks such as high overlay network and routing table maintenance overhead, or not being completely self-organizing. As a result, they may result in poor scalability as the number of nodes in the system grows.

Laptop constructs a location-aware overlay network without pre-determined landmarks and adopts a routing cache scheme to avoid maintaining the routing table periodically. In addition, Laptop significantly reduces the overlay maintenance overhead by making each node maintain only the connectivity between parent and itself. Mathematical analysis and simulations are conducted to evaluate the efficiency, scalability, and robustness of Laptop. Our mathematical analysis shows that 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 is the number of nodes in the system, and d is the maximum degree of each node on the overlay tree. Our simulation results show that the average latency stretch is 1.6 and the average routing path length is only about three in 10 000 Laptop nodes, and the maximum degree of a node is bounded by 32. Copyright © 2006 John Wiley & Sons, Ltd.

Received 1 June 2005; Revised 1 January 2006; Accepted 1 March 2006

KEY WORDS: overlay networks; peer-to-peer networks; application level routing

1. INTRODUCTION

The value of sharing the computing power of billions of personal computers on the Internet is becoming increasingly obvious as the computing capability of the personal computer is getting more and more powerful. This makes the peer-to-peer (P2P) network model become more attractive to a number of client/server Internet applications. Unlike the traditional client/server

*Correspondence to: D.-K. Liu, Computer Science and Information Engineering, National Chung Cheng University, Taiwan.
† E-mail: dkliu@cs.ccu.edu.tw

model where clients only retrieve data from the server, each host (peer) acts as both a server and a client in a P2P network. This feature enables a node to obtain data of interest from any other node that holds it, solving the poor scalability problem in the client/server model. P2P networks have been used in various distributed applications, such as distributed file systems, file sharing systems, and content distribution systems.

P2P networks in the literature can be roughly divided 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 files of interest 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. On the other hand, 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 which maps identifiers of nodes and data items onto an m-bit circular identifier space. The data discovery procedure can be viewed as a routing process that starts from the querier to the node that holds the identifier of requested data item. To forward the query message, each Chord node maintains a routing table called a *finger table*. For a node n, the kth entry of the finger table consists of the address information of the node with the smallest identifier equal to or greater than $n + 2^{k-1}$ in the circular identifier space, where $1 \leqslant k \leqslant m$. Nodes in node n's finger table are so-called *successors* of n. Figure 1 shows a simple example of Chord which consists of three nodes. Node 1's successors are nodes with the least identifiers equal to or greater than $(1 + 2^0), (1 + 2^1)$, and $(1 + 2^2)$, and they are node 4, node 4, and node 7. When a node wants to search for a data item, it generates a query within the corresponding identifier and starts the routing process. At each step, each node first checks whether it holds the requested data item. If yes, it will reply this query, otherwise, it forwards the query to its successor with the greatest identifier in its finger table that precedes the requested data item's identifier.

We believe that the decentralized but structured P2P network is most suitable for large-scale distributed applications since it searches for the node that keeps data of interest in a scalable and efficient way, and it can be widely deployed as an application level routing service for various P2P applications. However, the design of a decentralized but structured P2P network has to overcome two critical issues. The first issue is the long routing latency. Several proximity schemes [5–9] have been proposed to avert long routing latency in current structured P2P networks, but they require a high-complexity procedure to periodically maintain the routing table [7] (e.g. Pastry system) or they need pre-chosen landmarks to construct the overlay. However, the P2P system is by its very nature unstable since nodes join and leave frequently. For instance, the study of Gnutella [3] shows around approximately 1200 membership changes per minute in a 100 000 nodes P2P system. Another proximity scheme needs some pre-chosen landmarks or a complete BGP routing table support. As a result, they both increase the difficulty of the P2P system deployment.

The second issue is system maintenance overhead. The existing structured P2P networks allow nodes to keep some nearby nodes in their routing tables in order to achieve efficient routing. The routing table size corresponds with the capacity of the system size in some structured P2P networks, and the system size should be pre-defined to be large enough to avoid nodeID collision or for further capacity of system. As a result, the number of practical participants is significantly less than the system size and increases the redundant maintenance overhead.

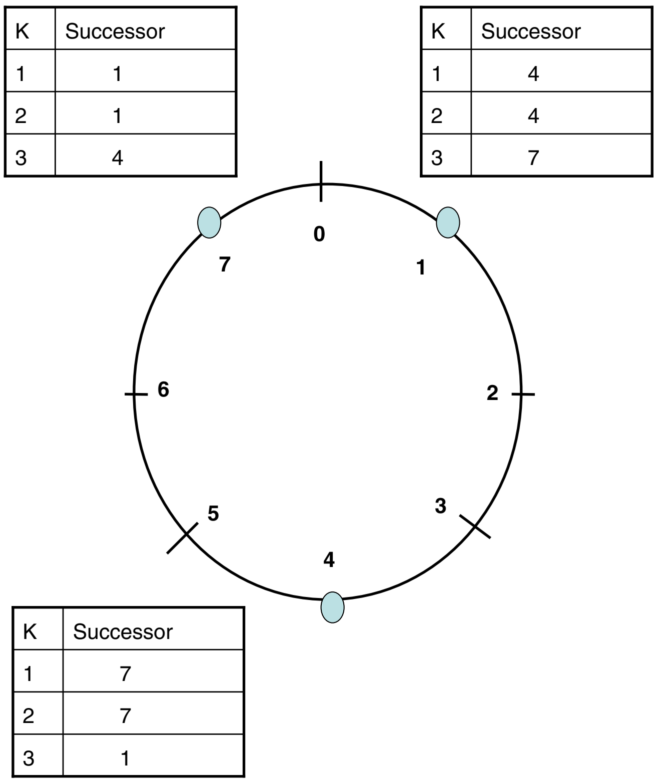


Figure 1. Chord system.

This work presents Laptop, a novel self-organizing P2P overlay network with efficient, scalable, robust lookup and routing schemes. Laptop adopts the location-aware concept to the overlay construction. Specifically, Laptop organizes nodes into a tree-based overlay network and uses a novel node addressing scheme to enable nodes to be roughly aware of its physical location in the underlying network when they join the overlay network. Besides, a routing cache learns routing entries during forwarding messages so as to reduce the significant routing table maintenance overhead. Therefore, the routing path length in a Laptop overlay network is bounded by $O(\log_d N)$ hops in the balance overlay tree, where N is the number of nodes, and d is the maximum degree of each node. Meanwhile, the tree-based overlay network enables each Laptop node to periodically contact with only *one* node, i.e. its parent node in the tree-based overlay network, for overlay maintenance. As a consequence, the overlay maintenance overhead is greatly reduced compared to the existing structured P2P networks.

Our simulation results show that the average routing path length is three, and that the average latency stretch is 1.6 in the case of 100 000 nodes in the overlay network, and the maximum degree of a node is bounded by 32. The maintenance overhead of Laptop is 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.

The rest of this paper is organized as follows. First, Section 2 reviews P2P networks related literature. Next, Section 3 presents an overview of Laptop and describes how Laptop approaches the three issues mentioned above. Section 4 then provides a detailed overview of Laptop. Section 5 shows our mathematical analysis and the simulation results for evaluating the performance of Laptop. Finally, we conclude our work in Section 6 and discuss our future research.

