Available online at www.sciencedirect.com

SCIENCE DIRECT®

J. Parallel Distrib. Comput. 65 (2005) 207–219

Journal of Parallel and Distributed Computing

www.elsevier.com/locate/jpdc

Analyzing and enhancing the resilience of structured peer-to-peer systems

Shengquan Wang[a], Dong Xuan[b,*], Wei Zhao[a]

[a]Department of Computer Science, Texas A&M University, College Station, TX 77843, USA

[b]Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210, USA

Received 10 September 2004
Available online 19 November 2004

Abstract

In this paper, we propose an approach to analyze the resilience of structured peer-to-peer (P2P) systems under failures. The approach is Markov-chain based, and can be applied to systems with relatively stable size and uniform distribution of nodes. We apply our approach to several well-known structured P2P systems. We find that different system features (types of neighbors of nodes) in P2P systems have different impacts on their resilience against failures. Following this observation, we propose to add some extra neighbor(s) to CAN using small-world model principles to form a so-called CAN-SW system. We then apply the proposed approach to analyze its resilience. We find that the performance is improved significantly, particularly, in terms of the average path length.

Keywords: Peer-to-Peer; Resilience; Small-world model

1. Introduction

A peer-to-peer (P2P) system is a group of Internet nodes which construct their own special-purpose networks on top of the Internet. Such a system performs application-level routing on top of IP routing. There are two types of P2P systems: structured and unstructured P2P systems. Structured P2P systems are ones in which nodes organize themselves in an orderly fashion while unstructured P2P systems are ones in which nodes organize themselves randomly. Structured P2P systems boast an efficient lookup mechanism by means of distributed hash tables (DHTs) while unstructured P2P systems mostly use broadcast search. This paper focuses on structured P2P systems and investigates their behavior in the event of failures occurring in the network.

Consider a few scalable and structured P2P systems that support a DHT functionality such as CAN/Chord/Tapestry/Pastry. Because of their rich structural arrangement, they have efficient key lookups and are also somewhat resilient to failures of nodes. However, almost all the protocols are designed with an ideal overlay structure under which the key lookups are efficient. P2P nodes are notoriously transient and the resilience of routing under node failures is a very important consideration. This is a serious concern since performance constraints in an ad hoc network are critical to the survival of the network itself. If the system is robust, more nodes tend to participate in the network and will assist in improving the robustness. So an analytical study of the resilience of P2P networks is necessary for the continued development of such systems. Our focus in this paper is to study the resilience of structured P2P systems under node failures.

In the framework of large distributed systems, some papers address system resilience under failures with minimal overhead. For example, the Anthill project attempts to explore the possibilities of improving the efficiency of large-scale distributed systems under failures [1]. However, to the best of our knowledge, there is no approach proposed to analyze resilience of structured P2P systems under failures which is what we want to address here. Our goal is to systematically analyze features of resilience of current structured P2P systems under failures in term of the average path

* Corresponding author. Fax: +1 614 292 2911.
E-mail addresses: swang@cs.tamu.edu (S. Wang), xuan@cis.ohio-state.edu (D. Xuan), zhao@cs.tamu.edu (W. Zhao).

0743-7315/$ - see front matter © 2004 Published by Elsevier Inc.
doi:10.1016/j.jpdc.2004.09.009

length and hit ratio, and to understand the features that lead to improved system resilience. The average path length is the average value of the path length taken by uniformly randomly generated queries in the P2P system. The hit ratio is the ratio of the number of queries that eventually can be routed to the correct destinations to the total number of queries and under node failures it is not necessarily 100 percent.

In our attempt to analyze the resilience of P2P systems, we first propose a general Markov-chain-based analytical approach, and then apply it to analyze the resilience performance (in terms of the average path length and the average hit ratio) for various structured P2P systems (CAN, Chord, Tapestry and Pastry) under node failures. We find that different system features (types of neighbors of nodes) in P2P systems have different impacts on their resilience under failures. Following this observation, we propose to add some extra neighbor(s) to CAN using small-world model principles to form a so-called CAN-SW system. We then apply the proposed approach to analysis its resilience. We find the performance is improved significantly, particularly in terms of the average path length.

The rest of the paper is organized as follows: Section 2 gives a brief introduction to the structure P2P systems. Section 3 provides our approach for analyzing the resilience to failure of the P2P systems. Section 4 presents the theoretical results for the resilience of various structure P2P systems. Section 5 introduces the *small-world phenomenon* and the methodology of implementing it in CAN. In Section 6, we conclude the paper by highlighting the scope for future work.

2. Structured P2P systems

In this section, we will review some of the existing well-known P2P systems. For the sake of brevity we consider four popular P2P systems.

Sylvia et al. [11] proposed the content addressable networks (CAN) as a distributed infrastructure that provides hash table-like functionality on internet-like scales. It models the participating peers as zones in a d-dimensional toroidal space. Each node is associated with a hyper-cubal region of this key space, and its neighbors are the nodes which are associated with the adjoining hybercubes. Routing consists of forwarding to a neighbor that is closer to the key (in the toroidal space). Its performance is determined by the number of dimensions used d. For a d-dimensional space partitioned into n-equal zones, the average routing path length is $\frac{d}{4}n^{\frac{1}{d}}$ hops and individual nodes maintain $2d$ neighbors. These scaling results mean that for a d-dimensional space, we can grow the number of nodes (and hence zones) without increasing per node state while the average path length grows as $\mathcal{O}(dn^{\frac{1}{d}})$. In CAN system, multiple paths exist between two points in the space and so, even if one or more of a node's neighbors were to crash, a node can automatically route along the next best available path.

Unlike CAN, Chord [15] uses a single-dimensional circular key space. The node responsible for the key is the one whose identifier most closely follows the key (numerically); that node is called the key's successor. Chord maintains two sets of neighbors. Each node has a successor list that immediately follows it in the key space. The neighbor list also called as the finger table is constructed with nodes which are at distance in powers of 2, i.e., the nodes at distances $(n + 2^{i-1})$ where $1 \leqslant i \leqslant \log n$. Chord's routing procedure can be thought of as an one-dimensional analog of the Grid location system [4]. Routing efficiency is achieved with the finger list of $\log n$ nodes-spaced exponentially around the key space. Routing consists of forwarding to the node closest, but not past the key; in the event of failures, the Chord protocol runs a stabilization algorithm to recover from it using successor lists.

In Pastry [14], nodes are responsible for keys that are the closest numerically (with the keyspace considered as a circle similar to Chord). The neighbors consist of a $LeafSet,\ L$ which is the set of L closest nodes (half-larger, half-smaller). Correct, not necessarily efficient, routing will be achieved with this leaf set. To achieve more efficient routing, Pastry has another set of neighbors spread in the key space. Also, Pastry takes into account network locality; it seeks to minimize the distance messages travel, according to a scalar proximity metric like the number of IP routing hops. Each Pastry node keeps track of its immediate neighbors in the node ID space, and notifies applications of new node arrivals, node failures and recoveries. Because node IDs are randomly assigned, with high probability, the set of nodes with adjacent node ID is diverse in geography, ownership, jurisdiction, etc. Applications can leverage this, as Pastry can route to one of k nodes that are numerically closest to the key. A heuristic ensures that among a set of nodes with the k closest node IDs to the key, the message is likely to first reach a closest node in terms of the proximity metric. Routing consists of forwarding the query to the neighboring node that has the longest shared prefix with key (and, in the case of ties, to the node with identifier closest numerically to the key). In Pastry, in the event of node failures, it uses the repairment algorithm discussed in [14].

