Fair Reputation Evaluating Protocol for Mobile Ad Hoc Network*

Zhu Lei[1], DaeHun Nyang[1], KyungHee Lee[2], and Hyotaek Lim[3]

[1] Information Security Research Laboratory, INHA University
[2] Department of Electrical Engineering, The University of Suwon
[3] Division of Computer Information Engineering, Dongseo University
koudai@seclab.inha.ac.kr, nyang@inha.ac.kr, khlee@suwon.ac.kr, htlim@dongseo.ac.kr

Abstract. Ad hoc network is a society of nodes who work in cooperative manner in accordance with self regulatory protocol. So reputation and trust should be built up and selfishness be dealt with a proper regulatory protocol. Selfish nodes are those which do not behave as the protocol specifies, with a wish to conserve power. This paper proposes an environmental compensation algorithm to the General Reputation Model. The algorithm provides a scheme as a mean to mitigate the detrimental effect of selfish nodes. And it deals for the first time with the security -the environment's influence on nodes' behavior. It also shows how to establish trusts in different areas with different environmental characteristics.

Keywords: Security, Ad Hoc, Environment, Trust.

1 Introduction

Reputation systems have been proposed for a variety of applications. Selection of good partners in a peer-to-peer communications and choice of faithful trade partners in online auctioning are among those. Under the mobile ad hoc networking architecture, the detection of misbehaving nodes provides the basis of reputation system.

There is a trade off between efficiency in using the available information and robustness against false ratings. If the ratings are made by others, the reputation system can be vulnerable to false accusations or praise. If it is established on the basis of one's own experience only, it does not provide a comprehensive rating neglecting other's experiences.

The goal of our model is to make neighborhood survailence systems both robust against selfishness and efficiency in detecting misbehavior. Our proposal is making use of all the available information, i.e. both positive or negative, and one's own or other's. And to guarantee the robustness of the reputation system, we show a way to deal with false ratings.

* This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement)(IITA-2006-C1090-0603-0028).

Y. Wang, Y. Cheung, and H. Liu (Eds.): CIS 2006, LNAI 4456, pp. 683–693, 2007.

2 The Reputation Methods in Mobile Ad Hoc Networks

The use of reputation systems in many different areas is increasing, not least because of their widely publicised use in online auctions and product reviews, see, for example eBay and Amazon [14]. Mui et al. [13] gave many examples of how reputation systems are used.

Reputation systems are used to decide whom to trust, and whom to encourage trustworthy behaviour. Resnick and Zeckhauser [12] identified three goals for reputation systems:

1. To provide information to distinguish between a trustworthy principal and an untrustworthy principal,
2. To encourage principals to act in a trustworthy manner, and
3. To discourage untrustworthy principals from participating in the service the reputation mechanism is present to protect.

Two reputation mechanisms that have been proposed to help protect ad hoc routing are the Cooperation of Nodes: Fairness in Dynamic Ad-Hoc NeTworks (CONFIDANT) protocol [1], and the Collaborative Reputation Mechanism(CORE) protocol [2], which work in a similar way.

But both of them have some problems. For example, by placing more weight on past behaviour, CORE scheme is vulnerable to an attack where a node can build up a good reputation before behaving maliciously for a period. Attacks involving 'building up credit' before behaving selfishly have less effect in CONFIDANT, as good behaviour is not rewarded, so all nodes are always under suspicion of bad behaviour. However, this makes CONFIDANT less tolerant of failed nodes, which may be exhibiting failed behaviour due, for example, to loss of power.

3 The General Reputation Method

3.1 Assumptions

We assume certain things:

- Each node has a unique id.
- Links are bidirectional.
- Nodes do not have a prior "trust" relationships.
- All nodes give correct reputation to others.
- Misbehaving nodes do not forward data packets, but act correctly for everything else(which is selfishness).
- There are no malicious nodes (who want to destroy the network).

3.2 Direct Trust (DT)

When we want to know if we can trust some node B, we can route some packets via B and see (by sniffing in promiscuous mode) if B forwards them correctly.