2. RELATED WORKS

Numerous approaches have been suggested to exploit network proximity in current DHT-based P2P overlay networks. Ratnasamy *et al.* [8] divided these approaches into three categories: geographic layout, proximity routing and proximity neighbour selection. We will first reviews the three kinds of proximity mechanisms. Then we will describe some related researches on the analysis of fault-tolerant mechanisms based on DHTs [7, 9, 10].

The concept behind the geographic layout approach is to consider the geographic layout of nodes during the overlay construction. For example, bin-CAN [9] proposed a scheme for node clustering and server selection by letting nodes measure RTTs to a set of pre-chosen landmarks. Essentially, the bin-CAN attempts to map a d-dimensional space onto the geographic layout of nodes. However, two problems emerge. First, this approach is not completely self-organizing since the landmark has to be chosen *a priori*. Second, the bin-CAN is designed for clustering nodes on the overlay network, and each node is therefore aware of nearby nodes in terms of their geographic co-ordinate space. As a consequence, the next hop selection during the routing process is based on greedy neighbour selection. The greedy neighbour selection may cause that a long route to a far node consists of consecutive hops located in a physically small region, increasing the number of hops traversed as compared to the routing in the underlying IP network.

The proximity routing approach improves routing efficiency by optimizing the choice at each routing step. Chord [4] is a typical example of proximity routing. A node in Chord chooses the next hop that is either closer to the key of the message or has a lower latency to the destination in its routing table during the routing process. Since a message is forwarded to the node that is most likely to be close to the destination at each routing step, each node is expected to be reached within a small number of hops. Nevertheless, the proximity routing constructs routing tables of nodes without considering the physical network, making the route to the next hop more likely to be a long route in the underlying physical network.

The proximity neighbour selection approach utilizes a similar scheme with proximity routing in the next hop selection during the routing process. Proximity neighbour selection and proximity routing differ mainly in that the proximity neighbour selection considers network proximity in the routing table construction. Pastry [11] and Tapestry [12] are two representatives of this kind of approach. For example, routing table entries of a Pastry node are chosen from nearby nodes in sense of latency among all nodes with proper node ids. A message is passed to the nearby node whose node id either shares a longest prefix with the key or is numerically closer to the key at each routing step. Pastry exploits network proximity to overlay construction by assuming that each Pastry node can estimate its proximity to any node in terms of an application-specific proximity metric. Additionally, it claims that it significantly reduces the number of routing hops on a path by choosing the next hop from a large number of nearby nodes during the routing process. Tapestry is very similar to Pastry but differs in the mechanism to locate the numerically closest node.

Recently, several projects have adopted the hierarchical approach to improve the routing efficiency of some decentralized but structured P2P overlay networks. For example, Brocade considers locality in the routing process at the autonomous system (AS) level by avoiding routing a message through unnecessary AS domains. The Brocade [13] builds a secondary overlay on top of a Tapestry network. The secondary overlay is composed of nodes near network access point, such as routers and gateways. These nodes are so-called supernodes. Messages across the wide area networks can takes advantage of the highly connected network infrastructure between these supernodes. Therefore, messages can be routed as directly as possible from one AS to another. Meanwhile, eCAN [14] builds an auxiliary overlay network called expressway by exploiting the AS-level topology from BGP routing reports and predefined landmarks to improve the routing performance of CAN.

Fault tolerance is another important issue in decentralized but structured P2P overlay networks. Most DHT-based P2P overlay networks make nodes maintain a routing table or neighbour table by periodically exchanging control messages with other nodes to achieve efficient routing and overlay connectivity. Consequently, the routing table maintenance overhead increases as the number of nodes increases. Furthermore, some of them may not be able to provide full recovery from multiple simultaneous failures. Recently, several researchers have started to analyse the failure recovery overhead. For example, Liben-Nowell *et al.* [10] showed that the per-node maintenance protocol bandwidth is lower bounded by $O(\log N)$ per half-life, for any P2P overlay network that wishes to remain connected with a high probability in an N-node network. Meanwhile, other research studied the robustness of multiple failures. For instance, Datar [15] exploited the concept of a butterfly network in a P2P overlay network. The proposed architecture retains good routing structure, even after the adversarial removal of a constant fraction of nodes. Datar also showed how to maintain the infrastructure, as multiple nodes fail in the case where the number of nodes joining the network is always sufficiently larger than the number of failures.

3. OVERVIEW OF LAPTOP

Laptop is a location-aware, self-organizing P2P overlay network that focuses mainly on reducing routing latency and providing high scalability as well as fault tolerance. Section 3.1 describes the motivation of Laptop by discussing the proximity schemes in the literature. Section 3.2 then presents the location-aware, tree-based overlay network of Laptop in detail.

3.1. Motivations

The proximity neighbour selection scheme is judged to be the best choice among the three kinds of proximity schemes discussed in the last section since it considers the network proximity in the overlay construction and selects the next hop from a large set of nearby nodes at each routing step. However, existing solutions that adopt the proximity neighbour selection scheme, such as Pastry and Tapestry, suffer from two problems. First, these approaches take the physical network proximity into account under the assumption that the triangle inequality feature holds true on the Internet. Nevertheless, Savage *et al.* [16] have shown that the triangle inequality may not hold true on the Internet. Second, these approaches depend highly on the ability that nodes can estimate their 'distances' in terms of an application-specific proximity metric to any node with a given IP address. This may complicate the joining and routing process.

Brocade and eCAN build secondary overlay networks on top of Tapestry and CAN overlay networks, respectively. The routing performance can thus be improved, especially in the case of long routes, since the secondary overlay consists of nodes near routers and gateways. Although these two approaches greatly reduce the routing latency to a far node, how to identify nodes that are close to routers and gateways to form the secondary overlay remains an open issue. In addition, these proposed approaches recognize these nodes based on BGP reports and pre-chosen landmarks. Therefore, they improve the routing performance but sacrifice the self-organizing feature of previous P2P overlay networks.

Laptop utilizes the concept behind the geographical layout approach and constructs a hierarchical overlay network just like Brocade and eCAN. However, the overlay network in Laptop is formed in a self-organizing way. Specifically, Laptop enables a newcomer to find its access point on the tree-based overlay network only by estimating the round trip time (RTT) to a small number of nodes on the overlay network. The access point on the tree-based overlay network for a newcomer is so-called its parent node. Additionally, a Laptop newcomer will keep nodes it has ever contacted during the joining process in its routing table and adopt the proximity routing scheme to the routing process. Laptop proposes a hierarchical addressing scheme to correlate the logical distance with the physical distance of nearby nodes, thus improving the weakness of the proximity routing approach.

3.2. Location-aware tree-based overlay network

Laptop organizes nodes into a tree-based overlay network and enables nearby nodes to be roughly aware of their physical distances through the proposed hierarchical addressing scheme. Based on the addressing scheme, Laptop utilizes a hierarchical routing scheme that mimics the nature of the IP routing, thereby achieving efficient routing. Each Laptop node is identified by a level label and a node address and they are formally defined as follows.