Tapestry [19] is an overlay location and routing infrastructure that provides location-independent routing of messages directly to the closest copy of an object or service using only point-to-point links and without centralized resources. The routing and directory information within this infrastructure is purely soft state. The nodes maintain the routing table called as $neighbormaps$. Its routing along the nodes, follow the longest prefix routing similar to the one in the CIDR IP address allocation architecture. The Tapestry location mechanism is similar to the Plaxton location scheme [10]. While multiple copies of data exist in Plaxton's scheme, each node en route to the root node only stores the location of the closest replica to it. Tapestry, however, stores locations of all

Table 1
The routing table size and path length of well-known structured P2P systems

Systems	Routing table size	Path length
CAN	$\mathcal{O}(d)$	$\mathcal{O}(dn^{\frac{1}{d}})$
Chord	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
Tapestry	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
Pastry	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$

such replicas to increase semantic flexibility. While the Plaxton mechanism always returns the first object within some distance, Tapestry location provides more semantic flexibility, by allowing the application to define the selection operator. Each object may include an optional application-specific metric in addition to a distance metric.

Table 1 provides a compilation of the routing table sizes and the lookup path lengths of the four systems discussed above. The bounds for the path length in Table 1 hold for a perfectly functioning system with all nodes operational. However, P2P nodes are notoriously transient and the resilience of routing to failures is a very important consideration.

3. An analytical approach to resilience of structured P2P systems

In this section, we will describe our approach to analyze resilience of structure P2P systems under failures. First, we will introduce our system model. Following that, we will present a Markov-chain-based approach and derive formulas to compute the average hit ratio and average path length with the model we describe. Finally, we will take CAN as an example to describe the steps in computing the transition probability.

3.1. Model

The structured P2P systems described above differ from each other mainly on how to organize peer nodes into neighbors. Neighborship is a key feature in P2P systems. Node B is said to be the neighbor of node A if node B is sending and receiving requests directly to and from node A. A node assumes its neighbor as a failed neighbor if the node cannot communicate with the neighbor and the neighbor cannot function. The failure of a neighboring node may be due to the departure or crashing of the neighbor or due to some error in the lower-layer communication. We are interested in analyzing the resilience so that routing can continue to function even with some failed neighbors.

For any node, there are two types of neighbors [12] as shown in Fig. 1: *special neighbor*—they are re-established shortly after a failure; *finger neighbor*—they are never re-established or re-established in a long while after a failure. The examples of special neighbors are the neighbors in CAN, the successor list in Chord and the leaf set in Pastry. The example of finger neighbors are the entries in the

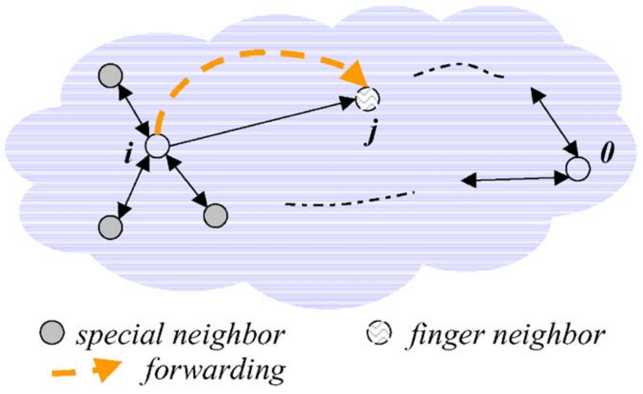

Fig. 1. Special and finger neighbors.

finger table of Chord and in the routing table of Pastry. Both types of neighbors are checked periodically for existence by means of polling messages. However, the time intervals of polling might differ between special and finger neighbors. A node sends polling messages to special neighbors more frequently than to finger neighbors. The presence of the special neighbors can allow one to prove the correctness of routing with high maintenance overhead. The finger neighbors can assist in improving the routing performance with small maintenance overhead, but are not guaranteed to be always functional. In this paper, we define τ_1 and τ_2 as the probabilities that a special neighbor and a finger neighbor are in failure states at any point of time, respectively. From the previous discussions, we then have $\tau_1 < \tau_2$.[1]

In this study, we assume the system is stable. By "stable", we mean that the number of nodes in the system are almost constant (the number of joining nodes and leaving nodes are almost same in a short period) and all nodes are uniformly distributed in the system (achieved by distributed hashing). With these assumptions, the neighborhood of any node in the system does not change very much over time. For any node, a departing neighbor will be replaced by a newly joining node or one of the other existing nodes as a result of recovery process. From the routing perspective, such a replacement can be regarded as an up/down process of a neighbor of that node. We are interested only in analyzing the average performance of a P2P system, and so the neighbors of any node can be regarded as being fixed with up/down state irrespective of whether the nodes keep leaving and joining. With these assumptions, a P2P system's whole routing process can be modeled as a stochastic process even though the nodes in the system are dynamic.

In this study, we are interested in studying the impact of different neighbors on the resilience of structured P2P systems under the failure model described above. The

[1] Generally, values of τ_1 and τ_2 vary node-by-node. In our evaluation, for simplicity, we choose the same value of τ_1 for all nodes and do the same for τ_2.

performance metrics are the average path length and the average hit ratio.

3.2. A Markov-chain-based analytical approach

As we know, in P2P systems, during the data looking-up process, a node forwards a message (data request) to the next node just based on its current status. The looking-up process can be formalized as a discrete absorbing *Markov chain* as follows:

- Define a stochastic process $\{X_h : h = 0, 1, \ldots\}$, where random variable X_h is the state of a message forwarding during a looking-up process. Specifically, X_h can be a "failure" state or the location of the message (i.e., the node ID of the message holder) after the message is forwarded to the hth hop.[2] Due to the property of the decentralized routing algorithm, $\{X_h : h = 0, 1, \ldots\}$ satisfies

$$\Pr\{X_h = i_h | X_{h-1} = i_{h-1}, \ldots, X_0 = i_0\} = \Pr\{X_h = i_h | X_{h-1} = i_{h-1}\}, \tag{1}$$

  where $i_h \in \mathcal{S}$ and $\mathcal{S}$ is the state–space which includes all node IDs and a "failure" state.
- The message is forwarded to the destination or dropped finally (We call these as "destination state" and "failure state", respectively). These two states are absorbing and the others are transient.[3]

In the following, we will use the feature of the Markov chain to compute the average hit-ratio and the average path length of any source to one particular destination node. We assume that in a P2P system, each node has a unique ID. In a system with n nodes, the nodes are named starting from 0 to $n-1$. Without loss of generality, we consider node 0 as the destination, and all other nodes $1, 2, \ldots, n-1$ as sources. We define state i as the state that the message is at node i, $i = 0, 1, \ldots, n-1$. We denote n as the "failure" state. That is, if the message cannot be forwarded to any node in the system, it will be dropped and virtually put into n. We know that $1, 2, \ldots, n-1$ are transient states and $0, n$ are absorbing states.

We define the *transition probability* as $p_{i,j} = \Pr[X_h = j | X_{h-1} = i]$. We have the matrix of transition probabilities $P = (p_{i,j})_{(n+1)\times(n+1)}$. We can write P as follows:

$$P = \begin{pmatrix} 1 & 0\cdots0 & 0 \\ U & Q & V \\ 0 & 0\cdots0 & 1 \end{pmatrix}, \tag{2}$$

where Q is an $(n-1)\times(n-1)$ matrix that delineates the transition rates between the transient states $1, 2, \ldots, n-1$, U is $(n-1)\times 1$ column matrix that delineates the transition rates between all transient states and the absorbing state 0, and V is $(n-1)\times 1$ column matrix that delineates the transition rates between all transient states and the absorbing state n.

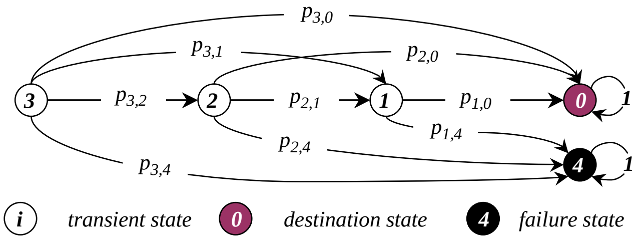

Fig. 2. A discrete absorbing Markov process.

