On Resilience of Structured Peer-to-Peer Systems

Shengquan Wang, Dong Xuan and Wei Zhao

Abstract—**In this paper, we propose an approach to analyze the resilience to failures of structured P2P systems. The approach is Markov-chain based, and can be applied to systems with relatively stable size and uniformly distributed nodes. We apply our method to several well-known structured P2P systems. We find that finger neighbors and special neighbors have different influences on the resilience features of P2P systems. More particularly, finger neighbors have significant influence on the average path length while special neighbors influence the hit ratio. Following the above observation, we propose to add some finger neighbor(s) to nodes of the CAN system which have no finger neighbor(s). We use the *small-world phenomenon* to form the CAN-Small-World (CAN-SW) system. We then apply the proposed approach to analyze its resilience. We find that the performance of the system under failures or not, has been improved significantly, particularly, in terms of the average path length.**

I. Introduction

A Peer-to-Peer (P2P) networked 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 systems 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 DHTs (Distributed Hash Tables) while unstructured P2P systems use mostly broadcast search. This paper focuses on structured P2P systems and investigate their behavior in the event of failures occurring in the network.

Consider a few scalable P2P systems that support distributed hash table (DHT) functionality (such as CAN [1], Chord [2], Tapestry [3] and Pastry [4]). Because of their rich structural arrangement they have efficient key lookups. and also somewhat resistant to failures of nodes. However, almost all the protocols are designed as an ideal overlay structure under which the key lookups are efficient. P2P nodes are notoriously transient and the resilience of routing to failures is a very important consideration. This is a serious concern since performance constraints in an ad hoc network is critical to the existence 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 resiliency of P2P networks is necessary for the continued development of such systems. Some papers address the issues concerning resilience to failures in distributed systems. For example, the Anthill project attempts to explore the possibilities of improving the efficiency of large scale distributed systems [5]. However, to the best of our knowledge, there is no approach proposed to analyze resilience to failures of structured P2P systems.

Our goal is to systematically analyze the features of resilience to failures of the current structured P2P systems in term of average hit ratio and path length, and understand the causes, which lead to better resilience features. In our attempt to analyze the resilient features of the P2P systems, we first propose a general model for describing the failures that possibly occur in a P2P system. Then we adopt a Markov-chain based approach to compute the hit ratio and average path length for various P2P systems under this failure model. After analyzing several P2P systems, we find that different types of neighbors in P2P systems have different impacts on the resilience of the P2P system to failures. Following this line of observation, we propose to add some extra neighbor(s) to CAN with small-world model to form the *CAN-SW system*. We then apply the proposed approach to analyze its resilience. We find that the performance improves significantly, particularly in terms of the average path length.

Shengquan Wang and Wei Zhao are with the Department of Computer Science, Texas A&M University, College Station, TX 77843, E-mail: {swang,zhao}@cs.tamu.edu. Dong Xuan is with the Department of Computer and Information Science, The Ohio State University Columbus, OH 43210. E-mail: xuan@cis.ohio-state.edu.

II. An Approach to Resilience to Failure of Structured P2P Systems

A. Model

A node say B is said to be the neighbor of A if it is sending and receiving requests directly to and from 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 crash of the neighbor or due to some error in the low-layer communication. We are interested in evaluating the resilience that routing can continue to function with some failed neighbors.

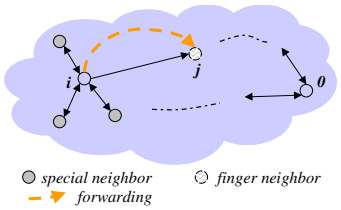


Fig. 1. Special and finger eighbors

For any node, there are two types of neighbors [6] 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 finger table of Chord and in the routing table of Pastry. Both types of neighbors are validated periodically for existence by polling messages sent from the node. However the intervals of polling might differ between specials 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 some point of time, respectively, to analyse the resilience to failure of neighbors. Basically, $\tau_1 < \tau_2$. [1]

In our model, we assume the system is stable. By "stable", we means 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 leaving neighbor will be replaced by a new 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 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 no matter whether the nodes keep leaving and joining. With these arguments, a P2P system's whole routing process can be modeled as a stochastic process even though the nodes in the system are dynamic.