Definition 1 (Level label)
A node p is assigned a label L_i if $\mathrm{dist}(\mathrm{pr}, p)$ falls in Segment(i), for all i, $1 \leqslant i \leqslant \mathrm{Max\_Level}$, where $\mathrm{dist}(\mathrm{pr}, p)$ is the RTT between node p and its parent node pr, and Segment(i), $1 \leqslant i \leqslant \mathrm{Max\_Level}$, are configurable RTT intervals. The root node, which refers to the first node in the overlay network, is initially assigned with label L_1. The root node is assumed to be a well-known node in the overlay network. Note that L_1 is considered to be the highest level, while $L_{\mathrm{Max\_Level}}$ is considered to be the lowest level.

Definition 2
Nodes can only have children nodes with a level lower than themselves except the $L_{\mathrm{Max\_Level}}$ node. An $L_{\mathrm{Max\_Level}}$ node only has $L_{\mathrm{Max\_Level}}$ children nodes. An $L_{\mathrm{Max\_Level}}$ node has children nodes only when its parent node has reached its maximum degree. The root node maintains an L_1 node list to keep information of all existing L_1 nodes. An L_1 node which is not the root node also keeps information of other L_1 nodes in the overlay network.

Definition 3 (Node address)
Each node obtains one node address in the form of a dotted decimal number, and each octet of a node address ranges from 1 to d, where d is the maximum degree of a node in Laptop. Each node, except the L_1 node, is assigned the node address by appending a unique octet to the node address of its parent node. The root node is initially assigned with the address, '1', and the addresses of the other L_1 nodes which have the format of x, $x > 1$, are also dynamically assigned by the root.

Definition 4 (Physical distance of nodes)
For a descendant node Y of a node X, dist(X, Y) is less than the lower bound of Segment(p), where p is the level label of node X, for all i, $1 \leqslant i < \text{Max\_Level}$.

Figure 2 shows an example of a Laptop overlay network while d and Max_Level are set to 4. The RTT intervals for these segments (from 1 to 4) are set to [175,∞], [75,175], [35,75], [0,35] in units of ms. Essentially, these four intervals can be defined according to the expected RTT of two nodes which are located in the same site, in different sites of the same ISP, in different ISPs of the same country, and in different countries. By Definition 4, descendants of node '1' can be considered to be located in the same country with node '1' since the level label of node '1' is L_1 such that the RTT between them is less than 175, the lower bound of the RTT interval of segment 1. Similarly, descendants of node '1.1' can be considered to be located in the same ISP with node '1.1' as the level label of node '1.1' is L_2 so that the RTT between them is less than the lower bound of the RTT interval of segment 2.

Definitions 3 and 4 enable the routing paths on Laptop to be close to those of the IP routing by adopting the longest-prefix matching scheme. Specifically, a message from the source node to the destination node travels along the overlay tree up to the nearest common ancestor of them,

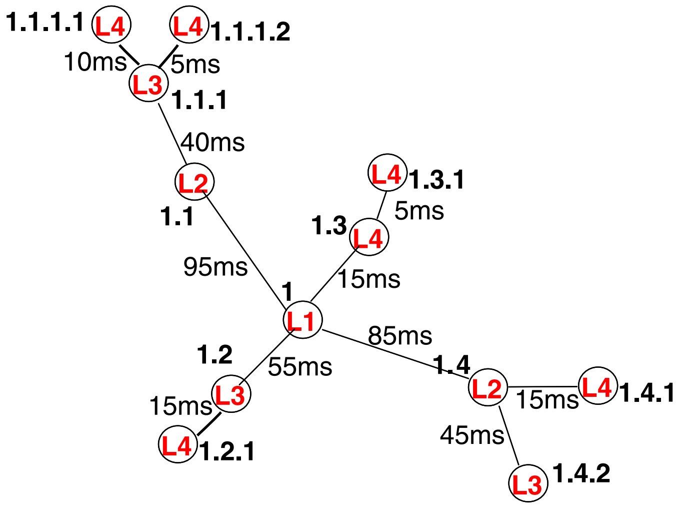
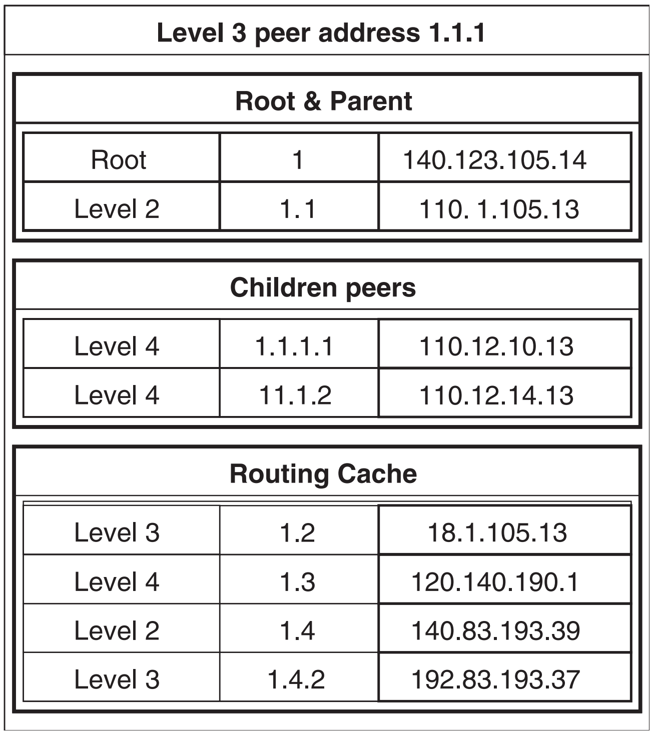


Figure 2. A Laptop overlay network.

and then travels down the tree to the destination node. Consider the case in Figure 1, a message sent from node '1.1.1' to node '1.4.1' is first routed to node '1', since the longest common prefix of '1.1.1' and '1.4.1' is '1'. Node '1' then forwards the message to '1.4' as it is the longest prefix entry in its routing table. Finally, node '1.4' forwards the message to its child node '1.4.1'. Conceptually, this example demonstrates that this message is first routed from the source node which is located in a local ISP, to a node which resides in a regional ISP that takes charge of local ISPs source and destination nodes. The node in the regional ISP then forwards the message to a node in the destination node's ISP. Finally, the message is sent to the destination node. In this way, Laptop adopts a hierarchical routing scheme that mimics the IP routing to achieve routing efficiency.