An example of a discrete absorbing Markov process with transition probabilities is illustrated in Fig. 2, where Q, U, and V are expressed as follows:

$$Q = \begin{pmatrix} 0 & 0 & 0 \\ p_{2,1} & 0 & 0 \\ p_{3,1} & p_{3,2} & 0 \end{pmatrix}, U = \begin{pmatrix} p_{1,0} \\ p_{2,0} \\ p_{3,0} \end{pmatrix}, V = \begin{pmatrix} p_{1,4} \\ p_{2,4} \\ p_{3,4} \end{pmatrix}. \tag{3}$$

We define $a_{i,j}$ as the probability that a message is successfully forwarded from a transient state i ($i = 1, 2, \ldots, n-1$) to an absorbing state j ($j = 0, n-1$), and $m_{i,j}$ as the average number of steps for the successful forwarding from state i to j. Obviously, these two values are the average hit ratio and the average path length from state i to j, respectively. Fortunately, computing these two values is a typical absorbing probability problem and a mean first passage time problem, respectively. By the analysis in [13], we have

$$a_{i,j} = p_{i,j} + \sum_{i'=1}^{n-1} p_{i,i'} a_{i',j}, \tag{4}$$

$$a_{i,j} m_{i,j} = a_{i,j} + \sum_{i'=1}^{n-1} p_{i,i'} a_{i',j} m_{i',j}. \tag{5}$$

Especially, for the state 0, if we define $A = (a_{1,0}, a_{2,0}, \ldots, a_{n-1,0})^{\perp}$ and $M = (m_{1,0}, m_{2,0}, \ldots, m_{n-1,0})^{\perp}$, then we have[4]

$$A = (I-Q)^{-1} U, \tag{6}$$
$$M = ((I-Q)^{-1} A) \div A. \tag{7}$$

By (6) and (7), we can obtain the average hit ratio and the average path lengths from transient state i ($i = 1, 2, \ldots, n-1$) to destination state j ($j = 0, n-1$). We assume that the sources are uniformly distributed to all nodes ($0, 1, \ldots, n-1$). Also we assume that the average hit ratio from 0 to 0 is 1, and the average path length from 0 to 0 is 0. Therefore the average hit ratio and the average path length from any source to the destination 0 are given in the following theorem:

[2] Here, we slightly abuse the notation of node ID and the state.

[3] If messages can retry after previous trials fail, we can model it as a discrete Markov chain with only one absorbing state (destination state). The analysis is almost similar except that we have to remove state n and replace $p_{i,i}$ with $p_{i,n}$ in computing transition probabilities.

[4] We define a new matrix operation $Z = X \div Y$ as $z_{i,j} = x_{i,j}/y_{i,j}$ for each i and j.

Theorem 1. *The average* hit ratio $\bar{a}$ *and the average path length* $\bar{m}$ *for the algorithm sending message from any source to the destination* 0 *can be described, respectively, as follows*:

$$\bar{a} = \frac{1}{n}(\pi^0 A + 1), \tag{8}$$

$$\bar{m} = \frac{1}{n}\pi^0 M, \tag{9}$$

where A, M *are defined in* (6) *and* (7), *respectively, and* $\pi^0 = (1, 1, \ldots, 1)$.

Based on (8) and (9), we can compute the average hit ratio and the average path lengths to node 0 from any other node. Recall that we made no assumption on node 0, and the derivation of the above formulas does not use any particular feature of this node. Hence, the above formula applies to compute the average hit ratio and the average path lengths to any node i from other nodes. If a system is symmetric such that all nodes are equivalent in term of looking up, the results of formulas are automatically the average hit ratio and the average path length of all the system. If a system is not symmetric, the overall performance can be obtained based the above formulas and the specific features of the systems.

The remaining problem is how to obtain the matrix of transition probability P. We will introduce the computation of P in the following sub section.

3.3. Computing transition probabilities

In this section, we will discuss how to compute the transition probability $p_{i,j}$ from node i to j for a given P2P system. For each node i, we need compute the individual probability that node i will forward the message directly to other node j. Obviously, if node j is not a neighbor, the probability of forwarding is 0. To the other nodes, the following steps need to be taken to compute the transition probabilities:

- Determine the set of neighbors, including special neighbors and finger neighbors. Recall that these two types of neighbors have different failure probabilities.
- Follow the specific routing semantics of the given P2P system to determine the probability of forwarding to each neighbor. Different P2P systems have different routing policy. Care has to be taken to make sure the routing policy is honored.

In the following, we would like to use CAN as an example (see Fig. 3) to illustrate how to compute the transition probabilities following the above steps:

- Given node i, first let us determine the neighbor set of node i. Recall that the destination is defined as 0. Define $l(i)$ as the lattice distance from i to destination 0 and $l(i,j)$ as the lattice distance from i to j. Define $N(i) = \{i' : l(i') = l(i) - 1 \wedge l(i,i') = 1\}$. $N(i)$ is the neighbor set of node i. As we know, all nodes in $N(i)$ are *special neighbors*, and each is operational with probability τ_1. Their lattice distances are same under our assumption.
- Having determined the neighbor set, let us discuss how to follow the routing semantics of CAN to determine the forwarding probability of different neighbors. As we know, in CAN system, node i can uniformly choose one of the operational neighbors, and each node in $N(i)$ will be chosen as the next hop with probability $(1-\tau_1^{|N(i)|})\frac{1}{|N(i)|}$. As we know, if all neighbors are down, the message will be dropped at node i, hence $p_{i,n} = \tau_1^{N(i)}$.

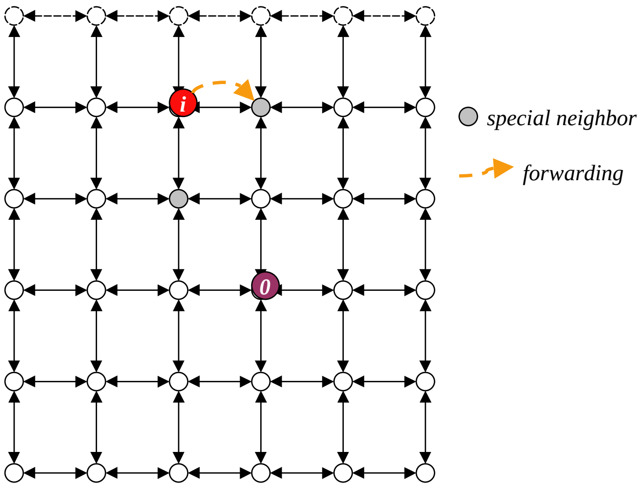

Fig. 3. An illustration of message forwarding in CAN.

Therefore, for CAN system, the transition probabilities can be described as

$$p_{i,j} = \begin{cases} (1-\tau_1^{|N(i)|})\frac{1}{|N(i)|}, & j \in N(i), \\ \tau_1^{N(i)}, & j = n, \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

for $i = 1, 2, \ldots, n-1$.

4. Analysis of structured P2P systems

4.1. Individual P2P systems

In the following section, we follow the steps proposed in the above to compute transition probabilities for some of the other well-known structured P2P systems. For each node i, we want to know, given any node j, the probability that the message will be forwarded from holder i to j directly. As we know, CAN, Chord, Tapestry and Pastry all use greedy forwarding, i.e., they forward the message to the next hop which is one of the closest neighbors to the destination. During the computation, we first consider the closest neighbors. The faults in routing tables introduced by node failures must be taken into account in the computation of transition probabilities.

