Published in IET Information Security
Received on 7th July 2010
Revised on 25th May 2011
doi: 10.1049/iet-ifs.2010.0160

ISSN 1751-8709

Light-weight trust-based routing protocol for mobile *ad hoc* networks

N. Marchang[1] *R. Datta*[2]
[1]*Department of Computer Science and Engineering, North Eastern Regional Institute of Science and Technology Nirjuli, Itanagar 791109, Arunachal Pradesh, India*
[2]*Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, West Bengal, India*
E-mail: ningrinla@yahoo.co.in

Abstract: Mobile *ad hoc* networks (MANETs) were originally designed for a cooperative environment. To use them in hostile environments, trust-based routing can be used, where instead of establishing the shortest routes as done in traditional routing protocols, most trusted routes are established. In this study, the authors present a light-weight trust-based routing protocol. It is light-weight in the sense that the intrusion detection system (IDS) used for estimating the trust that one node has for another, consumes limited computational resource. Moreover, it uses only local information thereby ensuring scalability. Our light-weight IDS takes care of two kinds of attacks, namely, the blackhole attack and the grey hole attack. Whereas our proposed approach can be incorporated in any routing protocol, the authors have used AODV as the base routing protocol to evaluate our proposed approach and give a performance analysis.

1 Introduction

Most of the Mobile *ad hoc* networks (MANETs) secure routing protocols [1] in the literature use cryptographic techniques to secure the routing protocol. However, the downside of using cryptographic tools is that they are known to be computationally very expensive, which does not lend well to incorporating them in resource-constrained mobile devices. Hence, in the attempt to prevent some attacks, these protocols create new avenues for denial of service (DoS) attacks [2]. Besides, several such secure routing protocols presume the existence of a centralised or distributed trusted third party in the network [3].

Thus, it is becoming more acceptable to consider trust-based routing as a viable security solution, in which a trust-based scheme is used to protect the routing protocol. Every node in the network independently executes a trust model to estimate the trust it has on other nodes in network. This estimated trust value is used during routing decisions. Trust-based routing protocols attempt to establish most trusted routes rather than shortest routes as is done in traditional routing protocols.

The following are some schemes used for estimating trust in MANETs found in the literature. Several trust estimation schemes such as [4–11] use an intrusion detection system (IDS) that monitors neighbourhood traffic to estimate trust. However, they require buffering of packets and search for packets in the buffer for a match, which consumes considerable computational resources. Our approach also monitors the neighbourhood traffic but without needing to buffer packets or search for them, thus reducing this overhead considerably. Although these schemes address several other issues (attacks), which our proposed approach does not, we can still compare our work with them as we find them similar in some ways. Our approach addresses only packet-dropping attacks such as blackhole and greyhole [12]. Whereas the afore-mentioned schemes may be aimed at handling different issues, for example, ensuring dependable routing [7–9] or enforcing cooperation of nodes [4–6, 10, 11, 13], neighbourhood watch for dropping of packets is a common mechanism used in all these techniques. Hence, our technique can be combined with any other technique (e.g. [7–9, 14]) that uses neighbourhood watch for dropping of packets for a more efficient solution. Moreover, it can be used to implement an efficient mechanism for achieving cooperation among nodes as in [4–6, 10, 11, 13]. In most of them, misbehaviour of a node is confirmed on reaching of a threshold, which requires judicious setting. Our approach does not use threshold detection. More significantly, they require the buffering of packets (or signature of packets) and subsequent searching of the buffer for a match, which is a significant overhead. This overhead is avoided in our work.

Our approach can also be used independently to establish dependable routes in the presence of blackhole and greyhole attacks or to complement crypto-based secure routing protocols. These attacks are significant since they are considered lethal in terms of hampering availability of network service [15] and they impact on the network connectivity much severer than other attacks [16]. Moreover, solutions based on cryptography are ineffective against insider attacks such as blackhole or greyhole [17].

Hence, several researchers have worked on studying the effect of these attacks and mitigating them [15, 18–22].

Our work uses trust metrics to avoid the inclusion of misbehaving nodes while establishing routes. Some other works on trust management and trust-based routing are in [23–28]. While these consider different factors such as software configuration, hardware configuration, battery power [23], link quality [24], quality-of-service parameters [25] and behaviour of neighbours [26–28], our work considers only the packet forwarding behaviour of a neighbour in evaluating its trust level. The trust model we have used is similar to that of [27]. However, the techniques for evaluating the components of the trust model are different.

