Collaborative Anomaly Detection For Structured P2P Networks

Wei Wang, Hong Man, Fangming He
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ 07030, USA
e-mail: {wwang3, hman, fhe}@stevens.edu

***Abstract*—Anomaly detection in Peer-to-Peer (P2P) networks is generally difficult due to the large number of users in the network. Exhaustive probing on each user is extremely unrealistic. Besides, unlike hierarchical systems, the infrastructure of a P2P network is flat, which makes multi-casting based probing schemes impossible. Most P2P security research focus on proactive prevention schemes to secure the system. In this paper, we aim to apply passive anomaly detection to estimate the proportion of malicious nodes in the network, without any network parameter information. Two deployment schemes are proposed for different network attacks. We deploy monitoring nodes which maintain both in- and out-of-band P2P communications. Monitoring nodes collaboratively probe one another periodically, and observations at each monitoring node are aggregated by a token message. Simulation results show that after applying our anomaly detection system, we can estimate the status of malicious nodes in a P2P network with high accuracy, and the delivery rate of the network is noticeably increased after successfully blocking suspicious nodes.**

***Index Terms*—Anomaly Detection, Peer-to-Peer Networks, Network Security.**

I. Introduction and Related Work

Structured P2P protocols, such as CAN, Chord, Pastry and Tapestry, can ensure that peer nodes efficiently route messages to the desired destination over a large-scale overlay network. Peer nodes are much more powerful than conventional network nodes. There is no authorization or authentication entity in the network, and all live nodes act as routers, maintaining their own routing tables. In this fully decentralized, dynamic and self-organized network environment, no node can easily trust others. Thus, the assignment and use of identifiers (IDs) is essential to ensure operations in the network. This weakness in P2P networks has been exploited by malicious nodes to launch attacks, such as sybil attacks [1], in which a malicious node generates a large number of shadow identities and controls many network operations.

Many previous P2P security work focuses on proactive prevention to secure the P2P system. Miguel Castro and Peter Druschel [2] presented several techniques for secure node joining, routing table maintenance, and correct message forwarding in structured P2P networks. For secure ID assignment, they used a set of trust certification authorities (CAs) with well known public keys to assign IDs and sign certificates. For secure routing table maintenance, they imposed strong constraints on routing table entry and exploited network proximity information for efficient routing. They applied a routing failure test to determine whether the routing works, or a redundant route discovery scheme is needed for correct message delivery. In [3], a challenge-based scheme was presented to prevent malicious nodes from updating routing table entries. The authors proposed to use anonymous auditing to bound node degrees in the network. Those nodes with large in- or out-bound traffic degrees, would be excluded from updating legitimate nodes' routing entries.

Some other works study the prevention of one user from easily obtaining many identities. Hosam and William [4] proposed an admission control system. A node wishing to join the network would be challenged by nodes from many hierarchical cooperative peer nodes. A self-registration strategy was proposed in [5]. By modifying the join and stabilize phases of the P2P protocol, the joining node had to register at r specified registration nodes using its IP and port information. Only when over a half of the registration nodes validated its identity, could the new node be accepted into the network.

In this paper, we present two collaborative anomaly detection methods to passively detect anomalies. We attempt to estimate the proportion of malicious nodes in the network, without knowing any network information, such as the number of live nodes or the packet delivery ratio. The deployment schemes are designed to have low operational cost. Finally, we may block suspicious nodes, by aggregating information gathered at different monitoring nodes, to improve the network performance.

II. Proposed Node Deployment Schemes

In this section, we study how to deploy a small number of monitoring nodes which only consume a small amount of network resources, and generate an accurate estimation on the proportion of malicious nodes in the structured P2P network. Without losing generality, we choose Chord [6] as our P2P routing protocol. Chord has a simple and efficient routing scheme. In an m-bit ID space with a maximum number of 2^m nodes, each node maintains a routing table ("finger table") consisting of m entries ("fingers"). In the finger table of node n, the i^{th} entry refers to the live node with the smallest nodeId clockwise from $(2^i + n) mod(2^m)$ where $i \in [0, m-1]$. Nodes forward messages only along the clockwise direction in a circular ID space. Each node in Chord also maintains pointers to its predecessor and k successors (the neighbor set). When a file is published at one node, nodes in its neighbor set will store file replicas for fault tolerance and load balance.