Assume all nodes in Chord are located in the ring in clock-wise direction with indices ordered increasingly. Each node i will have a routing table which maintains all special neighbors[5] $(i-1)|_n$, $(i+1)|_n$, and finger neighbors $(i+2^k)|_n$, where $k = 1, 2, \ldots, \log n$. Given node i, assume that $2^{k_0} \leqslant n - i < 2^{k_0+1}$, the candidates of the next hop is $\{(i+2^k)|_n : k = 1, \ldots, k_0\}$. During forwarding, it will choose one of the closest operational neighbors in clockwise direction, i.e., in the order: $(i+2^{k_0})|_n, \ldots, (i+2^1)|_n, (i+1)|_n$. Therefore, for transition probability $p_{i,j}$, consider these kind of order and the fault probability τ_1 (special neighbors: successor nodes) and τ_2 (finger neighbors), we have

$$p_{i,j} = \begin{cases} (1-\tau_2)\tau_2^{k_0-k}, & j=(i+2^k)|_n,\ k = 1, \ldots, k_0, \\ (1-\tau_1)\tau_2^{k_0}, & j=(i+1)|_n, \\ \tau_1\tau_2^{k_0}, & j = n, \\ 0 & \text{otherwise.} \end{cases} \quad (11)$$

Both Pastry and Tapstry are based on Plaxton's tree [10]. Pastry uses the prefix-matching mechanism with base b to find the next hop in a routing table. A node in Pastry maintains a set of *leaf* neighbors, which consists of L nodes closest to itself. For node i, IDs of its leaf neighbors are $i \pm l$, where $l = 1, 2, \ldots, \frac{L}{2}$. During routing, the node will choose the closest operational neighbors in a clockwise direction.[6] Here, we assume $b = 2$. For transition probability $p_{i,j}$, assuming that $2^{k_0} \leqslant i < 2^{k_0+1}$ and $2^{k_1} \leqslant \frac{L}{2} < 2^{k_1+1}$, we have

- $k_0 > k_1$:

$$p_{i,j} = \begin{cases} (1-\tau_2)\tau_2^{k_0-k}, & j = (i-2^k)|_n, \\ & k = k_1+1, \ldots, k_0, \\ (1-\tau_1)\tau_1^{\frac{L}{2}-l}\tau_2^{k_0-k_1}, & j = (i-l)|_n, \\ & l = 1, \ldots, \frac{L}{2}, \\ \tau_1^{\frac{L}{2}}\tau_2^{k_0-k_1}, & j = n, \\ 0 & \text{otherwise,} \end{cases} \quad (12)$$

- $k_0 \leqslant k_1$:

$$p_{i,j} = \begin{cases} (1-\tau_1)\tau_1^{i-l}, & j = i-l,\ l = 1, \ldots, i, \\ \tau_1^i, & j = n, \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

Tapestry also uses the prefix-matching mechanism with base b to find the next hop in a routing table. If we choose $L = 1$ in Pastry, we can map the routing in Pastry into Tapestry. The transition probabilities in Tapestry are the same as the ones in Pastry. We note that, due to the symmetry of the ring, as $b = 2$, the matrix of transition probability is totally same as that of Chord.

4.2. Analysis results and observations

We apply the above formulas to compute the average path length and the average hit ratio for CAN, Chord, Pastry and Tapestry systems, using MATLAB. We run the systems with the number of nodes from 64 to 4096. The data of other systems reflect the same features. In our computation, for Pastry and Tapestry, we choose 2 as the base. In this case, Chord and Tapestry are special cases of Pastry on routing performance under this condition. So, we only report the results for Pastry including this special case. We vary the dimension number d for CAN and the leaf set size L for Pastry to change the average number of special neighbors in a given system. There are no finger neighbors for CAN system but there exist $\mathcal{O}(\log n)$ finger neighbors for Pastry.

Figs. 4 and 5 illustrate the sensitivity of the average path length and the average hit ratio to the failure of the special neighbors (τ_1 is the probability that the special neighbor is in the failure state) and the finger neighbors (τ_2 is the probability that the finger neighbor is in the failure state) for CAN system and Pastry system, respectively.[7]

From these figures, we have the following observations:

- *Finger neighbors:* For CAN system, since there are no finger neighbors, both the average hit ratio and the average path length remain constant as τ_2 changes for different dimensions, which is shown in Fig. 4(a). Fig. 5(a) shows the sensitivity of the routing performance to τ_1, given τ_2 for Pastry system. For different sizes of the leaf sets, the average hit ratio is not sensitive to the change of τ_2, but the average path length is very sensitive to such a change.
- *Special neighbors:* Figs. 4(b) and 5(b) show the sensitivity of routing performance to τ_1, given τ_2. As τ_1 increases, the average hit ratio for both CAN and Pastry decreases for different number of special neighbors (i.e., $2d$ in CAN and L in Pastry), but the average path length remains almost constant.

In summary, we can conclude that

- *Resilience to failures of finger neighbors:* The average path length is very sensitive to the failure of finger neighbors, but not the average hit ratio. This is independent of the system model and the number of special neighbors used. Hence, the finger neighbors have significant impact on resilience to failures in terms of the average path length.
- *Resilience to failures of special neighbors:* The average hit ratio is very sensitive to the failure of special neighbors, but not the average path length. This is independent of the system model and the number of special neighbors used. Hence, the special neighbors have significant impact on resilience to failures in terms of the average hit ratio.

As we know, among the four systems we study in this paper, i.e. CAN, Chord, Pastry and Tapestry, only CAN has no finger neighbors. In the following sections, we propose to add some finger neighbors to CAN system to enhance its resilience to failures in terms of the average path length.

[5] We use $x|_n$ to denote $x \bmod n$.

[6] The neighborhood set will not be addressed in this paper.

[7] Generally speaking, $\tau_1 < \tau_2$. For the purpose of illustration, we also show the data in the case that $\tau_1 > \tau_2$.

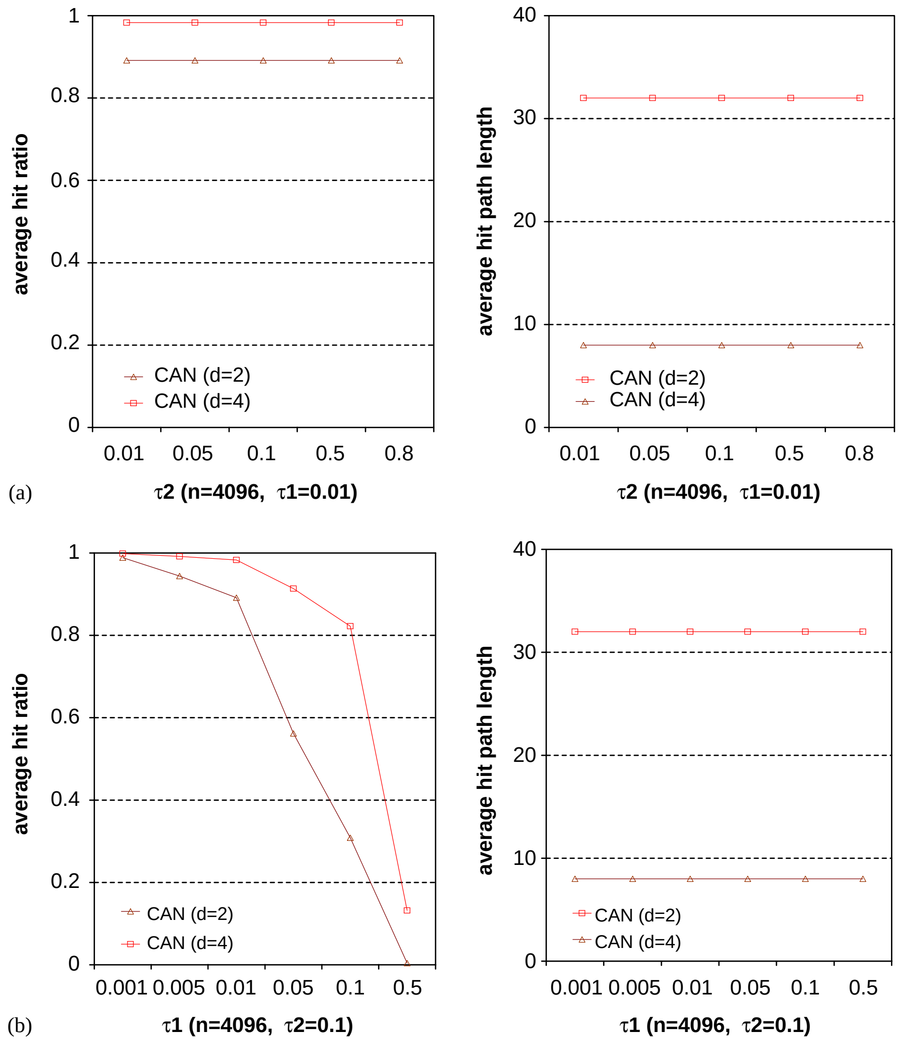

Fig. 4. Performance evaluation of resilience to failure of finger and special neighbors for CAN: (a) finger neighbor and (b) special neighbor.

