Journal of Computer Science 7 (8): 1157-1166, 2011
ISSN 1549-3636

Acknowledgment based Reputation Mechanism to Mitigate the Node Misbehavior in Mobile Ad Hoc Networks

K. Gopalakrishnan and V. Rhymend Uthariaraj
Ramanujan Computing Centre, College of Engineering Guindy,
Anna University, Chennai-600025, Tamil Nadu, India

Abstract: Problem statement: The cooperation between the nodes is essential to discover and maintain routes in mobile ad hoc network. Due to the presence of misbehaving nodes, the node cooperation is not always guaranteed as the nodes agreed during the route discovery phase. This phenomenon results in frequent network partitioning and it makes the routing process difficult. It also degrades the overall network performance and increases the control overhead due to frequent route discovery. **Approach:** This study proposed an Acknowledgment based Reputation Mechanism (ARM) to detected and isolated the misbehaving links in mobile ad hoc networks. The proposed system introduced a novel dynamic acknowledgment ratio to make the nodes to request its second hop successor in the source route to acknowledge the receipt of the packet. This makes the nodes to detect the behavior of the next hop links and to select the trusted routes for its transmissions. **Results:** The measured parameters such as packet drop ratio, malicious drop, false detection and send buffer drop are reduced greatly. The results also compared with the existing scheme and with original routing protocol performance. **Conclusion/Recommendations:** The proposed scheme performs better in detecting and isolating the misbehaving links. The overall network packet drop was decreased which in turn increase the overall network throughput. This shows that the proposed acknowledgment scheme discovers the trusted routes with the presence of misbehaving nodes.

Keywords: Routing security, node misbehavior, mobile ad hoc network, reputation mechanism, dynamic source routing, computational resources, misbehaving link list, routing protocol

INTRODUCTION

Mobile Ad hoc Networks (MANETs) is an autonomous collection of mobile devices which self organize to create a network by exploiting their wireless interfaces without a requirement for an existing infrastructure or any centralized administration. The routing protocol plays a vital role in establishing route between the mobile nodes and maintenance of the routes in these networks. All the nodes in an ad hoc network have to work mutually for executing the basic networking functions such as route discovery, route maintenance and multi-hop forwarding of packets. So the network performance becomes highly dependent on collaboration of all the participating nodes. The mobile ad hoc network has a wide range of applications in diverse fields ranging from low power military wireless sensor networks to large scale civilian applications, emergency search and rescue operations (Conti and Giordano, 2007; Khalid *et al*., 2009).

The network keeps on functioning when each node in the network executes the functions of a routing protocol in a proper manner. Misbehaving nodes comes into existence in a network due to scarcely available resources of mobile nodes such as battery power and computational resources (Marti *et al*., 2000). So the mobile user does not rely on the presumption that every terminal relay packets mutually, because the terminals are in the hands of users. If certain users tamper with their nodes to make them behave selfishly and they will not relay packets on behalf of other nodes while they send or receive their own packets. They will deprive the network of resources such as others battery power and bandwidth. The network service might eventually become unavailable for the users and makes the fairness of the network will be in danger.

The mobile ad hoc network lacks a centralized monitoring and control point, making it a challenging task to detect such misbehaving nodes effectively. Non-cooperative actions are usually termed as selfishness, which is notably different from malicious behavior. Selfish nodes use the network for their own communication but simply refuse to cooperate in forwarding packets on behalf of other nodes in order to

Corresponding Author: K. Gopalakrishnan, Ramanujan Computing Centre, College of Engineering Guindy, Anna University, Chennai-600025, Tamil Nadu, India Tel: +914422358024 Fax: +914422201160

save its battery power and computational resources. They have no intention of damaging the network. In other hand, the malicious nodes injected by adversaries will actively spend battery power to cause harm to the entire network (Deng *et al*., 2002; Jun and Hua, 2010).

Node misbehavior problem have been studied by many researchers and proposed various techniques to prevent it. The schemes (Buttyan and Hubaux, 2003; Zhong *et al*., 2003) provides virtual currency or nuggets to the nodes in order to perform the networking operations. Nodes get paid for forwarding packets on behalf of other nodes. When they request other nodes to forward their packets, they use the same payment system to pay for such services. The main problem with virtual currency based schemes is that they need a tamper proof hardware or a similar kind of mechanism for managing the payment system.

