The Exclusion of Malicious Routing Peers in Structured P2P Systems

Bong-Soo Roh, O-Hoon Kwon, Sung Je Hong, and Jong Kim

Dept. of Computer Science and Engineering,
Pohang University of Science and Technology
{saintroh,dolphin,sjhong,jkim}@postech.ac.kr

Abstract. We propose a scheme which excludes malicious routing peers from the normal routing process of structured P2P systems such as Chord. This scheme prevents continuous routing overheads from malicious peers. Simulation results show that the proposed scheme reduces the average routing length compared with the routing algorithm only using the alternate lookup path.

1 Introduction

Structured peer-to-peer (P2P) systems such as Chord [1], CAN [2], and Pastry [3] provide good characteristics such as load balance, decentralization, scalability and availability when their algorithms are executed correctly. The routing algorithms are especially important because the peers place and lookup data deterministically using robust routing algorithms. Therefore, *incorrect lookup routing* is a serious problem in structured P2P systems. It means that malicious peers deliver query messages to incorrect or non-existing nodes. Even a small number of malicious peers can prevent correct message delivery and cause large overheads. Existing approaches for solving *incorrect lookup routing* are based on the concept of *secure message forwarding*. It is to deliver the message to good peers who are responsible with the query message in the presence of malicious peers. For example, there are techniques such as the iterative routing [4], the redundant routing [5] and the alternate lookup path [6]. However, even though these techniques securely delivere messages at once, malicious peers can participate again in the routing protocols. Therefore, lookups would continue to be routed to the malicious peers, which would increase the routing overheads.

In this paper, we propose a scheme which excludes the malicious routing peers from the normal routing process of structured P2P systems. This scheme prevents continuous routing overheads from existing malicious nodes. The proposed scheme has four characteristics. First, it is a fully distributed scheme to exclude malicious routing peers. Second, it makes the system work well even in the high ratio of malicious peers in the overlay. Third, it allows the arbitrary behavior of malicious routing peers. Fourth, it uses *the alternate lookup path* [6] and *the query observation* [4].

The rest of this paper is organized as follows. In Section 2, related works and their differences with our work are discussed. In Section 3 and 4, the adversary model and the characteristics of the proposed system are described. Section 5 describes the exclusion routing protocol. Section 6 shows the simulation results. Finally, we summarize this paper and discuss concluding remarks in Section 7.

2 Related Works

In this section, we briefly discuss the previous works which deal with *incorrect lookup routing* in structured P2P systems.

Sit and Morris [4] proposed the *iterative routing* using a query observation to defend against incorrect lookup routing. At each hop, the querier checks if the lookup gets closer to the key identifier. If an incorrect lookup is detected, the querier might recover by backtracking to the last good hop and asking for an alternative step that offers less progress. However, they had no experimental data to prove their claim. Castro et al. [5] proposed *redundant routing* using a routing failure test for secure routing in structured P2P Systems. The redundant routing technique is invoked when the failure test returns positive. The idea is simply to route copies of the message over multiple routes towards each of the destination key's replica roots. Their techniques allow P2P systems to tolerate up to 25% malicious nodes while providing good performance when the fraction of malicious nodes is small. Srivatsa and Liu [6] emphasized the importance of multiple *alternate lookup paths* for secure routing in structured P2P Systems. If the query originator detects an incorrect lookup using the query observation [4], it can choose an alternative (possibly sub-optimal) lookup path towards the destination identifier. Even though the above techniques can support secure message forwarding, malicious peers can continuously cause routing errors and routing overheads. Therefore, we will propose a scheme which excludes malicious routing peers from the normal routing process.

3 Adversary Model

In this paper, adversaries refer to those peers, which do not follow the routing protocol of the system and mislead good peers by providing them with incorrect routing information or no response.

We assume that most of peers can be malicious nodes in the overlay. In the worst case, almost all of the routing entries will be incorrect paths. As a result, the routing overheads such as the lookup failure, path re-computation and network bandwidth wastage can be increased, but the routing operation will work correctly. Generally, a malicious node behavior is assumed to be consistent. But, we assume that malicious nodes can perform arbitrary behaviors. We consider that malicious nodes may intentionally upgrade their trustworthiness by performing normal routing from time to time. However, if their behavior is accumulated, their trustworthiness will be evaluated accordingly.

We also assume that the malicious nodes cannot collude. Since general P2P architectures guarantee anonymity, the collusion attack by malicious nodes is a very complex problem. Although many research groups have worked on this problem, a complete solution has not been proposed. This problem is related to the authentication for P2P nodes. We assume that the underlying network layer is secure. That is, an adversary node can only access the packets that have been addressed to it. If the packet is not encrypted, the malicious node may modify it. Also, the domain name service, the network routers, and the related networking infrastructure are completely secure. Therefore, we assume that these infrastructures cannot be compromised by malicious nodes.