5. Improving CAN with small world

The CAN system as discussed in the previous sections uses its local neighbors to route the information towards the destination. This means that the request gradually approaches the destination which in other words would mean that it consumes more number of hops. However, if we redesign the CAN system to have bypass routes or shortcuts, it is intuitive that the average number of hops would decrease considerably. For redesigning the CAN system with such bypass routes we augment CAN with the *small-world* phenomenon.

5.1. Small-world model

The small-world phenomenon, the principle that most of us are linked by short chains of acquaintances was first investigated as a question in sociology by Milgram in the 1960s [7]. It is now a feature in a range of networks arising in nature and technology. A network can exhibit the small-world phenomenon thanks to the presence of a few long-range/remote edges that link parts of the network that would otherwise be far apart. One network construction that gives rise to small-world behavior is one in which each node in the network knows its local neighbors, as well as a small number of randomly chosen remote nodes. The latter represent shortcuts in the network. It has been shown that this construction leads to graphs with small diameter, leading to a small routing distance between any two individuals [16].

Kleinberg [3] proved that there exists only one model within an infinite family of network models for which a decentralized algorithm exists to find the shortest paths with high probability. In Kleinberg's model, a set of nodes are identified with the set of lattice points in an mesh. In this paper, we extend the mesh to a d-dimensional torus. Assume that torus is of size m^d, where $m^d = n$ and m is the width of each dimension for the torus. The lattice distance between

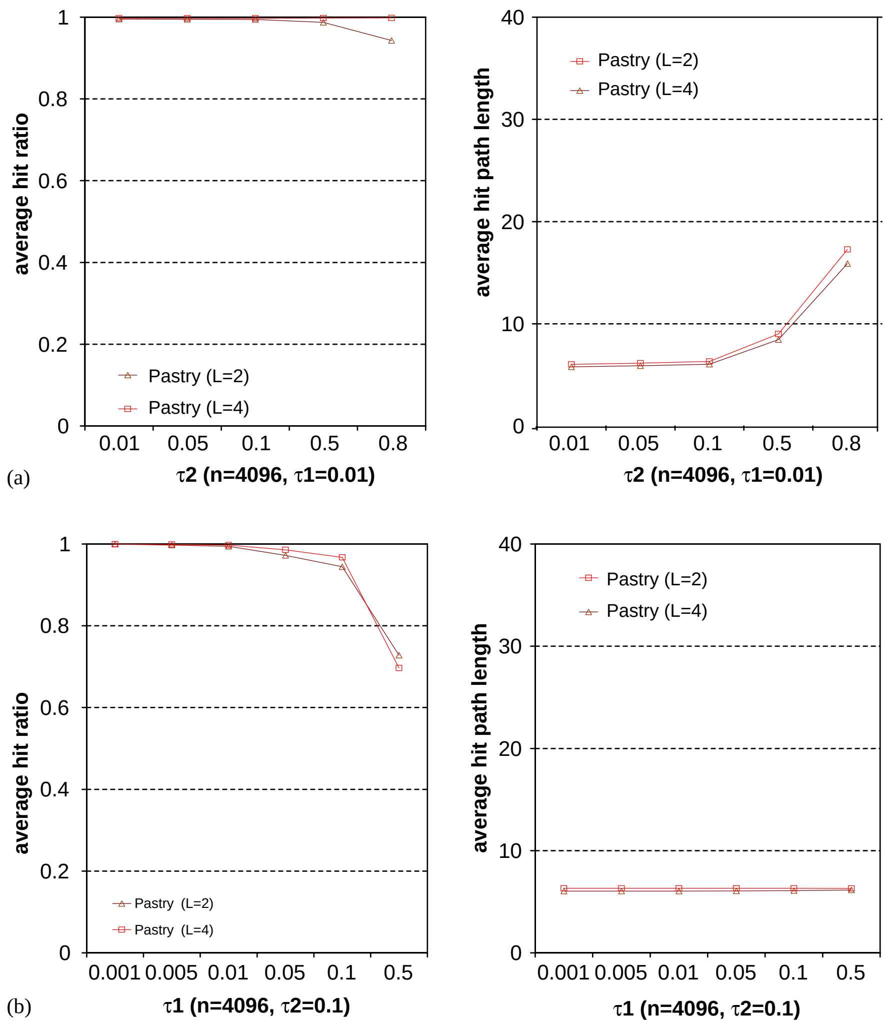

Fig. 5. Performance evaluation of resilience to failure of finger and special neighbors for Pastry: (a) finger neighbor and (b) special neighbor.

two nodes i and j is defined to be the minimum number of "lattice hops" separating them: $l(i, j) = \sum_{k=1}^{d} \|i_k - i_k\|$.[8] Each node i has a directed edge to every other node within lattice distance 1, which are its local neighbors. It also has a directed edge from i to another node j with probability proportional to $l^{-r}(i, j)$.

The remote neighbors can improve the average path length dramatically as shown in [3]: *As $r = d$, this mechanism can reduce the average path length to polynomial in $\mathcal{O}(\log^2 n)$.*

Having said the advantages of using the *small-world* phenomenon, we now propose a method of using it in CAN and investigate its effect on the performance of CAN in the following sections.

5.2. CAN-SW

In the basic CAN system, there are three key issues in the looking-up system: (1) CAN construction; (2) routing mechanism; (3) CAN maintenance. In our CAN-SW, we also focus on these issues and address differences of our design.

5.2.1. Construction

The process of CAN construction can be done in four steps: (i) First, the new node must find a node already in the CAN system. Then a point is randomly chosen from the key space (i.e. the d-dimensional torus). (ii) Next, using the routing mechanisms, it must find the node which owns that point and its zone is split. (iii) The split portion of the original zone where the point lies is now taken by the new node. (iv) Finally, the neighbors of the split zone must be notified so that routing can include the new node.

Here construction of the routing-table is a critical issue. In the original design of the CAN system, each node maintains as its set of neighbors, the IP addresses of those nodes that hold coordinate zones adjoining its own zone. In our designed system, besides local neighbors, each node will

[8] We define $\|x\| = \min\{|x|, m - |x|\}$.

maintain an additional remote neighbor. The remote neighbor v of node u is chosen randomly with probability $\alpha_{i,j} = \frac{l^{-d}(i,j)}{\Delta}$, where $\Delta = \sum_{i' \neq i} l^{-d}(i, i')$.

In order to choose remote neighbor, we need to know the size of system (i.e., the number of nodes/zones) and the topology of the basic CAN system. Due to the dynamics of the construction of CAN, the topology of the basic CAN system is not regular. For simplicity, we assume that the topology is a d-dimensional torus. We can use the volume of individual zone to measure the system size. For example, for node i associated with zone Z_i, define $[Z_i]$ as its volume. Now, we know that the overall system volume is 1. So the system size is estimated to be $n \approx m^d$, where $m^d \leqslant \frac{1}{[Z_i]} < (m+1)^d$. Once n is known, based on node i, a remote point v can be generated with probability $\alpha_{i,j}$. Using the routing mechanism, the remote zone where j is located can be found.