In our work, every monitoring node firstly publishes a file with the fileID equals to its nodeID. Consequently, the k nodes whose IDs are numerically close to the fileID will replicate the file onto them. Monitoring nodes then periodically probe other monitoring nodes by sending in-band P2P query messages, asking for the previously published files. Under the malicious-free condition, the monitoring node that has the file will successfully receive the query message, and return a source list containing itself and k neighbor nodes that have the file replicas to the query monitoring node. But if there are malicious nodes in-the-middle, they will either drop the query message or return a false source list, pretending they have the file replicas. Thus in our work, every query monitoring node will verify the returned source list and check the consistency of the received file by comparing it with an out-of-band message. Failing to receive the requested file after a timer expires or finding any alterations on the requested file will be treated as a positive test which need further investigation; otherwise we define it as a negative test. We will propose two estimation schemes for different malicious behaviors. Scheme I is suitable for message dropping misbehavior, and scheme II is for coalitions among faulty nodes that return false source lists.

A. SCHEME I

We consider a random node launching a random query. If we represent the distance from the source node to the destination as d, and express it as $d = \lambda_1 2^{m-1} + \ldots + \lambda_{m-1} 2^1 + \lambda_m 2^0$, where λ_i =0 or 1, the number of relay nodes will be $\sum_{i=1}^{m} \lambda_i$. It has been proved in [6] that with high probability, the maximum number of nodes that are contacted to find another node in an N-node network is $\lceil log_2 N \rceil$. Since the distance is random, we expect $\lceil 0.5 log_2 N \rceil$ to be the average number of relay nodes for a random query. Assume the proportion of malicious nodes that will launch message dropping attacks is f, and h is the number of relay nodes between the source and destination nodes. The average successful delivery probability from one random source to one random destination monitoring node is

$$P = (1-f)^h = (1-f)^{\lceil 0.5 log_2 N \rceil}. \qquad (1)$$

We will utilize equation (1) to estimate f. Then we want to explore a reasonable deployment method which requires only a few monitoring nodes but can produce accurate estimation.

1) Method I: Liang and Sencun [7] proposed a random-sampling scheme to detect sybil attacks, and a path-resolving scheme to identify attacker nodes in a multi-cast network. Their schemes take advantage of the hierarchical tree topology in video streaming applications, which is not the case for a P2P network. However, all structured P2P routing protocols will ensure messages being forwarded to the node closest to the destination after each hop. In Chord, for example, each node has finger entries at the power of two intervals around the ID space and can forward a query at least halfway along the remaining distance between the node and the target node [6]. We utilize this property to construct a comparative virtual tree topology as the base for our method I.

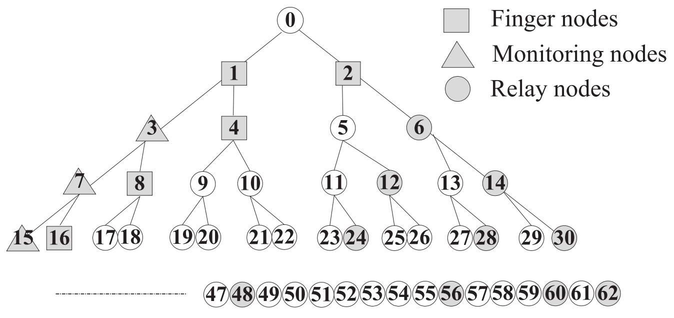

Fig. 1. Scheme I method I monitoring nodes deployment

We assume node 0 as the node which will send query messages to other monitoring nodes for certain files. We deploy the rest of the monitoring nodes as those gray triangle nodes in Figure 1. The gray square nodes are the finger nodes in node 0's routing table. Based on the Chord routing protocol, the gray circle nodes and finger nodes are the middle relays.

All the relay nodes are our candidate malicious nodes. Assume we deploy M monitoring nodes, and the probing will start from node 0 to other monitoring nodes with the IDs $M(i) = 2^{i+1} - 1$ where $i \in [1, m-2]$. Thus the distance between consecutive nodes is increased exponentially. Figure 1 shows that there are at most i intermediate steps to reach the $M(i)$ monitoring node. Because the P2P network is always large with typically thousands of live nodes, this is an approach using only a few monitoring nodes to traverse most nodes in the network. For the best case, when the ID space is full of live nodes, the maximum total number of relay nodes in forwarding the messages equals $1 + 2 + 3 + \ldots + (m-1) = m(m-1)/2$.