4 Characteristics of the System

In this section, we describe the characteristics of the system extended from general structure P2P systems in order to support the proposed exclusion scheme. The proposed scheme can be applied to all DHT-based P2P systems, but we will explain it for Chord [1].

4.1 Extended Routing Table

In structured P2P systems, each node uses the per-node routing tables for routing the query message. The routing table consists of references to other neighbors. In Chord [1], it is called the finger table. The k-th finger node is the first node that succeeds the current node by at least 2^{k-1}, where $1 \leq k \leq m$ and the identifier is a m-bit number. The finger table is used for efficient routing. In order to exclude malicious routing peers from the finger table, we add one column to each row. The column represents the ratio of how much incorrect routing the corresponding neighbor had done. The column is called the TCR (Total Claim Ratio). In our system, every peer forwards the query message to a peer with the minimum TCR value.

4.2 TCR(Total Claim Ratio)

The claim ratio (CR) is a ratio of the claim count (CC) to the forwarding count (FC). FC increases when the query is forwarded to each routing entry. CC increases when a peer receives the claim. Periodically, CR is reseted at every time interval of TCR. By doing so, the system is not influenced by unintentional routing failures such as malicious claims or short-term path errors.

The total claim ratio (TCR) reflects the node's historic behavior. The proposed scheme assumes that malicious peers can fabricate their CR. For example, if malicious nodes have normal behaviors in many routing steps initially, their FC increases enough for malicious behaviors to have no effect on their CR later on. Therefore, recent malicious behavior has a stronger influence on TCR that the old behaviors.

$$TCR = \sum_{k=1}^{n} \alpha_n * CR = \alpha_1 * CR_1 + \ldots + \alpha_n * CR_n \tag{1}$$

where $\alpha_1 + \alpha_2 + \ldots + \alpha_n = 1$ and $\alpha_n > \alpha_{n-1} > \cdots > \alpha_2 > \alpha_1$. CR_n is the most recent CR and CR_1 is the oldest CR, where n is the number of time interval.

4.3 Query Observation and Alternate Lookup Path

In the proposed scheme, the query originator checks if the lookup is correct by using the query observation [4] at each hop. Thus, each step of query process must be visible to the querier. In our system, the receivers of the query report *the identifier of the current node* and *the identifier of the next node* to the query originator at each hop. Using this information, the query originator can check for incorrect routing because the lookup is always supposed to get closer to the key identifier in the clockwise motion. Therefore, if an incorrect lookup is detected, the query originator can recover the lookup by backtracking to the last good hop for another path.

Upon detecting an incorrect routing by using the query observation, the query originator asks the previous good node along the lookup path for an alternate lookup path toward the destination identifier [6]. Due to the characteristics of the finger table in Chord [1], it is likely that the alternate lookup path proceeds only by half the distance along the identifier circle compared to the original path.

5 The Exclusion Routing Protocol

In this section, we describe the exclusion routing protocol against malicious routing peers. The exclusion routing protocol consists of the claim process and the verification process.

5.1 Claim Process

A misrouting node (MRN) is a node which misroutes a message intentionally or unintentionally. A previous good node (PGN) is a good node which delivers a message to a MRN. If the query originator detects an incorrect lookup routing, it gives a notice to MRNs and PGNs, which is called a *claim*. The purpose of this process is to leave the history table on the neighbors' routing trustworthiness as a column of the extended routing table. The *claim* includes the following information.

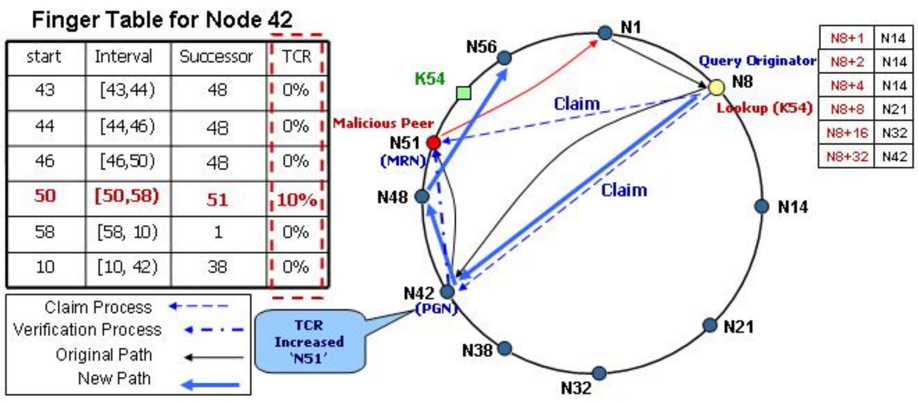
– the identifier and the IP address of the querier
– the identifier of a target node for the claim
– the destination key identifier

The claim process is as follows.