The schemes (Marti *et al*., 2000; Buchegger and Le Boudec, 2002) are based on next hop monitoring, in which the nodes except the destination and its previous hop in the source route of the packet have to monitor the behavior of its next hop in order to identify the node misbehavior but the monitoring method employed by these schemes have the same disadvantages as mentioned in (Marti *et al*., 2000). Whereas the schemes (Li and Lee, 2006; Gopalakrishnan and Rhymend Uthariaraj, 2011) employs neighborhood monitoring approach, which adds flexibility in monitoring by allowing a node to monitor the neighboring transmissions even if those transmissions does not involves it. All these overhearing based schemes uses a threshold based reputation mechanism to detect and isolate the misbehaving nodes. The main drawback of these overhearing schemes is that the nodes always promiscuously listens the channel in order to overhear the packet and to identify the node misbehavior. The schemes (Balakrishnan *et al*., 2005; Liu *et al*., 2007) detect the node misbehavior based on the acknowledgment sent by a node in response to receiving a packet. These schemes do not consider the identity spoofing and packet modification behavior. Moreover (Balakrishnan *et al*., 2005) suffers from control overhead due to acknowledge of every received packet and the modified version (Liu *et al*., 2007) also uses a static ratio of acknowledgment.

Almost all the related works discussed above considered only packet droppers but where as in the case of proposed system the packet modification and identity spoofing are also considered. In addition, the proposed scheme reintroduces the misbehaving links after a timeout period and disseminates the misbehaving links by piggybacking it in the route discovery packet without incurring additional control overhead. The proposed scheme introduces a dynamic acknowledgment ratio which guarantees the fast detection of misbehaving links rather than the static ratio approach (Liu *et al*., 2007).

MATERIALS AND METHODS

This study assumes bidirectional communication symmetry on every link between the nodes. This assumption is often valid because many wireless MAC layer protocols including IEEE 802.11 require bidirectional communication for reliable transmission.

Notations: The following notations are used in this study:

- N_{AckReq}: Number of acknowledgment requested from the second hop successor
- N_{PktSnd}: Total number of packets sent or forwarded
- R_{Ack}: Acknowledgment ratio decides whether to request an acknowledgment from the second hop successor or not
- α: Increment ratio of trust value of the monitored link upon successful reception of an acknowledgment
- β: Decrement ratio of trust value of the monitored link upon not receiving an acknowledgment
- γ: Decrement ratio of trust value of the monitored link upon identifying the packet modification
- R_{Part}: Ratio of the fraction of packets to be acknowledged
- R_{Full}: Ratio of acknowledgment for every packet

Acknowledgment scheme: In the proposed system, all nodes in the source route except the destination and its previous hop have to request an explicit acknowledgment termed as TwoHopAck for the packets expect the RREQ from its second hop successor based on the acknowledgment ratio R_{Ack}. Similarly, the nodes except the source and its immediate successor have to send a TwoHopAck packet based on the acknowledgment request received from its second hop predecessor. The check sum of the received packet is also piggybacked along with the acknowledgment; up on receiving the acknowledgment the second hop predecessor compares the stored checksum with the received one in order to identify the packet modification behavior. The packet header is modified in order to accommodate the Acknowledgment Request (AckReq) field. As shown in Fig. 1, the source node S communicates with destination node D via the intermediate nodes I1→I2→I3.

In this scenario, the nodes except I3 and destination D have to request a TwoHopAck from its second hop successor in the source route by including its identity in the AckReq field. The nodes I2, I3 and D act as a TwoHopAck sender and S, I1 and I2 act as a TwoHopAck receiver. The data structure maintained by the TwoHopAck requester is shown in Fig. 2. A dynamic acknowledgment ratio R_{Ack} is used to detect the misbehaving links in a timely manner and it also reduces the control overhead occurred due to TwoHopAck.

Each node maintains a One Hop Connectivity List (OHCL) which contains the neighboring node information (IP/MAC address) based on the packets overheard / received. Whenever a node receives a control packet, it checks the transmitting node information of the received packet with the stored information in OHCL in order to detect the identity spoofing. The Algorithm 1 shows the procedure executed by each node for identifying the IP/MAC spoofing.

