Policy-based Malicious Peer Detection in Ad Hoc Networks

Wenjia Li, Anupam Joshi, Senior *Member, IEEE*, and Tim Finin

***Abstract*— Mobile Ad hoc Networks (MANETs) are susceptible to various node misbehaviors due to their unique features, such as highly dynamic network topology, rigorous power constraints and error-prone transmission media. Significant research efforts have been made to address the problem of misbehavior detection. However, little research work has been done to distinguish truly malicious behaviors from the faulty behaviors. Both the malicious behaviors and the faulty behaviors are generally equally treated as misbehaviors without any further investigation by most of the traditional misbehavior detection mechanisms. In this paper, we propose and develop a policy-based malicious peer detection mechanism, in which context information, such as communication channel status, buffer status, and transmission power level, is collected and then used to determine whether the misbehavior is likely a result of malicious activity or not. Simulation results illustrate that the policy-based malicious peer detection mechanism is able to distinguish malicious peers from faulty peers with high confidence. Moreover, the mechanism converges to a consistent view of malicious nodes amongst all the nodes with a limited communication overhead.**

I. Introduction

A MOBILE ad hoc network (MANET) is a self-configuring network of mobile devices that are connected by wireless links. In a MANET, each device is willing to serve as a router and share its transmission power with other devices because it is required to forward traffic that is irrelevant to its own interest.

Unlike the traditional wired networks, MANETs are generally more susceptible to malicious attacks as well as failures. Moreover, there are various sophisticated attacks that are difficult to identify [1, 2, 3]. Another threat comes from the compromised nodes that are taken over by an adversary. These compromised nodes can interfere with almost all of the network operations, such as route discovery, secure key management, and packet forwarding. Therefore, misbehavior surveillance and detection is a crucial method that has been widely used in MANETs to protect them from both external attackers and internal malicious nodes (see, for instance, [4, 6, 7]).

The misbehaviors observed by neighboring peers typically include dropping, modification, and misrouting of packets at the network layer, as well as false Request/Clears in the MAC layer [8]. However, many of these events may also occur due to environmental and mobility related reasons, not just malicious intent. For instance, a packet may be dropped when a node's buffer becomes full because of an inability to forward packets on a noisy channel. Even if they are both regarded as misbehaviors, malicious behaviors are far more dangerous than the faulty behaviors, because the goal of the malicious attackers is to disturb the network operations by carrying out the misbehaviors, whereas faulty nodes do not aim to intentionally disrupt the network and their effects are generally self limiting. Hence, it is essential that malicious attackers and faulty nodes be properly classified.

Significant research has been done on various node misbehaviors [4, 5, 6] and the corresponding countermeasures that can be done against them [7, 8, 9]. However, little research has been done to distinguish truly malicious behaviors from faulty behaviors, which is the problem that we address in this paper. We use *context information*, such as communication channel status (busy/idle), buffer status (full/not full), and transmission power level, to judge whether the misbehavior is the result of malicious intent or not. In our proposed mechanism, the peers in a MANET observe and record the abnormal behaviors of their neighbors in a manner similar to existing methods [4, 7, 8]. In contrast to most existing approaches however, each peer also simultaneously collects the *context information* within which the abnormal behaviors occur. When each peer decides if a node is malicious based on observing abnormal behaviors, it factors in the context information in a manner specified by a policy. In other words, the policy specifies, based on the context, how "abnormal" is defined. Moreover, all the nodes will exchange their observed abnormal behaviors as well as the observed context information with their neighbors. Therefore, each node can then make use of both local context information and remote context information to better understand the circumstance under which the misbehavior has occurred.

The remainder of this paper is organized as follows. In Section II, we present related work on misbehavior detection in MANETs. The malicious peer detection mechanism is described in Section III. In Section IV, we validate our proposed mechanism by various simulation scenarios, followed by conclusions in Section V.

Wenjia Li is with the Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County (UMBC), Baltimore, MD 21250 USA (phone: 410-455-3971; e-mail:wenjia1@ umbc.edu).

Anupam Joshi is with the Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County (UMBC), Baltimore, MD 21250 USA (e-mail: joshi@cs.umbc.edu).

Tim Finin is with the Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County (UMBC), Baltimore, MD 21250 USA (e-mail: finin@cs.umbc.edu).

II. Related Work

Misbehavior detection is a long studied topic in the security research, and the work on misbehavior detection (may also be called as intrusion detection in some cases) has produced a very rich literature in traditional [9, 10], P2P [11] and ad hoc networks [4, 7, 8]. In the latter, most contributions assume that there is no fixed network and security infrastructure that misbehavior detection mechanism can rely on.