Each Laptop node maintains a routing table for message forwarding. Figure 3 shows an example of the routing table of node '1.1.1' in Figure 2. Each entry consists of the level label and node address of a node, as well as its public IP address. Entries in a routing table can be divided into two categories: *default routes* and *routing cache*. Default routes consist of the address information of the root node, parent node, and children nodes. Information of root and parent nodes is retrieved during the joining process, while information of a child node is obtained when a new node become the node's child node. The default routes enable a Laptop node to adopt a proposed hierarchical routing scheme to forward messages. Additionally, the Laptop applies the routing cache to further improve the routing efficiency. Entries in the routing cache are learned when forwarding a message. Entries of the routing cache are initially set to addresses



Level 3 peer address 1.1.1

Root & Parent

Root	1	140.123.105.14
Level 2	1.1	110. 1.105.13

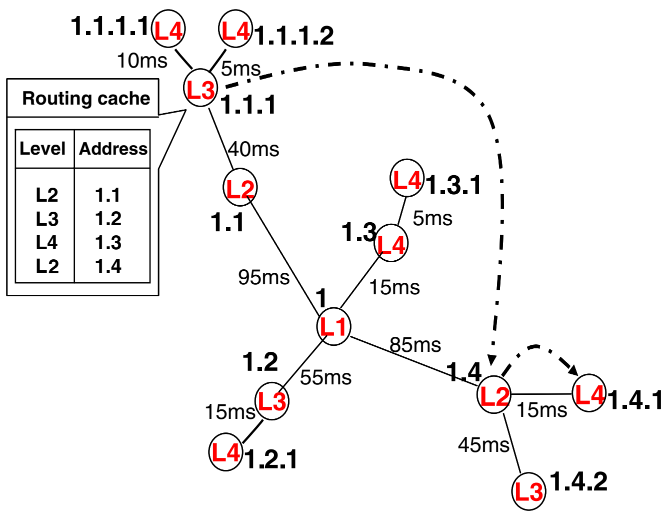
Children peers

Level 4	1.1.1.1	110.12.10.13
Level 4	11.1.2	110.12.14.13

Routing Cache

Level 3	1.2	18.1.105.13
Level 4	1.3	120.140.190.1
Level 2	1.4	140.83.193.39
Level 3	1.4.2	192.83.193.37

Figure 3. Routing table of peer '1.1.1'.



Level	Address
L2	1.1
L3	1.2
L4	1.3
L2	1.4

Figure 4. Routing acceleration by routing cache.

information of children nodes of the root node and nodes contacted during the joining process. The routing cache can improve the routing efficiency and achieve good load balance. Figure 4 shows an example of routing acceleration by routing cache. The routing table of node '1.1.1' includes information of node '1.4' because node '1.4' is a child node of the root node. In this case, when node '1.1.1' sends a message with key '1.4.1', it routes the message to node '1.4' instead of to node '1', since node '1.4' has a longer common prefix with the key. As a consequence, a shorter path is taken which excludes node '1'. The detail of the routing cache maintenance will be described later. In Laptop, the number of children nodes a node can have, as well as the size of routing cache is limited by d and $C_$Size, respectively, where d is the maximum degree of a Laptop node mentioned before, and $C_$Size is a predefined parameter. This feature makes Laptop become scalable. A replacement policy is therefore applied to the routing cache such that it can keep more useful entries, such as far away peers or peers with a high-level label.

The Laptop simplifies the overlay maintenance by letting the parent node take charge of its children nodes, making it very efficient. A Laptop node periodically sends heartbeat messages to its parent node. Therefore, the parent node will know the absence of the child node, either because the child node gracefully notifies its leaving or if it fails abruptly. The overlay maintenance procedure is used for either notifying the parent node about the departure of its child node when the child node is a leave node, or assigning an active node to take over its child node when the child node is not a leave node. The overlay maintenance procedure is not only light-weight, but also efficient since only parent and children nodes of a failed or leaving node are involved, and they are physically close due to the location-aware feature. Furthermore, the searching and routing service will not be disrupted in the case of a node leaving or failure, because the parent node will take the place of its failed or leaving child node until a new node is selected to take over the place of this child node, enhancing the routing robustness of Laptop.

4. DESIGN OF LAPTOP

In Laptop, a newcomer can join the overlay network by invoking the node joining procedure. In addition, a node can invoke a graceful departure procedure to leave the overlay network. On the other hand, the failure recovery procedure will be invoked when a node fails abruptly. Meanwhile, a prefix-based routing scheme is provided for search and routing service on the overlay network. This section will describe these algorithms in detail, along with their theoretical analysis.

4.1. Node joining

Laptop adopts a tree traversal scheme for a newcomer to find its parent node on the overlay network. Due to location-aware and hierarchy features of the Laptop overlay network, a newcomer is likely to consider a node physically close to it as its parent node. The newcomer is assigned its level label as well as its node address by its parent node. Additionally, the newcomer keeps nodes that it has contacted during the joining process in its routing table for further routing processes. The joining procedure works as follows.

(1) The newcomer N sends a joining request to the root node and gets a list of L_1 nodes.
(2) N determines the closest L_1 node by measuring RTTs to nodes in the list.
(3) If the measured RTT for the closest L_1 node falls in Segment(1), then N becomes a new L_1 node and the joining process is completed.
(4) Otherwise, N sets its potential parent node as the closest L_1 node.
(5) If the potential parent node does not have any children nodes, then N sets its parent node as that potential parent node and gets its level label as well as the node address from its parent node according to Definitions 1 and 3. The joining process is then completed.
(6) Otherwise, N gets a list of L_i children nodes that do not violate Definitions 1 and 2 if N selects them as parent nodes, for $2 \leqslant i < L_{\text{Max\_Level}}$, from the potential parent node.
(7) N measures RTT for the list of children nodes and selects the closest one as new potential parent node. Go back to Step 5.

Figure 5 shows an example of the joining procedure, where Max_Level is set to 4. RTT intervals for these segments (from 1 to 4) are set to $[175,\infty]$, $[75,175]$, $[35,75]$, $[0,35]$ in units of ms. The newcomer first contacts root node '1' to obtain the L1 node list and measures RTTs between itself and these L_1 nodes. In this scenario, it finds that node '1' is the nearest L_1 node and that the measured RTT does not fall in Segment(Max_Level). The newcomer then asks node '1' for a list of L_X children nodes, for $2 \leqslant X < 4$. It turns out that node '1.2' is the nearest node, and the RTT between the newcomer and node '1.2' falls in Segment(Max_Level). Therefore, the newcomer chooses it as its parent node. Finally, the newcomer becomes an L_4 child node of node '1.2' and obtains a node address of '1.2.1'. Note that a newcomer does not measure RTTs to $L_{\text{Max\_Level}}$ nodes to reduce the joining latency as well as the message overhead. Since $L_{\text{Max\_Level}}$ nodes are very close to their parent nodes and are likely to appear more frequent than nodes with other level labels, without measuring RTTs to $L_{\text{Max\_Level}}$ nodes, they will not greatly affect the locality feature of Laptop.

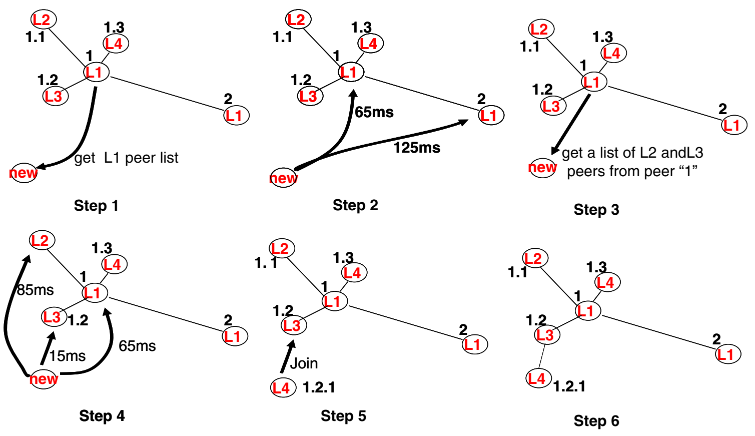


Figure 5. Example of the joining procedure.