2 Trust model

In our trust model, every node maintains a value (which we call trust value) for each of its neighbours (nodes that are within its radio range). This value is a measure of the level of trust it has on its neighbour. For scalability, we have designed our trust model such that the trust value is calculated using only local information. Let $T_i(j)$ denote the level of trust of node i on neighbour j. The values of $T_i(j)$ range from 0 (denoting absolutely no trust) to 1 (denoting full trust) $0 \le T_i(j) \le 1$. We have taken $T_i(j)$ to be the weighted average of two components

$$T_i(j) = \alpha T_{i(\text{self})}(j) + \beta T_{i(\text{neighbour})}(j) \tag{1}$$

$T_{i(\text{self})}(j)$ represents the trust of node i on node j, based on node i's observation of node j's behaviour (e.g. by monitoring traffic of node j). $T_{i(\text{neighbour})}(j)$ represents the trust that neighbours of node i has on node j. These neighbours of node i are also neighbours of node j. $\alpha + \beta = 1$ and $0 \le \alpha, \beta \le 1$. Let $a_1, a_2, a_3, \ldots, a_n$ be the neighbours of node i (where n is the number of neighbours) such that they are also neighbours of node j. Then $T_{i(\text{neighbour})}(j)$ is given by

$$T_{i(\text{neighbour})}(j) = \frac{1}{n}\sum_{k=1}^{n} T_{a_k}(j) \tag{2}$$

By varying the values of α and β, we can thus vary the weight of self-trust as compared to neighbours' trust in evaluating the overall trust. It is clear that $T_{i(\text{neighbour})}(j)$ is the average of the existing trusts of the neighbours. Thus, in our node trust model, the past history is also taken into account. This is important when we want to evaluate trust based not only on present observations but also on past behaviour.

In our trust model, apart from the nodes maintaining trust values for their neighbours, the routes that are established also have trust values associated with them. This is needed so as to facilitate a node while establishing routes. It enables a node to decide whether a new route to a destination node that is showing up at the moment is better (has more trust value) than the existing one. The trust value of a node can be considered as a measure of the reliability of that node. A route is nothing but a sequence of nodes. Consequently, the reliability (trustworthiness) of a route depends on the reliability of all the nodes in the route. Let a route r consisting of l nodes be represented by a sequence $a_1, a_2, \ldots, a_l$ where a_i is the ith node in the sequence. Then, the trust value of route r, denoted by R_r is given as

$$R_r = T_{a_1}(a_2)T_{a_2}(a_3)\ldots T_{a_{l-2}}(a_{l-1}) = \prod_{i=1}^{l-2} T_{a_i}(a_{i+1}) \tag{3}$$

3 Trust estimation

In the trust model of the previous section, the trust values evaluated by watching the behaviour of the neighbours ($T_{i(\text{self})}(j)$ in (1)) form the basic blocks upon which the model is built. In this section, we present the technique that is used for estimating this node trust. For simplicity and also to minimise the overhead in a resource-constrained environment as that of a MANET, we have used only passive monitoring of forwarded data traffic to evaluate the behaviour of a node. Subsequently, this behaviour is translated to an estimate of the trust, which the monitoring node has on the monitored node.

For each neighbour of a node, we define three datastructures: (i) ToForward, (ii) Forwarded and (iii) Source list. The ToForward and Forwarded datastructures store the number of packets to be forwarded and the number of packets already forwarded, respectively. Each of them is divided into N slots or windows. Each window stores a positive integer value, which represents number of packets. The maximum number that each window can store is a fixed-size number, M. N and M are configurable parameters. The windows are accessed in a circular fashion. For these two datastructures, we also define an index CurrentWindow, which points to the window in both datastructures, which is currently being accessed. The initial value of the CurrentWindow can be set to any number between 0 and $N-1$. The use of only one index allows the windows in both datastructures to be accessed in a lock-step manner. Finally, we also define a list of sources (sourcelist) of packets to be forwarded, the significance of which is explained later.

Assume that node i is monitoring the traffic of its neighbour j. All packets that are sent out by node j are seen by node i assuming that a promiscuous mode of communication is in place. Thus, node i maintains the afore-mentioned data structures for node j. Node i watches for two kinds of actions of node j. First, node i watches for the packets that it has sent to node j to be forwarded further. Second, node i watches for packets, which are sent to node j by a neighbour of node i to be forwarded by node j.

Thus, in both cases, whenever node i finds that node j has received a packet to be forwarded further, it increments the ToForward count of node j by one. Similarly, whenever node i finds that node j has forwarded a packet it has to forward, it increases the Forwarded count by one. If any of the ToForward or Forwarded counts exceeds the limit M, then the content of the next window (i.e. (CurrentWindow + 1) mod N) is initialised to 0 and made the current window.

We now look at how node trust is calculated by observing the behaviour of a neighbour. The node trust $T_{i(\text{self})}(j)$ is calculated as

$$T_{i(\text{self})}(j) = \frac{\sum_{k=0}^{N-1} \text{Forwarded}(k)}{\sum_{k=0}^{N-1} \text{ToForwarded}(k)} \tag{4}$$

The ToForward and Forwarded datastructures used in (4) correspond to those that are kept by node i for node j.

