Security through Collaboration in MANETs

Wenjia Li, James Parker and Anupam Joshi

Department of Computer Science and Electrical Engineering,
University of Maryland, Baltimore County (UMBC),
1000 Hilltop Circle, Baltimore MD 21250
{wenjia1, jparke2,joshi}@cs.umbc.edu

Abstract. It is well understood that Mobile Ad Hoc Networks (MANETs) are extremely susceptible to a variety of attacks, and traditional security mechanisms do not work well. Many security schemes have been proposed that depend on cooperation amongst the nodes in a MANET for identifying nodes that are exhibiting malicious behavior such as packet dropping, packet modification, and packet misrouting. We argue that in general, this problem can be viewed as an instance of detecting nodes whose behavior is an outlier when compared to others. In this paper, we propose a collaborative outlier detection algorithm for MANETs that factors in a nodes reputation. The algorithm leads to a common outlier view amongst distributed nodes with a limited communication overhead. Simulation results demonstrate that the proposed algorithm is efficient and accurate.

Keywords: outlier detection, mobile ad hoc network, misbehavior, gossiping

1 Introduction

Outliers are generally defined as data points that are very different from the rest of the data with respect to some measure [1, 2]. Outlier detection can be used for two purposes. The first is to eliminate outliers to potentially reduce the noise in the data. The other usage of outlier detection is to expose the outliers for further analysis, for example in intrusion detection [3, 4], fraud analysis [5] and habitat monitoring for endangered species [6].

Several factors make Mobile Ad Hoc Networks (MANETs) extremely susceptible to various attacks such as intrusions [7], greyholes [8], and blackholes [9]. First of all, data in MANETs are transmitted via RF broadcasts, which can be easily eavesdropped on or even modified. Second, nodes in MANETs have limited power supply, and consequently their performance is severely degraded when power is exhausted. Third, when they are used for security and military purposes, nodes in MANETs are vulnerable to compromise and manipulation by adversaries. Hence, it is obvious that misbehavior detection should be an indispensable component of any security solution that aims to safeguard the mobile ad hoc networks. The misbehavior typically observed includes dropping of packets, misroutes, false Requests/Clears in the MAC layer etc. However, many of these events can also happen due to environmental and mobility related reasons, not just malicious intent. Most

of the current misbehavior detection mechanisms rely on a predefined threshold to decide if a node's behavior is malicious or not. However, it is rather difficult to set an appropriate threshold because the network is quite dynamic and unpredictable, and environmental conditions such as ambient RF noise can vary. In contrast, we do not need to rely on any previous knowledge to find anode that is an outlier with respect to a given observable. Given the fact that a malicious node will behave differently when compared to other nodes, we can detect the node misbehaviors by means of outlier detection.

In this paper, we propose and evaluate a collaborative, gossip-based outlier detection algorithm for mobile ad hoc networks. In our approach, as in many others [22,24,28,33], all the nodes in MANETs observe the behavior of their neighbors. Unlike most existing approaches however, each node calculates its local views of outliers amongst the neighboring nodes. In the next step, the nodes exchange their local views with their immediate neighbors. Then they will update their local views if they find that outlier lists from other nodes are more accurate than theirs. This process continues, with each node updating its neighbors when its current view of the outliers changes, and halts when there are no more changes. Some important features of our algorithm are: (1) it is compatible with different outlier detection heuristics; (2) it is resilient to attempts by misbehaving nodes to defeat it; (3) it is resilient to the motion and failure of nodes in MANETs; (4) it is efficient in terms of communication overhead; and (4) all the nodes will ultimately have a coincident view of outliers unless the nodes change their behaviors very fast.

2 Related Work

2.1. Outlier Detection

Outlier detection is a hot topic in the data mining research, and various definitions of outliers have been proposed in the literature. Our proposed algorithm takes two popular distance-based definitions into account: (1) distance to the nearest neighbor (*NN*) [10], and (2) average distance to *k* nearest neighbors (*k-NN*) [11].

One major motivation of outlier detection research is to efficiently identify outliers in a large-scale database [10, 12, 13, 14]. Nevertheless, the situation in mobile ad hoc networks is significantly different from that in large-scale central databases: in mobile ad hoc networks, data are generated and stored in scattered nodes and transmitted via wireless channels, which are unreliable and bandwidth and power-constrained. Outlier detection methods for the large-scale databases cannot be seamlessly used in mobile ad hoc networks because they will cause a large communication overhead.

Several outlier detection algorithms have been recently proposed for wireless sensor networks (WSNs) [6, 19, 20, 21]. Palpanas et al. propose a model-based outlier detection algorithm in sensor networks [20]. In their algorithm, normal behaviors are first characterized by predictive models, and then outliers can be detected as the deviations. Subramaniam et al. [21] propose an online outlier detection mechanism for sensor networks. In this mechanism, every sensor node will keep a sliding window of the historic data and approximate the data distribution to detect the outliers. In a recent paper by Sheng et al. [6], a histogram-based outlier detection algorithm is

studied, and sensor data distribution is estimated by the histogram-based method. This method can reduce communication cost under two different detection schemes. Moreover, a histogram refinement technique for some crucial portion of data distribution has been applied to obtain more information about outliers. Branch et al. [19] propose an in-network outlier detection scheme to detect the outliers based on data exchange among neighbors. In this scheme, all the sensor nodes will first calculate the local outlier(s). Then some messages, which contain the local outlier(s) as well as some other supportive information, will be exchanged among all the nodes. The message exchanging process will not halt until all the nodes have the same global view of outlier(s). Our proposed outlier detection algorithm is somewhat similar to the method proposed by Branch et al. However, there are two significant differences between the two methods. First, the method by Branch et al. does not consider the mobility of the nodes, whereas our proposed method takes the mobility issue in consideration. Second, there is no malicious behaviors in the discussion of the method by Branch et al., i.e., the nodes will not deliberately fabricate fake local views or alter incoming local views in their method. On the contrary, we have considered the malicious behaviors of the nodes, and applied the knowledge of trust and reputation as the countermeasure to the malicious behaviors.

2.2. Misbehavior Detection in Mobile Ad Hoc Networks

In mobile ad hoc networks, all network operations such as routing and forwarding rely on cooperation of the nodes because there is no centralized infrastructure. Hence, if some nodes choose not to participate in the network operations, then these network services may be incomplete or even unavailable. These non-cooperative nodes are generally called *selfish nodes* [22]. Besides selfishness, ad hoc network misbehavior may also be conducted by malicious nodes, which aim to harm the whole ad hoc networks. A malicious node can perform different attacks to either compromise individual node(s) or degrade the performance of the overall network [23]. The existence of selfishness and malicious behaviors has motivated research in the area of misbehavior detection for mobile ad hoc networks.