Note that a newcomer may find its parent node, but the parent node cannot afford more child nodes due to the maximum degree limitation. When the parent node has reached the maximum degree, the following algorithm is used to resolve the exception:

(1) The parent node P notifies the newcomer N that it cannot afford more child nodes and asks it to consider node C, the child node of P that has the lowest level label, to be its parent node. If there are multiple children nodes qualified, the nearest one will be chosen.
(2) If C can support more child nodes, it asks N to be its $L_{\text{Max\_Level}}$ child node.
(3) Otherwise, the above procedure is repeated with C as the new parent node.

This algorithm may cause an exception of Definition 1 since the newcomer is assigned a label $L_{\text{Max\_Level}}$ even though the RTT between it and its parent node may not fall in Segment(Max_Level). Essentially, there is a trade-off between definition compliance and system scalability. Without the maximum node degree limitation, Definition 1 will never be violated. However, considering the scalability of a Laptop system, a node should not have too many children nodes since this may make it become the system bottleneck. Therefore, the definition violation is inevitable when designing a Laptop system. To take the locality property into account, when the parent node can accept more child nodes, the newcomer shall consider the child node that is nearest to the original parent node to be its parent node. Consequently, the newcomer will select the child node that has the lowest level label to be its parent node. Noted that an $L_{\text{Max\_Level}}$ node will not be contacted during the joining process. To avoid interfering with the joining process of the late newcomers, a newcomer that violates Definition 1 is therefore assigned to be an $L_{\text{Max\_Level}}$ node. Now we will show that the control overhead of the joining process is $O(d \log_d N)$.

Lemma 1
The control overhead of the joining process is $O(d \log_d N)$ in terms of the number of nodes to contact, where N is the number of nodes in the system and d is the maximum degree of a node.

Proof
A newcomer initially chooses an L_1 node from the L_1 node list given by the root node. The number of L_1 nodes is bounded by d. After the nearest L_1 node is found, the newcomer then must measure the RTTs of all its L_x children nodes, for $2 \leqslant x < \text{Max\_Level}$. The number of measurements is also bounded by d. In the worst case, the newcomer finds its parent node after traversing the whole overlay tree. As a consequence, the measurements may be repeated as many times as the height of the overlay tree. Assume that the overlay tree is a balanced tree, the height is therefore bounded by $O(\log_d N)$. In general, since a parent node can only have children nodes with a lower level label and N is much larger than Max_Level, a linear chain structure cannot be formed. In the worst case, most of nodes are $L_{\text{Max\_Level}}$ nodes or children nodes of $L_{\text{Max\_Level}}$ nodes. Since the degree of a Laptop node is d, an $L_{\text{Max\_Level}}$ node cannot have a grandchild node unless it reaches its maximum degree. Therefore, the height of a Laptop overlay tree is bounded by $c + \log_d N$, where c is some constant (to reflect some high level nodes that do not have the degree of d). In other words, the height of a Laptop overlay tree is bounded by $O(\log_d N)$. Therefore, the control overhead for the joining process is $O(d \log_d N)$. □

4.2. Node departure

The tree structure and hierarchical self-addressing scheme enables Laptop to efficiently handle an overlay network change due to the departure of a node, be it either gracefully or abnormally. Overlay maintenance only involves the grand parent, parent and children nodes of the leaving or failed node. This implies that the overlay network partition due to nodes' departure will be recovered soon, which shows the robustness of Laptop. This subsection will discuss two node departure cases: graceful departure and abnormal departure.

The graceful departure procedure is proposed as follows.

(1) The leaving node checks the number of children nodes in the overlay network.
(2) If the leaving node does not have any children nodes, it simply notifies its parent node of its leaving. After receiving the notification, the parent node removes the leaving node from its routing table. The departure procedure is completed.
(3) Otherwise, the leaving node selects the child node with the lowest RTT to it to take over its position in order to preserve the location-aware property.
(4) All children nodes of the leaving node other than the takeover node changes its parent node to the takeover node, while the takeover nodes changes its parent node to the leaving node's parent node.

The failure recovery procedure is invoked when a node fails accidentally. In Laptop, each node shows its availability by periodically sending control messages, called HEARTBEAT messages, to its parent node. Upon receiving a HEARTBEAT message, the parent node should acknowledge it. In the case that children nodes have not received acknowledgment messages from their parent node for a preset Active_Period milliseconds after sending HEARTBEAT messages, the parent node is assumed to be dead and the failure recovery procedure is invoked. The failure recovery procedure works as follows.

(1) The children nodes of the failed node send out CONTENTION messages to their grandparent node, start a FAILURE RECOVERY timer, and wait for a response from the grandparent node. If the FAILURE RECOVERY timer times out and, the new parent node is still not found, the children nodes invoke the joining procedure to find their

new parent node. Note that the children nodes have contacted with their grandparent node during the joining procedure so that the IP address of the grandparent node is stored in their routing caches.

(2) The parent of the failed node starts a CONTENTION timer after receiving the first CONTENTION message from its grandchild node and continues receiving CONTENTION messages from other grandchildren nodes.
(3) The parent of the failed node chooses the grandchild node with the lowest level label as the takeover node, after the CONTENTION timer times out. The tie is broken by the lower RTT.
(4) The parent node of the failed node then updates its routing table and child list, and sends a TAKEOVER message to the takeover node.
(5) The parent node of the failed node sends an UPDATE_PARENT message to all of the nodes that had sent CONTENTION messages, except the takeover node, to inform them to consider the takeover node to be their new parent node.

Now let us analyse the control overhead of the departure procedures.

Lemma 2
The graceful departure overhead is bounded by $O(d)$ in terms of the number of nodes to contact.

Proof
It is trivial that the control overhead is $O(1)$ if the leaving node is a leaf node. Otherwise, the leaving node needs to inform its parent and all of its children nodes. In this case, the control overhead is $O(d)$ since it has at most d children and one parent nodes. □

Lemma 3
The abnormal departure overhead is bounded by $O(d \log_d N)$ in terms of the number of nodes to contact.

Proof
In the case that the new parent node can be found before the CONTENTION timer expires, the recovery procedure involves d children nodes of failed nodes and the parent of the failed node. Therefore, the control overhead is bounded by $O(d)$. Otherwise, a child node may need to perform the joining procedure where the control overhead is bounded by $O(d \log_d N)$. □

4.3. Routing

Laptop utilizes the longest-prefix matching routing scheme when forwarding messages. The routing entry that matches the longest prefix with the key of the message is retrieved, and the IP address of the next hop is then used to forward the message at each routing step. A routing table consists of a set of default routes and a routing cache. At each routing step, a node first checks whether there is a shortcut to the destination by comparing the key of the message with addresses of routing cache entries. If yes, the message is passed on to the next hop of the routing cache entry, based on the longest-prefix matching scheme, otherwise, the message is passed on to the next hop of the default route that is nearest to the destination in accordance with the longest-prefix matching scheme. Note that the routing cache entries likely to become unreachable due to the node's leaving or failure. If a node passes the message to the next hop of a routing cache entry and finds that it is unreachable, it will reselect the next hop from the default routes.