The fraction of correctly forwarded packets in relation to the total amount of packets then gives us some idea on how trustworthy B is.

$$DT(A,B) = \frac{\sharp forwarded}{\sharp sent} \tag{1}$$

$\sharp$ forwarded = number of packets (coming from A) correctly forwarded by B.
$\sharp$ sent = number of packets sent to B (by A).

3.3 Indirect Trust (IDT)

What happens if a new node comes? If A now wants to get references for B, he creates a reputation request, sets himself as source, sets B as target and broadcasts it to his neighbors (ttl = 1). Every node N receiving this request then looks if he has a direct trust value for B and if yes creates a reputation reply (from him to A) which is carrying this value. After some time A can then combine the received values to a reputation value for B:

$$IDT(A,B) = \frac{\sum_{i=1}^{n} DT(A,N_i) \times DT(N_i,B)}{n} \tag{2}$$

N_i : node A's i-th neighbor node

This indirect trust value depends on when it is calculated and how many answers (route replies) have been received (and from whom). The question is how we combine all the *direct trust values* from the reputation replies together to one *indirect trust value*. One possibility is to weight them with the *direct trust values* we have (as in equation(2)). Another possibility is to look at the answers and compare them.

3.4 Reputation

Now we have some direct trust values and some indirect trust values. They can be combined in the following way:

$$REP(A,B) = \omega \times DT(A,B) + (1-\omega) \times IDT(A,B) \quad (0 < \omega < 1) \tag{3}$$

ω : the weight we put on DT(A,B).

4 Reputation Compensation Protocol

There are a lot of reputation methods for mobile ad hoc networks. But none of them had concerned about environment's influence on the behavior of nodes yet. For example, consider that the network is formed by several parts. Each part has different environment (it's easy for nodes to communicate with each other when they are in flat areas compare to hilly fields). If we use the same rule to all nodes, obviously it's unfair. Some nodes may be punished not because they misbehaved, but for the environmental reason that forces they have low trust value.

We propose a new method in order to compensate those nodes who are in the bad areas. This method can be applied to other protocols such as CONFIDANT[1] or CORE[2]. We take the general reputation protocol as an example to show how the environment can affect the nodes' behavior.

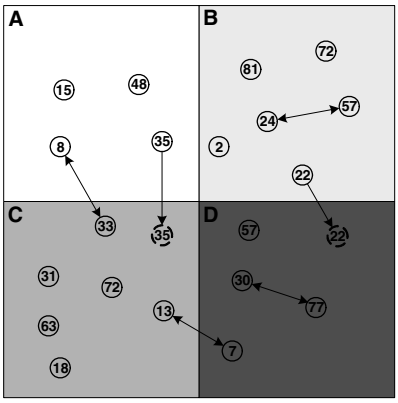


Fig. 1. The Network Model

In our scheme, the whole network would be divided into several parts depending on their environment. See Figure 1 as an example. The whole network is divided into four parts: A, B ,C and D. Within each part, environment is all the same(nodes have the same radio coverage or other parameters. The environment has the same influence on each node). Suppose the part A is the best environment for nodes to communicate with each other, the part B is the second, then the part C, the area of the part D is the worst amongst these four parts. So nodes in this part are hardest to communicate with each other. Take node No.8 and node No.33 for example, and assume that node No.33 is in the worse part. Then it may have a higher packet drop rate because of the environment's influence not because of its own wish. Then if node No.8 calculate node No.33's *direct trust* value, it may have a low *direct trust* value. Node No.33's *reputation* value may below the threshold and be considered as a misbehaving node(So as node No.13 etc.). So we must have a method to compensate the nodes that are in the "bad" part of the network.

4.1 Compensated Direct Trust(CDT)

The compensated direct value would be like this:

$$CDT(A,B) = \alpha_{A,B} \times \frac{\sharp forwarded}{\sharp sent} \tag{4}$$

α : the compensating factor to direct trust value

Situation 1. **Nodes in the same part**

Because node No.24 and No.57 are in the same part, they share the same environment. Then it's no need to compensate them. Thus, for them, the α value would be 1(The same for node No.30 and No.77).

