PERFORMANCE ANALYSIS OF THE NEIGHBOR WEIGHT TRUST DETERMINATION ALGORITHM IN MANETS

Ali Abu Romman[1] and Hussein Al-Bahadili[2]

[1]King Hussein Faculty for Computing Sciences, Princess Sumaya University for Technology, Amman, Jordan
[2]Faculty of Information Technology, University of Petra, Amman, Jordan

ABSTRACT

Mobile ad-hoc networks (MANETs) are susceptible to attacks by malicious nodes that could easily bring down the whole network. Therefore, it is important to have a reliable mechanism for detecting and isolating malicious nodes before they can do any harm to the network. One of the possible mechanisms is by using trust-based routing protocols. One of the main requirements of such protocols is to have a cost-effective trust determination algorithm. This paper presents the performance analysis of a recently developed trust determination algorithm, namely, the neighbor-weight trust determination (NWTD) algorithm. The performance of the algorithm is evaluated through simulation using the MANET simulator (MANSim). The simulation results demonstrated the reliability and effectiveness of the algorithm in identifying and isolating any maliciously behaving node(s) in a timely manner.

KEYWORDS

NWTD; trust determination; trust-based routing protocols; malicious node; MANET; MANSim.

1. INTRODUCTION

A MANET is defined as a set of wireless mobile nodes communicate with one another for a purpose of data exchange without relying on any pre-existing or centralized infrastructure[1]. Early MANET research assumed a friendly and cooperative environment and focused on problems such as wireless channel access, multi-hop routing, power consumption, while ignoring any network security issues. Thus, early MANETs are venerable and susceptible to attacks by malicious nodes that could easily bring the network down. Therefore, network security becomes a major issue to protect network infrastructure from being attacked by adversary or malicious nodes and to provide protected communication between nodes in a potentially hostile MANET environment [2].

MANETs heavily suffer from malicious nodes, which could easily degrade the network stability by exhibiting one or more of the following behavior: packet drop or delaying, buffer overflow, battery drained, bandwidth consumption, link break, message tampering or modification or discarding, route modification, node isolation, stealing information, session capturing, etc. Therefore, it is important to have a reliable mechanism for detecting and isolating malicious nodes before they can impair the network. One of the potential mechanisms that has been developed is the trust-based routing mechanism. In which only trusted nodes are accepted for forwarding data/control packets, so that each node to be part of the routing table, it should have a trust above a certain value known as minimum acceptable trust (MAT). The main requirement and challenge to these protocols is the availability of an appropriate trust determination [3 -5].

DOI: 10.5121/ijnsa.2016.8403

One of the earliest approaches for trust determination is Marsh's formalism [6], in which the outcomes of direct interactions among nodes is used by each node to calculate the trusts of other nodes in the network. After each interaction, a node considers whether the other node fulfilled its obligations. If fulfilled, then trust increases otherwise decreases. Based on this formalism, a number of trust determination algorithms have been developed throughout the years [7,8].

This paper analyzes the performance of a recently developed trust determination algorithm, namely, the Neighbor-Weight Trust Determination (NWTD) algorithm [9-11], which is based on the weighted voting concept. In NWTD, each node in the network periodically broadcasts messages containing the IDs and trusts of its one-hop neighbors. Each node after receiving these messages from its one-hop neighbors, extracts the IDs and trusts of each node on the message. It is expected that a node receives different trusts for the same one-hop neighbor from different nodes. Thus, the receiving node calculates the average of the received trusts using a weighted-average formula to represent the new trust of the node. The weight here is the weight of the one-hop neighbors, therefore it is referred to as the NWTD algorithm. The receiving node participates in the averaging process by giving itself a trust and weight of unity.

The algorithm defines two types of nodes: the master node and the monitoring node. A master node (hereinafter referred to as master only) is any trusted node in the network that can take the responsibility of testing new arriving nodes, determines their initial trust, and then broadcasts the initial trust to its one-hop neighbors. A monitoring node is any node in the network that has the capacity to detect the malicious behavior of other nodes within its neighborhood, updates and broadcasts the trust of the detected malicious nodes. The algorithm allows one or more monitoring nodes to be active at the same time.

The NWTD algorithm is implemented and integrated with the MANET simulator (MANSim)[12] to evaluate the performance of the algorithm. In particular, this paper uses MANSim to simulate three different scenarios. The first one simulates a network having no malicious nodes; the second scenario simulates a network having one monitoring and one malicious node; and the third scenario simulates a network having one monitoring and two malicious nodes.