Once a message has reached the destination node, the destination node records the mapping of the node address of the sender and the IP address in its routing cache. The Laptop utilizes a routing cache maintenance scheme that is similar to the ARP cache of IP networks. Each routing cache entry has an expiration timer, and the content of the entry is removed after the timer expires. Since the size of the routing cache in Laptop is bounded by a predefined constant, *C*_Size, the next hop of a routing cache entry with the lowest level label is removed in the case that the node wishes to add an entry in the routing cache but the routing cache is full. The tie is broken by the *least recently used* (LRU) policy. Now we will show that the length of a routing path in Laptop in bounded by $O(\log_d N)$.

Lemma 4
The length of a routing path is bounded by $O(\log_d N)$.

Proof
In the worst case, a message is first routed to the root, and then routed to a leaf along the overlay tree due to the longest-prefix matching routing. For a balanced tree, the height of the overlay tree is bounded by $\log_d N$. In general, the height of the overlay tree is bounded by $O(\log_d N)$. Therefore, the length of a routing path in Laptop is also bounded by $O(\log_d N)$. □

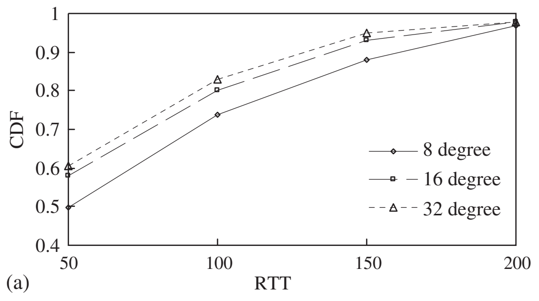
5. PERFORMANCE EVALUATION

This section describes the simulation results for evaluating the performance of Laptop. We designed four experiments to evaluate the performance of Laptop. The first experiment evaluates the clustering effect of Laptop in terms of RTTs and hop counts between nodes. The second experiment shows the routing performance of Laptop under various overlay network sizes. The third experiment presents the location-aware property of Laptop based on the routing stretch. The routing stretch refers to the ratio of the average inter-node latency on the overlay network to that on the underlying IP network. Finally, the performance of failure recovery of Laptop will be shown in the last experiment. Note that node failures are only simulated in the last set of experiments.

The underlying physical network topology is generated by the BRITE topology generator [17]. The network topology consists of 1 000 000 nodes which are uniformly distributed in 50 Autonomous Systems, while each AS consists of 200 local area networks (LANs). Since the BRITE topology generator creates each node within a two-dimension co-ordinate, the delay between any two nodes in our simulation is set proportionally to their Euclidean distance. In addition, Max_Level is set to 4, and RTT intervals for these four segments (from 1 to 4) are set to $[175,\infty]$, $[75,175]$, $[35,75]$, $[0,35]$ in units of ms. The size of the routing cache in each Laptop node, *C*_Size, is set to 64.

The first experiment evaluates the clustering effect of the Laptop overlay network. The Laptop performs well in terms of the clustering effect if each Laptop node finds a physically nearby parent to join. For this reason we evaluate the cumulative distribution function (CDF) of the RTT and the number of hops between nodes and their parent nodes under various values of *d*, the maximum degree limitation of each node. The experiment consists of 10 independent runs, and 1 000 000 Laptop nodes are randomly selected to join the overlay network at each run. Figure 6 shows the simulation results of the RTTs and hop counts between any two nodes under various values of *d*, respectively. As expected, the Laptop performs better in terms of the clustering effect as *d* increases, since a parent node can afford more nearby children nodes for a larger *d*. Figure 6 shows that about 80% of the nodes are within 3 hops from their parent nodes, and 60% of the nodes have RTTs of less than 50 ms to their parent nodes, in the case of $d = 32$. Most nodes are generally within 5 hops from their parent nodes, as shown in Figure 6(b). Therefore, Laptop organizes nodes into a hierarchical, tree-based structure while taking the physical locality into consideration.

The second experiment first examines the average length of the routing paths between any two Laptop nodes in terms of hop count in order to evaluate the routing performance. Figure 7 shows the average path length under various numbers of Laptop nodes in the overlay network. The experiment also includes 10 runs. Within each run 500 000 routing requests are generated, and the source and destination nodes of a routing request are randomly assigned. The routing cache size is set to 64. The average routing path length within 1 000 000 nodes and maximum



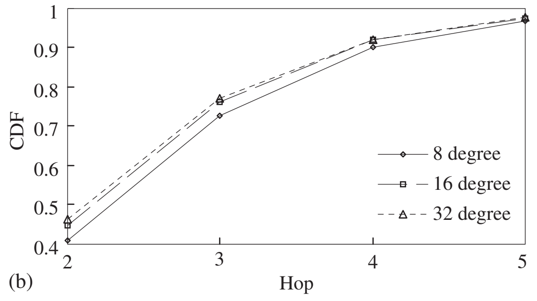


Figure 6. Distribution of RTT and hop counts between a Laptop node and its parent node: (a) CDF of RTT under various degree of peer; and (b) CDF of hop counts under various degree of peer.

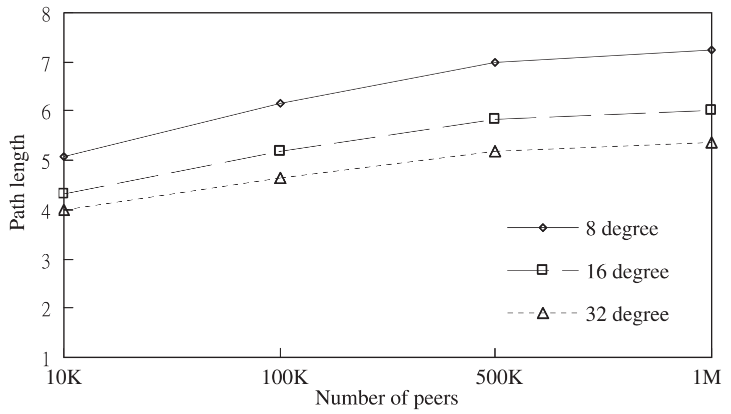


Figure 7. Average routing path length under various system sizes.

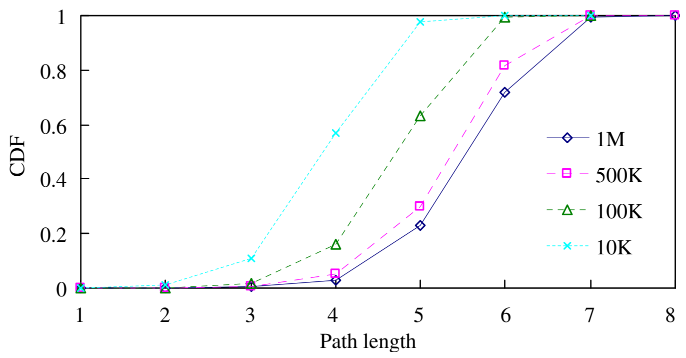


Figure 8. CDF of routing path length under various system sizes for degree of 16.

degree of 32 is only about 5.4, as shown in Figure 7. This shows that routing in Laptop is very scalable in terms of routing path length.