Intrusion Detection Systems (IDS) provide an important framework for detecting various node misbehaviors in MANETs. Several approaches have been proposed to build IDSs on each individual peer due to the lack of a fixed infrastructure [7, 13, 14]. In these, each node is equipped with an IDS sensor, and each IDS sensor is assumed to be always monitoring the network traffic, which is not energy efficient given the limited battery power that each node has in MANETs. On the contrary, Huang et al. [15] propose a cooperative intrusion detection framework in which clusters are formed and the nodes in each cluster will take over the intrusion detection operations in turn. This cluster-based approach can definitely reduce the power consumption for each node.

Routing misbehavior is another kind of malicious activity that is common in ad hoc networks. Marti et al. [4] introduce two related techniques, namely *watchdog* and *pathrater*, to detect and isolate misbehaving nodes, which are nodes that do not forward packets. There are also other proposed solutions that aim to cope with the routing misbehaviors [16, 17, 18].

In previous work [12, 19], we have described a gossip-based misbehavior detection algorithm in which the outlier detection method is adopted to identify various node misbehaviors. Weighted voting and the Dempster-Shafer Theory of evidence (DST) are used to combine multiple local views of misbehaving nodes from different nodes. However, none of these previous works endeavored to reveal the difference between the malicious peers and the faulty peers, both of which may be treated as misbehaving nodes with no difference.

III. Malicious Peer Detection Using Context Information

The goal of the malicious peer detection system is to properly identify the malicious peers in MANETs by using the distributed misbehavior detection mechanism as well as the context information collection scheme. The distributed misbehavior detection mechanism is similar to the gossip-based outlier detection mechanism [12] in which a certain number of outliers are identified in terms of some abnormal behaviors observed by neighbors, such as packet drops, misroutes or modifications. Gossiping in MANETs generally refers to the repetitive probabilistic exchange of messages between two peers in MANETs. In the context collection scheme, each node observes and records the current network and node context information, such as channel status, buffer status and transmission signal strength. When the distributed misbehavior detection mechanism attempts to identify the malicious peers, the collected context information will be utilized to help decide which nodes are truly malicious nodes and which nodes are merely faulty nodes that randomly exhibit misbehaviors.

A. System Architecture

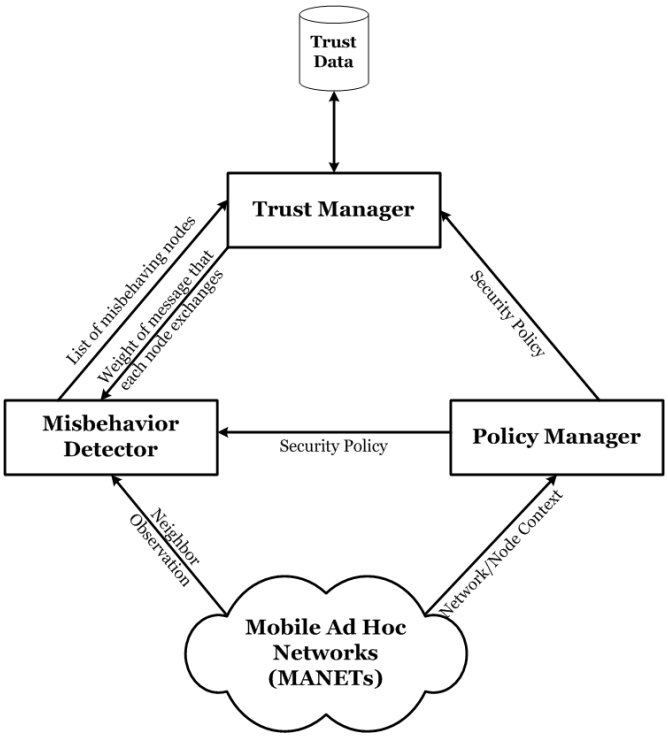


Figure 1. The system architecture comprises three main compoenents: a misbehavior detector, a policy manager and a trust manager.