This section introduces the general domain of the paper. The rest of this paper is organized as follows: Section 2 reviews some of the most recent and related on trust determination. Section 3 describes the NWTD algorithm. The implementation of the NWTD algorithm and the simulation environment used in this paper are described in Section 4. The results and discussions are presented in Section 5. Finally, in Section 6, based on the simulation results, conclusions are drawn and a number of recommendations for future work are pointed-out.

2. LITERATURE REVIEW

Many trust determination models have been developed for peer-to-peer systems [13], which are based on sharing recommendation information to establish trust and reputation. Applying these models to MANETs faces two main problems; these are significant network overhead due to the additional information exchanged, and requirement of a trusted third party (or a computationally expensive public key infrastructure (PKI)), which are against the nature of MANETs. However, later on, many trust determination models have been developed for MANETs [6, 7].

Ferdous et al. [4] developed a novel scheme for trust management in MANETs, namely, the Network Trust Management (NTM) scheme that is based on the nodes' own responsibility of building their trust level and node-level trust monitoring. Hughes et al. [5] proposed a dynamic trust-based resources (DyTR) system, which applies a dynamic notion of trust to network resources. DyTR continuously assesses the trustworthiness of nodes over time based on network environment. DyTR utilizes a socio-cognitive model of trustfor dynamic trust assessment.

Gong et al. [8] presented a routing protocol called secure and energy aware routing protocol (ETARP) designed for energy efficiency and security for wireless sensor networks (WSN). The key part of ETARP is to simultaneously consider energy efficiency and trustworthiness of routes. Simulation results show that ETARP keeps the same security level while achieving more energy efficiency for packet delivery as compared to previously developed routing protocols.

Sabater & Sierra [14] and Ramchurn et al. [15] developed approaches that are based on Marsh formalism[6], namely, ReGreT and FIRE, which add reputation information provided by third parties and knowledge of social structures to arrive at overall trust assessments. However, whilst powerful, such sophisticated models are not appropriate for MANETs, where resources are limited and knowledge of social relationships between nodes is unlikely to be accessible. Pirzada & McDonald [16] developed a trust determination mechanism, where nodes calculate situational trust according to observed events and then use an aggregated general trust for routing decisions.Sun et al. [17] presented an information theoretic framework to quantitatively measure trust. They developed four axioms and based on these axioms, they presented two trust models: entropy-based and probability-based models, which satisfy all the axioms. Simulations showed that these models could significantly improve the network throughput as well as effectively detect malicious behaviors in MANETs.

Sylvia et al. [3] proposed a trust based routing scheme for MANETs under adverse environment and compare the performance of the proposed scheme against existing schemes. The simulation results demonstrate that in adverse environment, the proposed scheme improves network throughput and delay;however, it incurs higher overhead.Cordasco & Wetzel [7] compared the performance of two MANET routing protocols; these are: the Secure AODV (SAODV) and Trusted AODV (TAODV), which address routing security through cryptographic and trust-based means respectively.

A dynamic trust prediction models to evaluate the trustworthiness of nodes were also developed by K. Haldaret al. [18], Park et al. [19], Saini and Gautam [20], Xia et al. [21], Gowda and Hiremath [22], and Patil et al. [23].A trust establishment and management framework for hierarchical wireless sensor networks was developed by Zhang et al. [24], and also a trust management framework (TMF) for MANETs was developed by Guo et al. [25].

3. THE NWTD ALGORITHM

Trust determination algorithms should resolve three main issues[6]:

a. Determine an initial trust value for any new arriving node, so that it will be either trusted by other nodes in the network or discarded.
b. Periodically determine/update the trust of all nodes in the network.
c. Broadcast the newly determined trusts to other nodes in the network with minimum overheads.

Trust determination models are usually used in dynamic and multivariable environment, so that it is important to define some configuration parameters that should be carefully adjusted at each node to suit the environment and to optimize the performance of the trust determination model. These parameters include[6, 10, 11]:

a. Minimum Acceptable Trust (MAT). The minimum trust a node should have in order to be trusted by other nodes, which is usually between 0 and 1.
b. Trust Update Time (TUT).The minimum duration before updating current trust of any node on the network.

c. Minimum Trustable Participants (MTPs).The minimum number of trustable neighbors that should participate in determining the trust of any other node in the network. For example if MTP=3, then for Node $i(N_i)$ to determine the trust of N_j ($T_{i,j}$), N_i should get trusts for N_j from at least 2 other trustable nodes as it already has a trust for N_j.