Intrusion Detection Systems (IDS) are an important means to detect node misbehavior. Several mechanisms have been proposed to build IDS on individual nodes due to the lack of a centralized infrastructure [24, 25, 26, 27]. In these mechanisms, every node is equipped with an IDS, and each IDS is assumed to be always on, which is not energy-efficient given the limited battery power of nodes in ad hoc networks. On the other hand, Huang et al. [28] propose a cooperative intrusion detection framework in which clusters are formed in ad hoc networks and all the nodes in one single cluster will cooperate in intrusion detection operation.

Routing misbehavior is another kind of malicious activity that is common in ad hoc networks. When an adversary aims to degrade the network service of ad hoc network, he can try to compromise some nodes in the ad hoc network, and use them to disturb the routing service so as to make part of or the entire network unreachable. Marti et al. [22] introduce two related techniques to detect and isolate misbehaving nodes, which are nodes that do not forward packets. In the “watchdog" approach, a node forwarding a packet verifies whether the node in the next hop also forwards it or not.

If not, a failure tally is incremented and misbehavior will be recognized if the tally exceeds a certain threshold. The “pathrater" module then utilizes this knowledge of misbehaving nodes to avoid them in path selection. There are also some other proposed solutions that aim to handle the routing misbehavior [29, 30, 31].

3 Gossip-based Outlier Detection Algorithm

In this section, we describe our gossip-based distributed outlier detection algorithm. The goal of the algorithm is to find the top k outliers in terms of some observed behaviors (such as packet drops or misroutes) from all the nodes in mobile ad hoc networks (Here k is a user-defined parameter). The algorithm leads to a coincident global view of the top k outliers in all the nodes as long as these nodes do not change their behavior significantly during the convergence time of the algorithm. By using constrained gossiping, the algorithm avoids flooding the network.

3.1. Algorithm Description

The proposed outlier detection algorithm contains the following four steps, namely local view formation, local view exchange, local view update, and global view formation. We have adopted two local view update methods in our algorithm: one is the simple averaging method, in which all the local views are merged by simply averaging them; the other method is the trust-based weighted method, in which the local views are merged incorporating the trust in other nodes.

The first step of our algorithm is the formation of local views. The nodes monitor and record the possible malicious behaviors of other nodes within their radio range. Each node generates its local view of outliers based on their own observations.

Once all the nodes form their local views, they will broadcast the local views to all of their immediate neighbors, i.e., all the nodes that are one hop away from them. Upon reception of a local view from another node, the recipient will update its local view based on the received view. The first local view update method we employ is the simple averaging method, which is shown in the Subroutine 1 below. Here n_i denotes the i-th node in the mobile ad hoc networks. V_i denotes the initial view of n_i. V_i’ denotes the updated view of n_i.

Subroutine 1 Update of Local View for node *i* Using the *Simple Averaging* Method

Input of n_i: V_i
Output of n_i: V_i’
Upon reception of V_j from node n_j:
if $V_j \neq V_i$
—merge the V_i and V_j according to the following rules:
——if node m is in BOTH V_i AND V_j, then calculate the average of the corresponding columns for node m in BOTH V_i and V_j, and store the average of node m to an intermediate list $TEMP_i$ as an entry.

——if node m is in EITHER V_i OR V_j, but NOT BOTH, then add a virtual entry of node m to the view that previously does not contain m, and set all the columns of this virtual entry as 0. Then calculate the average between the true entry of m and virtual entry of m for each column, and then store the average values of node m to an intermediate list $TEMP_i$ as an entry.

—calculate the top k outliers from $TEMP_i$, and assign these k top outliers to V_i'.

—broadcast V_i' to all of its immediate neighbors (number of hop = 1).

else keep V_i unchanged, and not send any message out

The averaging is necessary due to the existence of malicious nodes that may produce false views to mislead other nodes. Suppose a malicious node randomly generates some entries reporting large misbehaviors for a good node, and sends this false view to others. If the recipients simply take the false view it will miss the true outliers. Averaging the information from all neighbors helps avoid this. Another heuristic is that if a recipient receives information about any node that has never been seen before, it will use only half of the reported value in computing the average. In other words, it will treat this new information conservatively. On the other hand, the true outliers will not be influenced by either of the heuristics because several nodes will report their observed outlier values. Of course, this scheme will be vulnerable in a locality where most of the nodes are malicious, but in such circumstances most misbehavior detection algorithms fail anyway.

Another possibility is to use the trust-based weighted method during the local view update process. Unlike the simple averaging method, the trust-based weighted method relies on the reputation of a node to determine how to merge the view it sends out with the local view of the receiver. The trust-based weighted method is listed in the Subroutine 2 below. Again, n_i denotes the i-th node in the mobile ad hoc networks. V_i denotes the initial view of n_i. V_i' denotes the updated view of n_i. w_{ik} denotes the weight of local view sent from node k to node i.

Subroutine 2 Update of Local View for node *i* using the *Trust-based Weighted* Method

Input of n_i: V_i

Output of n_i: V_i'

Upon reception of V_k from node n_k:

if $V_j \neq V_k$

—merge the V_i and V_k according to the following rule:

——if node m is in BOTH V_i AND V_k, then calculate the weighted average WA_i of the corresponding columns for node m in BOTH V_i and V_k according to the following formula:

$$WA_i = \left(w_{ii} * m_i + w_{ik} * m_k\right) / \left(w_{ii} + w_{ik}\right)$$

and then store the weighted average WA_i of node m to an intermediate list $TEMP_i$ as an entry.

——if node m is in EITHER V_i OR V_k, but NOT BOTH, then we simply set m_i or m_k to be zero, and the calculation of WA_i follows the formulae below:

$$WA_i = \begin{cases} w_{ii} * m_i, \text{ when } m_k = 0 \\ w_{ik} * m_k, \text{ when } m_i = 0 \end{cases}$$

and then store the weighted average WA_i of node m to an intermediate list $TEMP_i$ as an entry.

—calculate the top k outliers from $TEMP_i$, and assign these k top outliers to V_i'.

—broadcast V_i' to all of its immediate neighbors (number of hop = 1).

else keep V_i unchanged, and not send any message out

Note that unlike traditional gossiping, the more the nodes that accept the same view of outliers, the less the number of new messages that are sent out. Ultimately, when all the nodes hold the same view of outliers, the algorithm will halt, and the view that all the nodes hold is regarded as the global view of outliers.

The pseudo-code of the algorithm is given in Table1 and uses the same notation as described earlier. In addition, *GV* denotes the ultimate global view.