Figure 1 illustrates the system architecture with three components: the Misbehavior Detector, a Trust Manager, and a Policy Manager. As this figure shows, both the Misbehavior Detector and Policy Manager actively sense the network. However, the Misbehavior Detector aims to observe and record the abnormal behaviors of neighbors, whereas the Policy Manager attempts to detect changes in network context. Once the Misbehavior Detector identifies misbehaving nodes, it notifies the Trust Manager of its findings. Then, the Trust Manager will update the trustworthiness value based on the information about misbehaviors. The Misbehavior Detector will get feedback from both the Trust Manager and Route Manager once they finish updating the relevant information. On the other hand, the other information stream collected from the network, which is the network context, will be processed and reasoned by the Policy Manager. If it finds out that the policies should be updated, the Policy Manager will propagate the updated policies to all other components.

B. Formulation

A *node* in MANETs is defined as a system entity that can observe the behaviors of other entities within its radio transmission range. A *neighbor* of a node *N* is defined as a node that is located within *N*'s radio transmission range. We assume that each node transmits at full power. If the underlying MAC protocol uses variable power settings, then a neighbor is a node that can listen to the transmission.

A node not only observes the abnormal behaviors that its neighbors exhibit, it will also keep track of the total number of

packets that each node has received. The rate of abnormal traffic over the total traffic is an appropriate criterion for a node to decide if its neighbor is potentially a malicious peer or not. For example, if all the nodes agree to observe the behaviors of packet drop, misroute and modification, then the packet drop rate (PDR), packet modification rate (PMOR) and packet misroute rate (PMIR) can be defined as follows, respectively.

$$PDR = \frac{Number\ \ of\ \ Packets\ \ Dropped}{Total\ \ Number\ \ of\ \ Incoming\ \ Packets}$$

$$PMOR = \frac{Number\ \ of\ \ Packets\ \ Modified}{Total\ \ Numer\ \ of\ \ Incoming\ \ Packets}$$

$$PMIR = \frac{Number\ \ of\ \ Packets\ \ Misrouted}{Total\ \ Number\ \ of\ \ Incoming\ \ Packets}$$

We define the *trustworthiness* of a node N_k as a real value θ_k that reflects the probability with which the node will perform the exact actions that it is supposed to take. θ_k can be assigned any real value in the range [0, 1], and the higher the value of θ_k, the node N_k is more reliable and has a higher probability to take the correct actions. The trustworthiness θ_k of a node N_k can be defined as a function of all misbehaviors that other nodes have observed for the node N_k. In other words, the trustworthiness θ_k can initially be derived as follows.

$$\theta_k = 1 - \sum_i M_{ki}$$

We note that different misbehaviors may be caused by different reasons. For instance, packet dropping and packet modification are both viewed as 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 should vary the punishment for different misbehaviors according to the context of their occurrence. Namely, the calculation of trustworthiness θ_k is adjusted as follow.

$$\theta_k = 1 - \sum_i P_i * M_{ki}$$

Here P_i denotes the *punishment factor* for the i-th misbehavior, which indicates the severity degree of its outcome. The punishment factor is the function of the context in which the misbehavior occurs. In other words, the same misbehavior may be punished differently when it occurs in different circumstances. The punishment factor is determined by the *Policy Manager*, which will be discussed later in more details. M_{ki} represents the rate of this misbehavior over the total observed behaviors. For example, if packet drop, packet modification, and packet misroute are the three exact misbehaviors we are observing, then θ_k can be derived as follow. The punishment factors for different misbehaviors will need to be selected so that θ_k always falls in the range of [0, 1]. This does not however necessitate that P_is sum to 1.

$$\theta_k = 1 - P_{drop} * PDR - P_{mod\ ification} * PMOR - P_{misroute} * PMIR$$

C. Misbehavior Detection

As we have discussed in the previous section, the gossip-based outlier detection algorithm is used in the Misbehavior Detector to identify misbehaving nodes. The outlier detection algorithm has the following four steps, viz. local view formation, local view exchange, view combination, and global view formation. The basic functionality of the Misbehavior Detector is similar to the outlier detection algorithm that we have proposed earlier [12]. However, the context information offered by the Policy Manager is added to the outlier detection algorithm, and the context information is used to decide the circumstance under which the misbehaviors occur. In this way, a node may distinguish a malicious node from other faulty nodes because they carry out the misbehaviors under different circumstances.

D. Trust Management

A variety of trust and reputation management approaches have been studied during the past decades, for instance [20, 21, 22]. All of these trust management approaches can fit our system. For the experiments presented in this paper, we adopt a simple but well-defined trust management scheme, in which each nodes trustworthiness θ_k is initially set to a default value. A peers θ_k is modified whenever we obtain any novel information regarding its trustworthiness in terms of both direct observation results from the node itself and indirect observation results from other nodes. Direct observation results and indirect observation results are generally called *first-hand information* and *second-hand information*, respectively [23].