3.1. Trust Determination

In the NWTD algorithm, each node in the network periodically broadcasts an extended HELLO (EHELLO) message to its one-hop neighbors. The EHELLO message has the same format of standard HELLO message usually broadcasted by the network routing protocol and some additional data appended at the end of the message; these are the number of one-hop neighbors and the ID and trust of each of the one-hop neighbors.

On the other hand, each receiving node then performs the following tasks:

- Compare the trust of the broadcasting node with the pre-defined MAT;if it is ≥MAT, then accept message, otherwise discard the message.
- Extract the IDs and trusts of the one-hop neighbors of the broadcasting node that pass the above test.
- Construct a table listing the one-hop neighbor(s) and the trusts they have for each other. The node is fully trusted by itself (i.e., $T_{x,x}$=1)
- Determine the trust of the one-hop neighbors using the mathematical model described below.
- Remove from the routing table any node for which the determined trust is <MAT.
- Broadcast the newly determined trusts to all one-hop neighbors and waits for broadcasts from its neighbors.

A node $s(0)$ calculates the trust of each of its one-hop neighbors as follows:

a. Calculates the average trust of the one-hop neighbor $x$as follows:

$$\bar{T}_{s(0),s(j)} = \frac{1}{k_j}\sum_{i=0}^{k_j} T_{s(i),s(j)} \qquad (j=1 \text{ to } n)\ (k_j \geq \text{MTP}) \tag{1}$$

Where $T_{s(i),s(j)}$ is the trust of $s(j)$ as determined by $s(i)$, where $s(i)$ and $s(j)$ are one-hop neighbors of $s(0)$; $\bar{T}_{s(0),s(j)}$ is the average trust of $s(j)$ as calculated by $s(0)$; k_j is the number of nodes that have trust for $s(j)$, n is the number of the one-hop neighbors of $s(j)$; and MTP is the minimum trustable participants.

b. Calculates the new trust of $s(j)$ as the sum of the product of the trust of $s(j)$ as determined by $s(i)$ and the weight of $s(i)$ as determined by $s(0)$, which is mathematically expressed as:

$$T_{s(0),s(j)} = \sum_{i=0}^{k_j} w_{s(0),s(i)}^{s(j)} \cdot T_{s(i),s(j)} \qquad (j=1 \text{ to } n)\ (k_j \geq \text{MTP}) \tag{2}$$

Where $w_{s(0),s(i)}^{s(j)}$ is the weight of $s(i)$ as determined by $s(0)$ for a particular $s(j)$, and it is calculated as:

$$w_{s(0),s(i)}^{s(j)} = \frac{\bar{T}_{s(i),s(j)}}{\sum_{m=0}^{k_j} \bar{T}_{s(m),s(j)}} \qquad (i=0 \text{ to } k_i,\ j=1 \text{ to } n)\ (k_j \geq \text{MTP}) \tag{3}$$

The average trusts assist to compute the weights of the nodes that determines the new trust of their one-hop neighbors, which means that the one-hop neighbors share their ideas before deciding the trust of any of their neighbors, and the contribution of each neighbor depends on its weight as determined by the receiving node.

3.2. Master Node

A master node is responsible for testing a new arriving node to determine its initial trust; and pass the result of this test to its one-hop neighbors. Each node in the network must determine a masterfor itself by applying a certain criteria on its one-hop neighbors and itself. If a node could not determine a master for itself, it can be a master for itself. The main properties of a master are it is one-hop neighbor for both the new node and node searching for a master, and the new node must trust it.If there is more than one node having the same properties, then the one with the smallest ID is selected as the master [9].

Let us consider a scenario with four nodes (A, B, C and D) trust each other and trusted by their one-hop neighbors. They are all in same range except for D, which is in range only with C. New Node *x* arrives to the network. It is one-hop neighbor for A, B, C and D as shown in Fig.1. All four nodes can detect that there is a new node arrive to the network. When detected, each node will compare its table of one-hop neighbors with the neighbor table received from *x*in order to choose a master [10, 11].

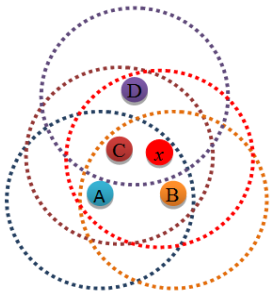

Fig.1. Master node selection scenario.