Algorithm 1 Gossip-based Outlier Detection

Input of n_i: V_i
Output of n_i: GV
For each node n_i:
broadcast V_i to all of its immediate neighbors
Upon reception of V_j from node n_j:
invoke **Subroutine1** ***OR*** **Subroutine 2**
When no more message exchange occurs:
$\forall k$, GV = V_k

3.2. Trust Establishment and Management

Several trust and reputation management schemes have been proposed in the past decades [34, 35, 36].Any of them can be used in our system. For our experiments, we chose a simple approach that starts with a default trust for unknown nodes, and modifies it upon each encounter with that node.

Initially, we define all the trust value to be 1. Whenever the node observes any misbehavior of its neighbors, the node reduces the trust value of the entry for the misbehaving neighbor according to the punishment factors. We set different reduction factors for different misbehaviors when we adjust the trust value. For example, packet dropping and packet modification are both misbehaviors. However, packet dropping may be caused either by intentional malicious behavior or by power failure. On the other hand, when we find that a node is modifying the incoming packets, we can safely conclude that it is malicious. Hence, we set a higher reduction factor for packet modification than packet dropping.

During the local view update process, when a node i gets local view from its neighbor k, the node uses the trust value of its neighbor as the weight w_{ik}, and its own weight w_{ii} will always be 1. In this way, we apply the knowledge of trust and reputation to the local view update process, and we can ensure that the fake local views spread by the malicious nodes will not influence the formation of the global view.

3.3. An Example Scenario

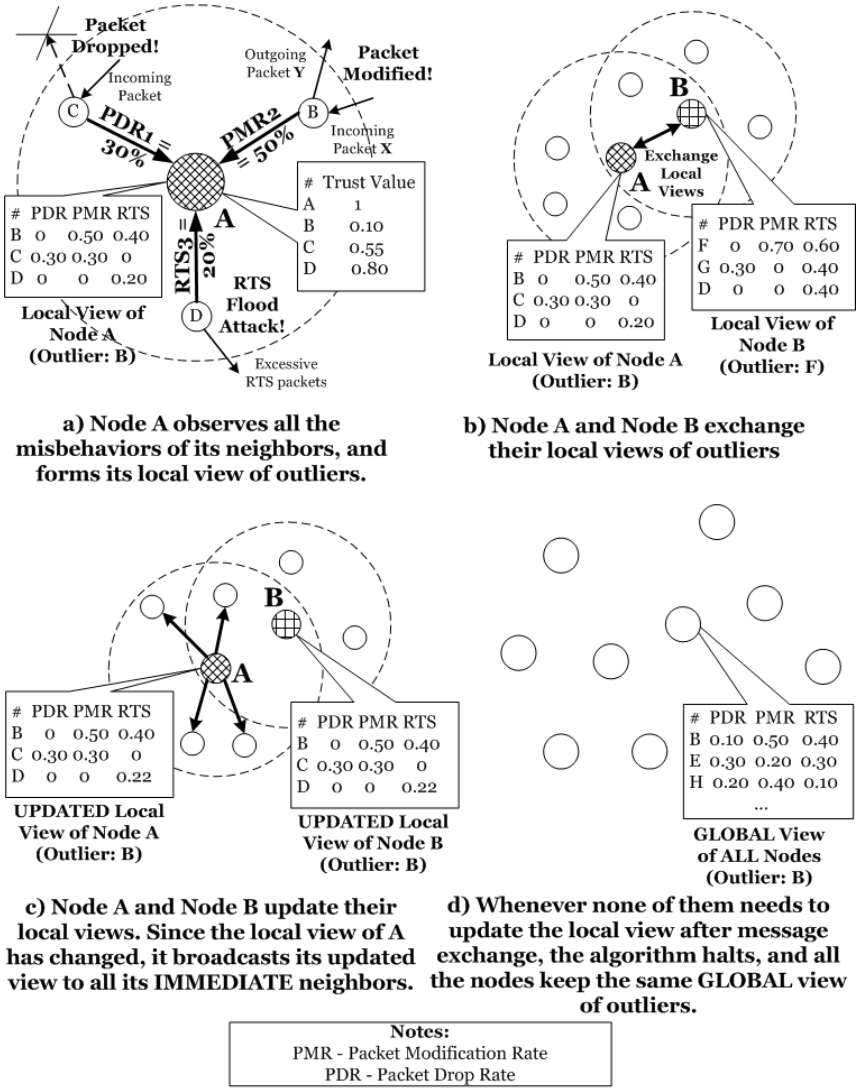

Fig. 1. Gossip-based Outlier Detection Algorithm

To help better understand the proposed algorithm, an example is presented in Fig. 1. In Fig. 1a, node A observes all the misbehaviors of it neighbors, and then forms its local view based on its own observation. Node A will also construct its initial trust table based on its observation to its neighbors. All other nodes are simultaneously collecting their neighbors' misbehavior information, and generate their local views as well as trust tables. The outlier candidates in the local views are sorted according to the distances between their nearest neighbors and themselves, and the top three outliers are picked in this example. We note that as long as all the nodes are observing

the same set of behaviors, our approach can handle anything defined as a misbehavior.

The next step is the initial exchange of the local views, which is demonstrated in Fig. 1b. In this step, all the nodes send their local views to all of their *immediate* neighbors, which are defined as the nodes that are located one hop away from them. From Fig. 1b we find that the local views of node A and node B are not consistent.

Fig. 1c exhibits the view update and optional rebroadcast step. Both node A and node B update their local views according to the view they have received. We note that node A applies the knowledge of trust to the local view update process. In this way, node A ensures that its updated local view contains the least fake information from node B, who is likely to be a malicious node since its trust value is quite low. Then, they rebroadcast their updated views to all the immediate neighbors. We should also be aware that node B may send out any arbitrary view to its immediate neighbors regardless of the true updated view it gets, because node B seems to be malicious.

The view update and optional rebroadcast process will continue until all the nodes hold the same view of the top three outliers, and this final state is shown in Fig. 1d. We find from Fig. 1d that the composition of the outliers has been significantly altered for both node A and B when compared to their initial views.

4 Evaluation

In this section, we present the experimental evaluation results to verify the performance of the algorithm. There are two goals for the performance evaluation: the first is to compare the performance of our algorithm with that of a centralized outlier detection algorithm; the other is to observe the performance of our algorithm under different parameter configurations.