Situation 2. **Nodes in the different parts**

Now, concern about node No.8 and No.33. They are in the different parts. Obviously 33 has a worse environment. Then we have to compensate it(The same for node No.13 and No.7). Also, when node No.35 and No.22 move to other part, their α value should be changed.

The α value would be like this:

$$\alpha_{A,B} = \frac{avg\ A}{avg\ B} \tag{5}$$

$avg\ A$: average reputation values of the nodes belonging to the part of A
$avg\ B$: average reputation values of the nodes belonging to the part of B

4.2 CIDT and CREP

Since the α value has already been added in the $DT(A, N_i)$ and $DT(N_i, B)$, it's no need to compensate the *indirect trust value* and the *reputation value* with α.

Then compensated CIDT and CREP are following:

$$CIDT(A, B) = \frac{\sum_{i=1}^{n} CDT(A, N_i) \times CDT(N_i, B)}{n} \tag{6}$$

and

$$CREP(A, B) = \omega \times CDT(A, B) + (1 - \omega) \times CIDT(A, B) \quad (0 < \omega < 1) \tag{7}$$

ω : the weight we put on CDT(A,B).

5 The Whole Scenario

The whole network is divided into several parts according to the environment. A node maintains a direct trust table which consists of entries for every neighbor and their direct trust value for performing a certain function. Nodes temporarily send *DT_update* message, which contains the source node 's direct trust value of other nodes and its own α value. On receiving this message, other nodes check sender's id and see if it is misbehaving or not. If the sender is dependable, nodes will accept the message and then update other nodes' indirect trust value (the most voted) and calculate other nodes' reputation value. The reputation value *rep*, is initially set to the variable *startdtv* (start direct trust value)

When a node requests a service from a neighbor, it gives the neighbor x opportunities to respond, where initially x is equal to *startdtv*. If the response is positive, x is increased by *cv* (change value). While x is positive, the value

of x should be returned to the initial starting value after a timeout period, and thus, the value has to be earned again. After a certain number of consecutive timeout periods where no negative behavior has occurred, the *rep* value should be increased by *cv*.

Where there is no response or the response is negative, x is decreased by $2cv$. The node should keep trying until x reaches zero, then the corresponding *direct trust value* is decreased by $2cv$. In this event, the node should look to request the service from a different node. If later on, the node wishes to try and request the service from the same neighbor again, it performs the same algorithm, where the *rep* value is less and thus the number x of opportunities is now less, i.e. the neighbor is given less chances. The node should perform exponential back off to allow the neighbor to recover from any temporary problems(i.e. suddenly lose power). Neighbor nodes should be given some chance of recovery. Thus, if a node has no other option but to try a selfish node, the node can just request the service with an initial x value of 1. This, along with a decreasing *direct trust value*, results in less resources being wasted on a neighbor who is selfish or failed. Also, to discourage unwanted behavior, service requests from nodes with reputation values below a threshold should be ignored.

5.1 Which Nodes Are Misbehaving?

First, we need to observe that it is not possible for us to differentiate the different types of misbehavior. We cannot say if a node is misbehaving because he is malicious, just selfish, has no battery left and so on. We - in the following - just try to somehow determine which nodes are misbehaving without too many false positives.

In 4.2, we calculated trust values. But how to use them? When do we trust a node for routing packets?

The idea is to exclude misbehaving nodes from the network. Nobody wants to send his packets via a misbehaving node where one can not be sure if it reaches its destination (unchanged), but when nobody sends packets via misbehaving nodes they are relieved from the burden of forwarding packets, and therefore rewarded for their misbehavior. Many proposed protocols work like this, but we do not want to encourage misbehavior. We want to enforce cooperation. This can be achieved by dropping packets of misbehaving nodes by the other nodes (instead of forwarded). In this way, misbehaving nodes are completely excluded from the network. Because we want to give misbehaving nodes a chance of changing their behavior, we will route some of our packets through them (so that we can monitor their behavior), but we will not forward packets for them.