For example, A will choose itself as a master, B and C will choose A as a master and D will choose C as a master. The reason why D chooses C as a master is that it doesnot see A or B. However, C will not become master, as it already knows that A will test *x*. Therefore, it will just wait for the result and then forward it to D.

In the current version of NWTD, the master performs two types of test on any new arriving node, namely, the FirstTest and the SecondTest. In the FirstTest, the master randomly chooses a destination node with a known and reachable route. It informs the destination that test will start. Then, it sends a route request (RREQ) message for that destination through *x*. The first RREQ has intentionally increased sequence number, so that the new node should not reply. If it replies it indicates a malicious behavior (Test1()). While, the second RREQ has a normal sequence number and the new node should reply back telling the master that it has a route to the destination. However, if it doesnot reply, this indicates a malicious behavior (Test2()).

This test replays with different destinations for a number of times, and the success and fail ratios are calculated. The success ratio (*S*) is ratio between the number of successful test divided by the total number of tests, and the fail ratio (*F*) is the ratio between the number of failed tests divided by the total number of tests; so that $S+F=1$. If *S* of the new node is less than a pre-set value, then it is announced as a non-trusted node. Fig. 2 outlines the pseudocode for the procedures FirstTest(), Test1(), and Test2().

If the new node made a success then a second phase of testing will begin (SecondTest()). In the SecondTest, the master sends data packet to some destination via the new node. At the same time, the master sends the hash value of the transmitted data packet to the destination by using its old route (not going through the new node).

The SecondTest replays for a number of times. Each time, the destination calculates the hash of the received data packet and compares it with the hash received from the master. If they match then it is a successful test, otherwise, it is a failed test. If both phases have completed successfully then the master announces that the new node is fully trusted. Fig. 3 outlines the pseudocode for the procedures SecondTest() and Testee(). These tests should be performed periodically between nodes to ensure stability of the network.

```
FirstTest()
Failed=0
For (trials<nTests)
    If (Test1() == False)
        Failed++
    End If
    If (Test2() == False)
        Failed++
    End If
    If (Failed ≥ 25% of Trials)
        BlockNode ();
    Else
        SecondTest()
    End If
```

```
Test1()
SendRREQ(IncrSeqNo)
ReceiveRREQ()
If (RREP Received)
    Return False;
Else
    Return True;
End If
```

```
Test2()
SendRREQ(NormSeqNo)
ReceiveRREQ()
If (RREP received)
    Return True;
Else
    Return False;
End If
```

Fig. 2. The pseudo code for procedures FirstTest(), Test1(), and Test2().

```
SecondTest()
packet = GenerateDataPacket()
hash = CalculateHash(packet)
Destination = Testee
OriginalRoute = RoutingTable(Testee)
TestedRoute = RetrieveRouteFrom(RREP)
Send(hash, OriginalRoute)
Send(data, TestedRoute)
Response = ReceiveResponse(Testee)
If (Response == False)
    Return False;
Else
    Return True;
End If
```

```
Testee()
hash = ReceiveHash() // Receive hash from master node
pkt = ReceivePacket()// Receive packet from tested node
If (pkt.IsReceived)
    newHash = CalculateHash(pkt)
    If  (hash == newHash)
        Send(True)// Send response to the master node
    Else
        Send(False)
    End If
Else
    Send (False)
End If
```

Fig. 3. The pseudocode for procedures SecondTest() and Testee().

3.3. Monitoring Node

In addition to the master discussed above, NWTD designates as many nodes as possible as monitoring nodes. The monitoring nodes monitor and classify the behavior of its one-hop neighbors into positive and malicious behaviors, update the trust of its one-hop neighbors by upgrading the trust of the positively behaving neighbors and degrading the trust of the maliciously behaving neighbors, and use the updated trust of its one-hop neighbors in the forthcoming trust determination procedure described above.

The monitoring node updates the trust of its one-hop neighbors using the following simple linear equation:

$$T_{updated} = \alpha \cdot T_{current} \quad (4)$$

Where $T_{current}$ and $T_{updated}$ are the trust of the one-hop neighbor before and after the update; and α is the update factor (α>0, α>1 for positively behaving nodes, and α<1 for maliciously behaving nodes). The value of α can be determined dynamically by the monitoring node for each of its one-hop neighbors separately; which means different values of α can be determined for different nodes at the same time depending on their behavior.