4.1. Experimentation Setup

We use Glomosim 2.03 [32] as our simulation platform, and the simulation setup is shown in Table I. We use three parameters to assess the correctness and efficiency of our algorithms: Correctness Rate (*CR*), Total Number of Packet for Outlier Detection (*TNPOD*), Communication Overhead (*CO*), and Convergence Time (*CT*). They are defined as follows:

$$CR = \frac{\text{Number of } \textit{True} \text{ Outliers Found}}{\textit{Total} \text{ Number of Outliers}}$$

$$TNPOD = \textit{Total Number of} \text{ Packets for Outlier Detection}$$

$$CO = \frac{TNPOD}{\textit{Total} \text{ Number of Packets in the network}}$$

$$CT = \textit{Time} \text{ taken to form a } \textit{consistent} \text{ global view of outliers}$$

TABLE 1
SIMULATION SETUP

Parameter	*Value*
Simulation area	150m × 150m, 300m × 300m, 450m × 450m, 600 m × 600m
Number of nodes	15, 25, 50, 100, 200
Transmission range	45m, 60m, 90m, 120m
Simulation duration	900 s
Mobility pattern of nodes	random waypoint
Maximum motion speed	5m/s, 10m/s, 20m/s
Number of bad nodes	5

Here we want to keep track of *CO* since we want to see the ratio of network traffic that outlier detection consumes over the whole network traffic. However, we also have interest in exploring the possible relationship between *TNPOD* and the number of nodes in the network.

We compare the performance of our collaborative outlier detection algorithm with that of a centralized algorithm. All nodes send their observations of misbehaviors to a designated *fusion* node, which then calculates the global outliers and floods the results out to all nodes in the network. An example of the centralized algorithm is shown in Fig 2.

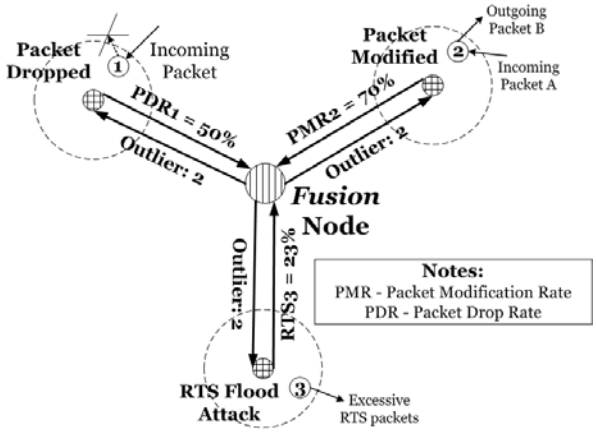

Fig. 2. Centralized Outlier Detection Algorithm

4.2. Experimentation Results

The first series of the experiments aim to compare the performance of our algorithm with that of the centralized algorithm. We use two different definitions to the outliers, which are distance to the nearest neighbor (*NN*) and average distance to *k* nearest neighbors (*k-NN*) in our experiments. All the experiments are simulated for thirty runs. The results are shown in Fig. 3 and Fig. 4.

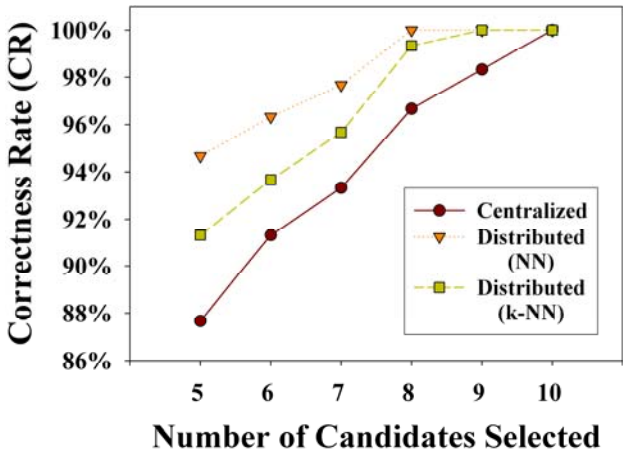

Fig. 3. CR of Various Algorithms in a 50-node MANET

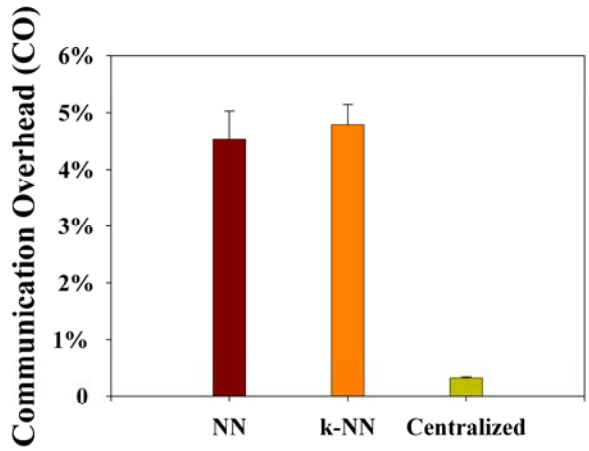

Fig. 4. CO of Various Algorithms in a 50-node MANET

From Fig. 3 we find that the Correctness Rate of our algorithm is higher than that of the centralized algorithm. This is true because of the robustness introduced by local gossiping in our algorithm. The centralized algorithm requires reliable communication links between the fusion node and other nodes, which cannot be guaranteed due to the node mobility and limited radio range. Moreover, the misbehavior of some nodes will also prevent some observations from successful delivery to the fusion node. Hence, the calculation of the global outliers ends up being based on a subset of the observations that the fusion node gets. In contrast, our algorithm is more resilient to various misbehaviors. By use of gossiping method, a node can receive the observations of its neighbors through different routes. Even if some of the observation messages are blocked by malicious nodes, a node may still get the blocked observations that are forwarded by some other nodes in its neighborhood. We also note that the Correctness Rate of *NN* is slightly higher than that of *k-NN*. By definition, *k-NN* finds *k* distinct supporting points to identify one outlier, whereas *NN* simply looks for the nearest neighbor to get one outlier. Hence,

k-NN is more prone to the noise brought by multiple supporting points, and consequently it will produce a lower Correctness Rate that *NN*.

Fig. 4 shows that while our algorithm produces higher communication overhead than the centralized algorithm, it is still within 5% of the total messages. Given that our algorithm produces higher correctness rate, and is more resilient to misbehaviors, the communication overhead of our algorithm is acceptable.