How do we determine if a node is misbehaving? A trust value can be small if a node dropped packets, but also if they never reached him or if we have not seen the correct forwarding. For the forwarding of packets it does not matter why a node has a small trust value. We, therefore, choose nodes with high trust values to maximize the probability of reaching the destination. In the other case we want to drop packets of misbehaving nodes only.

All this can not be achieved at 100%, but the errors should be minimized. So we need some thresholds. However, we use α value to compensate nodes in bad environment, so the whole network can use the same threshold. Where all nodes with $CREP < \tau$ will be treated as misbehaving.

5.2 The Bottleneck Problem

We use reputation system in order to find a relatively stable route to the destination. However, if a node has a high reputation and all the nodes want to send their packets through it, the congestion would happen, and it would be the bottleneck of the network. The route is "safe" but may not be efficient at all.

We use the following rule to select node:

- PDR_x : node x's packet drop rate
- $avg(PDR_x)$: average packet drop rate of the part the node x belongs to
- REP_x : node x's reputation value
- $avg(REP_x)$: average reputation value of the part the node x belongs to

If $\frac{PDR_x}{avg(PDR_x)} > \frac{REP_x}{avg(REP_x)}$, we shouldn't give too much bandwidth to this node. Else $\frac{PDR_x}{avg(PDR_x)} \leq \frac{REP_x}{avg(REP_x)}$, we can give more bandwidth to this node

6 Performance Analysis

To evaluate our protocol, we run NS-2 simulations with our implementation[10].

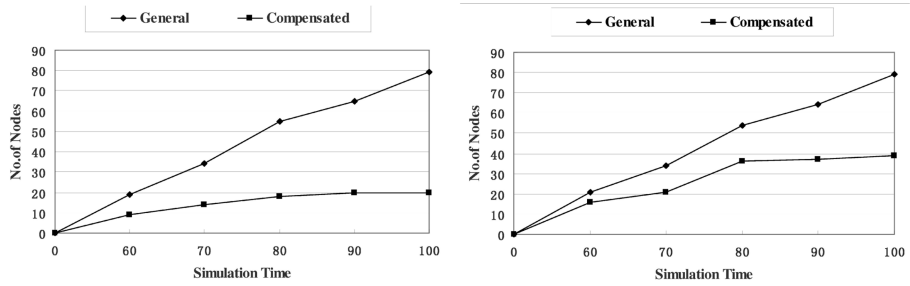
6.1 Definitions

- Original DSR : The original DSR protocol without reputation systems
- General : The DSR protocol with the general reputation scheme
- Compensated : The DSR protocol with the reputation compensation protocol
- Goodput : The ratio of received to sent packets
- Overhead : The ratio of number of reputation messages to routing messages

We simulated our protocol with the following parameters: Area 1000m $\times$ 1000m,Placement is uniform, Application is CBR, The number of nodes is 100, Maximal speed is 50 m/s, Packet size is 128 B, Pause time is 0,Percentage of Selfish nodes is 20%, Weight $\omega = 0.5$, and finally, Threshold τ is 0.4.

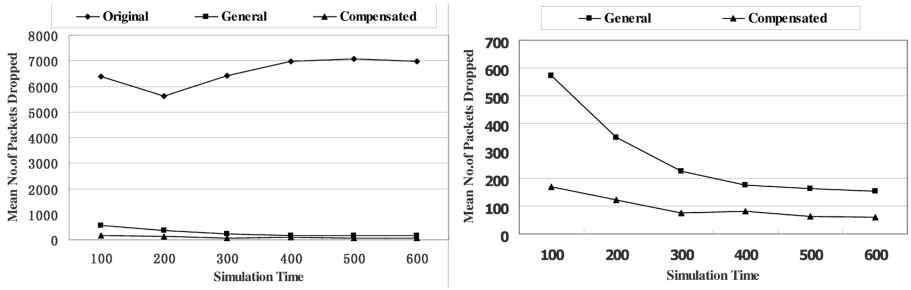
6.2 Simulation Results

Figure 2 shows the number of nodes are convinced to be misbehaving node varying the simulation time. We set 20 selfish node in Figure 2(a) and 40 in Figure 2(b). It's obvious the reputation compensation scheme is better than the general scheme. The reputation compensation scheme can catch out every selfish node without treating other good node unjustly. However, the general scheme consider almost 80% of nodes are selfish node because it has no compensation