It is important to recognize that any node in the network can act a monitor node as long as it has the capabilities to monitor the behavior of its neighbors, classify their behaviors into positive and malicious behaviors and update their trust accordingly. Furthermore, it can be easily seen that the trust of the node depends on the number of one-hop neighbors, the trusts of the node as determined by its one-hop neighbors, and the weight of the one-hop neighbors.

4. IMPLEMENTATION AND SIMULATION ENVIRONMENT

In order to evaluate the performance of the NWTD algorithm, it is implemented and integrated with MANSim, which is a network simulator written with C++ programming language for evaluating the performance of various MANETs protocols[12]. In particular, three main functions are developed and integrated with MANSim, these are:

a. TrustDetermination(), where the trust determination model described in Section 3.1 is implemented.
b. InitialTrust. As it has been discussed in Section 3.2 that the master is responsible for testing and determining the initial trust of any new arriving node, and broadcast this initial trust to its one-hop neighbors. This function simulates the function of the masters, where it determines the trust of the one-hop neighbors for each of the nodes in the network using the following initial trust distribution function: $T_{i,j} = 0.5 + 0.5\xi$, where $T_{i,j}$ is the trust of N_j as determined by N_i, and ξ is a randomnumber between 0 and 1. Accordingly, the initial trust lies between 0.5 and 1. However, $T_{i,j}$ can be determined using any other linear or non-linear function.
c. TrustUpdate(), where the trust update procedure described in Section 3.3 is implemented.

4.1. Simulation Environment

The simulation environment that will be used throughout this paper simulates a network area of 150x150m with 49 nodes distributed across the network using the semi-regular node distribution of MANSim [12]. Each node has a transmission radius (R) of 30m. The actual nodes locations across the network are shown in Fig. 4. This distribution is chosen because it is easy to predict the NWTD performance variation according to the nodes behavior. Furthermore, using the same nodes distribution keeps the focus on the effect of nodes behavior. Furthermore, the nodes are assumed to be fixed (non-mobile with speed u=0) throughout the simulations.

The main attributes of the nodes (e.g., location, R, u, initial trust, and simulation time (T_{sim}))are determined. The simulation time is divided into discrete intervals (loops). During each loop, the nodes calculate the new trusts for its one-hop neighbors. In addition, the monitoring nodes may update (degrade or upgrade) the trust of its one-hop neighbors based on their behavior. Therefore, we refer to the number of discrete intervals as number of updates (U). In this work, each simulation carries 25 updates (U=25), which is enough to demonstrate the trust variation.

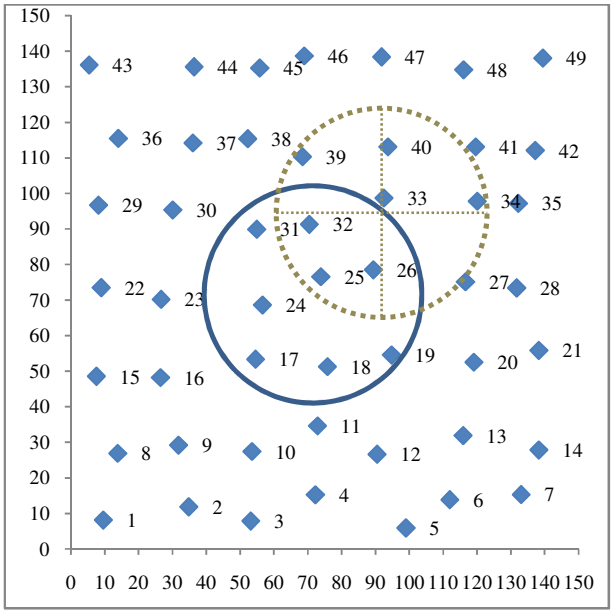

Fig.4. Nodes distributed in 150x150m network area for all scenarios.

5.RESULTS AND DISCUSSIONS

In order to evaluate the performance of the NWTD algorithm in identifying and isolating a malicious node, the MANSim simulator has been used to simulate three different scenarios considering the simulation environment described above.

Scenario #1 simulates a neighborhood that confined no malicious nodes. It assumes all nodes across the network are behaving positively (i.e., $\alpha_{i,j}$=1). The trust of the one-hop neighbors of N_{25} ($T_{25,x}$) (e.g., Nodes: 17, 18, 19, 24, 26, 31, 32, 33) are estimated and plotted in Fig. 5.The results demonstrate that after an initial fluctuation for few updates, the trust of the nodes almost stabilizes and none of the nodes introduces any changes in the trusts of its one-hop neighbors.