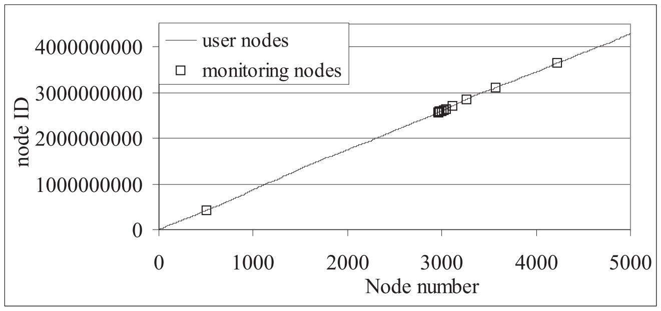

Fig. 2. Scheme I method I nodeID distribution in a sparse ID space

However, method I is ideal for closely packed P2P networks. In practice, a P2P network assigns each node and file with an m-bit ID using a hash based function such as SHA-1. The huge m-bit ID space will assure distinct ID assignment and map the expected inputs as evenly as possible over its output range. From Figure 2, we can see that if $N = 5000$ nodes are assigned IDs in a 32-bit ID space using a hash function, when using method I with $M = 30$ monitoring nodes (as the square points), some monitoring nodes will cluster together since there are no live nodes with assigned IDs in between. In this way, some monitoring nodes will be the immediate successors which will sharply reduce the number of intermediary relay nodes. Thus, in such a sparse network ($2^m >> N$), method I is inefficient in detecting abnormal nodes.

2) Method II: As mentioned in method I, the distinct ID assignment algorithms will generate IDs for users as evenly as possible for load balance. If N node IDs are uniformly distributed random variables over the m-bit ID space, the distances of consecutive node-IDs are approximately exponential random variables with the mean of $2^m/N$ when N is large [2]. Figure 3 is the ID distance histogram of consecutive N (5000/10000) nodes in an m(32)-bit ID space. If we set $F(x_0) = 1 - e^{-\lambda x_0} \geq \tau$ with $\lambda = N/2^m$, then $x_0 \geq \frac{-\ln(1-\tau)2^m}{N}$ which means τ percent of the consecutive node distance is less than x_0. We can treat $-\ln(1-\tau)$ as a small constant compared to 2^m, assuming $2^m >> N$, thus most nodes will have consecutive ID distances less than or equal to $2^m/N$, while a few nodes have longer distances.

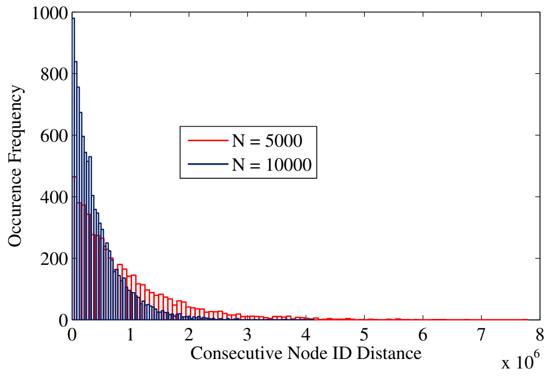

Fig. 3. The ID distance histogram of consecutive nodes with $m = 32$

Because of this distance pattern, we propose our method II, with all monitoring nodes being placed uniformly in the ID space. We deploy a predetermined number of monitoring nodes in the network to keep deployment expense to a minimum. Also we can see in section IV that there is no need to deploy a large number of monitoring nodes for good estimation on the malicious proportion. Assume we deploy M monitoring nodes, we want to assure that the distance d between any two nodes k_1, k_2 is not equal to the power of two. This is to avoid the situation that k_1 will forward messages directly to k_2 without any relay nodes. Thus we calculate d as

$$d = 2^m/M * k_1 - 2^m/M * k_2 \neq 2^l$$

which gives

$$2^{m-l} \neq M/(k_1 - k_2).$$

Under this constraint, we choose M to be an odd number, which decreases the possibility that the monitoring nodes are in each other's finger table.