Algorithm 1: Identity spoofing detection:

```
if (Received a Control Packet) then
    Search the OHCL for identifying the sender
     identity
     if (found) then
        Check the IP and MAC address of the packet
        with the stored one
        if (match) then
            Process the Packet
       else
           Drop the packet
       end if
    else
        Create an entry in OHCL and store the Node
        Information
        Process the packet
    end if
end if
```

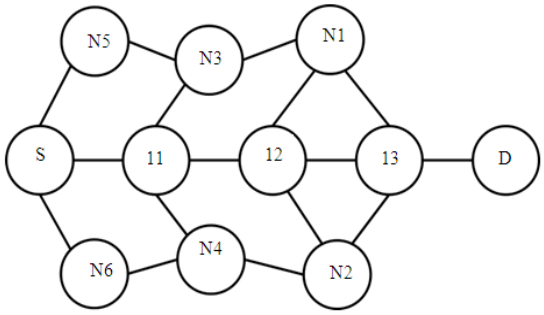

Fig. 1: Scenario of two hopack

Components of acknowledgment scheme: This scheme consists of three main components namely: a monitor responsible for monitoring the behavior of next hop link, trust manager for maintaining the trust value of next hop link and a path manager to maintain the route without containing misbehaving link in it as shown in Fig. 3.

These components are added as an add-on into the Dynamic Source Routing (DSR) (Johnson *et al*., 2007) protocol which enables each node to execute these functions along with its usual routing functionality. The monitor component is responsible for registering the sent packet id along with its checksum and it has a set of detectors which are used to identify the misbehaving link based on TwoHopAck. When a TwoHopAck is not received by the monitoring node within an acknowledgment timeout period or if the received checksum is not matched with the stored one then a negative event is registered by the monitor. A positive event is registered by the monitor only when the received checksum carried by the TwoHopAck matches with the stored one along with its packet id. The trust manager maintains the trust value of the monitored link based on the event reported by the monitor. When a positive event is reported by the monitor then the trust value of the monitored link will be incremented by α. If a negative event is received by the trust manger then the trust value of the monitored link will be decremented by β for missing acknowledgment and by γ for checksum mismatch. The packet modification misbehavior is a serious threat to the routing process so the monitored link will be punished as twice as packet dropping. When the trust value of the monitored link reaches the Negative Threshold limit then the link will be added into the Misbehaving Link List (MLL) and it is communicated to the path manager in order to prune the routes which have the misbehaving link in it. Once the monitored link is added into the MLL then all the traffic to and from the misbehaving link will be rejected.

A second chance timer is initiated for the misbehaving link and an explicit route error packet is sent to the packet originator to inform about the misbehaving link. The data structure of the explicit route error packet is shown in Fig. 4.

Once the second chance timer expires then the misbehaving link is removed from the MLL by reducing its trust value by half. The reason for not resetting the trust value of reintroduced link is that the link might continue to misbehave. If it continues to misbehave then it will be detected quickly. The MLL is disseminated using a Route Request (RREQ) packet so that the misbehaving link information is widely

spread over the network as well as it does not incur extra control overhead for disseminating the MLL. A variable length list is added into RREQ packet in order to accommodate the MLL. When a node receives a RREQ packet, it extracts the MLL and stores it into its Misbehaving Link Table (MLT). If the MLL is already received from the same link then the existing list will be replaced by the new one else a new entry will be created for that link in the MLT. If the received node is not a destination or an intermediate node that has a route to the destination then it will merge its own MLL into the MLL in the RREQ packet and then rebroadcast it. The data structure of route request packet after the inclusion of MLL is shown in Fig. 5.

The MLL stored in the MLT are checked against the source route of the control packet, whenever a node receives a control packet destined to it. If it matches then the packet is dropped else it is accepted. Whenever a monitored link is added into the MLL then its corresponding MLT entry will be deleted. The procedure executed by TwoHopAck sender and receiver is shown in Algorithm 2 and 3 respectively.

Algorithm 2: TwoHopAck sender:

```
Publish h_n
i←n
while true do
  if (Received packet is not a RREQ) then
      Search second hop predecessor in AckReq List
       if (Present) then
          Prepare MAC with h_i-1
          Prepare TwoHopAck with ID, Checksum,
          MAC and h_i
          Send TwoHopAck
          Remove the second hop predecessor from
         AckReq List
         i --
      end if
  end if
end while
```