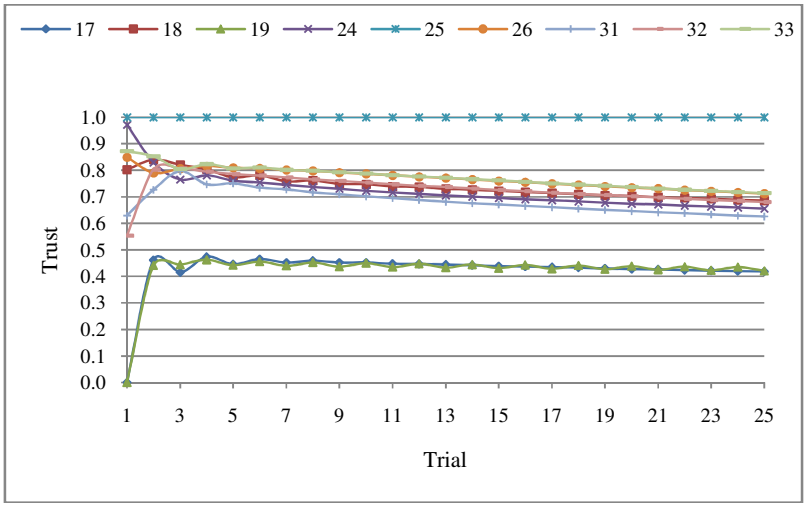

Fig.5. Trusts of some neighbors of N_{25} ($T_{25,x}$) ($\alpha_{i,j}$=1).

Scenario #2 simulates a neighborhood that confines one monitoring node (N_{25}) and one malicious node (N_{31}). It assumes all nodes across the network are behaving positively ($\alpha_{i,j}$=1) except for N_{31}, which presents some malicious behavior as recognized by the monitoring node (N_{25}). The trust of some neighbors of N_{25} ($T_{25,x}$) are estimated considering two different update factors ($\alpha_{25,31}$) of 0.9and 0.7. The results obtained for these two different cases are shown in Figs. 6 and 7.

In this scenario, after an initial fluctuation for few updates, the trusts of the nodes almost stabilize with some decreasing rates for all nodes. The reduction rate depends on the nodes locations with respect to N_{25} and N_{31}, and the update factor. For example, after 25 updates, N_{18}has a trust of 78.6%, 74.8%, and 68.6% for $\alpha_{25,31}$ 1.0, 0.9, and 0.7, respectively. This means that the trusts of N_{18} decreases as $\alpha_{25,31}$ increases. This is because the reduction in the trust of N_{31} reduces its weight and consequently the trust of any nodes received through it. Furthermore, it has been found that if MAT=0.5, N_{25} requires around 50 updates to isolate N_{31} with $\alpha_{25,31}$=0.6 and around 40 updates with $\alpha_{25,31}$=0.5.

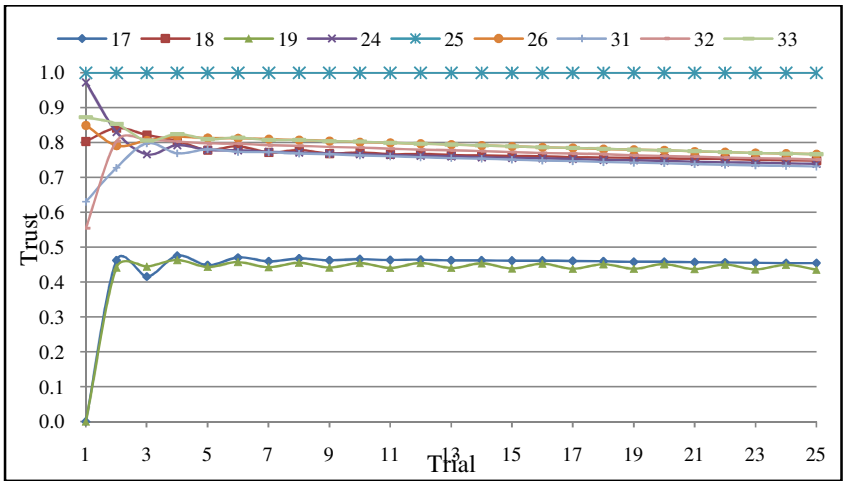

Fig.6. Trusts of one-hop neighbors of N_{25} ($T_{25,x}$) ($\alpha_{25,31}$=0.9)