Thus, $T_{i(\text{self})}(j)$ is the ratio of the number of packets forwarded to the number of packets to be forwarded. Every node uses the above formula periodically (after every 'NodeTrustUpdate' interval) to update the node trust of its neighbours. Note that in (4), only the counts of the packets to be forwarded and the packets forwarded are used for calculation of the trust value.

The compartmentalisation of the ToForward and Forwarded datastructures into windows and accessing them in a circular fashion enable our approach to capture the effect of the passage of time. Thus, a node, which behaves maliciously in the beginning may start behaving properly. Such nodes will be considered non-malicious after some time as the evidence of malicious behaviour in the datastructures gets overwritten with that of benign behaviour. On the other hand, a node that has been behaving well for a long period of time will not escape punishment if it starts behaving maliciously. How quickly one wants the trust evaluation scheme to respond to good or bad behaviour depends on the values of the configurable parameters – N and M. Large values of N and M would enable the behavioural changes of a neighbour to be reflected in its trust value very slowly, whereas small values would do so very quickly.

During the calculation of the trust value, the source nodes of the packets are also considered. This is required because not all packets that are forwarded by node j are accounted for in the ToForward count of node j (Fig. 1). When packet p_1 is forwarded to node C by node D, node A does not see it since node D is not its neighbour. Consequently, the ToForward count of node C is not updated on account of p_1. However, when node C subsequently forwards packet p_1 to node B, node A sees it but it would be an error to increase the Forwarded count of node C since packet p_1 is not accounted for in the ToForward count. To handle such a situation, the source nodes of the packets are also considered.

For every neighbour, the sources of packets that it has to forward are maintained in a list (we call it sourcelist). Thus, whenever node i sees a packet sent to node j to be forwarded further, it not only increases the ToForward count of node j but also records the source from which the packet originated in the source list. Later, when it sees a packet being forwarded by node j, it first checks whether the source of this packet is in the source list. If it is indeed in the source list, the Forwarded count of node j is incremented by one, otherwise not.

Since nodes are mobile, there may arise situations in which the above scheme fails. For instance, in Fig. 1, assume that node C is malicious and node S is a fast-moving mobile node. Suppose node B sends some packet p_2, originating at node S, to node C, which node C has to forward to node D. Also suppose node C conveniently drops p_2. Consequently, node A includes node S in the source list and updates the ToForward count of node C. Since node S is a fast-moving node, it finds itself now in the position as shown in Fig. 2. Now, suppose source S sends another packet p_3 to node C to be forwarded to node D. Now node C acts non-maliciously and this time forwards packet p_3 faithfully. When it does that, node A checks whether node S is in the source list. It indeed is there, and consequently it increases the Forwarded count of node C by one. Thus, node C may escape punishment for the packet p_2 that it has dropped. Moreover, it may happen that for subsequent such packets that originate from node S, the Forwarded count of node C is updated mistakenly, since node S is still in the source list, while it actually has already moved away. This can be taken care of by assigning some expiry time ('SourceExpireTime') to each node in the source list, such that after the expiry time, a node is no longer considered to be in the source list.

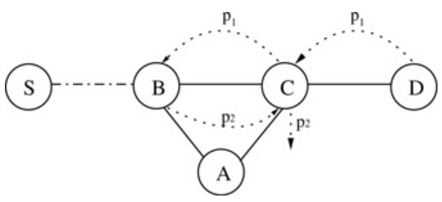

Fig. 1 *A overhears packet p_1 forwarded by C but not when D sent p_1 to C; A overhears packet p_2 sent by B to C, but C drops it; a solid edge between two nodes denotes they are neighbours; a dashed edge denotes there is a path between them.*

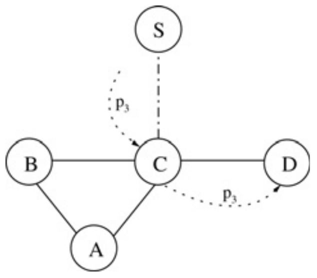

Fig. 2 *Node S moves its location*

4 Trust-based routing

In this section, we look at how the node trust can be used in routing. While our proposed technique can be incorporated in any routing protocol, we have chosen ad-hoc on-demand vector routing (AODV) [29]. In AODV, the shortest route is established during route establishment. We make modifications in AODV so that the most trusted route is chosen instead of the shortest route. When there is a tie, that is when two or more routes have the same trust value, the shortest path among them is chosen.

Equation (1) is used to calculate the trust value node i has for a neighbour j (i.e. $T_i(j)$). For our experimentation, we have taken two cases: (a) when $\alpha = 1$ and (b) when $\alpha < 1$.

4.1 Evaluation using only self-trust

First, we consider the case when $\alpha = 1$ (and $\beta = 0$). Thus, in this case, $T_i(j) = T_{i(\text{self})}(j)$. In other words, the trust that a node i has on another node j is solely dependent on the observations node i has on node j.