B. A Markov-chain based Approach

In a stable P2P system, a node holding a query can be regarded as a state and routing behaviors can be regarded as transition among the different states. As we know in P2P systems, during the data look-up process, a node forwards a query to the next node based on its current routing table. The routing process can be modeled as a discrete absorbing *Markov chain*:

- Define a stochastic process $\{X_h : h = 0, 1, \cdots\}$, where random variable X_h is the state of a message forwarding during a routing process. Specifically, X_h can be a "failure" state or the ID of the node where the message is located after it is forwarded to the h-th hop [2].
- The message is forwarded to the destination or dropped finally ("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 from any source to one specific 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 a destination, and all other nodes $1, 2, \ldots, n-1$ as sources. We define the state i ($i = 0, 1, \ldots, n-1$) as the state when the message is at node i. We denote n as the "failure" state, i.e., if the message cannot be forwarded to any nodes in the system, it will be dropped and virtually put into n. Obviously, $1, 2, \ldots, n-1$ are transient states and $0, n$ are absorbing states.

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

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

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 and V are $(n-1) \times 1$ column matrices that delineate the transition rates between the transient states and the absorbing state 0 and n, respectively.

We define hit ratio $a_{i,j}$ to be the probability that a message is successfully forwarded from a transient state i ($i = 1, 2, \ldots, n-1$) to an absorbing state j ($j = 0, n$), and average path length $m_{i,j}$ to be the average number of steps for the corresponding successful forwarding. Computing these two values are nothing but the absorbing probability problem and the mean first passage time problem, respectively. Especially, for the state 0, if we define $A = (a_{1,0}, a_{2,0}, \ldots, a_{n-1,0})^{\perp}$ and $M = (m_{1,0}, m_{2,0}, \ldots, m_{n-1,0})^{\perp}$, by the analysis in [7], we have[4]

$$A = (I - Q)^{-1} U, \quad M = ((I - Q)^{-1} A) \div A. \tag{2}$$

By (2), we can obtain the hit ratio and the average path length from transient state i ($i = 1, 2, \ldots, n-1$) to destination state j ($j = 0, n$). Assume that a source is uniformly randomly chosen, then we have

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

$$\bar{a} = \frac{1}{n}(\pi^0 A + 1), \quad \bar{m} = \frac{1}{n}\pi^0 M, \tag{3}$$

where A, M are defined in (2), and $\pi^0 = (1, 1, \ldots, 1)$.

Based on (3), we can compute the hit ratio and the average path lengths to node 0 from any other nodes. Recall that we made no assumption on node 0, and the derivation of the above formulae does not use any particular feature of this node. Hence, the above formulae are applicable to any node i as the destination. If the system is symmetric such that all nodes are equivalent in terms of looking up, the results of formulae are automatically the hit ratio and the average path length of the system. If the system is not symmetric, the overall performance can be obtained based on the above formulae and the specific features of the system.

III. Analysis of structured P2P systems

A. Computation of Transition Probability

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

- The set of neighbors, including special neighbors and finger neighbors are determined first. Recall that these two types of neighbors have different failure probabilities.

[1] Generally, values of τ_1 and τ_2 depend on node ID. In our evaluation, for simplification, we choose unique τ_1 and τ_2 for all nodes.

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

[3] If message 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 you must remove state n and replace $p_{i,i}$ with $p_{i,n}$ on computing transition probabilities.

[4] Define $Z = X \div Y$ as $z_{i,j} = x_{i,j}/y_{i,j}$ for any i, j.

• Then, the routing semantics specific to the given P2P system are followed to determine the probability of forwarding to each neighbor. Different P2P systems have different routing policy. Care is taken to make sure that the routing policy is honored.

In the following, we use CAN as an example to describe how to compute the transition probabilities. Sylvia et.al [1] proposed 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 has only special neighbors, which are the nodes associated with the adjoining hybercubes. Routing consists of forwarding to a neighbor that is closer to the key(in the toroidal space). Let us compute the transition probability $p_{i,j}$ from node i to node j:

• 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. All nodes in $N(i)$ are *special neighbors*, and each is operational with probability τ_1. Their lattice distances from node i 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. 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 $(1-\tau_1^{|N(i)|})\frac{1}{|N(i)|}$. Now, if all the neighbors are down, the message will be dropped at node i, with probability $p_{i,n} = \tau_1^{N(i)}$. Therefore, for $i = 1, 2, \ldots, n-1$, we have