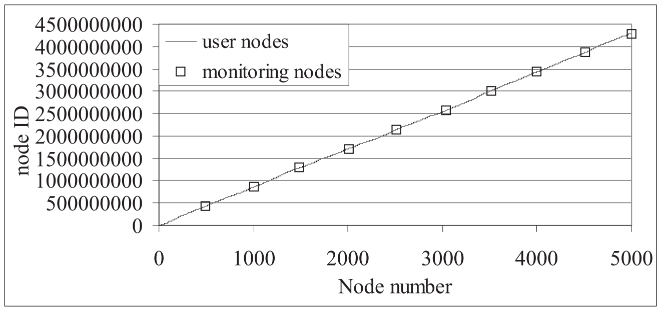

Fig. 4. Scheme I method II node ID distribution in a sparse ID space

Figure 4 represents the node ID distribution for method II. We can see that when the network is sparse with a low ID density per unit in the ID space, unlike method I, uniformly deployed monitoring nodes can always cover a certain amount of intermediary relay nodes.

In method II, we have each monitoring node recursively probe other monitoring nodes in several rounds. In round i, the s^{th} node will send a probing message to the $((s + i)mod(M))^{th}$ node ($s \in [1, M], i \in [1, M-1]$). Since the distances between monitoring nodes are uniform for every round, the number of relay nodes for round i at the s^{th} monitoring node will be $h_i = \lceil log_2 \frac{iN}{M} \rceil$. Thus the average hops at each monitoring node for all rounds will be $h = \frac{1}{M-1} \sum_{i=1}^{M-1} \lceil log_2 \frac{iN}{M} \rceil$. With this probing method, equation (1) changes to

$$P = (1 - f)^{\frac{1}{M-1} \sum_{i=1}^{M-1} \lceil log_2 \frac{iN}{M} \rceil}. \quad (2)$$

B. SCHEME II

Many works studied the case of coalition attacks which involves an attacker controlling many peer nodes [3], [4], [8]. The Byzantine failure model is used in [2] which also assumed that all faulty nodes were grouped into multiple coalitions to prevent correct message delivery. In scheme II, we aim at estimating the proportion of coalition malicious nodes. Scheme I is not capable of dealing with this misbehavior because the coalitions will pretend to be the legitimate nodes and perform "normal" message forwarding.

A successful query result will be a source list of legitimate replica nodes (negative test). But if a query message is forwarded by a malicious node, it can easily return a coalition of malicious nodes it controls (positive test). As in Figure 5, we can see that since the nodeID assignment is random, malicious nodes can not choose their desired IDs; therefore the average distance of the sender's neighbor set is smaller than that of the coalition neighbor set's, while approximately equals to that of the prospective legitimate node neighbor set's. Based on this fact, we propose our scheme II by comparing the density of nodeIDs in the neighbor set of the sender with the density of nodeIDs in the returned list of replica nodes for the requested files.

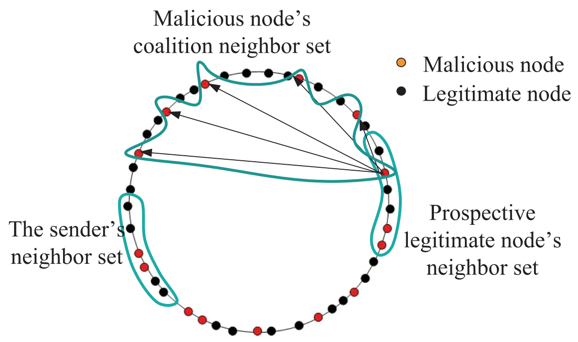
Malicious node's coalition neighbor set
● Malicious node
● Legitimate node
The sender's neighbor set
Prospective legitimate node's neighbor set