The addition of a new node affects only a small number of existing nodes in a very small locality of the coordinate space. The number of neighbors a node maintains depends only on the dimension d of the coordinate space and the number of remote neighbors,[9] and is independent of the total number of nodes in the system. Thus, node insertion affects only $\mathcal{O}(d)$ existing nodes, which is important for CAN-SW with huge numbers of nodes.

It can be easily inferred that there is no big difference between the update messages and data-request forwarding messages. In the operational system, the number of the former is significantly smaller than the number of the latter. Therefore, the introduced overhead from these messages is negligible. But the number of nodes whose routing table should be updated is a big overhead for joining nodes, which will affect the system performance.

In CAN-SW, if each node constructs multiple remote neighbor independently, it can be inferred that the performance would improve, however, only with an increased overhead in system maintenance. As we know, in the original CAN system, during node's joining and departure, the corresponding zone will be split or will be merged with another zone, and the routing tables of this zone and its neighbor have to be updated correspondingly. Obviously, this zone's volume will change. As we mentioned above, the change of a zone's volume will affect the measure of the system size. Therefore, the original node and the new node (in the case of node joining) and the taking over node in the case of node departure along with their neighbors (including both local and remote neighbors) have to regenerate their remote neighbor.

Shortcuts introduced by the remote neighbors, in real life are hot links. Message will be forwarded through these links with higher probability than other links. Introducing multiple remote neighbors for any node, on the other hand, can also improve load balancing very much, although it has been achieved in some degree due to the randomness of choosing remote neighbor in the single remote neighbor case.

[9] So far, we assume the number of remote neighbors is 1. In the extended version, it may be a constant number larger than 1.

5.2.2. Routing

Intuitively, routing in CAN works by following the straight line path through the Cartesian space from source to destination coordinates.

In the CAN system, each node maintains a coordinate routing table that holds the IP address and virtual coordinate zone of each of its immediate neighbors in the coordinate space. In a d-dimensional coordinate space, two nodes are neighbors if their coordinate spans overlap along $d-1$ dimensions and abut along the other dimension. This purely local neighbor state is sufficient to route between two arbitrary points in the space: A CAN message includes the destination coordinates. Using its neighbor coordinate set, a node routes a message towards its destination by simple greedily forwarding to the neighbor with coordinates closest to the destination coordinates. It is a decentralized greedy routing algorithm.

In the enhanced system, each node, apart from maintaining the information about its local neighbors, stores the information about a remote node (i.e., remote neighbor) determined using our technique described above. However, we still follow the same routing mechanism: *in each step, the current message-holder chooses as the next hop a neighbor* (*a local one or a remote one*) *that is as close to the destination t as possible, in the sense of lattice distance*. For the neighbors with same distance to destination, the current message-holder can randomly pick one. In case that one or more of a node's neighbors were to crash, a node can automatically route along the next best available path.

5.2.3. Maintenance

In the CAN system, when nodes leave a CAN system, the zones they occupied are taken over by one of the remaining nodes. The CAN system also needs to be robust to node or network failures, where one or more nodes simply become unreachable. This is handled through an immediate takeover algorithm that ensures one of the failed node's neighbors takes over the zone. However, in this case the $(key, value)$ pairs held by the departing node are lost until the state is refreshed by the holders of the data. In the enhanced CAN system, the only difference is that the maintenance of remote neighbor nodes has to be taken into account same as local neighbors.

5.3. Analysis

In the following section, we follow the steps proposed before to compute transition probabilities CAN-SW. To compute $p_{i,j}$, first, we consider the neighbor, especially the finger neighbor (remote one). Although small-world

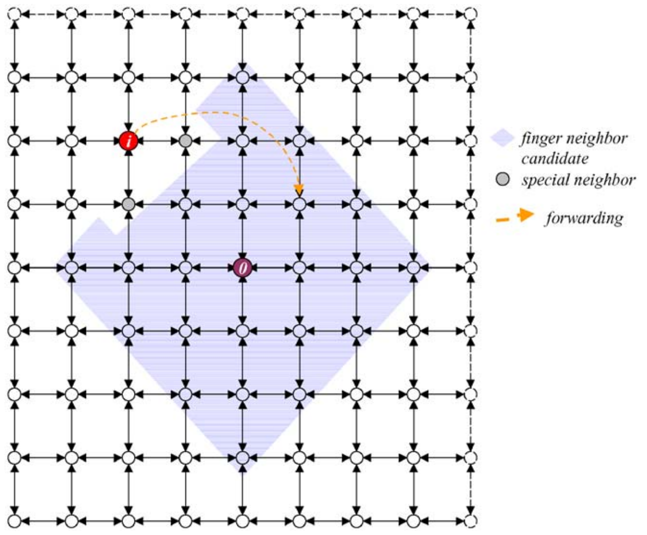

Fig. 6. An illustration of the message forwarding in CAN-SW.

model considers all remote neighbors as being generated initially, at random, we invoke the "Principle of Deferred Decisions"—a common mechanism for analyzing randomized algorithms [8]—and assume that the remote neighbors of a node i are generated only when the message first reaches j.

We know that the probability that i choose j as remote neighbor is

$$\alpha_{i,j} = l^{-d}(i,j)\frac{1}{\Delta}, \tag{14}$$

where $\Delta = \sum_{i' \neq i} l^{-d}(i,i')$. Define the set of nodes that can be next hop as a remote neighbor as $B(i) = \{i' : l(i') \leqslant l(i) - 1\} \wedge l(i,i') > 1\}$. $N(i) = \{i' : l(i') = l(i) - 1 \wedge l(i,i') = 1\}$ is defined as the set of local neighbors that can be the next hop.

As shown in Fig. 6, each node $j \in B(i)$ will be chosen as the next hop with probability $(1-\tau_2)\alpha_{i,j}$. If no remote neighbor is available, a local neighbor must be chosen. Therefore, each node $j \in N(i)$ will be chosen as the next hop with probability $(1-\tau_1^{|N(i)|})\frac{1}{|N(i)|}(1-(1-\tau_2)\sum_{i' \in B(i)} \alpha_{i,i'})$. As we know, if all neighbors are down, the message will be dropped at node i with probability $\tau_1^{|N(i)|}(1-(1-\tau_2)\sum_{i' \in B(i)} \alpha_{i,i'})$.

Therefore, the transition probability $p_{i,j}$ can be described as follows:

- $j \in B(i)$: j must be the remote neighbor, then

$$p_{i,j} = (1-\tau_2)\alpha_{i,j}. \tag{15}$$

- $j \in N(i)$: j must be one of i's local neighbors, then

$$p_{i,j} = (1-\tau_1^{|N(i)|})\frac{1}{|N(i)|}\left(1-(1-\tau_2)\sum_{i' \in B(i)} \alpha_{i,i'}\right). \tag{16}$$

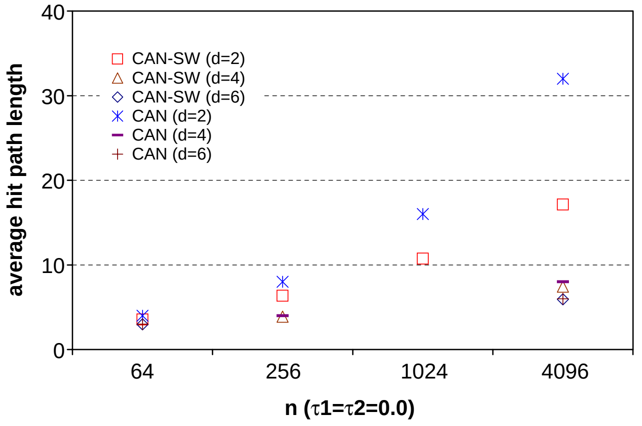

Fig. 7. Comparison of resilience to failure of finger and special neighbors for CAN and CAN-SW.