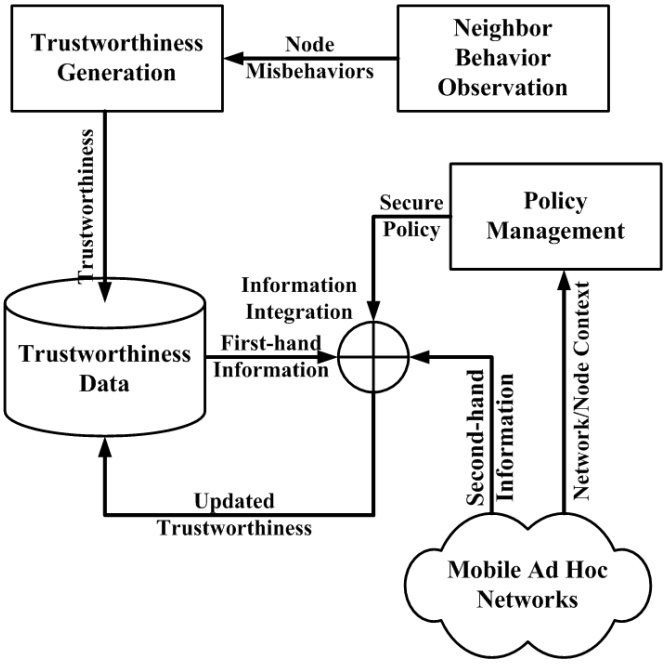


Figure 2. Trust Management

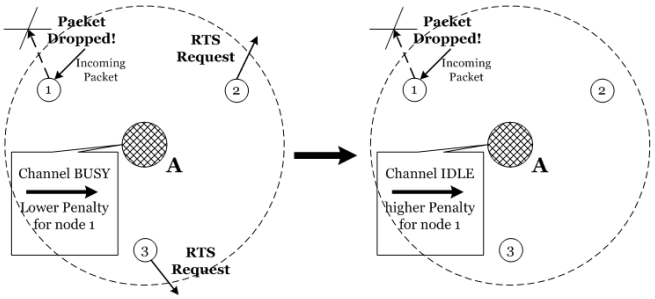
The trust management scheme is shown in Figure 2. In the trust management scheme, we adopt DST to combine first-hand information and second-hand information. Furthermore, Policy Manager collects the network and node contexts, and then the context information will feed to the view combination in terms of secure policy. In this way, trustworthiness θ_k is reduced for various misbehaviors according to various contexts.

We should also note that there is no "global" information on trustworthiness, and the algorithm converges with a global picture of outliers, not with a global picture of trustworthiness. Each node has its own trust measures which evolve, and could be different from that of the others.

E. Policy Management

The Policy Manager is responsible to collect and record network/node context information, and then enforce the corresponding security policies so that the Misbehavior Detector and Trust Manager can make use of the context information when they identify the malicious nodes.

For example, we can make use of the context information when we need to punish a node because it has dropped a certain number of packets. Packet dropping is generally carried out by two types of nodes: malicious nodes and selfish nodes. Malicious nodes aim to destroy network operations by dropping some portion or all of the packets that flow through them; whereas selfish nodes simply want to preserve their resource, and consequently they occasionally refuse to forward packets for others. However, packet dropping may also be caused by external factors, such as (1) overflowing buffers caused by overwhelming incoming traffic, or (2) channel collision caused by too many simultaneous Request-To-Send (RTS) packets by different nodes. Because nodes themselves are unable to control these external factors, packet dropping resulting from these should be punished less than packet dropping induced by malicious or selfish intents.



a) Node A observes that the channel is busy, so it reduces the penalty factor for node 1

b) Node A later finds that the channel is idle now, so it sets higher the penalty factor for node 1

Figure 3. Punishing Misbehaving Nodes in Different Contexts

Figure 3 demonstrates an example of how context information can be utilized when punishing node misbehaviors. In Figure 3a), a node *A* senses that the communication channel is busy, because node *2* is sending out RTS packets and attempting to occupy the channel. At the same time, node *A* also finds that node *1* drops some incoming packets. In this case, node *A* may find that the channel is busy when node *1* drops packets. As a result, node *A* decides to decrease the punishment for node *1* because it may be forced to drop packets. A rule demonstrating this policy is shown below, which is written using SWRL [24].