Algorithm 3: TwoHopAck receiver

```
While true do
   if (Received h_n from the TwoHopAck sender) then
       Record h_n, i←n
   end if
end while
N_PktSnd←0, N_AckReq←1, R_Ack←1, R_Part←0.2, R_Full←1
while true do
 if (Packet is ready to send) then
   if (Packet is not a RREQ) then
     N_PktSnd ++
    if (N_AckReq / N_PktSnd < R_Ack) then
       Compute the Checksum
       Add its own id into AckReq list and append
       it into the packet header
       N_AckReq ++
       Transmit Packet
       LIST←LIST ∪ (Packet_ID, Checksum)
       Setup timer for Packet_ID
    else
       Transmit Packet
    end if
  end if
end if
if (TwoHopAck packet received) then
   Search for the Packet_ID carried by TwoHopAck
   in LIST
   if (found) then
    Check the checksum carried by TwoHopAck is
    equal to stored checksum
    if (match) then
     Check validity of h_i
     LIST←LIST - (Packet_ID, Checksum)
     Clear timer for Packet_ID
     TrustValue_Link ++
     if (R_Ack=R_Full) then
       R_Ack←R_Part
     end if
    else
     TrustValue_Link --
     LIST←LIST - (Packet_ID, Checksum)
     if (TrustValue_Link <=Negative_Threshold) then
       Send Misbehavior Report to Packet
       Originator
       MisbehavingLinkList←MisbehavingLinkList
       ∪ Link
       Setup the second chance timer for the Link
     end if
     if (R_Ack!=R_Full) then
        R_Ack←R_Full
    end if
    end if
  end if
end if
if (TwoHopAck Timeout Event Occurred) then
    LIST←LIST − (Packet_ID, Checksum)
    TrustValue_Link --
    if (TrustValue_Link <= Negative_Threshold) then
```

```
            Send Misbehavior Report to Packet Originator
            MisbehavingLinkList←MisbehavingLinkList
            ∪ Link
            Setup the second chance timer for the Link
        end if
        if (R_Ack!=R_Full) then
            R_Ack←R_Full
        end if
    end if
    if (Second Chance Timer Expires) then
        MisbehavingLinkList←MisbehavingLinkList−Link
        TrustValue_Link←TrustValue_Link / 2
    end if
end while
```

Authenticating the two HopAck packet: Since, the TwoHopAck packets are forwarded by an intermediate node without proper protection, a misbehaving intermediate node can fabricate TwoHopAck packets and claim that they were sent by TwoHopAck originator. Therefore, an authentication technique is needed in order to protect TwoHopAck packets from being forged. A simple way to stop intermediate nodes from forging the TwoHopAck packets is to use the digital signature algorithm. A digital signature is a small number of extra bits of information attached by TwoHopAck originator. The signature is unique and usually computationally impossible to forge unless the security key of TwoHopAck originator is disclosed. Further, the signature may be used to assure the integrity of the transmitted data. An asymmetric cryptography technique such as RSA (Johnson, 1999) is used to implement the digital signature. However, such asymmetric operations are too expensive for the mobile nodes in ad hoc networks which are usually resource constrained. Hu *et al*. (2003), an efficient algorithm termed one-way hash chain (Lamport, 1981) was used to guard against security attacks such as DoS and resource consumption attacks in the Destination Sequenced Distance Vector (DSDV) routing protocol (Perkins and Bhagwat, 1994).

Next hop receiver	Second hop receiver	N_{PktSnd} packet transmitted	N_{AckReq} acknowledgment requested	list list of packet IDs and their checksum	Trust $value_{link}$