Fig. 5. Replica nodeID desity

First consider a simple case where all malicious nodes form a single conspiracy. Assume F_1 is the distribution for consecutive distances of the legitimate node's neighbor set, and F_2 is that for the malicious coalition neighbor set. Therefore we have two exponential distributions F_1 and F_2 with different means $\mu_1 = D/N$ and $\mu_2 = D/(Nf)$. f is the fraction of malicious nodes, and D is the ID space length 2^m. Our objective is to estimate f, in a situation where the total number of live nodes in the network N is unknown. Scheme II will apply the same recursive probing method as that in scheme I method II. Moreover, by using an out-of-band P2P message, the query monitoring node will obtain the correct neighbor set list directly from the prospective destination monitoring node. Thus the query monitoring node can identify the coalition node's neighbor set by comparing the neighbor sets obtained by the in and out-of-band P2P messages. We compute the ID distances of consecutive nodes in each monitoring node's neighbor set. For the monitoring node i ($i \in [1, M]$), we represent consecutive ID distances in its neighbor set as $\mathbf{X} = (x_{i1}, \ldots, x_{ik})$, where k is the size of the neighbor set. Denote $\mathbf{Y} = (y_{i1}, \ldots, y_{ik})$ as the consecutive ID distances of replica nodes from the returned source list. All probing trials are independent, and for each node i, y_{ij} are k independent identically distributed (i.i.d) R.V.s ($j = 1, \ldots, k$), drawn either from the F_1 or F_2 distribution. sDefine $\lambda = 1/\mu_2$, then we have the joint log-likelihood function of λ with respect to the consecutive ID distances from the returned list at the monitoring node i

$$\begin{aligned} L(y_{i1}, y_{i2}, \ldots, y_{ik}|\lambda) &= \ln(\Pi_{j=1}^{k} f(y_{ij})) \\ &= \ln(\Pi_{j=1}^{k} \lambda e^{-\lambda y_{ij}}) \\ &= \ln(\lambda^k e^{-\lambda \Sigma_{j=1}^{k} y_{ij}}). \end{aligned} \quad (3)$$

The maximum likelihood estimator (MLE) of the unknown λ is

$$\hat{\lambda} = \arg\max_{\lambda} L(\lambda). \quad (4)$$

Take the gradient of the log-likelihood function with respect to λ in equation (3) and we have $\frac{\partial L(\lambda|y_{i1}, y_{i2}, \ldots, y_{ik})}{\partial \lambda} = 0$ which gives

$$\hat{\lambda} = \frac{k}{\Sigma_{j=1}^{k} y_{ij}}. \quad (5)$$

Thus, our estimation on f is

$$f_e = \frac{N}{Dk} \Sigma_{j=1}^{k} y_{ij}. \quad (6)$$

With the unknown parameter N, we estimate N using the observations from $\mathbf{X}$ as

$$\hat{N} = \frac{Dk}{\Sigma_{j=1}^{k} x_{ij}}. \quad (7)$$

We will only calculate the data from the q positive tests in which y_i are from the F_2 distribution. For convenience, we define a δ function which can represent positive and negative tests, as

$$\delta(y_i) = \begin{cases} 1, & \text{positive test;} \\ 0, & \text{negative test.} \end{cases}$$

Substituting equation (7) into (6) and averaging over the $q = \Sigma_i \delta(y_i)$ positive tests, we have our final estimation on f as

$$f_e = \frac{1}{M} \sum_{i=1}^{M} [\delta(y_i) \frac{\sum_{j=1}^{k} y_{ij}}{\sum_{j=1}^{k} x_{ij}}]. \quad (8)$$

As a more complicated attack, G independent coalition groups operate as multiple conspiracies. Similar to the single conspiracy case, we assume the consecutive distances of the i^{th} group ($i \in [1, G]$) coalition have an exponential distribution with different means $\mu_i = D/(Nf_i)$. Then the real proportion of malicious nodes in the network should be $f = \Sigma_{i=1}^{G} f_i$. The estimation on f will take the same form as equation (8). Assume R_i is the number of times the i^{th} coalition group with f_i being hit by the test, then our final estimation will be $f_e = R_i f_{ei}$, where f_{ei} is the estimation value for the i^{th} coalition group by equation (8). Because it is easier to hit the malicious coalition with the high proportion f_i, the final estimation f_e will be greater than the real f. But the estimation f_e is still reasonable because the malicious coalitions with larger f_i represent the majority of all the coalitions. Therefore, our scheme II can estimate the total f with multiple coalition groups in the network, though with a higher number.