Implementation of the trust estimation scheme requires the following changes on the datastructures of AODV. To the existing fields in the original AODV neighbour table entry, the following fields are added: (i) ToForward, (ii) Forwarded, (iii) current window (C_Window), (iv) source list (Sc_List) and (v) neighbour trust (Nb_Trust). All the names of the fields are self-explanatory. The field neighbour trust denotes the trust on the neighbour. For the route table entry, only one field route trust (Rt_Trust) is added to the existing fields. The field route trust denotes the trust value of the route. Similarly, for the route request (RREQ) and the route reply (RREP) packets also, a field each denoting the trust value of the route represented by these control packets is added.

The route trust values are calculated as shown in (3). Route discovery is as in AODV except for the following changes. In AODV, when a node receives a route request, which it has already seen, it drops it. In our proposed technique (which we call light-weight trust-based (LTB)-AODV), when this situation arises, it finds the product of the route trust value in the RREQ packet and the node trust value of the neighbour from which it received the RREQ packet. In case the new route represented by this RREQ packet and the existing route have the same sequence number, the product is compared with the trust value of the existing route (available in the routing table). If the new route has a better trust value, the routing table is updated to this new route. In the case of the two routes having the same sequence number and the same trust value, the shorter route is selected. RREP packets are also handled in the same way. When an RREP packet is seen advertising a new route, the same decision rules are applied as is done for RREQ packets.

In LTB-AODV, only the destination node is allowed for generating an RREP packet, unlike in AODV in which an intermediate node can respond to an RREQ packet, by generating an RREP packet in case it has a fresh enough route to the destination.

4.2 Evaluation using both self-trust and neighbour trust

Next, we consider the case when $\alpha < 1$ (and $\alpha + \beta = 1$). Thus, the trust that a node i has on another node j is dependent on the observations node i has on node j and the trusts neighbours of node i has on node j. The first case(when $\alpha = 1$ and $\beta = 0$) is a simplified version of the second one, and it is relatively easier to implement since it is not required to communicate with the neighbours. However, implementation of the second case is more complicated as the trust values maintained by the neighbours are also are required for estimating the trust [see (1)].

The neighbour table entry of the previous subsection is augmented with two more fields: (i) Average of Trusts contributed by neighbours (Nb_Avg_Trust) and (ii) The number of neighbours who contributed the trusts (Nb_Count_Neighbor). One way of getting the trust values maintained by the neighbours is for every node to periodically broadcast to its one-hop neighbours a packet containing the trust values that it has for its neighbours. Thus, a new control packet is required, which we call TRUST packet. The first and second fields of the TRUST packet are used to send the type of the packet and the number of neighbours for which the sender node is sending the trust values, respectively. The remaining fields are used to send the addresses of the neighbours and the corresponding trust values the sender node has on the neighbours. Periodically, a node not only broadcasts a TRUST packet to its one-hop neighbours but also calculates the average trust contributed by its neighbours.

It may be noted here that when trust information from neighbours are also considered (i.e. when $\alpha < 1$ in (1)) during the node trust calculation, the overhead increases as compared to when it is not [i.e. when $\alpha = 1$ in (1)]. Apart from adding two more fields in the neighbour table entry, there is the added overhead of broadcasting an extra packet (TRUST) for sharing the trust information periodically. Accordingly, the performance is expected to increase. We clearly see a trade-off here between performance and overhead. Simulation results are shown in the next section.

5 Simulation and performance evaluation

Simulations were done using ns2.32, the underlying MAC protocol being 802.11. 15 nodes of 250 m radio range were randomly deployed in an area measuring $700\text{ m} \times 300\text{ m}$, which move using the random waypoint model with a pause time of 0 s. Every point in the graphs is an average of the results obtained from simulating 10 different topologies for a period of 9600 s. The traffic is of type CBR at 4 packets/s, the packet size being 64 bytes. To obtain a fair result, all possible connections are included in the traffic. They last for a part of the simulation time and are repeated over the simulation period. Another reason why all possible connections are taken into consideration is to allow convergence within the simulation period. Although the bit sending rate is less for each connection, the traffic load of the whole network is not. This is because the number of traffic connections is very high as all possible connections from a source to destination are taken. The N, M and 'SourceExpireTime' parameters are set to 20, 5 and 20 s, respectively. The 'NodeTrustUpdate' interval is set to 1 s.

We simulated both the LTB-AODV and the AODV protocols in the situation when some nodes are made to exhibit malicious behaviour by dropping all data packets (that they are supposed to forward) that come their way. We considered three cases when 3 (20%) nodes, 5 (33%) nodes and 7 (46%) nodes are malicious. In LTB-AODV, node trust values are updated after every 'NodeTrustUpdate' interval. Thus, after a period of time, the malicious nodes would have exhibited their malicious behaviour and their neighbours would have noted it and consequently updated their node trust values. Since the most trusted routes are established, these malicious nodes would be avoided during subsequent route establishments. If there were no alternate paths from a source to a destination except through a malicious node, the malicious node cannot be avoided. On the other hand, in AODV, since the shortest routes are always chosen, the malicious nodes would drop data packets if they happen to be in the shortest routes.