Fig. 2: Data Structure maintained by twohopack requester

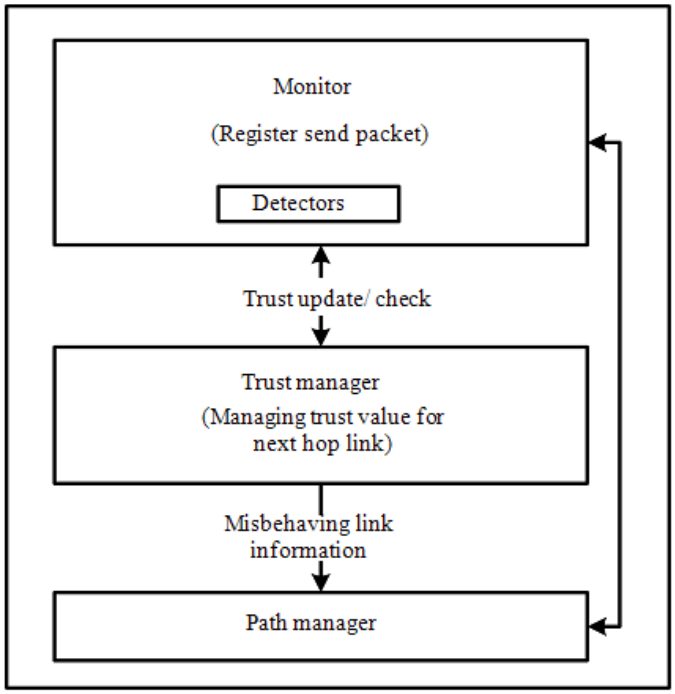

Fig. 3: Components of acknowledgment scheme

Option type	Opt data len	Error type (report misbehavior)	Reserved	Salvage	Error source address (misbehavior report sender)	Error destination address (misbehavior report receiver)	Type-specific information (misbehaving link)

Fig. 4: Data structure of explicit route error packet

Option type	Opt data len	identification	Target address	address (1)	address (2)	···	address (n)	Misbehaving link list (variage length)

Fig. 5: Data Structure of RREQ with misbehaving link list

Next hop receiver	Destination TwoHopAck receiver	ID sequence number	Checksum	MAC signature	h_i has release

Fig. 6: Packet Format of two HopAck

A one-way hash chain can be constructed based on a one-way hash function *H*. The hash function is a transformation that takes a variable-length input and returns a fixed-length bit string, that is, $H:\{0,1\}^{*}\rightarrow\{0,1\}^{p}$, where P is the length in bits, of the output of hash function. An ideal hash function H should have the following properties:

- The input can be of any length
- The output has a fixed length
- H(x) is relatively easy to compute for any given input x
- It is computationally infeasible to calculate x from H(x)
- H(x) is collision free

To create a one-way hash chain, a node picks up a random initial value $x\in\{0,1\}^{\rho}$ and computes its hash value. The first number in the hash chain h_0 is initialized to x. By using the general formula $h_i = H(h_{i-1})$ for $0<i\leq n$, for some n, a chain of h_i is formed:

$$H_0, h_2, h_3,\ldots h_n \quad (1)$$

It can be proved that, given an existing authenticated element of a one-way hash chain, it is feasible to verify the other elements preceding it. For example, given an authenticated value of h_n, a node can authenticate h_{n-3} by computing H (H(H(h_{n-3}))) and comparing the result with h_n (Hu *et al*., 2003).

The proposed scheme uses the above one-way hash chain to protect the TwoHopAck packets against fabrication. In order to use the one-way hash chain in (1) to authenticate TwoHopAck packets, the TwoHopAck Originator must distribute the h_n, element to two Hop Ack Requester. A traditional approach for such information distribution is through a trusted certificate authority. However, in a MANETs, nodes move from one place to another and there is usually no central server or base station to act as a trusted certificate entity. The proposed system uses the transmission extension technique mechanism as mentioned in (Liu *et al*., 2007). Using this technique, the Originator increases the transmission power to send the h_n element directly to the Requester. This technique bypasses the intermediate forwarding node, the potential threat to the distribution of h_n. While such a technique consumes more energy from the Originator but it takes place rather infrequently. The distribution of a new h_n element is only needed when the entire chain has been used. Once the h_n element is distributed from Originator to Requester, the Originator can use h_i ($0\leq i<n$) sequentially to sign the TwoHopAck packets to be sent to Requester. The h_i elements will be disclosed by Originator one at a time. Let's assume that h_{i+1} has been disclosed initially, (I = n-1). When the Originator needs to send a TwoHopAck packet, it calculates a Message Authentication Code (MAC) based on h_i-1 and attaches the MAC and the h_i value to the TwoHopAck packet. The packet format of TwoHopAck packet is shown in Fig. 6.

Since h_{i+1} is known to Requester, it compares $H(h_i)$ with h_{i+1}. If the results match, the h_i element is accepted and recorded. The TwoHopAck packet must have been sent from the Originator. However, the integrity of the TwoHopAck packet can only be proved when the next TwoHopAck packet arrives (with h_{i-1}).