(a) 100 nodes, 20 are selfish (b) 100 nodes, 40 are selfish

Fig. 2. No.of nodes are convinced to be selfish versus time



(a) 100 nodes, 20 are selfish (b) 100 nodes, 20 are selfish

Fig. 3. Mean No.of packets dropped versus time

to nodes in the bad parts. So when they communication with other nodes and would be convinced to be bad.

Figure 3 shows mean number of packets dropped varying the simulation time. In the original DSR protocol, there are about 7000 packets dropped due to the selfish node. And both of the general and the reputation compensation scheme have a far better result than the original scheme. They just dropped a few packets because they can detect selfish nodes effectively. Then we take the general and the reputation compensation part out of the Figure 3(a). As showing in Figure 3(b), the reputation compensation scheme has a better performance than the general scheme because fewer nodes are convinced to be selfish.

Figure 4 shows mean number of packets dropped versus the percentage of selfish nodes. We can see that in the original DSR, even a small percentage of selfish nodes can work havoc. There is not much difference in the number of intentionally dropped packets as the percentage of selfish nodes increases. This can be explained by the fact that it does not matter where on the path a packet is lost. Our scheme still keeps the number of deliberately dropped packets low even in a very hostile environment as given by more than half the population acting selfishly - given that there are enough nodes to provide harmless alternate partial paths around selfish nodes.

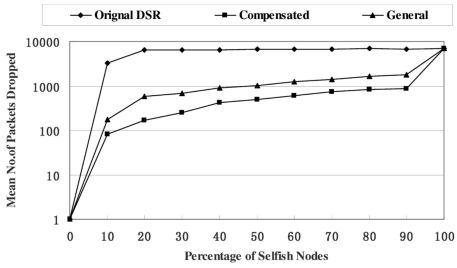
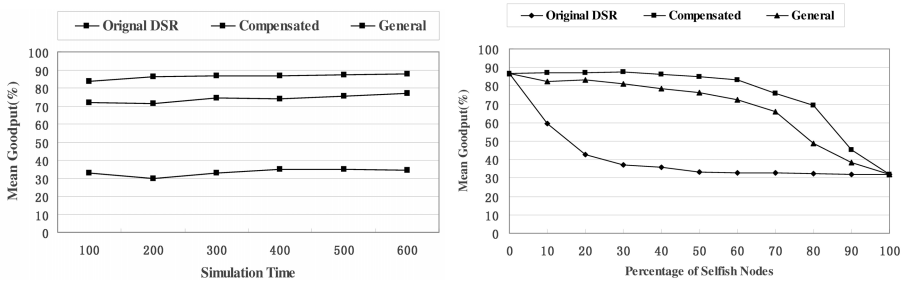


Fig. 4. Mean No.of packets dropped versus percentage of selfish nodes, 100 nodes, 20 are selfish



(a) 100 nodes, 20 are selfish (b) 100 nodes

Fig. 5. Mean Goodput versus time and percentage of selfish nodes

Figure 5(a) shows mean goodput varying the simulation time. The original DSR has a very bad performance, and the mean goodput is between 30% to 40%. The general protocol has a better performance, and the mean goodput is between 70% to 80%. Then the reputation compensation protocol has the best performance at the end of the simulation, and it almost reaches 90%.