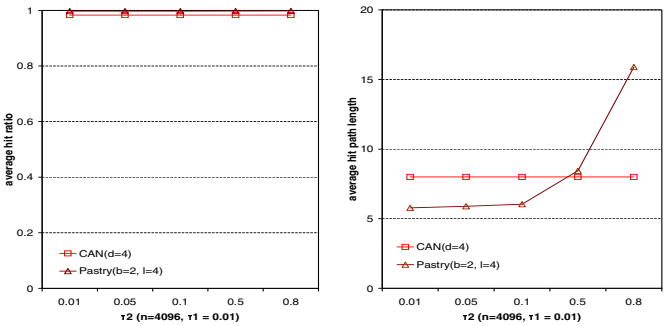
$$p_{i,j} = \begin{cases} (1-\tau_1^{|N(i)|})\frac{1}{|N(i)|}, & j \in N(i) \\ \tau_1^{N(i)}, & j = n \\ 0, & \text{otherwise} \end{cases} . \quad (4)$$

B. Analysis Results and Observations

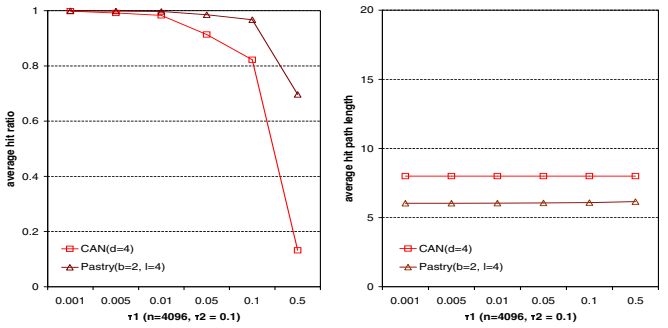
We apply the above formulae to compute the average path length and the hit ratio for CAN, Chord, Pastry and Tapestry systems, using MATLAB. We also ran simulation for the systems with the number of nodes ranging from 64 to 4096. Here we only report the data of CAN and Pastry.[5] Note that d is the dimension number for CAN and l is the leaf set size for Pastry. Note that there are no finger neighbors for CAN system but there exist $\mathcal{O}(\log n)$ finger neighbors for Pastry.

Fig. 2 illustrate the sensitivity of the average path length and the hit ratio to the failure of the special neighbors and the finger neighbors for CAN and Pastry, respectively.[6] We have the following observations:

• *Resilience to failures of finger neighbors:* The average path length is very sensitive to the failure of finger neighbors, but the average hit ratio is not, 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. E.g., in Fig. 2(a), for CAN, both the hit ratio and the average path length remain constant as τ_2 changes for different dimensions since there are no finger neighbors in CAN; for Pastry, given different sizes of the leaf sets, the average hit ratio is not sensitive to the change of τ_2, unlike the average path length which is very sensitive to such a change.



(a) finger neighbor



(b) special neighbor

Fig. 2. Performance evaluation of resilience to failure

• *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, 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 hit ratio. E.g., in Fig. 2(b), 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.

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

IV. Improving CAN with Small World

CAN uses its special neighbors (local neighbor) 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 with extra remote neighbor as finger neigbor to form shortcuts, intuitively the average number of hops would decrease considerably.

A. Small-World Model

The small world phenomenon, the principle that most of us are linked by short chains of acquaintances was first investigated

[5]The data of other systems reflect the same features. For further reading our on-line report [11] can be referred.

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

as a question in sociology by S. Milgram in the 1960s [8]. A network is said to exhibit the small-world phenomenon if any two nodes in the network are connected with a shorter hop distance. 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 randomly chosen remote nodes. Kleinberg

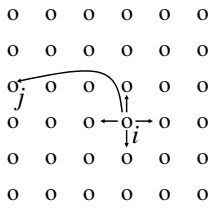


Fig. 3. An Illustration of Kleinberg's Small-World model

[9] 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 a mesh. In this paper, we extend the mesh to a d-torus. Assume that torus is of size $n = m^d$, where m is the width of each dimension of the torus. The lattice distance between two nodes i and j is defined to be the minimum number of "lattice hops" separating them: $l(i,j) = \sum_{k=1}^{d} \|i_k - j_k\|$ [7]. Each node i has a directed edge to every other node within lattice distance 1, which are its local neighbors (special 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 [9]: *As $r = d$, this mechanism can reduce the average path length to a polynomial in $\mathcal{O}(\log^2 n)$.*

B. CAN-Small-World (CAN-SW)

We apply the small-world model in CAN to form CAN-SW.[8] In the basic CAN-SW system, there are three key issues in the looking-up system: *CAN-SW construction*, *routing mechanism*, and *CAN-SW maintenance*. In CAN-SW, we still follow the greedy routing mechanism: In each step, the current message-holder chooses as the next hop a neighbor (either a local one or a remote one) that is as close to the destination as possible, in the sense of lattice distance. The maintenance mechanism are almost same as the ones in CAN system. The only difference is that remote neighbor nodes has to be taken into account in the same manner as local neighbors. In the following, we focus on CAN-SW construction.