Table 1: Simulation parameters

Parameter	Value
Simulation area	1000 m * 1000 m
Simulation time	900 s
Number of nodes	50
Node mobility	5 m/s
Pause time	30 s
Transmission range	250 m
Antenna	Omni Directional
Maximum connections	15
Seed value	1-20
Traffic type	CBR (UDP)
Positive threshold	1
Negative threshold	-1
Initial trust value of node	0
α	0.025
β	0.05
γ	0.1
Acknowledgment timeout	150 ms

When h_{i-1} is disclosed to Originator, it can be used to verify the integrity of the TwoHopAck packet received last time by calculating the MAC and comparing it with the received one. This is the so-called "delayed disclosure" technique due to Hu *et al*. (2003). The overhead caused by the authentication of the TwoHopAck packets is not studied in this study but compared to traditional security measures; the computation cost of the one-way hash function is relatively low Hu *et al*. (2003). The communication overhead depends on the length of each element and the value of n (size of the one-way hash chain). When n and the size of each element are chosen reasonably then the overhead occurred due to the transmission of h_n will be low.

Simulation study: The proposed system is implemented in ns2.34 as an add-on to the DSR. The simulation utilizes Random Waypoint (RWP) mobility model to mimic the real world movement of the mobile nodes and evaluated the performance of the proposed system. The simulation parameters that are used in the simulation are shown in Table 1.

Modeling the misbehavior: The proposed system is simulated by introducing five different kinds of misbehavior as mentioned below:

- Misbehavior Type 1: These nodes participate in the DSR Route Discovery and Route Maintenance phases, but refuse to forward data packets on behalf of other nodes.
- Misbehavior Type 2: These nodes participate in neither the Route Discovery phase, nor forwarding data packets. They only use their energy for their own packet transmission.
- Misbehavior Type 3: These nodes behave differently based on their energy levels. When the energy lies between initial energy E_i and a threshold T_1, the node behaves properly. On the other hand, if the energy level lies between T_1 and another lower threshold T_2 then it behaves like a node of Misbehavior Type 1. Finally for an energy level lower than T_2, it behaves like a node of Misbehavior Type 2.
- Misbehavior Type 4: These nodes modify the packet forwarded on behalf of other nodes.
- Misbehavior Type 5: This type of nodes spoofs its identity in order to get back the network resources.

Performance metrics: The performance of the proposed system has been measured by using the following parameters:

- Packet loss ratio (%): The packet Loss Ratio is measured in terms of the ratio of data packets not delivered to the destinations to those generated by the Constant Bit Rate (CBR) sources
- Normalized routing load (packets): The number of routing packets including TwoHopAck control packets transmitted per data packet delivered at the destination
- False detection (%): The percentage of nodes detected falsely as a misbehaving node
- Average end to end delay (sec): This includes all possible delays caused by buffering during route discovery latency, queuing at the interface queue, retransmission delays at the MAC, propagation and transfer times
- Average energy dissipation (joules) - The average amount of network energy dissipated over the simulation period
- Malicious drop (packets): The total number of packets dropped by the misbehaving nodes
- Send buffer drop (packets): The number of data packets dropped in the send buffer of the packet originated node due to delay in finding the route to the destination

RESULTS

The simulation results of the proposed system were compared with DSR and the existing scheme PLRSA (Li and Lee, 2006). The packet loss ratio has been decreased by 18-50% and 7-18% when compared to DSR and PLRSA respectively as shown in Fig. 7. As shown in Fig. 8, the normalized routing load has been increased when compared to DSR and PLRSA. The false detection has been decreased from 38-58% when compared to PLRSA as shown in Fig. 9.

As shown in Fig. 10, the send buffer drop has been decreased by 77-86% and 77-84% when compared to DSR and PLRSA respectively. The malicious drop has been decreased by 75-81% and 27-54% when compared to DSR and PLRSA respectively as shown in Fig. 11. As shown in Fig. 12, the average end-end delay has been reduced when compared to DSR and PLRSA respectively. The average energy dissipation has been gradually increased when compared to DSR and PLRSA as shown in Fig. 13.