1. When the querier receives a wrong routing result, the querier gives a claim message to MRN and PGN, respectively.
2. PGN delivers the query through the minimal TCR path.
3. Using the verification process, PGN verifies if the claim is correct or not.
4. If the claim is correct, the receivers of the claim reflect the TCR value.
5. If MRN is a good node, MRN also does the steps 3) and 4).

As a result of the above claim process, nodes with the higher TCR value are excluded in each node's routing entries. Each node always forwards the query to the minimal TCR node among available paths. If more than two TCR values are the same, a PGN forwards the query to the neighbor which is closer to the key identifier. If a node with the minimal TCR value is malicious, a PGN forwards the query to the next minimal TCR node.

Fig. 1 is an example of the claim process. We assumed that N8 is the querier and N51 is the malicious node. N8 learns its query trace by using the query observation.


Finger Table for Node 42

start	Interval	Successor	TCR
43	[43,44)	48	0%
44	[44,46)	48	0%
46	[46,50)	48	0%
50	[50,58)	51	10%
58	[58, 10)	1	0%
10	[10, 42)	38	0%

N8+1	N14
N8+2	N14
N8+4	N14
N8+8	N21
N8+16	N32
N8+32	N42

Fig. 1. An Example for the Claim Process

Pseudo Code of ALP_{basic}**(query_result)**

```
if (query_result is false) {
 while !(query_result is correct) {
   current_node = PN;
   Decrease finger_table_index by 1;
   Send a query using current_node[finger_table_index];
 }
```

Fig. 2. The Algorithm of Alternate Lookup Path

If N51 incorrectly routes to N1, N8 detects the malicious node (N51) using the query observation. Then, N8 sends the claim to N42 (PGN) and N51 (MRN). N42 verifies a claim message using the verification process. If the claim message is verified to be correct, N42 reflects a TCR value for N51 (MRN).

This algorithm is based on the query observation. The observation of queries can inform the querier on where the message is currently arriving from. Using this mechanism, the querier can go back to the last good node. Then, the querier selects the sub-optimal path within the routing entries. However, this algorithm has a problem. It has not considered another querier who will pass by this location. Therefore, many of them go through the same situation.

Fig. 2 shows the pseudo code of the alternate lookup path (ALP). Fig. 3 shows the pseudo code of finding an alternate lookup path using TCR values. These algorithms are based on the query observation [6]. Unlike the alternate lookup path, the proposed algorithm checks a routing history column to determine the routing path. Using this accumulated data, every P2P node forwards the query message to a minimum TCR node. This procedure means that all of the P2P nodes exclude the higher TCR nodes from the normal routing process.

Pseudo Code of ALP_{TCR}(query_result)

```
if (query_result is false) {
 while !(query_result is correct) {
   MRN = current_node;
   current_node = PGN;
   Decrease finger_table_index
     by next_minimum_TCR_entry;
   if (next_minimum_TCR_entry does not exist)
     Increase finger_table_index;
   Send a query;
   if (predecessor of key is malicious)
     Send a query using replica;
   verification_process(faulty_node,key_value);
   if (verification_process is correct)
     Increase TCR_of_faulty_node;
 }
```

Fig. 3. Algorithm for finding an Alternate Lookup Path Using TCR values

5.2 Verification Process

Since the query originator can also be malicious, the receivers of the claim need to verify the claim before reflecting it to their TCR column. If PGN receives the claim, they send the same query to a MRN using the identifier of the target and the destination key identifier. Then, if PGN receives the same claim from the querier, it updates its TCR column. Since malicious nodes do not know whether the query is the PGN's message or not, it is impossible for MRN to selectively perform different routings.

5.3 Replication

In Chord [1], all lookup queries for a key pass only through the predecessor of the responsible node for the key. If the predecessor node is malicious, all lookups for the key will always fail. Therefore, all data should be replicated on several nodes. There are some research works on replication schemes [7,8]. They used neighbors of the responsible peer as the replication nodes. In our case, when there are 2^r replication nodes for a key k in m-bit identifier space, the data are replicated on the successors of the following keys: $\{(k+2^{m-r}), (k+2^{m-(r-1)}), \cdots, (k+2^{m-2}), (k+2^{m-1})\} \pmod{2^m}$.

If the query receiver finds the responsible node in the routing entry but the query cannot be forwarded to the node, the query originator forwards the query towards the next responsible node of the replication group.