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#rule1"/>
 <ruleml:_body>
  <swrlx:individualPropertyAtom swrlx:property="isDropPacket">
   <ruleml:var>x2</ruleml:var>
  </swrlx:individualPropertyAtom>
  <swrlx:individualPropertyAtom swrlx:property="senseChannelBusy">
   <ruleml:var>x1</ruleml:var>
   <ruleml:var>x2</ruleml:var>
  </swrlx:individualPropertyAtom>
 </ruleml:_body>
 <ruleml:_head>
  <swrlx:individualPropertyAtom swrlx:property="reducePunishment">
   <ruleml:var>x1</ruleml:var>
   <ruleml:var>x2</ruleml:var>
  </swrlx:individualPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

In contrast, as is shown in Figure 3b), node *A* finds that the channel is idle when node *1* still drops some more packets. In this case, node A decides to increase the punishment factor for node *1* because there is no external factor for its packet dropping at this time. A SWRL [24] rule below illustrates this policy.

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#rule2"/>
 <ruleml:_body>
  <swrlx:individualPropertyAtom swrlx:property="isDropPacket">
   <ruleml:var>x2</ruleml:var>
  </swrlx:individualPropertyAtom>
  <swrlx:individualPropertyAtom swrlx:property="senseChannelIdle">
   <ruleml:var>x1</ruleml:var>
   <ruleml:var>x2</ruleml:var>
  </swrlx:individualPropertyAtom>
 </ruleml:_body>
 <ruleml:_head>
  <swrlx:individualPropertyAtom swrlx:property="addPunishment">
   <ruleml:var>x1</ruleml:var>
   <ruleml:var>x2</ruleml:var>
  </swrlx:individualPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

IV. PERFORMANCE EVALUATION

In this section, we examine the performance of the policy-based malicious peer detection method. We compare the performance of our mechanism with the baseline algorithm, which is the outlier detection algorithm studied in [12].

A. Simulation Setup

We use GloMoSim 2.03 [25] as the simulation platform. Table I lists the parameters used in the simulation scenarios.

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Simulation area	600m × 600m
Number of nodes	50, 100, 200
Transmission range	60m, 90m, 120m
Mobility pattern	Random waypoint
Node motion speed	10m/s
Number of malicious nodes	5, 10, 20
Simulation time	900 s

Here we also assume that transmission range equals to radio

range. In other words, the definition of neighbor in our simulation is someone who is always within the transmission range of the node.

Three parameters are employed to evaluate the correctness and efficiency of our algorithm: Correctness Rate (*CR*), Communication Overhead (*CO*), and Convergence Time (*CT*). They are defined as follows.

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

$$CO = \frac{Number\ \ of\ \ Packets\ \ for\ \ Outlier\ \ Detection}{Total\ \ Number\ \ of\ \ Packets\ \ in\ \ network}$$

$$CT = Time\ \ taken\ \ to\ \ form\ \ a\ \ consistent\ \ global\ \ view$$

Each simulation scenario has 30 runs with distinct random seed, which ensures a unique initial node placement for each run. In the simulation, each node observes and records the channel status (busy/idle), and exchanges this context information with other nodes.

B. Adversary Model

In our simulation, nodes either abide by various MANET protocols, such as AODV routing protocol, or their behaviors deviate from the protocol definition either intentionally (i.e. attackers) or unintentionally (i.e. faulty nodes). Both attackers and faulty nodes can do harm to the network functionalities, and consequently we regard them both as adversaries. In general, adversaries can partially or completely drop, modify or misroute any packet that is sent to them. We also assume that they can deploy the Denial-of-Service (DoS) attack by continuously sending out Request-To-Send (RTS) packets so as to improperly occupy the communication channel all the time, which is also regarded as the RTS flood attack.

The adversaries may mix all these misbehaviors so that it will be more difficult to identify their misbehaviors if observed only from one or two perspectives. More importantly, the adversaries are capable of deliberately injecting faulty data and spreading these fake data to other benign nodes. In this way, the benign nodes may be induced to generate faulty reports in which benign nodes can be misclassified as misbehaving nodes.

C. Simulation Results

The goal of the simulations is to observe the performance of our algorithm under different parameter configurations. We have compared the performance of our algorithm under the following five conditions: different number of nodes, different radio ranges, and different percentage of malicious nodes. The simulation results are showed in Figures 4 through 6.