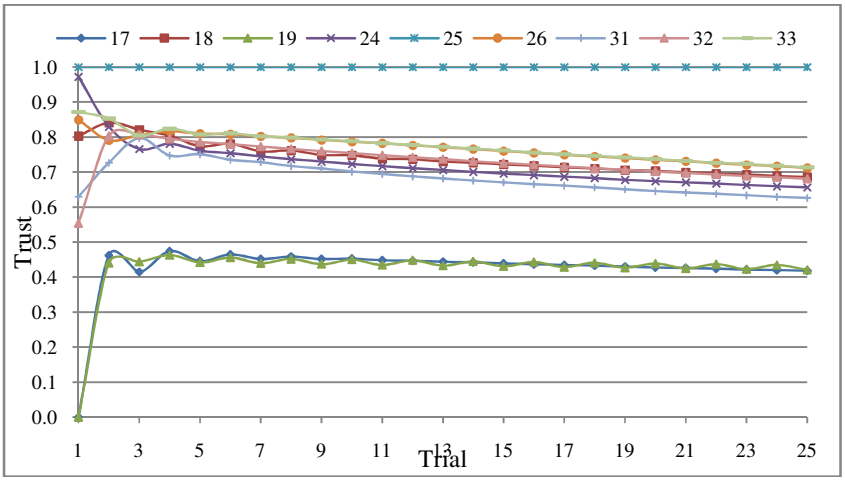

Fig. 7. Trusts of some neighbors of N_{25} ($T_{25,x}$) ($\alpha_{25,31}$=0.7).

Scenario #3 simulates the same network configuration in Scenario #1. However, it assumes different nodes behavior in which two of the first hop-neighbors of N_{25} are acting maliciously, namely, N_{31} and N_{26}as detected by N_{25}. Due to their malicious behavior, N_{25} reduces their trusts by a certain percentage. In particular, we shall investigate two different cases, in the first, the trust of N_{31}=N_{26}=0.8 ($\alpha_{25,31}$=$\alpha_{25,26}$=0.8)and, in the second,0.7 ($\alpha_{25,31}$=$\alpha_{25,26}$=0.7).The variation of $T_{25,31}$ and $T_{25,26}$ for the first and second cases are shown in Figs.8 and 9. The figures also show the variation of $T_{25,31}$ and $T_{25,26}$ for $a_{25,31}$=0.8 and $a_{25,31}$=0.7 (one malicious node) for the sake of comparison between Scenarios #2 and #3.

Scenario #3 demonstrates that with two malicious nodesless converges time or number of updates is required to isolate malicious nodes. In this scenario,N_{25} reduces the trust of N_{31} and N_{26} each update, and consequently reduce the trust and weight of their neighbors (N_{32}). Because of that, these neighbors contribute to further reduction in the trust of the N_{31} and N_{26}. For example, after 25 updates, $T_{25,31}$ is equal to 0.626 when only N_{31} acting maliciously ($\alpha_{25,31}$=0.7), and equal to 0.551 when both N_{31} and N_{26} are acting maliciously ($\alpha_{25,31}$=$\alpha_{25,26}$=0.7). Also, $T_{25,26}$ is equal to 0.712 when only N_{31} acting maliciously ($\alpha_{25,31}$=0.7), and equal to 0.524 when both N_{31} and N_{26} are acting maliciously ($\alpha_{25,31}$=$\alpha_{25,26}$=0.7).

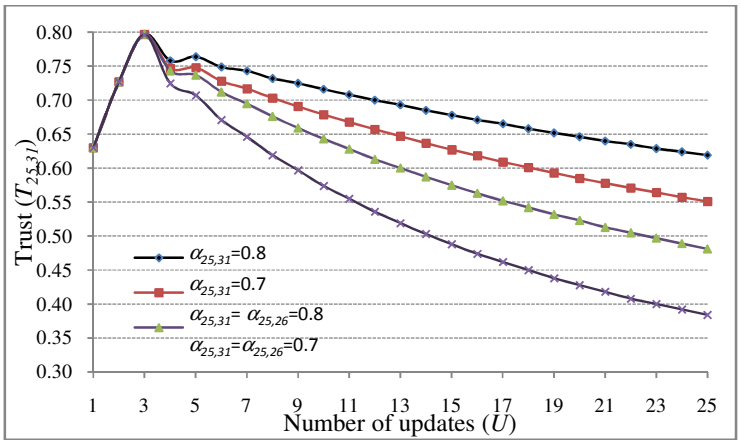

Fig. 8. Variation of $T_{25,31}$ for various $\alpha_{25,31}$ and $\alpha_{25,26}$