As CAN, the process of CAN construction can be put forward in steps: (i) First, the new node must find a node already in the CAN. Then a point is randomly chosen from the key space (i.e. the d-dimensional torus). (ii) Next, using the routing mechanisms, the node which owns that point is found 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 system, besides local neighbors, each node will 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 base CAN. Due to the dynamics of the construction of CAN, the topology of the base CAN is not regular. For simplicity, we assume that the topology is a d-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 $m^d \leq \frac{1}{[Z_i]} < (m+1)^d$. Once n is known, based on node i, a remote point v can be generated with probability $\alpha_{i,j}$. Using the routing and lookup mechanism, the remote zone where j is located can be found.

Note that 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.

C. Analysis

In the following section, we follow the step proposed before to compute the transition probabilities of CAN-SW. To compute $p_{i,j}$, first, we consider the neighbor, especially the remote neighbor (remote one). Although small-world 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 [14] – 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 $\alpha_{i,j} = l^{-d}(i,j)\frac{1}{\Delta}$, where $\Delta = \sum_{i' \neq i} l^{-d}(i,i')$. Define the set of remote neighbors that can be the next hop as $B(i) = \{i' : l(i') \leq l(i) - 1\} \wedge l(i,i') > 1\}$. $N(i) = \{i' : l(i') = l(i) - 1 \wedge l(i,i') = 1\}$ is defined as the set of local neighbors that can be the next hop. 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, hence $\tau_1^{|N(i)|}(1-(1-\tau_2)\sum_{i' \in B(i)} \alpha_{i,i'})$. Therefore, for $i = 1, 2, \ldots, n-1$, we have

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

[8] We were not the first ones to apply the small-world model in P2P systems. [10] applied this model in an unstructured P2P system, freenet.

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

$$p_{i,j} = \begin{cases} (1-\tau_2)\alpha_{i,j}, & j \in B(i) \\ (1-\tau_1^{|N(i)|})\frac{1}{|N(i)|} & \\ \quad \cdot (1-(1-\tau_2)\sum_{i' \in B(i)} \alpha_{i,i'}), & j \in N(i) \\ \tau_1^{|N(i)|}(1-(1-\tau_2)\sum_{i' \in B(i)} \alpha_{i,i'}), & j = n \\ 0, & \text{otherwise} \end{cases} \quad . \quad (5)$$

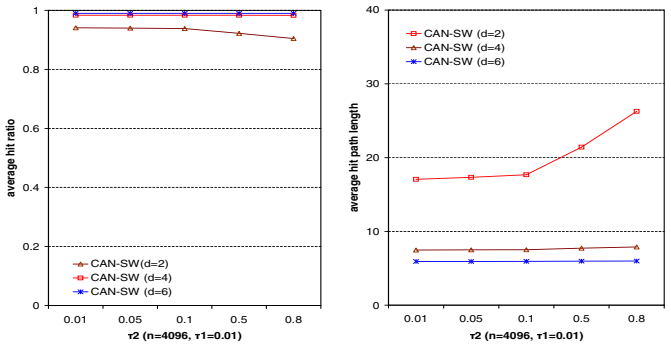
We apply the above formula to compute the average path length and the hit ratio for CAN-SW and compare it with CAN. Table I illustrates the comparison between CAN-SW and CAN

TABLE I

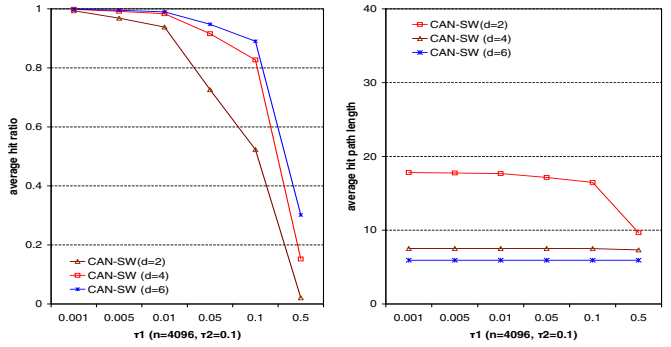
AVERAGE PATH LENGTH FOR CAN AND CAN-SW ($d = 2$)