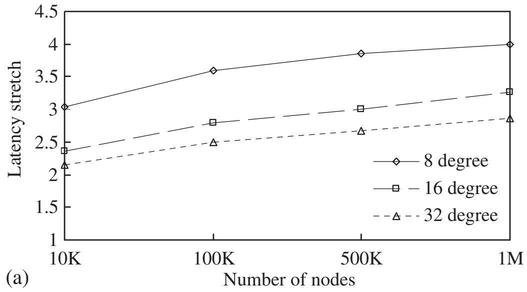
Figure 8 examines the CDF of the routing path length within a maximum degree of 16 under various nodes in the overlay network. It shows that 70% of the routes have a length less than 6 and that the maximum routing path length is about 7 in the case of 1 000 000 nodes in the overlay network and maximum degree of 16. Figure 8 shows the average routing path length is about 5 in this scenario. However, Lemma 4 indicates that the routing path length should be bounded by $\lceil \log_d N \rceil$, which is equivalent to 4 in the case of a balanced tree. Apparently, this variance is due to the unbalanced tree structure of the overlay network.

The longest-prefix matching scheme tends to pass messages toward the upstream of the overlay tree first and then travel down the tree to the destination node. This implies that Laptop may suffer from load imbalance, since nodes within the higher level labels are more likely to forward most messages onto the overlay network. Essentially, Laptop adopts a routing cache to alleviate this load imbalance. We designed an experiment to examine how the routing cache alleviates the load imbalance by observing the average number of forwardings performed by nodes of each level labels. The experiment was run on a 100 000 node network within 1 000 000 randomly generated routing requests. Table I shows the average number of forwarding performed by nodes of each level labels within the experiment of 10 runs. The second column shows the number of nodes at each level. Most nodes are L_4 nodes due to the location-aware property of Laptop. In the case of a cache size of 0, messages are more likely to travel toward the root node and then travel down to the destination nodes. As a result, almost all messages need to pass through the root node. The load can be alleviated significantly with a cache size of 64 or larger as shown in the last two columns. With a routing cache of 64, about 92, 74 and 25%, of the loads are saved at the root, L_1 nodes, and L_2 nodes, respectively. Although the load at the higher level nodes is still much higher than at the L_4 nodes, it is effectively reduced by the routing cache. For instance, only 8.4% of the 1 000 000 routing requests needed to be routed through the root node.

The third experiment examines the routing stretch in terms of delay and hop count. As mentioned before, the routing stretch refers to the ratio of average inter-node latency on the overlay network to that on the underlying IP network. The third experiment is run on the overlay network with 1 000 000 Laptop nodes. The experiment is run 10 times while 500 000

Table I. Load balance of laptop.

Node level	Number of nodes	Cache size 0	Cache size 64	Cache size 256
Root	1	999 996	84 186	77 457
Level 1 Node	12.2	136 884	35 189	17 465
Level 2 Node	26.2	23 653	15 282	9 534
Level 3 Node	286.4	3 183	3 138	3 137
Level 4 Node	99 675.2	23	23	23



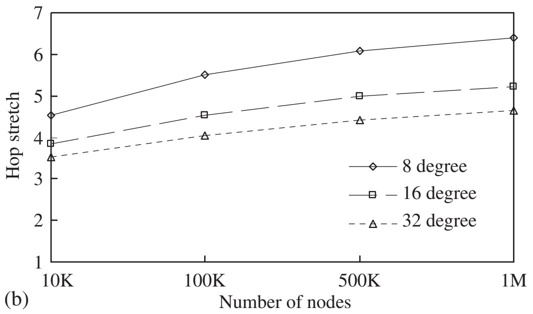


Figure 9. Routing stretch under various network sizes: (a) latency stretch; and (b) hop-count stretch.

routing requests are randomly generated during each run. Figure 9 shows the simulation results of the routing stretch in terms of latency and hop count, respectively. Again, a higher degree will yield a better overlay structure and thus a better routing stretch. Laptop performs quite well at a routing stretch for a system of 1 000 000 nodes. The latency stretch is only about 2.9, and hop-count stretch is about 4.6. Furthermore, as the number of nodes increase, the routing stretch does not grow exponentially. In general, Laptop performs quite scalable in the routing stretch.

The last experiment explores Laptop's behaviour in the presence of node failures. Node failures are simulated on a Laptop network with 1 000 000 nodes and a maximum degree of 32. Failed nodes are randomly selected and only single node failures are simulated, i.e. a new node failure is generated only after the recovery procedure for the previous node failure has been completed. (A node that fails will be failed for the rest of the simulation time.) In the simulation, all single node failures can be recovered so that all messages are routed correctly to their destinations. In this experiment, we evaluate the average routing path as the percentage of failed nodes in the system increases from 0 to 50%. The simulation results are the average of 10 runs. During each run, after the desired number of nodes have failed, 500 000 routing requests are randomly generated. Figure 10 shows the impact of failures on the route quality. The first bar shows the average routing path length (hop count) prior to those failures; the rest of the bars show the average length of the routing path after 10, 25 and 50% of the nodes have failed, respectively. As is evident, even if 50% of the nodes failed in the system, messages are still routed correctly to their destinations with routing paths that are on average only about 1.5 hops longer.

Figure 11 shows the CDF of the routing path length after various percentages of nodes failed. As we can see, for a Laptop system with 1 000 000 nodes, and with 50% of the nodes

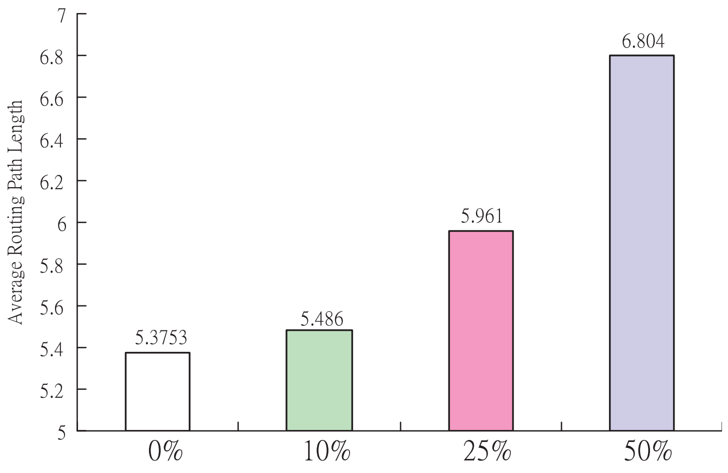


Figure 10. Average routing path under various percentages of nodes failed.

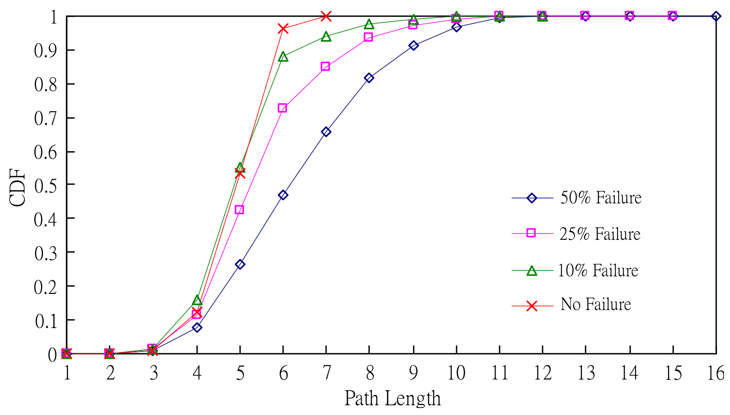


Figure 11. CDF of routing path length under various percentages of nodes failed.

failed, 96.6% of the routes have a length of less than 10, and the maximum length of a routing path is about 16. In summary, the results of Figures 10 and 11 show that Laptop is quite robust to node failures.