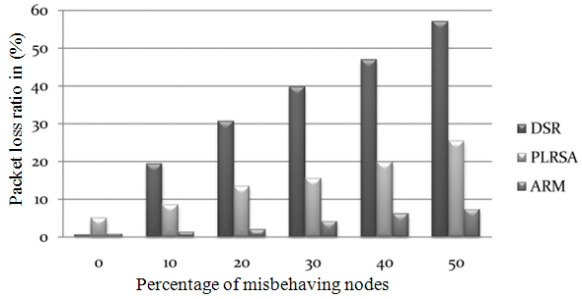

Fig. 7: Packet loss ratio in %

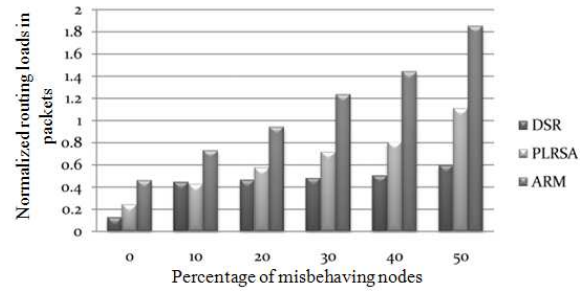

Fig. 8: Normalized routing load in packets

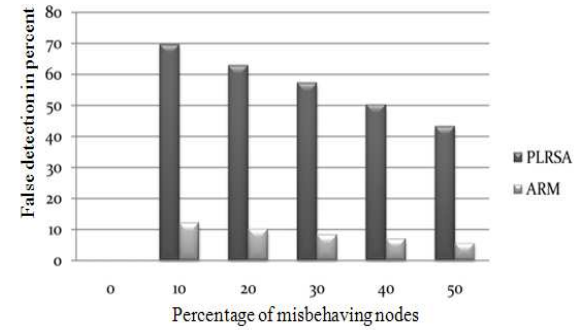

Fig. 9: False detection in %

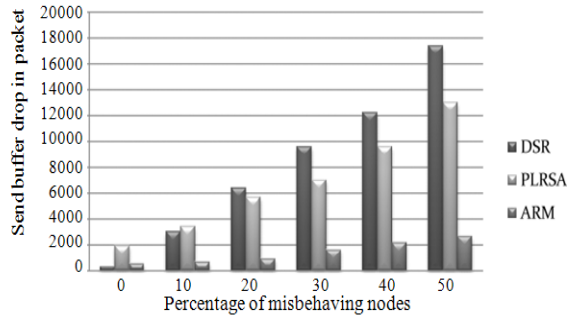

Fig. 10: Send buffer drop in packets

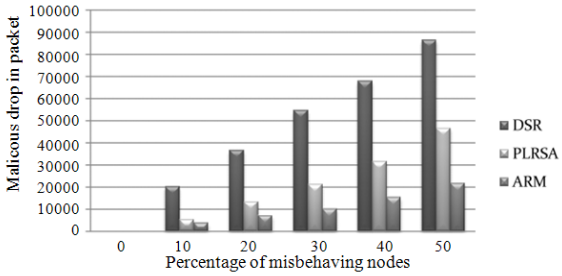

Fig. 11: Malicious drop in packets

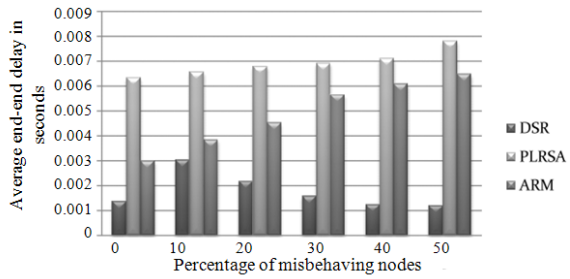

Fig. 12: Average end-end delay in seconds

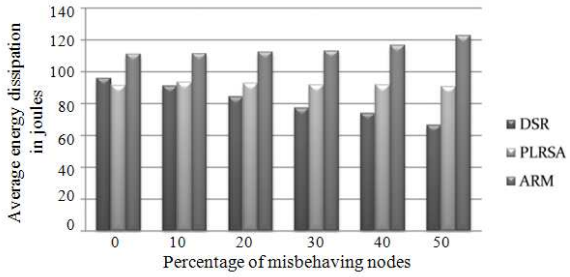

Fig. 13: Average energy dissipation in joules

DISCUSSION

The result shows that the proposed system has enough alternative routes to the destination even with the presence of misbehaving nodes. Since the average energy dissipation is directly proportional to overall network throughput and control packets spent, the energy dissipation is higher in the case of proposed system. This scheme is immune to overhearing technique drawbacks mentioned in (Buttyan and Hubaux, 2000) due to explicit TwoHopAck packet.

CONCLUSION

This scheme employs a dynamic acknowledgment mechanism based reputation system which detects and isolates the misbehaving links. The simulation result shows that the packet loss ratio, malicious drop, false detection and send buffer drop were greatly reduced. It shows the effectiveness of the proposed system in detecting, isolating the misbehaving links and finding out the alternative routes.