	$n = 64$	$n = 256$	$n = 1024$	$n = 4096$
CAN	4	8	16	32
CAN-SW	3.65	6.36	10.74	17.12

in terms of the average path length when there is no failure in the system. We can clearly find that CAN-SW outperforms CAN. Particularly, the improvement turns to be more significant as the number of nodes increases. We expect that the impact will increase as the number of remote neighbors increase, however it will cost more maintenance overhead.



(a) finger (remote) neighbor



(b) special (local) neighbor

Fig. 4. Resilience to failure of remote and local neighbors for CAN-SW

Fig. 4 illustrates the sensitivity of the average path length and the hit ratio to the failure probabilities of the local neighbors (the special neighbors) (τ_1) and the remote neighbors (the finger neighbors) (τ_2) of different systems. It confirms the observation in Sec. IV that the average path length is sensitive to the remote neighbors, while the hit ratio is sensitive to the local neighbors. Once $\tau_2 = 1$, CAN-SW becomes CAN. Fig. 4(a) (where τ_1 is fixed, and $\tau_2 \to 1$) shows that CAN-SW outperforms CAN both in terms of hit ratio and the average path length.

V. FINAL REMARKS AND FUTURE WORK

In this paper, we propose an approach to analyze resilience to failures of structured P2P systems. The approach is Markov-chain based, and can be applied to systems with relatively stable size and uniformly distributed nodes. We apply our method to several well-known structured P2P systems. We find that the finger neighbors and special neighbors have different influences on the resilience features of P2P system. More particularly, the finger neighbors have significant impact on the performance of the average path length while the special neighbors influence the hit ratio. Following the above observation, we propose to add some remote (finger) neighbor(s) to CAN using the *small-world phenomenon* to form the 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.

The future work lies in two directions: i) Analysis: our approach can be applied to systems with relatively stable size and uniformly distributed nodes. It will be interesting to extend this approach to more complex and dynamic systems. ii) Enhancement: we can enhance CAN-SW by introduce multiple remote neighbors for each node. It can be inferred that the path length would be decreased and load balancing can be improved, however, with an increased overhead in system maintenance. The tradeoff needs to be investigated

REFERENCES

[1] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, *A scalable content addressable network.* In Proceedings of ACM SIGCOMM, August 2001.

[2] I. Stoica, R. Morris, D. Karger, M. F.Kaashoek, and H. Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications.* In Proceedings of ACM SIGCOMM 2001, August 2001.

[3] Y. Zhao, J. Kubiatowicz, and A. Joseph, *Tapestry: An infrastructure for fault-tolerant wide-area location and routing*, Technical Report UCB/CSD-01-1141, University of California, Berkeley, 2000.

[4] A. Rowstron and P. Druschel, *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems.* In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, Nov. 2001.

[5] O. Babaoğlu, H. Meling, and A. Montresor, *Towards Adaptive, Resilient and Self-Organizing Peer-to-Peer Systems.* In International Workshop on Peer-to-Peer Computing, 2002.

[6] S. Ratnasamy, S. Shenker and I. Stoica, *Routing Algorithms for DHTs: Some Open Questions.* In proc. of IPTPS, March 2002.

[7] A. Ravindran, Don T. Phillips, James J. Solberg, *Operations Research: Principles and Practice, 2nd Edition*, John Wiley & Sons.

[8] S. Milgram, *The small-world problem.* Psychology Today, 1967.

[9] J. Kleinberg, *The Small-World Phenomenon: An Algorithmic Perspective*, In Proceedings of the 32nd ACM Symposium on Theory of Computing, 2000.

[10] H. Zhang, A. Goel, *Using the Small-World Model to Improve Freenet Performance.* In Proceedings of IEEE INFOCOM, 2002.

[11] S. Wang, M. Krishnamoorthy and D. Xuan, *Analyzing Resilience of Structured Peer-to-Peer Systems*, Technical Report, Department of Computer Science, Texas A&M University, http://students.cs.tamu.edu/swang/papers/wang-resilence.pdf.