- $j = n$:

$$p_{i,j} = \tau_1^{|N(i)|}\left(1-(1-\tau_2)\sum_{i' \in B(i)} \alpha_{i,i'}\right). \tag{17}$$

- Otherwise

$$p_{i,j} = 0. \tag{18}$$

5.4. Analysis results and observations

We apply the above formulas to compute the average path length and the average hit ratio for CAN-SW and compare it with CAN. We also run the computation for systems with the number of nodes ranging from 64 to 4096. Here we report the data in the case of 4096.

Fig. 7 shows the comparison between CAN-SW and CAN in terms of the average path length when there is no failure in the system. We can clearly find that for all combinations of dimensions and number of nodes, CAN-SW outperforms CAN. Particularly, once the dimension is fixed, the improvement turns to be more significant as the number of nodes increases. However, when the number of nodes is fixed, the improvement turns to be less significant as the dimension increases. This observation can be explained by the fact that as the dimension increases, the number of local neighbors increases, accordingly, and the impact of the single remote neighbor will decrease. We expect that the impact will increase as the number of remotes neighbors increases, however it will cost more maintenance overhead.

Figs. 8 and 9 show the sensitivity of the average path length and the average hit ratio to the failure probabilities of the special neighbors (τ_1) and the finger neighbors (τ_2) in different systems. It confirms the observation in Section 4 that the average path length is sensitive to the finger neighbors, while the average hit ratio is sensitive to the special neighbors. Fig. 9 compares the performance of CAN-SW and CAN when τ_2 is fixed, and τ_1 is

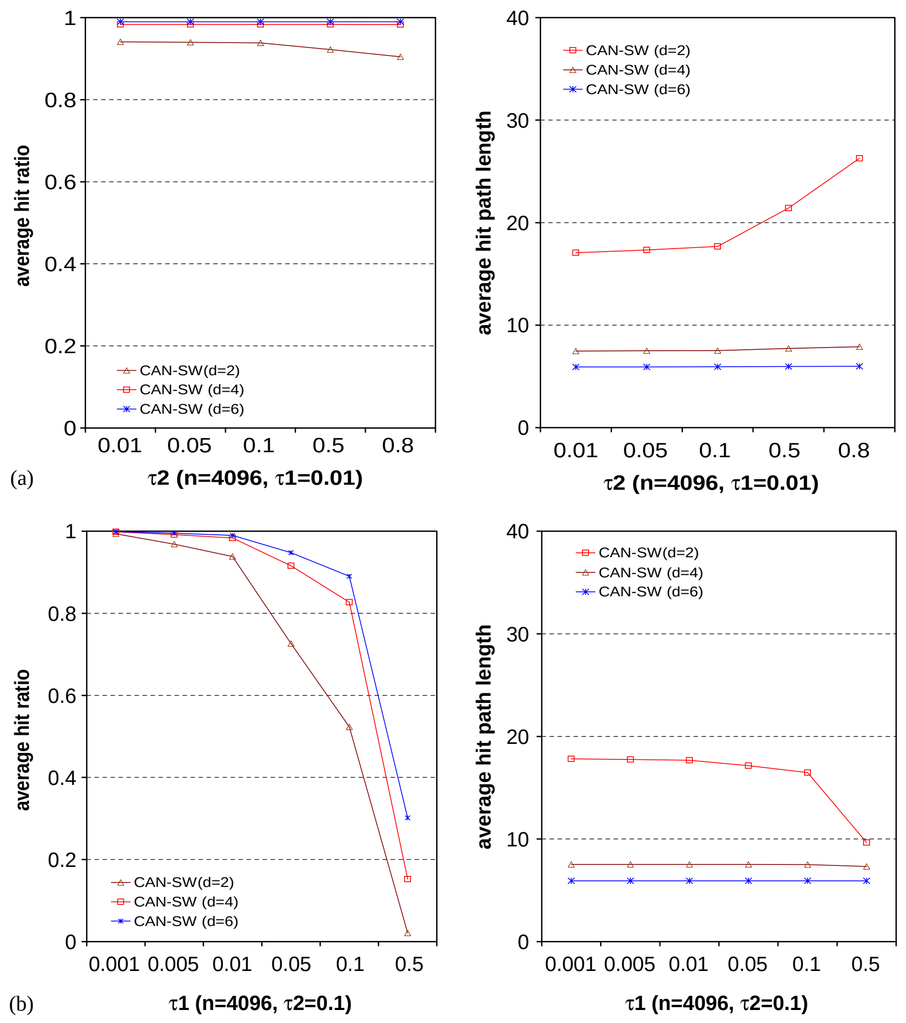

Fig. 8. Comparison of resilience to failure of finger and special neighbors for CAN and CAN-SW: (a) finger neighbor and (b) special neighbor.

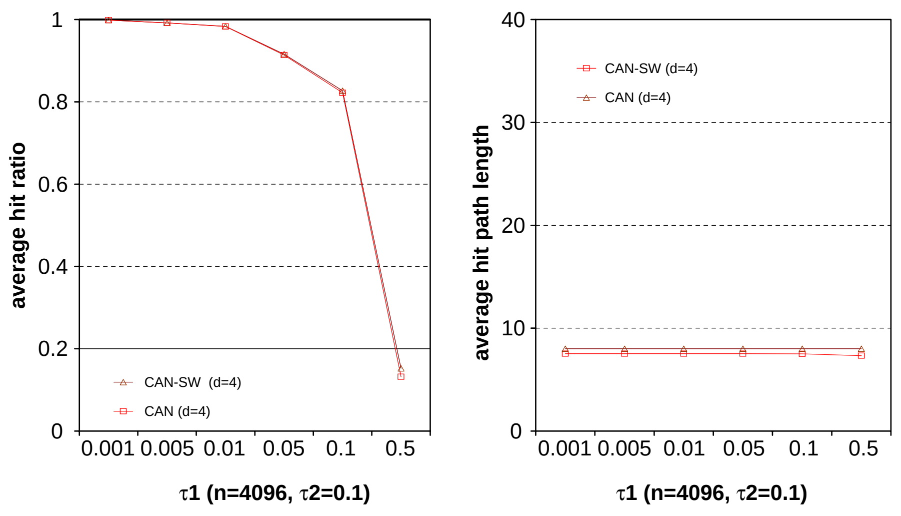

Fig. 9. Comparison of resilience to failure of finger and special neighbors for CAN and CAN-SW.

changed. The data shows that CAN-SW outperforms CAN both in terms of the average hit ratio and the average path length.

6. Final remarks

In this paper, we have systematically studied the resilience of structured P2P systems towards improving their performance in the presence of node failures. Particularly, we proposed an approach to analyze resilience of structured P2P systems under failures. The approach is Markov-chain-based, and can be applied to systems with relatively stable size and uniform distribution of nodes. To the best of our knowledge, this is the first general analytical approach to study the resilience of structured P2P systems. We apply our approach to several well-known structured P2P systems. We find that the finger neighbors and special neighbors have different impacts on the resilience of P2P systems. Particularly, the finger neighbors have a significant impact on the performance of the average path length while the special neighbors have an impact on the average hit ratio.

The research community of the DHT-based overlays usually uses two tools to analyze the resilience of a structured P2P system: simulation [2] and graph theory [5]. In [2], Gummadi et al. investigated how the underlying routing geometric approaches affect the resilience and proximity properties of DHTs. In [5], Loguinov et al. examined graph-theoretic properties of existing peer-to-peer architectures and proposed a new infrastructure based on optimal-diameter de Bruijn graphs. Our Markov chain-based approach basically is an analytical approach using probability and stochastic process theories which is different from the simulation and graph theory-based approaches. There are a rich set of probability and stochastic theories. We believe that our approach has a potential to incorporate with these theories to do more sophisticated analysis to P2P systems, which other approaches do not have. Also, the approach based on graph theory usually uses static metrics such as connectivity in [5] to measure the resilience, which is different from our resilience metrics, i.e., the average path length and the average hit ratio.