III. Information Aggregation

This aggregation process aims at estimating the proportion of malicious nodes and identifying the suspicious ones. For both schemes, a prior-defined "token" message is used for aggregating the information among the deployed monitoring nodes, via the out-of-band P2P communication.

For scheme I, to estimate the proportion of malicious nodes f_e, every monitoring node keeps the statistics of the number of probing messages it has sent s, the number of successful feedbacks from the destination monitoring node r, and the number of relay nodes for each successful feedback h. Each monitoring node will add its s, r, h in the payload of the token message. After a fixed sampling period T_s, the token message will have a summation of $S = \Sigma_{T_s}(s)$, $R = \Sigma_{T_s}(r)$, $H = \Sigma_{T_s}(h)$ in its payload. We use equation (2) to estimate f_e as

$$(1 - f_e)^{H/R} = R/S,$$

where R/S is the average successful delivery ratio and H/R is the average relay node number. f_e is a snap shot estimation on the current malicious node proportion in the network. s, r, h in the token message will be reset to zero for the next snap shot after each f_e estimation. In our experiment, we use three minutes as our sampling period T_s. Short sampling period will not give an accurate estimation, and long period can not respond to fast events.

Each monitoring node will send probing messages with a five seconds interval in our test. During the sampling period T_s, a node will be marked "suspicious" as long as one of the following two conditions is satisfied.

1) At the same monitoring node (time domain) or
2) different monitoring nodes (space domain),

if over γ times, a non-monitoring node w is reported as the last hop on the query or alters the requested file to the source node, then w will be marked as malicious and excluded from the network.

We set the threshold γ equal thirty, $\gamma = 30$. Because we have a five-second sample interval, the value of the threshold means either the node is detected as an anomaly for about two minutes at a single monitoring node, or multiple monitoring nodes report it as suspicious.

For scheme II, the integration is simpler since the estimation proportion is calculated at each monitoring node locally. The token message is used to pass the value to calculate the average by equation (8). Those nodes that initiatively return a faulty source list to the source monitoring node can be marked as malicious directly.

IV. PERFORMANCE EVALUATION

A. OVERSIM FOR P2P SIMULATION

In this paper, we choose OverSim as our simulation tool. OverSim is an open-source overlay network simulation framework for the OMNeT++ simulation environment [9]. It contains several models for structured and unstructured P2P protocols, and has a fully configurable network topology with realistic bandwidths, packet delays and packet losses. We modified the code of the Chord key based routing (KBR) application to verify our schemes.

Since we aim to estimate the malicious nodes proportion in the network without knowing any details about the network status, we make two assumptions without losing generality to verify our schemes. (1) Nodes can not join the P2P network without a valid credential, we only study malicious behaviors of authorized nodes. (2) The malicious nodes only passively drop messages or return false source list to the query nodes. They will not actively launch attacks to other nodes and they will maintain "normal" P2P overlay level communications, disguised as legitimate nodes.

B. Simulation Results And Analysis

We evaluate the performance for scheme I method II and scheme II in this section.

Figure 6 represents the simulated results for the malicious nodes proportion estimation for scheme I method II. We can

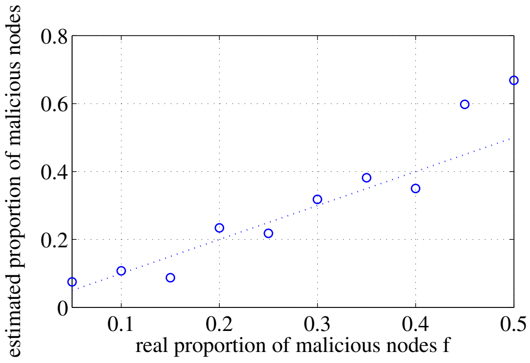

Fig. 6. Estimation under conditions with $M = 11, N = 5000, m = 32$ in a sparse ID space for different f

see that with f changing from 0.05 to 0.4, the estimated f_e is very close to the real f; but the estimation deviation is very large for $f > 0.4$ because the chance of successful message delivery is very slim under a large f. Method II will determine that the network is filled with malicious nodes under the large f and project a high f_e.