6. CONCLUSION

In this paper, we presented Laptop which provides a simple, efficient, scalable and robust peer-to-peer overlay routing service. The performance of Laptop was analysed both mathematically and via simulations. We have shown that the length of a routing path in Laptop is bounded by the height of the overlay tree which is $O(\log_d N)$. The complexity of the joining and leaving procedure is bounded by $O(d \log_d N)$. Simulation results showed that Laptop performs efficient routing for a system with 1 000 000 nodes. Specifically, the average routing path length in the case of $d = 32$ is only about 5.4. Of the routes, 70% have a length less than 6 and the maximum routing path length is about 7, when the maximum degree of a node is set as 16. In addition, locality was explored and taken into consideration in building the Laptop overlay structure. About 80% of the nodes are within 3 hops from their parent nodes, and 60% of the nodes have RTTs of less than 50 ms to their parent nodes, in the case of $d = 32$. The simulation results also demonstrated the robustness of the Laptop system. Even if 50% of the nodes failed in the system, messages are still routed correctly to their destinations, with routing paths that are on average only about 1.5 hop longer.

Laptop differs from other existing P2P systems by its self-organizing location-aware hierarchical overlay structure and its self-addressing, longest-prefix matching routing scheme. Specifically, the contribution of Laptop is that it is a feasible overlay network framework that adopts the location-aware concept. With this fully distributed, highly scalable addressing and routing scheme, Laptop is applicable to application-level multicast applications, such as Overcast [18] and NARADA [19], or other peer-to-peer systems, such as the group communication system [20] and the file storage system [21]. We believe that the concept of Laptop is also applicable to large-scale distributed computing systems, P2P applications in *ad hoc* networks, and others.

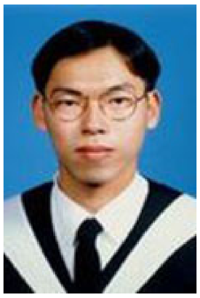
REFERENCES

1. Lv Q, Cao P, Cohen E, Li K, Shenker S. Search and replication in unstructured peer-to-peer networks. *ACM International Conference on Supercomputing*, NY, U.S.A., June 2002.
2. Napster. Napster, http://www.napster.com
3. Gnutella. Gnutella, http://gnutella.wego.com
4. Stoica I, Robert M, Karger D, Kaashoek MF, Balakrishnan H. Chord: a scalable peer-to-peer lookup service for Internet applications. *ACM Special Interest Group on Data Communications* (*SIGCOMM*), San Diego, U.S.A., August 2001.
5. Xu Z, Tang C, Zhang Z. Building topology-aware overlays using global soft-state. *ACM Conference on Principles of Distributed Computing* (*PODC*), Providence, U.S.A., May 2003.
6. Hildrum K, Kubiatowicz J, Rao S, Zhao BY. Distributed object location in a dynamic network. *Fourteenth ACM Symposium on Parallel Algorithms and Architectures* (*SPAA 2002*), Winnipeg, Canada, August 2002.
7. Castro M, Druschel P, Hu YC, Rowstron A. Exploiting network proximity in peer-to-peer overlay networks. *International Workshop on Future Directions in Distributed Computing* (*FuDiCo*), Bertinoro, Italy, June 2002.
8. Ratnasamy S, Stoica I, Shenker S. Routing algorithms for DHTs: some open questions. *First Workshop on Peer-to-Peer Systems* (*PTPS'02*), Cambridge, U.S.A., March 2002.
9. Castro M, Druschel P, Hu YC, Rowstron AI. Topology-aware routing in structured peer-to-peer overlay networks. *Future Directions in Distributed Computing*, Lecture Notes in Computer Science, vol. 2584. Springer: Berlin, 2003.
10. Liben-Nowell D, Balakrishnan H, Karger D. Analysis of the evolution of peer-to-peer systems. *ACM Conference on Principles of Distributed Computing* (*PODC*), Monterey, U.S.A., July 2002.
11. Rowstron A, Druschel P. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. *IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, Germany, November 2001.
12. Zhao BY, Kubiatowicz J, Joseph A. Tapestry: an infrastructure for fault-tolerant wide-area location and routing. *Technical Report*, 2001, UCB/CSD-01-1141.
13. Zhao BY, Duan Y, Huang L, Joseph AD, Kubiatowicz JD. Brocade: landmark routing on overlay networks. *Proceedings of the Workshop on Peer-to-Peer Systems* (*IPTPS 2002*), March 2002.
14. Xu Z, Mahalingam M, Karlsson M. Turing heterogeneity into and advantage in overlay routing. *22nd Annual Joint Conference of IEEE Computer and Communication Societies* (*INFOCOM'03*), San Francisco, U.S.A., March 2003.
15. Datar M. Butterflies and peer-to-peer networks. *10th European Symposium on Algorithms* (*ESA'02*), Rome, Italy, September 2002.
16. Savage S, Collins A, Hoffman E, Snell J, Anderson T. The end-to-end effects of Internet path selection. *SIGCOMM Computer Communication Review* 1999; **29**(4):289–299.

17. Medina A, Lakhina A, Matta I, Byers J. BRITE: an approach to universal topology generation. *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'01)*, Cincinnati, U.S.A., August 2001.
18. Jannotti J, Gifford D, Johnson K, Kaashoek F, O'Toole J. Overcast: reliable multicasting with an overlay network. *USENIC OSDI 2000*, San Diego, U.S.A., October 2000.
19. Chu YH, Rao S, Zhang H. A case for end system multicast. *ACM SIGMETRIC 2000*, Santa Clara, U.S.A., June 2000.
20. Castro M, Druschel P, Kermarrec A-M, Rowstron A. Scribe: a large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC) (Special Issue on Network Support for Multicast Communications)*, October 2002.
21. Kubiatowicz J, Bindel D, Chen Y, Eaton P, Geels D, Gummadi R, Rhea S, Weatherspoon H, Weimer W, Wells C, Zhao B. Oceanstore: an architecture for global-scale persistent storage. *Proceedings of ACM ASPLOS*, ACM, November 2000.

AUTHORS' BIOGRAPHIES

Chi-Jen Wu received his MS degrees in Communication Engineering from National Chung Cheng University, Taiwan in 2004. He is currently a Research Assistant in the Computer Systems and Communication Lab, Institute of Information Science, Academia Sinica, Taiwan. His research interests include overlay network, peer-to-peer network and *ad hoc* network.



De-Kai Liu received his BS and MS degrees from the National Chung-Cheng University, Taiwan, in Computer Science and Information Engineering, in 1997 and 1999, respectively. He joined the PhD program at National Chung-Cheng University in 1999. His current research is focused on P2P networks, mobile *ad hoc* networks, and real-time multimedia transmission.

Ren-Hung Hwang received his BS degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1985, and the MS and PhD degrees in computer science from University of Massachusetts, Amherst, Massachusetts, U.S.A., in 1989 and 1993, respectively.

He joined the Department of Computer Science and Information Engineering, National Chung Cheng University, Chia-Yi, Taiwan, in 1993, where he is a Professor and the chair of the Department of Communications Engineering. His research interests include peer-to-peer applications, *ad hoc* networks, e-learning, and 3G.