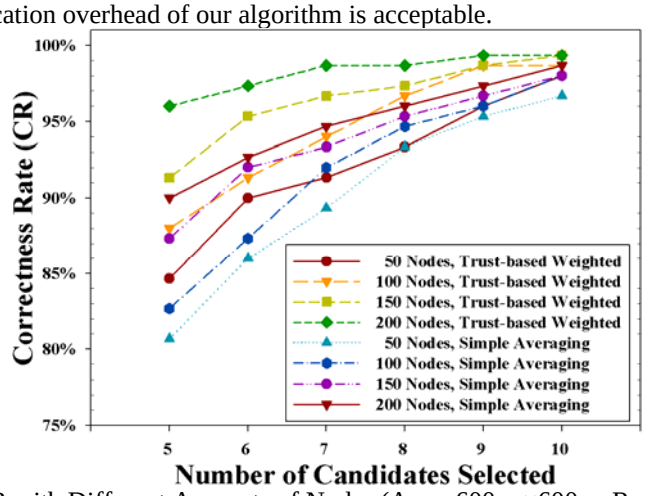

Fig. 5. CR with Different Amounts of Nodes (Area: 600m ×600m, Radio Range: 120m, Speed: 5m/s)

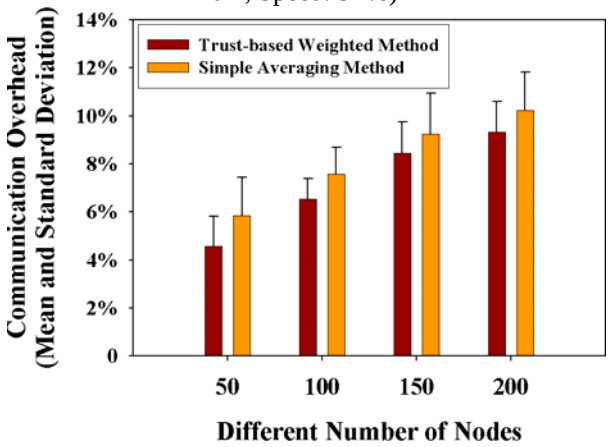

Fig. 6. CO with Different Amounts of Nodes (Area: 600m ×600m, Radio Range: 120m, Speed: 5m/s)

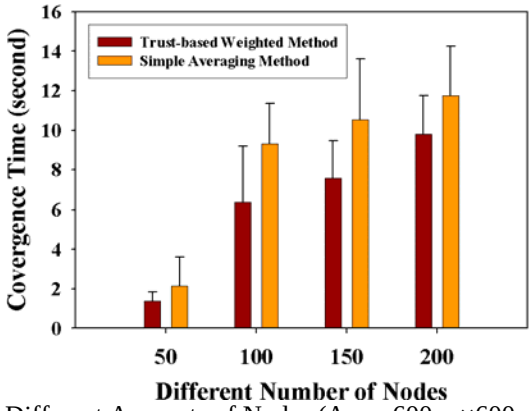

Fig.7. CT with Different Amounts of Nodes (Area: 600m ×600m, Radio Range: 120m, Speed: 5m/s)

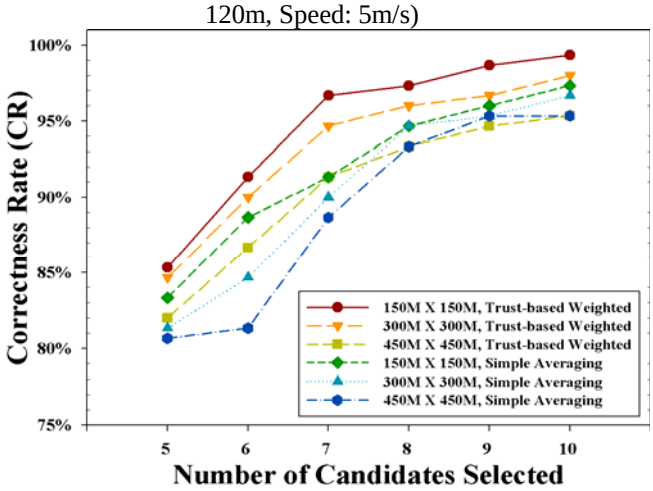

Fig. 8. CR with Different Simulation Areas (Num. of Nodes: 50, Radio Range: 60m, Speed: 5m/s)

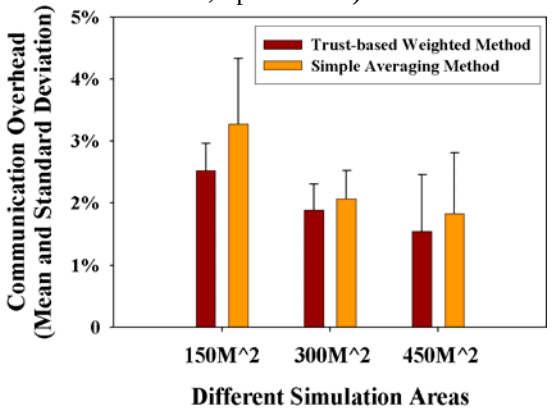

Fig. 9. CO with Different Simulation Areas (Num. of Nodes: 50, Radio Range: 60m, Speed: 5m/s)

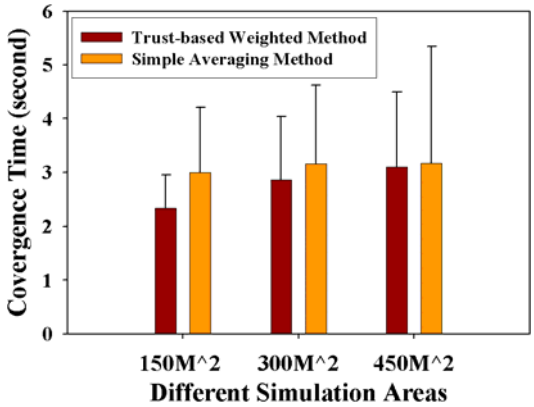

Fig. 10. CT with Different Simulation Areas (Num. of Nodes: 50, Radio Range: 60m, Speed: 5m/s)

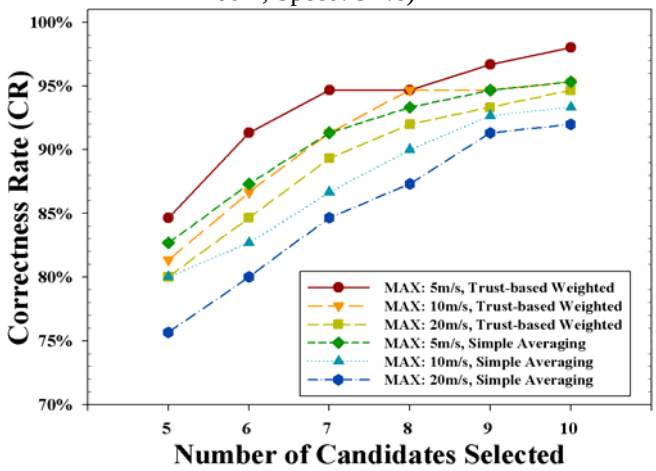

Fig. 11. CR with Different Maximum Motion Speed (Area: 600m ×600m, Num. of Nodes: 100, Radio Range: 60m)