Figure 5(b) shows mean goodput versus the percentage of selfish nodes. Obviously, our scheme has a better performance. The goodput of the original DSR

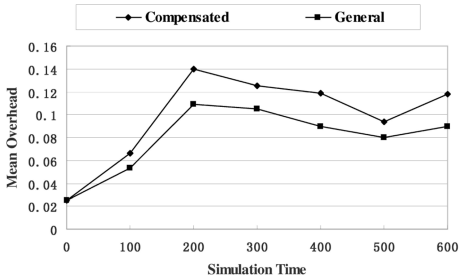


Fig. 6. Mean Overhead, 100 nodes, 20 are selfish

decreases sharply from the beginning and then decreases steadily. But our scheme keeps steadily at the beginning even half of nodes are selfish.

Figure 6 shows mean overhead varying the simulation time. Always when adding a new protocol, the overhead caused should not be too large. Our protocol adds less than 15% overhead but gain more than 50% in mean goodput, so that our protocol is worth to be added.

7 Conclusion

This paper tried to show how to incorporate reputation, trust and selfishness into the cooperative protocol of ad hoc networking. Its significance also lies in not only suggesting the reputation model, but also showing that its performance is promising.

The paper also proposed the General Reputation Model for mitigating detrimental effect of selfish nodes. To the model, we added the environmental influence attribute on nodes behavior and showed how it worked. The simulation by DSR proved our reputation- based trust management significantly improved the performance with a small amount of overhead increment. Goodput in a setup with 20% selfish nodes can be improved more than 50%, causing less than 15% overhead.

References

1. Buchegger, S., Le Boudec, J.-Y.: Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes: Fairnes. In Dynamic Ad-hoc NeTworks). In: Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC), Lausanne, CH, (June 2002)
2. Michiardi, P., Molva, R.: Core: A Collaborative Reputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks. In: Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security, pp. 107–121, (September 26-27) (2002)
3. Buchegger, S., Boudec, J.-Y.L.: Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In: Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands, pp. 403–410. IEEE Computer Society, Los Alamitos (2002)
4. Pirzada, A. A., McDonald, C.: Establishing trust in pure ad-hoc networks. ACM International Conference Proceeding Series; Vol. 56. In: Proceedings of the 27th conference on Australasian computer science - Volume 26
5. Dewan, P., Dasgupta, P., Bhattacharya, A.: On Using Reputations in Ad hoc Networks to Counter Malicious Nodes. QoS and Dynamic Systems in conjunction with IEEE ICPADS Newport Beach, USA (2004)
6. Marti, S., Giuli, T.J., Lai, K., Baker, M.: Mitigating Routing Misbehaviour in Mobile Ad Hoc Networks. In: Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking MobiCom (2000)
7. IETF MANET Working Group Internet Drafts. `http://www.ietf.org/ids.by.wg/manet.html`

8. Broch, J., Johnson, D.B., Maltz, D.A.: The dynamic source routing protocol for mobile ad hoc networks. Internet-Draft Version 03, IETF, (October 1999)
9. Zhou, L., Haas, Z.J.: Securing ad hoc networks. IEEE Network Magazine, vol. 13(6)(November/ December 1999)
10. The Network Simulator - ns-2 (2002), http://www.isi.edu/nsnam/ns/
11. The CMU Monarch Project. The CMU Monarch Projects Wireless and Mobility Extensions. (October 12, 1999) http://www.monarch.cs.rice.edu/cmu-ns.html
12. Resnick, P., Zeckhauser, R.: Trust among strangers in internet transactions: Empirical analysis of ebays reputation system. In: Baye, M. (ed.) Advances in Applied Microeconomics: The Economics of the Internet and E-Commerce, vol. 11, pp. 127–C157. Elsevier Science Ltd, Amsterdam (November 2002)
13. Mui, L., Mohtashemi, M., Halberstadt, A.: Notions of reputation in multi-agents systems: a review. In: Gini, M., Ishida, T., Castelfranchi, C., Johnson, W. (eds.) Proceedings of the first international joint conference on Autonomous agents and multiagent systems, Bologna, Italy, July 15-C19, 2002, ACM Press, New York (2002)
14. Resnick, P., Zeckhauser, R., Friedman, E., Kuwabara, K.: Reputation systems. Communications of the ACM, 43(12), 45–C48 (2000)