We have considered six performance measurement parameters:

1. Packet delivery ratio (PDR): Ratio of the total number of data packets received by the destinations to the total number of data packets sent.
2. Malicious packet drop ratio (MPDR): Ratio of the total number of data packets dropped by the malicious nodes to the total number of data packets sent.
3. End-to-end delay (EED): Average of the delay (received time minus sent time) of every data packet.
4. Route frequency (RF): The number of route requests generated per second.
5. Routing load (RL): The number of control packets transmitted (including the ones that are forwarded) for every data packet received.
6. Average throughput (AT): The total amount of data received by all nodes per second.

Fig. 3 shows the comparison of PDR of AODV and LTB-AODV. The label LTB-0.5-mal3 denotes the plot for LTB-AODV when $\alpha = 0.5$ (i.e. $\beta = 0.5$) and 3 out of the 15 nodes are malicious. We have chosen $\alpha = 0.5$ to analyse the situation when the estimated trust depends equally on self-trust and neighbour trust. Similarly, LTB-1-mal3 denotes the plot for LTB-AODV when $\alpha = 1$ and 3 out of

doi: 10.1049/iet-ifs.2010.0160

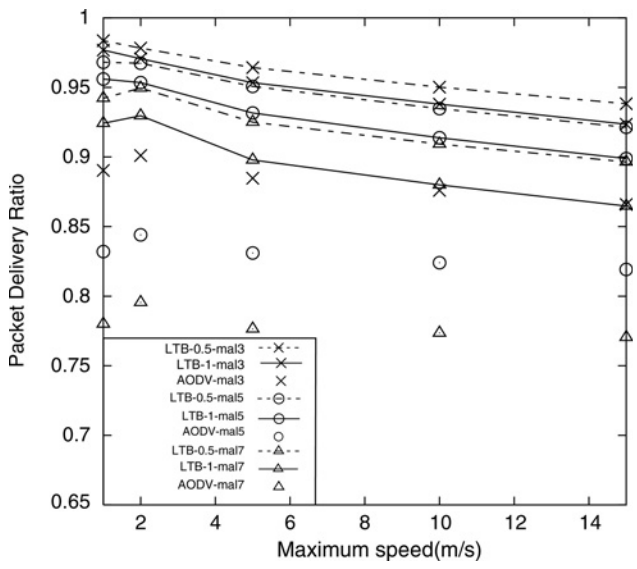

Fig. 3 *Packet delivery ratio against the maximum speed*

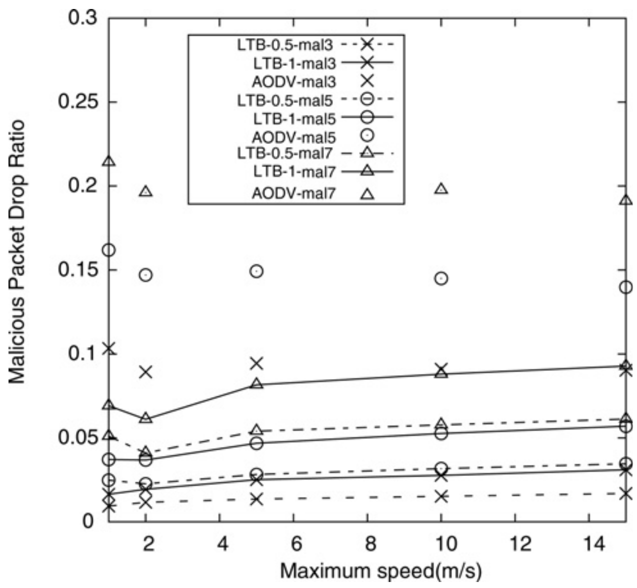

Fig. 4 *Malicious packet drop ratio against the maximum speed*

the 15 nodes are malicious. The label AODV-mal3 denotes the plot for AODV when 3 out of the 15 nodes are malicious. The other labels carry similar meaning. It is seen in all cases that there is a significant increase (about 10%) in the PDR of LTB-AODV as compared with that of AODV, even when about 46% (7 out of 15) of the nodes are malicious.

Moreover, it is seen that the PDR of LTB-AODV is more when the value of α is 0.5 than when it is 1, as expected. When $\alpha = 0.5$, the information supplied by the neighbours is also used during node trust calculation, which implies that information about the maliciousness of a neighbour which a node fails to accumulate purely by its own observations may be provided by the neighbours. Thus, the trust estimated is more accurate.

It may be noted here that under no circumstances the PDR of LTB-AODV can be 100% because the initial part of the simulation is spent by the neighbours to learn the behaviour of the malicious nodes. During this phase, some data packets are already dropped by the malicious nodes before they could be avoided. Thus, it is noteworthy that the highest PDR achieved is 0.9834 (LTB-0.5-mal3) when the maximum speed is 1 m/s. It is also noticed that the PDR decreases as the mobility increases, which is as expected.