6 Simulation Results

We have performed experiments to show that the proposed scheme reduce the average routing length even when the rate of malicious nodes is high. We simulated the alternate

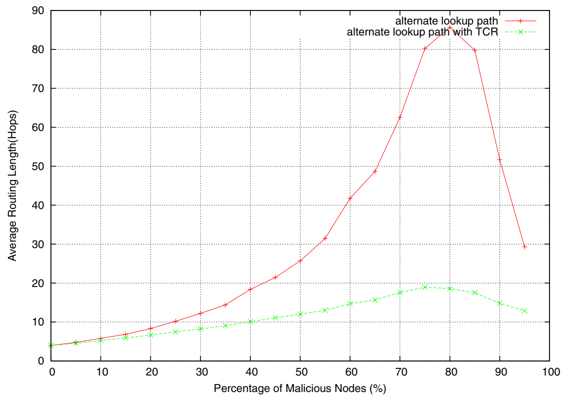


Fig. 4. Alternate Lookup Path Vs. TCR (B=5, Q=1000)

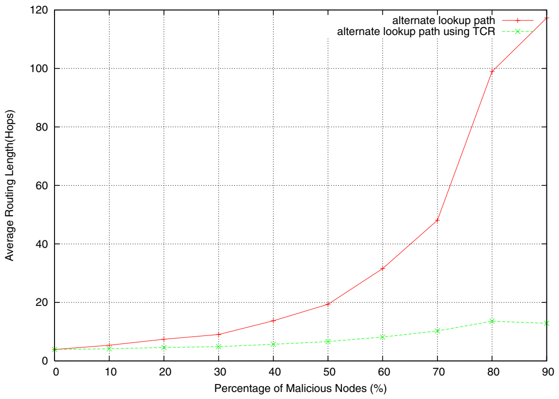


Fig. 5. Alternate Lookup Path Vs TCR (B=5, Q=1000000)

lookup path and the proposed scheme on a Chord system with 1024 nodes. The identifier of the nodes is 0 to 1023. The average routing length is 4 hop when there are no malicious nodes. The location of malicious nodes and the query originator is randomly selected.

Malicious nodes mislead good nodes to false successors. Therefore, if good nodes go through malicious nodes, the average routing length increases because they should find an alternate lookup path. When the querier can find the responsible node for a desired key in the routing entry, the routing successfully ends, while other cases are failures. Also, for a simple adversary model, we assumed that $\alpha = 1$ and $n = 1$.

If a ratio of malicious nodes increases, the number of nodes with false routing information in their routing table also increases. If all nodes in the routing entries are malicious, the querier cannot forward the query properly. To solve this problem, our system backtracks to a PGN and restarts the routing. Naturally, the average routing length increases, but the lookup failure rate decreases.

Fig. 4 shows the average routing length with backtracking, where the number of backtracking(B) is 5 and the number of queries(Q) is 1000. We can observe that the average routing length is doubled. The more the ratio of malicious nodes increases, the more the count of backtracking increases. Thus, the average routing length in the backtracking protocol is longer than that in the non-backtracking protocol. This result shows that our scheme reduces the average routing length even more by backtracking.

Fig. 5 shows the average routing length with backtracking, where the number of backtracking(B) is 5 and the number of queries(Q) is 1000000. This result shows that our scheme is more effective when the number of queries is large.

7 Conclusion

In this paper, we proposed a scheme which excludes malicious routing peers using TCR in a structured P2P system. Because our scheme excludes malicious nodes, the intermediate nodes along the routing path do not select a false routing path. The simulation results showed that the proposed scheme reduces the average routing length compared to the secure routing scheme which only uses the alternate lookup path.

Acknowledgments

This research was supported by the MIC(Ministry of Information and Communication), Korea, under the Chung-Ang University HNRC-ITRC(Home Network Research Center) support program supervised by the IITA(Institute of Information Technology Assessment).

References

1. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking (February 2003)
2. Ratnasamy, S., Francis, P., Handley, M., Karp, R.: A scalable content addressable network. In: Proceedings of ACM SIGCOMM 2001 Techinical Conference (August 2001)
3. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (November 2001)
4. Sit, E., Morris, R.: Security considerations for peer-to-peer distributed hash tables. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, Springer, Heidelberg (2002)
5. Castro, M., Druschel, P., Ganesh, A., Rowstron, A., Wallach, D.S.: Secure routing for structured peer-to-peer overlay networks. In: Proceedings of the 5th Usenix Symposium on Operating Systems Design and Implementation (OSDI) (December 2002)
6. Srivatsa, M., Liu, L.: Vulnerabilities and security threats in structured overlay networks: A quantitative analysis. In: Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC) (December 2004)
7. Maymounkov, Kademlia: A peer to peer inforamtion system based on the xor metric. In: Proceeding of 1st International Workshop on Peer-to-Peer Systems (March 2002)
8. Ratnasamy, Fancis: A scalable content addressable network. In: Proceeding of the ACM 2001 SIGCOMM Conference (August 2001)