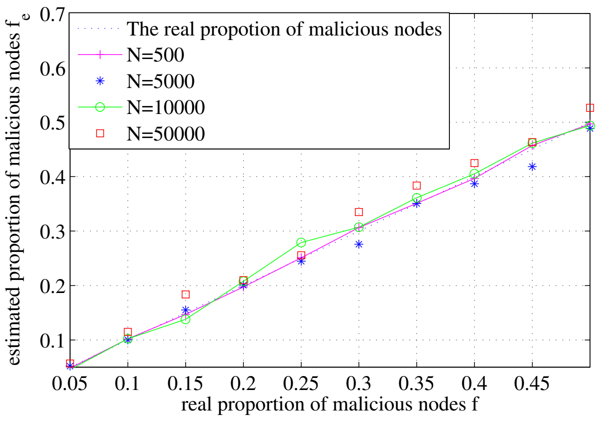

Fig. 7. Estimation under the condition of $m = 32, k = M = 20$

Figure 7 shows good estimation performance for scheme II with a fixed number of monitoring node $M = 20$. We have $k = 20$ since it is a default value for most P2P systems. For different values of live nodes N, f_e fluctuates slightly around the true f. From equation (8), we can see that the estimation accuracy deals nothing with N. But with a fixed number of M and k for different N, relatively the sample rate is bigger when N is a small number. Thus the accuracy for smaller N is higher as shown in the figure.

We also try to let M be proportional to N by applying equation (7) to estimate N. Figure 8 shows that with more samples for large N, the f_e is very close to f, and conversely for small N.

We further evaluate how much the network message delivery ratio will be increased by excluding the detected malicious nodes from the network. Figure 9 (a) (c) (e) (g) show the message delivery ratio with 10%, 20%, 30% and 40% malicious nodes in the network; (b) (d) (f) (h) show the ratios after taking

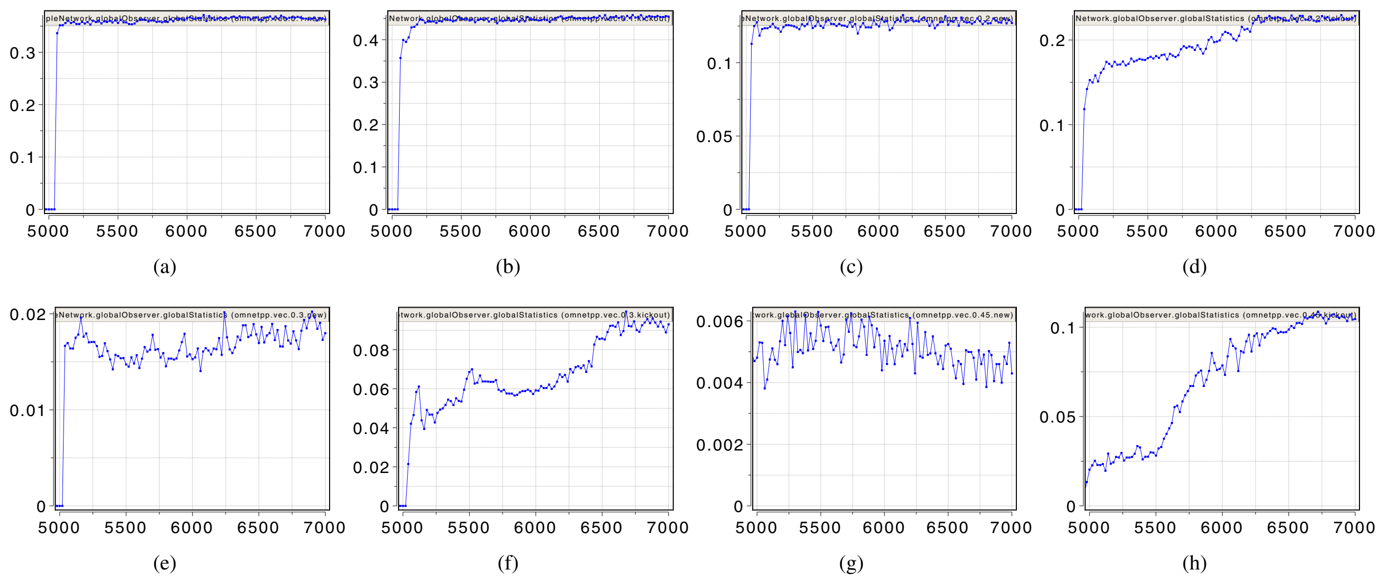
Fig. 9. Network message delivery ratio over time t (5000s - 7000s), before and after detection. $N = 5000, M = 11, 32bitsID$ with the real proportion of malicious nodes $f =$ (a) 10% before, (b) 10% after, (c) 20% before, (d) 20% after, (e) 30% before, (f) 30% after, (g) 40% before, (h) 40% after.