Not all packets that could not reach the destination are because of the dropping of packets by the malicious nodes. Therefore we have used the MPDR parameter (Fig. 4), which gives a measure of the packets that are actually dropped by the malicious nodes. LTB-AODV greatly reduces the malicious drop ratio in all cases. Besides, as the speed of the nodes increases, there seems to be no significant increase in the MPDR. This shows that our approach is invariant to the change in speed in that it is successful in avoiding the malicious nodes even in a rapidly changing topology. For reasons discussed earlier, the MPDR is less in LTB-AODV when $\alpha = 0.5$ as compared with the case when $\alpha = 1$.

As compared with AODV, we find that when $\alpha = 1$, LTB-AODV increases the delay only by about 1 ms for all cases (Fig. 5). The reason being that in LTB-AODV only the destination node replies to a route request, whereas in AODV intermediate nodes are allowed for replying to a route request. However, we find that the increase in delay in

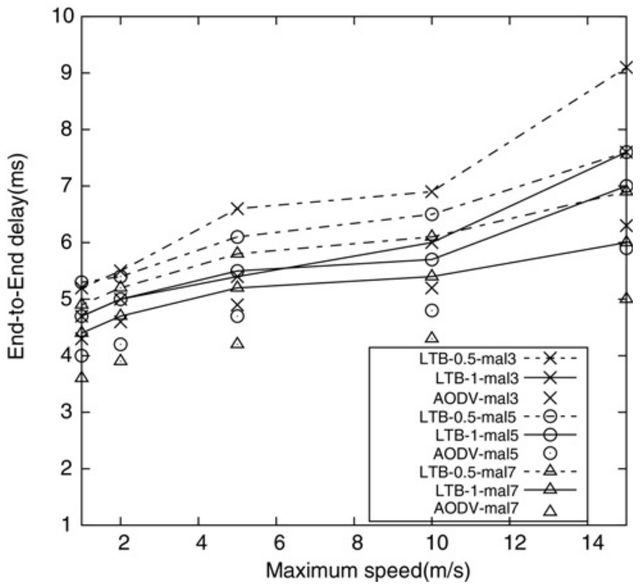

Fig. 5 *End-to-end packet delay against the maximum speed*

LTB-AODV when $\alpha = 0.5$ as compared with AODV is about twice that of the increase when $\alpha = 1$. This is because of the overhead of each node periodically broadcasting the control packet TRUST for sharing trust information. Since more packets are contending for the available channels, more delay occurs. For all cases, the delay tends to increase as the speed of nodes increases.

It is seen that the RF in LTB-AODV is less than that of AODV for all cases (Fig. 6). The routes (most trusted) established in LTB-AODV would potentially be longer than those (shortest routes) established in AODV. When a route is established between a source and a destination, all the intermediate nodes in between them have also consequently established the routes to the source and the destination. Thus, we can conclude that more routes per route request generated are established in LTB-AODV as compared with AODV.

A minimal increase in the RL is seen in LTB-AODV when $\alpha = 1$ as compared to AODV (Fig. 7). In LTB-AODV, only