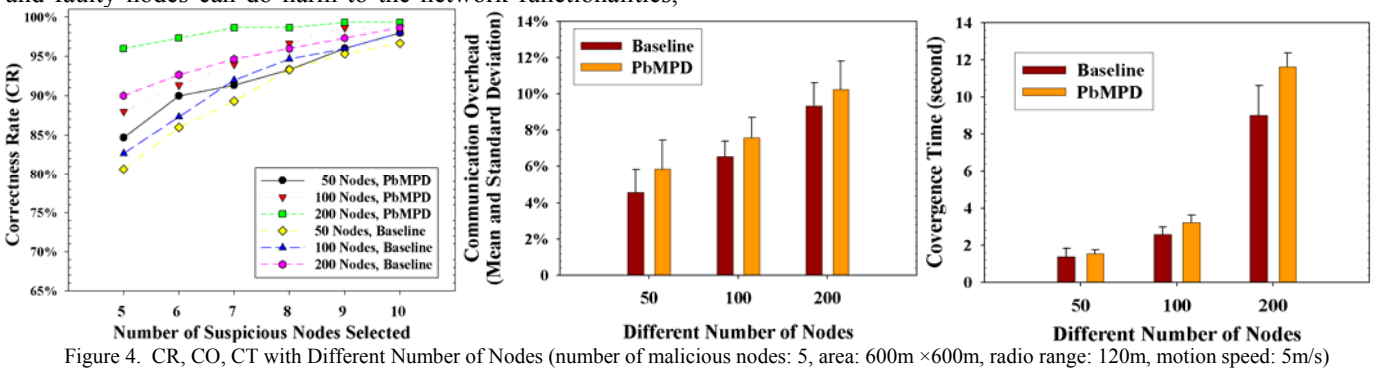


Figure 4. CR, CO, CT with Different Number of Nodes (number of malicious nodes: 5, area: 600m ×600m, radio range: 120m, motion speed: 5m/s)

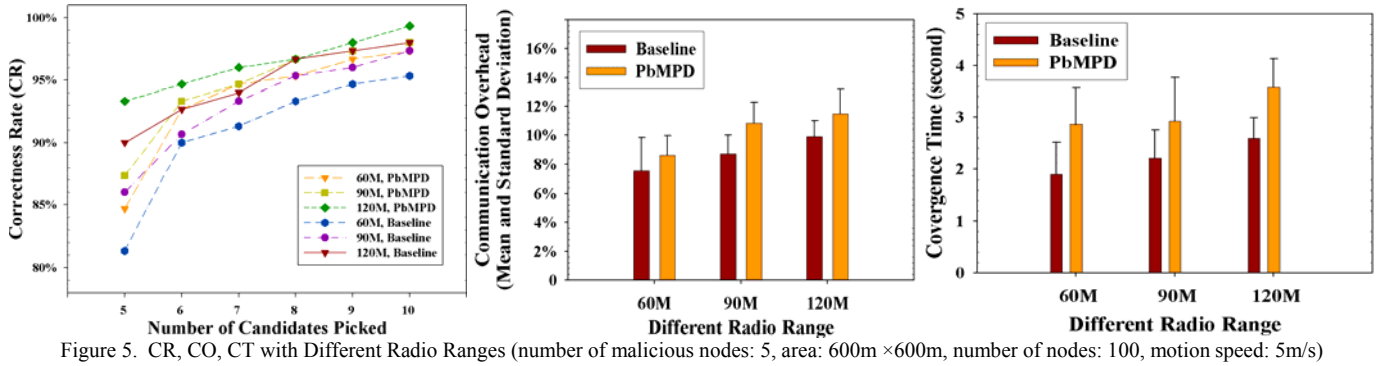


Figure 5. CR, CO, CT with Different Radio Ranges (number of malicious nodes: 5, area: 600m ×600m, number of nodes: 100, motion speed: 5m/s)