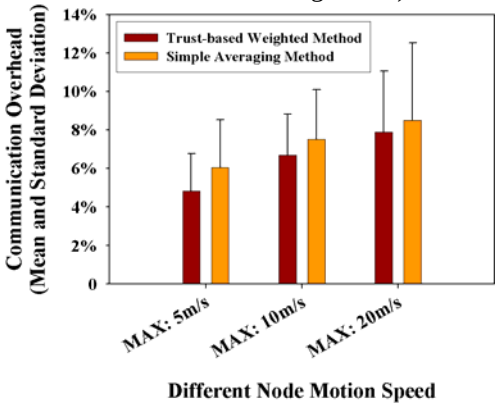

Fig. 12. CO with Different Maximum Motion Speed (Area: 600m ×600m, Num. of Nodes: 100, Radio Range: 60m)

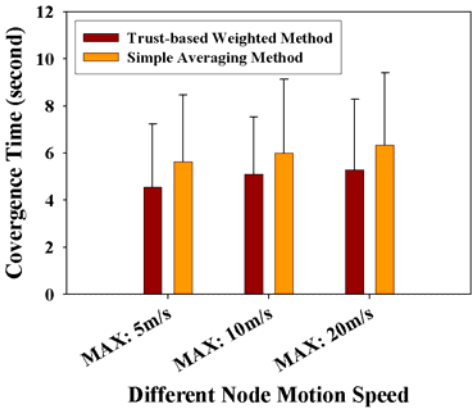

Fig. 13. CT with Different Maximum Motion Speed (Area: 600m ×600m, Num. of Nodes: 100, Radio Range: 60m)

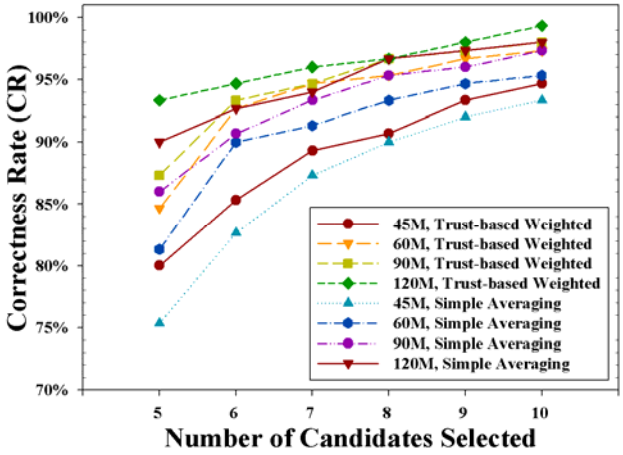

Fig. 14. CR with Different Radio Ranges (Num. of Nodes: 100, Area: 600m ×600m, Speed: 5m/s)

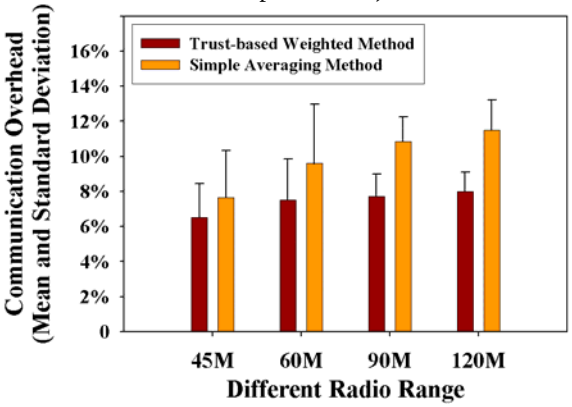

Fig.15. CO with Different Radio Ranges (Num. of Nodes: 100, Area: 600m ×600m, Speed: 5m/s)

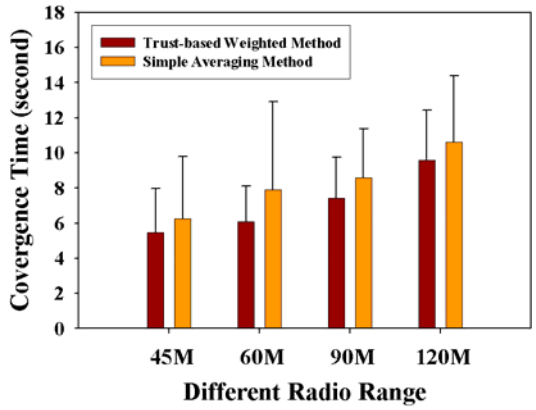

Fig. 16. CT with Different Radio Ranges (Num. of Nodes: 100, Area: 600m ×600m, Speed: 5m/s)

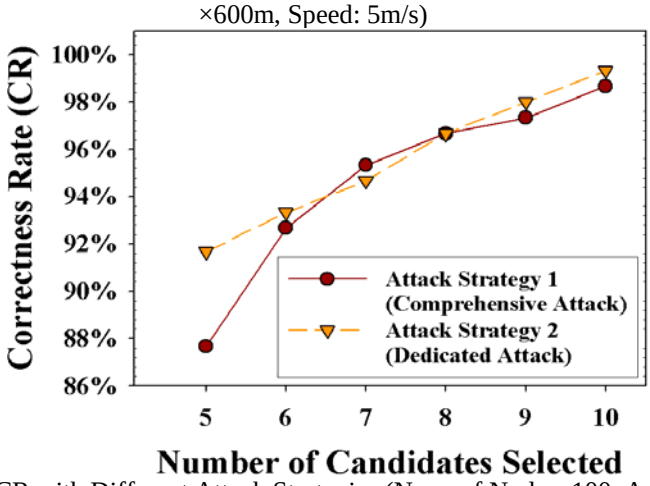

Fig. 17. CR with Different Attack Strategies (Num. of Nodes: 100, Area: 600m ×600m, Range: 90m, Speed: 5m/s)

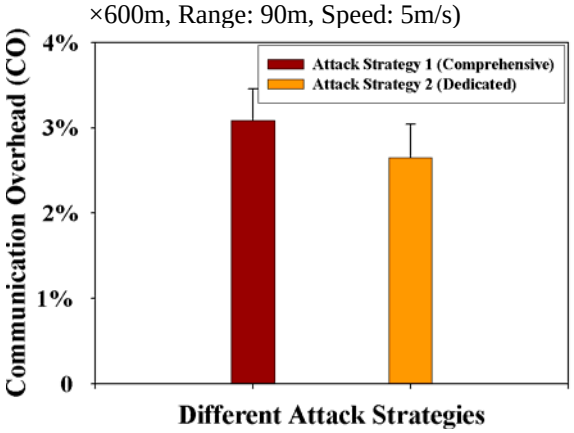

Fig. 18. CR with Different Attack Strategies (Num. of Nodes: 100, Area: 600m ×600m, Range: 90m, Speed: 5m/s)