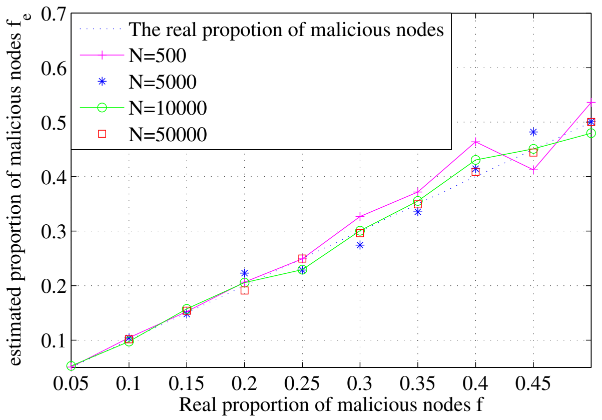

Fig. 8. Estimation under the condition of $m = 32, k = 20$

the detected suspicious nodes out of the network. With detecting and blocking malicious nodes gradually over time, the overall network performance is obviously enhanced. For $f = 0.1, 0.2, 0.3, 0.4$, the network message delivery ratio changed by $0.35 \rightarrow 0.45, 0.15 \rightarrow 0.23, 0.017 \rightarrow 0.09, 0.005 \rightarrow 0.12$ on average. For further study, the IP trace back technique can be applied afterwards if it turns out that many malicious nodes are from the same IP address, which means potential sybil attacks.

V. CONCLUSION

In this paper, we introduce two node deployment schemes to estimate the proportion of malicious nodes in the network, without knowing the network status. Scheme I places an odd number of exponentially or uniformly distributed monitoring nodes in an m-bit ID space and is targeted at message dropping misbehavior. Scheme II compares the faulty coalition nodeID density with that for the normal nodes. Both schemes can render accurate estimation of the malicious node ratio f with only a few monitoring nodes. Simulation shows that the network delivery ratio can be increased noticeably after blocking the misbehaved nodes.

REFERENCES

[1] J. Douceur, “The sybil attack,” in *First International workshop on peer-to-peer systems*, 2002, pp. 251–260.
[2] M. Castro, P. Druschel, A. J. Ganesh, A. I. T. Rowstron, and D. S. Wallach, “Secure routing for structured peer-to-peer overlay networks.” in *OSDI*, 2002.
[3] A. Singh, M. Castro, P. Druschel, and A. I. T. Rowstron, “Defending against eclipse attacks on overlay networks.” in *ACM SIGOPS European Workshop*, Y. Berbers and M. Castro, Eds. ACM, 2004, p. 21.
[4] H. Rowaihy, W. Enck, P. McDaniel, and T. L. Porta, “Limiting sybil attacks in structured p2p networks.” in *INFOCOM*. IEEE, 2007, pp. 2596–2600.
[5] J. Dinger and H. Hartenstein, “Defending the sybil attack in p2p networks: Taxonomy, challenges, and a proposal for self-registration.” in *ARES*. IEEE Computer Society, 2006, pp. 756–763.
[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” in *SIGCOMM ’01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2001, pp. 149–160.
[7] L. Xie and S. Zhu, “Message dropping attacks in overlay networks: Attack detection and attacker identification.” *ACM Trans. Inf. Syst. Secur.*, vol. 11, no. 3, 2008.
[8] S. Shitrit, E. Felstaine, N. Gilboa, and O. Hermoni, “Anonymity scheme for interactive p2p services.” in *Eighth IEEE International Symposium on Cluster Computing and the Grid*, 2008, pp. 33–40.
[9] I. Baumgart, B. Heep, and S. Krause, “OverSim: A Flexible Overlay Network Simulation Framework,” in *Proceedings of 10th IEEE Global Internet Symposium (GI ’07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, May 2007, pp. 79–84.