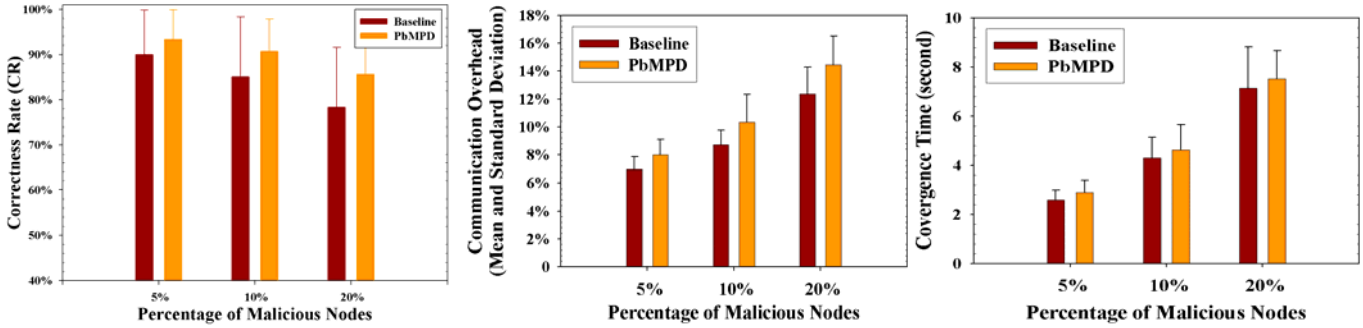


Figure 6. CR, CO, CT with Different Number of Malicious Nodes (radio range: 120m, area: 600m ×600m, number of nodes: 100, motion speed: 5m/s)

Figure 4 exhibits the performance of our PbMPD algorithm with different number of nodes. From Figure 4 we find that when the number of nodes is increased, the algorithm yields a higher correctness rate, but it also introduces more communication overhead. This is consistent with our analysis 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 more nodes. We also note that PbMPD demonstrates better correctness rate than the baseline algorithm. However, PbMPD also introduces slightly higher communication overhead, and PbMPD may take a longer time to converge. Because the context information needs to be exchanged amongst the nodes besides the local views, PbMPD is supposed to introduce more communication overhead. Similarly, it takes slightly more time for PbMPD to converge since it will take extra time for each node to exchange context information with other nodes.

Figure 5 shows how the simulation results differ with different transmission ranges. We find that with a smaller radio range, both PbMPD and the baseline algorithm suffer from performance degradation. When it is more difficult for nodes to exchange the local views, the correctness rate of the final global view will surely be degraded. On the other hand, we may also conclude from Figure 5 that PbMPD produces a higher correctness rate than the baseline algorithm with the same radio transmission range. This is the case because PbMPD has taken the context information into account, and consequently the faulty peers are less likely to be misclassified as malicious peers even if they both exhibit misbehaviors.

Figure 6 shows the simulation results with different percentage of malicious nodes. It is obvious that PbMPD can yield a much better performance than the baseline algorithm with a higher percentage of malicious nodes. This is true because with a higher percentage of malicious nodes, it will be more likely that the benign nodes are forced to drop packets because the malicious nodes consume a large portion of the channel time to conduct their malicious behaviors. Hence, when there are a higher percentage of malicious nodes, the performances of the baseline algorithm degrade noticeably. On the other hand, PbMPD can properly handle the malicious peer detection problem even in a more hostile environment because it relies on the context information to decide which nodes are truly malicious.

D. *Discussion*

We also note from the simulation that our PbMPD algorithm can properly distinct malicious peers from the faulty peers, whereas the baseline algorithm cannot do so at all times. For example, in the first simulation scenario in Figure 4, we observe that when there are 100 nodes in the network, node 24 and node 83 sometimes exhibit packet dropping misbehavior, and consequently they are misclassified as malicious peers by the baseline algorithm in some cases. Since neither is set as the malicious peers in our simulation setup, we further analyze the simulation output, and we find that some of their neighbors are malicious peers. Both of them are forced to drop some portion of the incoming packets because their malicious neighbors deliberately occupy the communication channel for a long period of time. Since the baseline algorithm neither collects the context information, nor does it utilize the policy to properly adjust the punishment in different context, node 24 and node 83 are sometimes misclassified as malicious peers.

On the other hand, our PbMPD algorithm first collects the context information, and then makes use of the context information as well as the corresponding policy to determine the punishment factor for each of the misbehaviors. Therefore, the faulty nodes can be correctly separated from the malicious nodes, and the correctness rate of PbMPD is surely higher than that of the baseline algorithm.

V. CONCLUSION

In this paper, a policy-based malicious peer detection algorithm is described discriminates the truly malicious attackers from the faulty nodes, both of which may exhibit misbehaviors. Through the use of context information, such as channel status, buffer status and transmission signal strength, a node can determine the circumstance under which the misbehaviors occur. As a result, the node can then tell whether a node is forced to act as a misbehaving node or not, and reveal the truly malicious attackers. The simulation results show that the approach is highly resilient to malicious attackers, and it can properly distinguish the malicious peers from the faulty peers with a limited communication overhead.

REFERENCES

[1] Y. Hu, A. Perrig and D. Johnson. Ariadne: A Secure On-demand Routing Protocol for Ad Hoc Networks. In *Proc. of MobiCom'02*, pages 12-23, 2002.