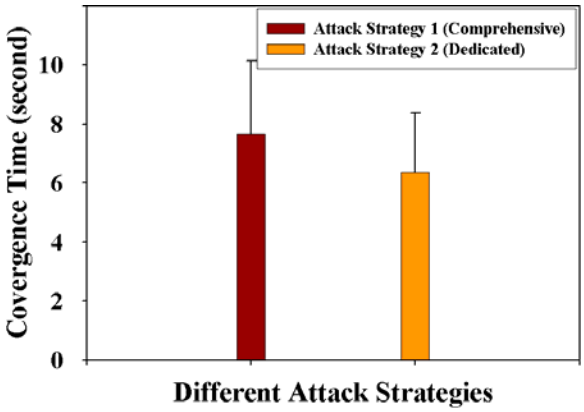

Fig. 19. CT with Different Attack Strategies (Num. of Nodes: 100, Area: 600m ×600m, Range: 90m, Speed: 5m/s)

The second series of experiments are designed to observe the performance of our algorithm under different parameter configurations. We have compared the performance of our algorithm under the following four conditions: different number of nodes, different simulation areas, different radio ranges, and different adversary strategies. The experimentation results are displayed in Fig. 5 through Fig. 19.

There are two adversary strategies that are used in our experiments. The first is called the *Comprehensive* strategy, in which the adversary simultaneously conducts several kinds of misbehaviors. This is used as a default in our simulations. In the second strategy, which is called the *Dedicated* strategy, the adversary conducts merely one kind of misbehavior at a time. Whereas most of the studies in the security area make the assumption that an adversary conducts one malicious behavior at a time, a recent study has shown that there may be cross-layer attacks [33] that involve misbehaviors at several layers of the protocol stack. In the *Dedicated* attack strategy, the malicious behaviors are less distinctive compared to other normal behaviors. In addition, the adversary may switch from various malicious behaviors from time to time, which makes it harder to keep track of each malicious behavior the adversary has conducted. Therefore, it is more difficult to identify an adversary if it deploys the dedicated attack strategy. We run experiments to compare the behavior of our approach against both.

From Fig. 5 through Fig. 7, we find that with an increase in the number of the nodes, the correctness rate increases, and the communication overhead also rises for both simple averaging method and trust-based weighted method. This is true because the information gathered to identify the outliers is generally more accurate if there are more observers. At the same time, more messages need to be exchanged amongst all the nodes to reach a consistent view when there are a larger amount of nodes. We also note that the trust-based weighted method yields better performance than the simple averaging method. Through the use of trust in the local view update process, a higher correctness rate can be produced in a shorter period of time together with a lower communication overhead

Fig. 8, Fig. 9, and Fig. 10 illustrate the results with different simulation areas. It is obvious that the correctness rate decreases as we increase the simulation area for the both methods. We also find that the communication overhead is reduced as the

simulation area becomes larger. Since the nodes have a lower probability to communicate with other nodes if the simulation area becomes larger, the correctness rate will surely become lower. Moreover, there will also be less communication overhead. As we have expected, the trust-based weighted method can also achieve better performance than the simple averaging method in different simulation areas.

The experimental results under different node motion speeds are demonstrated in Fig. 11 through Fig. 13. We find that with the increase to the maximum speed of nodes, the performance for both methods decreases. This is true because it is harder for the nodes to exchange their views when they are moving in a higher speed.

Fig. 14 through Fig. 16 shows how the experiment results differ with respect to different radio ranges. We conclude that the correctness rate significantly decreases as the radio range decreases. When it is more difficult for the nodes to exchange their local views when the radio range is smaller, the correctness rate of the final global view will surely be degraded. However, even if the radio range has been halved, the trust-based weighted method still yields a good performance.

In Fig. 17 through Fig. 19, the experiment results with two different attack strategies are discussed. We find that the algorithm will achieve a slightly higher correctness rate when the dedicated strategy is deployed. It is also clear that the communication overhead for the dedicated strategy will be lower than that for the comprehensive strategy. Nevertheless, our algorithm can achieve a satisfactory performance with both of the attack strategies.

5 Conclusion

In this paper, we propose a collaborative outlier detection algorithm for securing mobile ad hoc networks. The gossip-based outlier detection algorithm can help us identify the outliers, which are generally the nodes that have exhibited some kind of abnormal behaviors. Given the fact that benign nodes rarely behave abnormally, it is highly likely that the outliers are malicious nodes. Simulation results show that our algorithm is efficient and accurate with a small communication overhead.

References

[1] F. Grubbs, "Procedures for Detecting Outlying Observations in Samples", Technometrics, Vol. 11, No. 1, pp. 1-21, Feb. 1969.

[2] V. Barnett and T. Lewis, "Outliers in Statistical Data", New York, NY, John Wiley and Sons, 1994.

[3] A. Lazarevic, L. Ertoz, A. Ozgur, J. Srivastava and V. Kumar, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection", in *Proceedings of the Third SIAM International Conference on Data Mining*, San Francisco, CA, USA, May 2003.

[4] J. Zhang and M. Zulkernine, "Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection", in *Proceedings of IEEE International Conference on Communications (ICC 2006)*, Istanbul, Turkey, pp. 2388 – 2393, 2006.

[5] Z. Ferdousi and A. Maeda, "Unsupervised Outlier Detection in Time Series Data", in Proceedings of *the 22nd International Conference on Data Engineering Workshops (ICDEW06)*, Atlanta, GA, USA, Apr. 2006.

[6] B. Sheng, Q. Li, W. Mao and W. Jin, "Outlier Detection in Sensor Networks", in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc07)*, Montreal, Quebec, Canada, pp. 219 – 228, 2007.

[7] Y. Zhang and W. Lee, "Intrusion detection in wireless ad-hoc networks", in *Proceedings of the 8th International Conference on Mobile Computing and Networking (MobiCom)*, Atlanta, GA, USA, pp. 275-283, 2002.

[8] Y. Hu, A. Perrig and D. Johnson, "Ariadne: A Secure on-demand routing protocol for ad hoc networks", in *Proceedings of the 8th International Conference on Mobile Computing and Networking (MobiCom)*, Atlanta, GA, USA, pp. 12-23, 2002.

[9] B. Sun, Y. Guan, J. Chen and U. W. Pooch, "Detecting black-hole attack in mobile ad hoc networks", in *Proceedings of 5th European Personal Mobile Communications Conference* , Glasgow, Scotland, UK, pp. 490 – 495, 2003.

[10] S.Ramaswamy, R.Rastogi, and K.Shim, "Efficient Algorithms for Mining Outliers fromLarge Datasets", in *Proceedings of the 2000 ACM SIGMOD international Conference on Management of Data*, Dallas, Texas, USA, pp. 427-438, 2000.