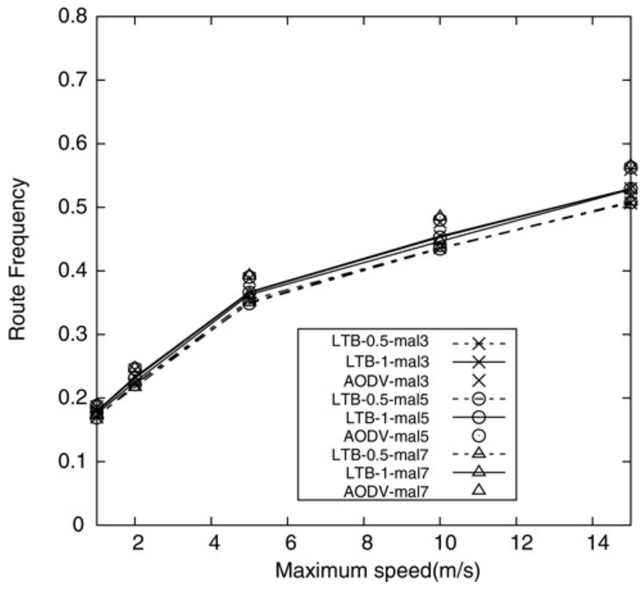

Fig. 6 *Route frequency against the maximum speed*

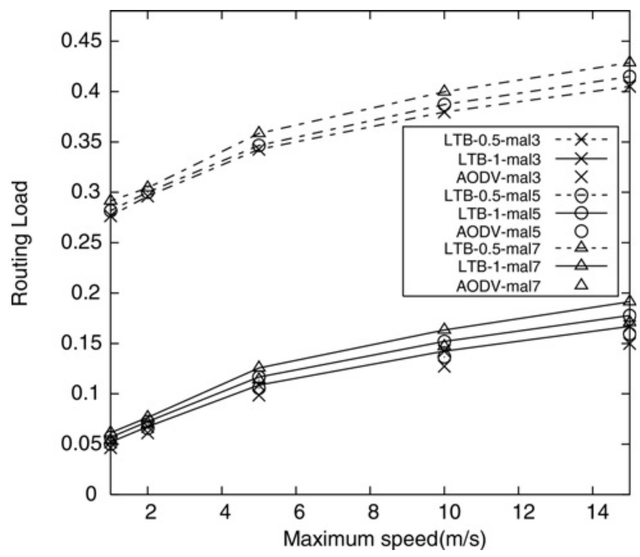

Fig. 7 *Routing load against the maximum speed*

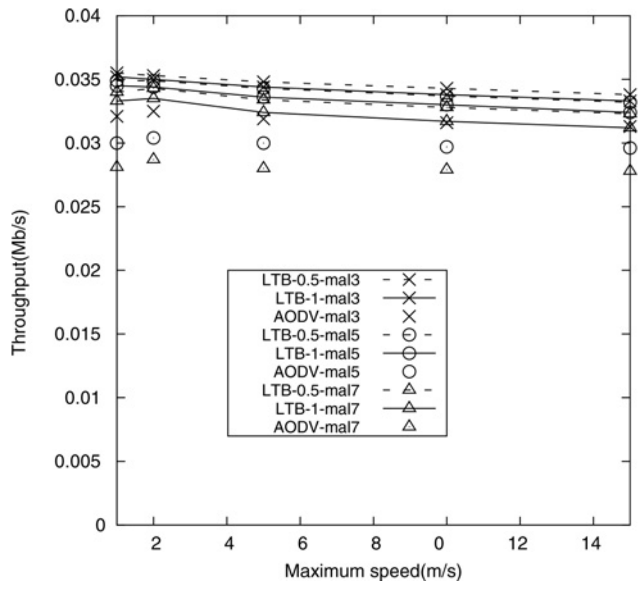

Fig. 8 *Average throughput(Mb/s) against the maximum speed*

the destination is allowed for generating a route reply whereas in AODV, intermediate nodes are allowed for doing so. Thus, control packets would be transmitted over more number of hops than in AODV. However, when $\alpha = 0.5$, the routing load of LTB-AODV is very high as compared with the case when $\alpha = 1$. The reason is that in the scenario when $\alpha = 0.5$, each node periodically broadcasts the control packet TRUST for sharing trust information. This overhead is absent when $\alpha = 1$.

Both variants of LTB-AODV give a better AT than AODV (Fig. 8). This is expected as LTB-AODV also gives a higher PDR than AODV (Fig. 3).

6 Conclusions

We have presented a light-weight trust-based routing protocol. The trust a node has for a neighbour forms the basic building block of our trust model. The proposed trust estimation technique, which is executed by every node in the network independently uses only local information thereby making it scalable. Moreover, unlike other techniques based on monitoring traffic that require a lot of space and time for buffering packets and searching for a packet match, our approach does not require such an overhead. Moreover, we presented two variants of the proposed approach. Depending upon one's priority and need, a variant can be chosen. Simulation results illustrate the effectiveness of our approach.

7 References

1 Gupte, S., Singhal, M.: 'Secure routing in mobile wireless ad-hoc networks', *Elsevier Ad Hoc Netw.*, 2003, **1**, pp. 151–174

2 Cordasco, J., Wetzel, S.: 'Cryptographic versus trust-based methods for MANET routing security', *Elsevier Electron. Notes Theor. Comput. Sci.*, 2008, **197**, pp. 131–140

3 Pirzada, A.A., McDonald, C., Datta, A.: 'Performance comparison of trust-based reactive routing protocols', *IEEE Trans. Mob. Comput.*, 2006, **5**, (6), pp. 695–710

4 Marti, S., Giuli, T.J., La, K., Baker, M.: 'Mitigating routing misbehaviour in a mobile ad-hoc environment'. Proc. Sixth Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking, August 2000

5 Buchegger, S., Le Boudec, J.: 'Performance analysis of the CONFIDANT protocol (cooperation of nodes – fairness in dynamic ad-hoc networks'. Proc. Third ACM Int. Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc'02), June 2002, pp. 226–336

6 Michiardi, P., Molva, R.: 'Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks'. Proc. Sixth IFIP Communication and Multimedia Security Conf. (CMS'02), September 2002

7 Pirzada, A.A., Datta, A., McDonald, C.: 'Incorporating trust and reputation in the DSR protocol for dependable routing', *Elsevier Comput. Commun.*, 2006, **29**, (15), pp. 2806–2821

8 Li, J., Lee, C.: 'Improve routing trust with promiscuous listening routing security algorithm in mobile ad-hoc networks', *Elsevier Comput. Commun.*, 2006, **29**, (8), pp. 1121–1132