The analytical results obtained by our proposed Markov-chain-based approach give insight to the resilience of structured P2P systems. They can provide important guidelines in the design of a structured P2P system, especially on choosing neighbors and routing mechanisms so that the system can have a good resilience feature in terms of the average path length and the average hit ratio.

Guided by the obtained analytical insight to P2P resilience, we propose to add some finger neighbor(s) to CAN following the *small-world model* to build the CAN-SW system. We then apply the proposed approach to analyze its resilience and find that the performance is improved significantly, particularly, in terms of the average path length. Zhang et al. [18] apply the small world model to enhance the routing performance of Freenet, which is an unstructured P2P system. Our work was inspired by Zhang et al. [18] on Freenet. However, our work differs from theirs in two directions: (i) the systems we study are structured P2P systems where there are more constraints in adding neighbor nodes; (ii) our purpose is to enhance the resilience of the system under node failures which is not the case in [18]. The authors of [6] propose to use the small-world model in a ring topology to build P2P systems, which is similar to our work. The work in [6] can be regarded as a special case of our work since the torus topology CAN-SW is based on turns to be a ring when its dimension is set to 1 [6] does not address the issues of the resilience in terms of the average path length and the average hit ratio.

There are several directions to extend our study: (i) We can extend our model to be applicable to dynamic systems by removing the assumption that the system size is stable. We are planning to incorporate other stochastic theories into our Markov-chain-based analytical approach to analyze dynamic systems. (ii) It will be an interesting and a worthwhile effort to extend our approach to analyze unstructured P2P systems, such as Freenet, etc. (iii) Another kind of extension is to further improve CAN-SW. The remote neighbor(s) can be added considering several factors: (a) *load balancing*: In CAN-SW, if each node constructs multiple remote neighbors independently, it can be inferred that the performance would improve. However, this may increase the overhead in system maintenance. (b) Considering the issue of popularity and importance of a node, we can adjust the number of remote neighbors for each node by assigning relative weights to it. (c) Consider network conditions and application needs in determining the remote neighbors using several methods like the archival technique proposed by Xu and Zhang [17].

Acknowledgements

This work was supported in part by the National Science Foundation under Contracts 0329155, 0081761 and 0324988, by the Defense Advanced Research Projects Agency under Contract F30602-99-1-0531, and by Texas A & M University under its Telecommunication and Information Task Force Program. Any opinions, findings, and conclusions or recommendations in this material, either expressed or implied, are those of the authors and do not necessarily reflect the views of the sponsors listed above.

References

[1] O. Babaoğlu, H. Meling, A. Montresor, Towards adaptive, resilient and self-organizing peer-to-peer systems, in: Proceedings of the International Workshop on Peer-to-Peer Computing, Pisa, Italy, May 2002.

[2] K.P. Gummadi, R. Gummadi, S.D. Gribble, S. Ratnasamy, S. Shenker, I. Stoica, The impact of DHT routing geometry on resilience and proximit, in: Proceedings of the ACM SIGCOMM, Karlsruhe, Germany, August 2003.

[3] J. Kleinberg, The small-world phenomenon: an algorithmic perspective, in: Proceedings of the ACM Symposium on Theory of Computing, Portland, OR, May 2000.
[4] J. Li, J. Jannotti, D. De Couto, D. Karger, R. Morris, A scalable location service for geographic ad hoc routing, in: Proceedings of the ACM MobiCom, Boston, MA, August 2000.
[5] D. Loguinov, A. Kumar, V. Rai, S. Ganesh, Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilienc, in: Proceedings of the ACM SIGCOMM, Karlsruhe, Germany, August 2003.
[6] G.S. Manku, M. Bawa, P. Raghavan, Symphony: distributed hashing in a small world, in: Proceedings of the USITS, Seattle, WA, March 2003.
[7] S. Milgram, The small-world problem, Psychol. Today 61 (1967) 60–67.
[8] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, 1995.
[9] R. Nelson, Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modelling, Springer, New York, 1995.
[10] C.G. Plaxton, R. Rajaraman, A.W. Richa, Accessing nearby copies of replicated objects in a distributed environment, in: Proceedings of the ACM SPAA, Newport, RI, June 1997.
[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker, A scalable content addressable network, in: Proceedings of the ACM SIGCOMM, San Diego, CA, August 2001.
[12] S. Ratnasamy, S. Shenker, I. Stoica, Routing algorithms for DHTs: some open questions, in: Proceedings of IPTPS, Cambridge, MA, March 2002.
[13] A. Ravindran, D.T. Phillips, J.J. Solberg, Operations Research: Principles and Practice, second ed., Wiley, New York, 1987.
[14] A. Rowstron, P. Druschel, Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems, in: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, November 2001.
[15] 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 the ACM SIGCOMM, San Diego, CA, August 2001.
[16] D.J. Watts, S.H. Strogatz, Collective dynamics of smallworld networks, Nature 393 (1998) 440–442.
[17] Z. Xu, Z. Zhang, Building low-maintenance expressways for P2P systems, Technical Report HPL-2002-41, HP Laboratories Palo Alto, 2002.
[18] H. Zhang, A. Goel, R. Govindan, Using the small-world model to improve freenet performance, in: Proceedings of the IEEE INFOCOM, New York, NY, June 2002.
[19] Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, J.D. Kubiatowicz, A resilient global-scale overlay for service deployment, IEEE J. Selected Areas Comm. 22 (2004) 41–53.

Shengquan Wang received his B.S. degree in Mathematics from Anhui Normal University (AHNU), China, in 1995, and M.S. degree in Applied Mathematics from the Shanghai Jiao Tong University (SJTU), China. In 1998, he received M.S. degree in Mathematics from Texas A&M University. Currently, he is a graduate student in Computer Science at Texas A&M University. He was in a graduate assistant in Department of Applied Mathematics at SJTU. From 1998 to 2003, he was a graduate assistant and a research assistant in Department of Mathematics and Department of Computer Science at TAMU, respectively. His research interests include real-time computing and communication, real-time distributed component systems.

Dong Xuan received his B.S. and M.S. degrees in Electronic Engineering from the Shanghai Jiao Tong University (SJTU), China, in 1990 and 1993, and Ph.D degree in Computer Engineering from Texas A&M University in 2001. Currently, he is an Assistant Professor in the Department of Computer and Information Science, the Ohio State University. He was on the faculty of Electronic Engineering at SJTU from 1993 to 1997. In 1997, he worked as a research assistant in the Department of Computer Science, City University of Hong Kong. From 1998 to 2001, he was a research assistant and then associate in Real-Time Systems Group of the Department of Computer Science, Texas A&M University. His research interests include real-time computing and communication, network security and distributed systems.

Wei Zhao is currently an Associate Vice President for Research at Texas A&M University. He completed his undergraduate program in physics at Shaanxi Normal University, Xian, China, in 1977. He received his M.S. degree and Ph.D. in Computer and Information Science from the University of Massachusetts, Amherst, MA, in 1983 and 1986, respectively. In 1990, he joined Texas A&M University where he has been a full professor in the Department of Computer Science since 1996. Between 1997 and 2001, he served as a department head. His current research interests includes secured real-time computing and communication, distributed operating systems, databases, and fault tolerant systems. Dr. Zhao is an inventor for two US patents and has published over 180 papers in journals, conferences, and book chapters. Dr. Zhao is active in professional services. He was an editor of the IEEE Transactions on Computers between 1992 and 1996. He currently is on the editorial board of the IEEE Transactions on Parallel and Distributed Systems. He was program and general chairs of the IEEE Real-Time Technology and Applications Symposia in 1995 and 1996, respectively. He served as program and general chairs of the IEEE Real-Time Systems Symposia in 1999 and 2000, respectively. He was the co-program chair for the IEEE International Conference on Distributed Computing Systems in 2001 and the co-general chair in 2003. He will be the guest editor for a special issue on security in parallel and distributed computing systems for the IEEE Transactions on Parallel and Distributed Systems, to be published in 2003.