[2] A. Perrig, Y.-C. Hu, and D. B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in wireless networks. In *Proc. of IEEE INFOCOM' 03*, pages 1976-1986, 2003.
[3] J. R. Douceur. The Sybil Attack. In *Proc. of the 1st International Workshop on Peer-to-Peer System, Lecture Notes In Computer Science (vol. 2429)*, Springer-Verlag, pages 251 – 260, 2002.
[4] S. Marti, T. J. Giuli, K. Lai and M. Baker. Mitigating Routing Misbehaviors in Mobile Ad hoc Networks. In *Proc. of ACM MobiCom '00*, pages 255-265, Aug. 2000.
[5] M. Hollick, J. Schmitt, C. Seipl and R. Steinmetz. On the Effect of Node Misbehaviors in Ad Hoc Networks. In *Proc. of IEEE ICC' 04*, volume 6, pages 3759-3763, Jun. 2004.
[6] B. Zhu, K. Ren and L. Wang. Anonymous Misbehavior Detection in Mobile Ad Hoc Networks. In *Proc. of ICDCS'08* Workshops, pages 358-262, Jun. 2008.
[7] Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad-hoc Networks. In *Proc. of MOBICOM'00*, pages 275-283, 2000.
[8] J. Parker, A. Patwardhan and A. Joshi. Cross-layer Analysis for Detecting Wireless Misbehavior. In *Proc. of CCNC '06*, pages 6-9, Jan. 2006.
[9] D. E. Denning. An Intrusion Detection Model. *IEEE Transactions on Software Engineering,* vol. SE-13, issue 2, pages 222-232, Feb. 1987.
[10] W. Lee and S. J. Stoflo. Data Mining Approaches for Intrusion Detection. In Proc. of *7th USENIX Security Symposium*, 1998.
[11] R. Janakiraman, M. Waldvogel and Q, Zhang. Indra: A Peer-to-peer Approach to Network Intrusion Detection and Prevention. In Proc. of *12th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2003),* pages 226-231, 2003.
[12] W. Li and A. Joshi. Outlier Detection in Ad Hoc Networks Using Dempster-Shafer Theory. In *Proc. of MDM '09*, May 2009.
[13] H. Deng, Q. Zeng, and D. P. Agrawal. SVM-based Intrusion Detection System for Wireless Ad Hoc Networks. In *Proc. of IEEE VTC'03*, vol. 3, pages 2147–2151, 2003.
[14] C. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, and K. Levitt. A Specification-based Intrusion Detection System for AODV. In *Proc. of SASN'03*, pages 125–134, 2003.
[15] Y. Huang and W. Lee. A Cooperative Intrusion Detection System for Ad Hoc Networks. In *Proc. of SASN'03,* pages 135-147, 2003.
[16] M. Kefayati, H. R. Rabiee, S. G. Miremadi, and A. Khonsari. Misbehavior Resilient Multi-path Data Transmission in Mobile Ad-hoc Networks. In *Proc. of SASN'06*, pages 91-100, 2006.
[17] Y. Xue and K. Nahrstedt. Providing Fault-Tolerant Ad hoc Routing Service in Adversarial Environments. *Wireless Personal Communication,* vol. 29, issue 3-4, pages 367-388, 2004.
[18] L. Anderegg and S. Eidenbenz. Ad hoc-VCG: A Truthful and Cost-efficient Routing Protocol for Mobile Ad Hoc Networks with Selfish Agents. In *Proc. of MOBICOM '03*, pages 245-259, 2003.
[19] W. Li, J. Parker and A. Joshi. Security through Collaboration in MANETs. In *Proc. of the 4th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom '08)*, Nov 2008.
[20] S. Buchegger, J. Le Boudec. Performance analysis of the CONFIDANT protocol. In *Proc. of the 3rd ACM international Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC '02)*, pages 226 - 236, Jun 2002.
[21] A. Patwardhan, F. Perich, A. Joshi, T. Finin, and Y. Yesha. Active Collaborations for Trustworthy Data Management in Ad Hoc Networks. In *Proc. of the 2nd IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2005)*, November 2005.
[22] 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.
[23] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas. Robust cooperative trust establishment for MANETs. In *Proc. of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '06)*, pages 23-34, Oct 2006.
[24] SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.w3.org/Submission/SWRL/.
[25] Glomosim 2.03, http://pcl.cs.ucla.edu/projects/glomosim/.