9 Balakrishnan, V., Varadharajan, V., Lucs, P., Tupakula, U.K.: 'Trust enhanced secure mobile ad-hoc network routing'. Proc. 21st Int. Conf. on Advanced Information Networking and Applications Workshops (AINAW'07), 2007

10 Buttyan, L., Hubaux, J.: 'Enforcing service availability in mobile ad hoc networks'. Proc. IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC), 2000, pp. 87–96

11 Buttyan, L., Hubaux, J.: 'Stimulating cooperation in self-organizing mobile ad hoc networks'. Proc. ACM/Kluwer Mobile Networks and Applications (MONET), 2003, vol. 8, pp. 579–592

12 Hu, Y., Perrig, A., Johnson, D.B.: 'Ariadne: a secure on-demand routing protocol for ad-hoc networks'. Proc. Mobicom 2002, National Science Foundation and the Polytechnic University, Atlanta, Georgia, USA, 23–26 September 2002

13 Balakrishnan, V., Varadharajan, V., Tupakula, U.K.: 'Fellowship: defense against flooding and packet drop attacks in MANET'. Proc. 10th IEEE/IFIP Network Operations and Management Symp., Vancouver, Canada, 2006, pp. 1–4

14 Pirzada, A.A., McDonald, C.: 'Establishing trust in pure ad hoc networks'. Proc. 27th Australian Computer Science Conf. (ACSC), 2004, vol. 26, pp. 47–54

15 Saha, H.N., Bhattacharyya, D., Banerjee, P.K.: 'A distributed administration based approach for detecting and preventing attacks on mobile ad hoc networks', *Int. J. Sci. Eng. Res.*, 2011, **2**, (3), Part(II)

16 Xing, F., Wang, W.: 'Understanding dynamic denial of service attacks in mobile ad hoc networks'. Proc. Military Communications Conf., (MILCOM 2006), 2006, pp. 1–7

17 Lima, M.N., da Silva, H.W., dos Santos, A.L., Pujolle, G.: 'Requirements for survivable routing in MANETs'. Proc. Wireless Pervasive Computing, (ISWPC 2008), 2008, pp. 441–445

18 Nguyen, H.L., Nguyen, U.T.: 'A study of different types of attacks on multicast in mobile ad hoc networks', *Elsevier Ad Hoc Netw.*, 2008, **6**, pp. 32–46

19 Zakhary, S.R., Radenkovic, M.: 'Reputation-based security protocol for MANETs in highly mobile disconnection-prone environments'. Proc. IEEE/IFIP WONS 2010 – Seventh Int. Conf. on Wireless On-demand Network Systems and Services, 2010, pp. 161–167

20 Su, M.-Y., Chiang, K.-L., Liao, W.-C.: 'Mitigation of black-hole nodes in mobile ad hoc networks'. Proc. Parallel and Distributed Processing with Applications (ISPA), 2010, pp. 162–167

21 Tamilselvan, L., Sankaranarayanan, V.: 'Prevention of blackhole attack in MANET'. Proc. Wireless Broadband and Ultra Wideband Communications, 2007 (AusWireless 2007), 2007, pp. 21–26

22 Bhalaji, N., Shanmugam, A.: 'Association between nodes to combat blackhole attack in DSR based MANET'. Proc. Wireless and Optical Communications Networks (WOCN), 2009, pp. 1–5

23 Abusalah, L., Khokhar, A., BenBrahim, G., ElHajj, W.: 'TARP: trust-aware routing protocol'. Proc. ACM IWCMC 2006, Vancouver, Canada, 3–6 July 2006

24 Rezgui, A., Eltoweissy, M.: 'TARP: a trust-aware routing protocol for sensor-actuator networks'. Proc. Fourth IEEE Int. Conf. on Mobile Ad Hoc and Sensor Networks, Pisa, Italy, October 2007

25 Umuhoza, D., Agbinya, J.I., Omlin, C.W.: 'Estimation of trust metrics for MANET using QoS parameters and source routing algorithms'. Proc. Second Int. Conf. on Wireless Broadband and Ultra Wideband Communications, (AusWireless 2007), 2007

26 Li, X., Lyu, M.R., Liu, J.: 'A trust model based routing protocol for secure ad hoc networks'. Proc. IEEE Aerospace Conf., 2004

27 Virendra, M., Jadliwala, M., Chandrasekaran, M., Upadhyaya, S.: 'Quantifying trust in ad-hoc networks'. Proc. IEEE Int. Conf. on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS), 2005, pp. 65–71

28 Meka, K.D., Virendra, M., Upadhyaya, S.: 'Trust based routing decisions in mobile ad-hoc networks'. Proc. Workshop on Secure Knowledge Management (SKM 2006), 2006

29 Perkins, C.E., Royer, E.M.: 'Ad-hoc on-demand vector routing'. Proc. Second IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90–100