ACKNOWLEDGMENT

The first researcher would like to acknowledge the support received from CSIR-HRDG (Council of Scientific and Industrial Research-Human Resource Development Group), India, through Senior Research Fellowship.

REFERENCES

Balakrishnan, K., J. Deng and P.K. Varshney, 2005. TWOACK: Preventing selfishness in mobile ad hoc networks. Proceedings of the Networking Conference on Wireless Communication and Networking, Mar.13-17, IEEE Xplore Press, USA., pp: 2137-2142. DOI: 10.1109/WCNC.2005.1424848

Buchegger, S. and J.Y. Le Boudec, 2002. Performance Analysis of the CONFIDANT Protocol. Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing, (MAHNC'02), ACM, New York, pp: 226-236. DOI: 10.1145/513800.513828

Buttyan, L. and J.-P. Hubaux, 2003. Stimulating cooperation in self-organizing mobile ad hoc networks. J. Mobile Networks Appli., 8: 579-592, DOI: 10.1023/A:1025146013151

Conti, M. and S. Giordano, 2007. Multihop ad hoc networking: The reality. J. IEEE Commun. Mag., 45: 88-95. DOI: 10.1109/MCOM.2007.343617

Deng, H., W. Li and D.P. Agrawal, 2002. Routing security in wireless ad hoc networks. J. IEEE Commun. Mag., 40: 70-75. DOI: 10.1109/MCOM.2002.1039859

Gopalakrishnan, K. and V. Rhymend Uthariaraj, 2011. Collaborative Alert in a Reputation System to Alleviate Colluding Packet Droppers in Mobile Ad Hoc Networks. In: Advances in Networks and Communications, Meghanathan, N., B.K. Kaushik and D. Nagamalai, (Eds.). Springer, Heidelberg, ISBN : 3642178774, pp: 135-146.

Hu, Y-C., D.B. Johnson and A. Perrig, 2003. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. Ad Hoc Networks, 1: 175-192. DOI: 10.1016/S1570-8705(03)00019-2

Johnson, D.B., 1999. ECC, Future Resiliency and High Security Systems. Citeulike. http://www.citeulike.org/user/mgran/article/887959

Johnson, D.B., D.A. Maltz and J. Broch, 2007. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks. IETF Trust. http://www.ietf.org/rfc/rfc4728.txt

Jun, D. and L.W. Hua, 2010. A security routing optimization scheme for multi-hop wireless networks. Inform. Technol. J., 9: 506-511. DOI: 10.3923/itj.2010.506.511

Kaabneh, K., A. Halasa and H. Al-Bahadili, 2009. An effective location-based power conservation scheme for mobile ad hoc networks. Am. J. Applied Sci., 6: 1708-1713. DOI: 10.3844/ajassp.2009.1708. 1713

Lamport, L., 1981. Password Authentication with insecure communication. J. Commun. ACM, 24: 770-772. DOI:10.1145/358790.358797

Li, J-S. and C-T.Lee, 2006. Improve routing trust with promiscuous listening routing security algorithm in mobile ad hoc networks. J. Elsevier Comput. Commun., 29: 1121-1132. DOI: 10.1016/j.comcom.2005. 06.025

Liu, K., J. Deng, P.K. Varshney and K. Balakrishnan, 2007. An acknowledgment-based approach for the detection of routing misbehavior in Manets. J. IEEE Transa. Mobile Comput., 6: 536-550. DOI: 10.1109/TMC. 2007.1036

Marti, S., T.J. Giuli, K. Lai and M. Baker, 2000. Mitigating routing misbehavior in mobile ad hoc networks. Proceedings of the 6th annual international conference on Mobile computing and networking, (MobiCom '00), ACM, New York, pp: 255-265. DOI: 10.1145/345910.345955

Perkins, C.E. and P. Bhagwat, 1994. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. J. ACM SIGCOMM Comput. Commun. Rev., 24: 234-244, DOI: 10.1145/190809.190336

Zhong, S., J. Chen and Y.R. Yang, 2003. Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks. Proceeding of the INFOCOM, 22nd Annual Joint Conference of the IEEE Computer and Communication, Mar. 30-Apr. 3, IEEE Xplore Press, USA,, pp: 1987-1997. DOI: 10.1109/INFCOM.2003