[11]F.Anguilla and C. Pizzuti, "Fast Outlier Detection in High Dimensional Spaces", in *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, Helsinki, Finland, Lecture Notes In Computer Science, vol. 2431. Springer-Verlag, pp. 15-26, 2002.

[12] E. M.Knorr and R. T.Ng, "Algorithms for Mining Distance-Based Outliers in Large Datasets", in *Proceedings of the 24thinternational Conference on Very Large Data Bases (VLDB98)*, New York, NY, USA, pp. 392-403, 1998.

[13] E. M.Knorr and R. T.Ng, "Finding Intensional Knowledge of Distance-Based Outliers", in *Proceedings of the 25thinternational Conference on Very Large Data Bases (VLDB99)*, Edinburgh, Scotland, UK,pp. 211-222, 1999.

[14] S. D.Bay and M.Schwabacher, "Mining Distance-based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule", in *Proceedings of the Ninth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (KDD03)*, Washington, D.C., USA, pp. 29-38, 2003.

[15] C.Aggarwal and S.Yu, "An Effective and Efficient Algorithm for High-dimensional Outlier Detection", *The VLDB Journal*, vol. 14, no. 2, pp. 211-221, 2005.

[16] C. C.Aggarwal, "Re-designing Distance Functions and Distance-based Applications for High Dimensional Data", *ACM SIGMOD Record*, vol. 30, issue. 1, pp. 13-18, 2001.

[17] C. C.Aggarwal and P. S. Yu, "Outlier Detection for High Dimensional Data", in *Proceedings of the 2001 ACM SIGMOD international Conference on Management of Data (SIGMOD01)*, Santa Barbara, California, USA, pp. 37-46, 2001.

[18] A.Lazarevic and V.Kumar, "Feature Bagging for Outlier Detection", in *Proceeding of the Eleventh ACM SIGKDD international Conference on Knowledge Discovery in Data Mining (KDD05)*, Chicago, Illinois, USA , pp. 157-166, 2005.

[19]J.Branch, B.Szymanski, C.Giannella, R.Wolff, and H.Kargupta, "In-network Outlier Detection in Wireless Sensor Networks", in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS2006)*,Lisbon, Portugal,2006.

[20] T.Palpanas, D.Papadopoulos, V.Kalogeraki, and D.Gunopulos, "Distributed Deviation Detection in Sensor Networks", *ACM SIGMOD Record*, vol. 32, issue. 4, pp. 77-82, 2003.

[21] S.Subramaniam, T.Palpanas, D.Papadopoulos, V.Kalogeraki, and D.Gunopulos, "Online Outlier Detection in Sensor Data Using Non-parametric Models", in *Proceedings of the 32nd international Conference on Very Large Data Bases (VLDB06)*, Seoul, Korea, pp. 187-198, 2006.

[22] S.Marti, T. J.Giuli, K.Lai, and M.Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks", in *Proceedings of the 6th Annual international Conference on Mobile Computing and Networking (MOBICOM00)*, Boston, MA, USA, pp. 255-265, 2000.

[23] Y.Liu, C.Comaniciu, and H.Man, "A Bayesian Game Approach for Intrusion Detection in Wireless Ad Hoc Networks", in *Proceeding of the 2006 Workshop on Game theory For Communications and Networks (GAMENET06)*, Pisa, Italy, 2006.

[24] Y.Zhang and W.Lee, "Intrusion Detection in Wireless Ad-hoc Networks", in *Proceedings of the 6th Annual international Conference on Mobile Computing and Networking (MOBICOM00)*, Boston, MA, USA, pp. 275-283, 2000.

[25]H.Deng, Q.Zeng, and D.P.Agrawal, "SVM-based Intrusion Detection System for Wireless Ad Hoc Networks", in *Proceedings of the IEEE Vehicular Technology Conference (VTC03)*, vol. 3, pp. 2147–2151, Orlando, FL, USA 2003.

[26]O.Kachirski and R.Guha, "Intrusion Detection Using Mobile Agents in Wireless Ad Hoc Networks", in *Proceedings of the IEEE Workshop on Knowledge Media Networking*, pp. 153–158, Kyoto, Japan, 2002.

[27]C.Tseng, P.Balasubramanyam,C. Ko, R.Limprasittiporn, J.Rowe, and K.Levitt, "A Specification-based Intrusion Detection System for AODV", in *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks(SASN03)*, Fairfax, VA, USA, pp. 125–134, 2003.

[28] Y.Huang and W.Lee, "A Cooperative Intrusion Detection System for Ad Hoc Networks", in *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN03)*, Fairfax, VA, USA, pp. 135-147, 2003.

[29] M.Kefayati, H.R.Rabiee, S. G.Miremadi, and A.Khonsari, "Misbehavior Resilient Multi-path Data Transmission in Mobile Ad-hoc Networks", in *Proceedings of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN06)*, Alexandria, VA, USA, pp. 91-100, 2006.

[30] Y.Xue and K.Nahrstedt, "Providing Fault-Tolerant Ad hoc Routing Service in Adversarial Environments", *Wireless Personal Communication*, vol. 29, issue 3-4, pp. 367-388, 2004.

[31] L.Anderegg and S.Eidenbenz, "Ad hoc-VCG: A Truthful and Cost-efficient Routing Protocol for Mobile Ad Hoc Networks with Selfish Agents", in *Proceedings of the 9th Annual international Conference on Mobile Computing and Networking (MOBICOM03)*, San Diego, CA, USA, pp. 245-259, 2003.

[32] Glomosim 2.03, http://pcl.cs.ucla.edu/projects/glomosim/.

[33] J. Parker, A. Patwardhan and A. Joshi, "Cross-layer Analysis for Detecting Wireless Misbehavior", in *Proceedings of the IEEE Consumer Communications and Networking Conference*(*CCNC 2006*), Las Vegas, Nevada, USA, Jan. 2006.

[34] Q. He, D. Wu, and P. Khosla, "SORI: A Secure and Objective Reputation-based Incentive Scheme for Ad hoc Networks", in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 2, pp. 825-830, 2004.

[35]A. Patwardhan, F. Perich, A. Joshi, T. Finin, and Y. Yesha, "Active Collaborations for Trustworthy Data Management in Ad Hoc Networks", in *Proceedings of the 2nd IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, November 2005.

[36] V. Srinivasan, P. Nuggehalli, C.-F. Chiasserini, and R. R. Rao, "An analytical approach to the study of cooperation in wireless ad hoc networks", *IEEE Transactions on Wireless Communications*,4(2):722–733, March 2005.