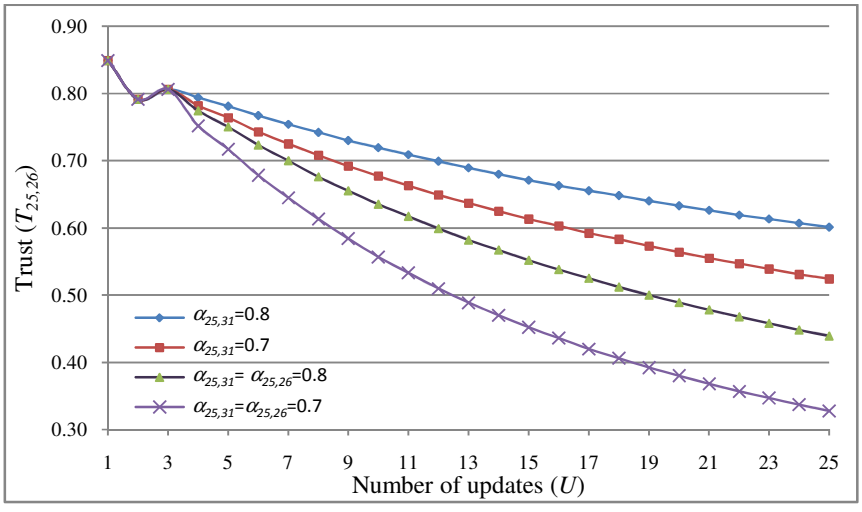

Fig. 9. Variation of $T_{25,26}$ for various $\alpha_{25,31}$ and $\alpha_{25,26}$.

6. CONCLUSIONS

The NWTD developed algorithm is an efficient and effective algorithm that can be reliably used to determine the trust of mobile nodes in MANETs, and consequently identify and isolate any malicious node. The simulation results demonstrate that for fully trusted network (no malicious node within the network), the nodes determine a certain trust for each other and the trust between the nodes remain steady as long as the monitoring node does not detect any malicious behavior from any node within the network (Scenario #1). However, if a malicious behavior is detected by the monitoring node for one (Scenario #2) or more nodes(Scenario #3) within the network, then the trust of all interactive nodes are negatively affected (reduced) and those of malicious behavior will be affected more, so that can be quickly detected and isolated.

The simulation results of Scenario #3 show that the algorithm can handle efficiently more than one malicious node at the same time by the same monitoring node, and can use different update factors for each malicious node. Furthermore, as we discussed above that smaller update factor can help with quicker identification and isolation of a malicious node.

REFERENCES

[1] C. S. R. Murthy and B. S. Manoj. Ad Hoc Wireless Networks: Architectures and Protocols. Prentice Hall PTR, 2004.

[2] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang. Security in Mobile Ad Hoc Networks: Challenges and Solutions.IEEE Transactions on Wireless Communications, Vol. 11, No. 1, pp. 38–47, 2004.

[3] D. Sylvia, J. Katiravan, and D. Srinivasa Rao. Trust based Routing in Wireless Ad Hoc Networks under Adverse Environment. International Journal of Computer Applications (IJCA), Vol. 136, No.10, pp. 23-28, 2016.

[4] R. Ferdous, V. Muthukkumarasamy, and A. Sattar. A Node-based Trust Management Scheme for Mobile Ad-Hoc Networks. Proceedings of the 4thInternational Conference on Network and System Security (NSS), pp. 275–280, 2010.

[5] T. Hughes, P. D, J. Denny, P. A. Muckelbauer, P. D, and J. Etzl. Dynamic Trust Applied to Ad Hoc Network Resources.Proceedings of Autonomous Agents & Multi-Agent Systems Conference, 2003.

[6] S. P. Marsh. Formalising Trust as a Computational Concept.PhD Thesis, Department of Computing Science and Mathematics, University of StirlingUSA, 1994.

[7] J. Cordasco and S. Wetzel. Cryptographic Versus Trust-based Methods for MANET Routing Security. Electron. Electronic Notes in Theoretical Computer Science, Vol. 197, No. 2, pp. 131–140, 2008.

[8] P. Gong, T. M. Chen, and Q. Xu. ETARP: An Energy Efficient Trust-Aware Routing Protocol for Wireless Sensor Networks. Journal of Sensors, Vol. 2015, Article ID 469793, 10 pages, 2015.

[9] K. El-Zayyat, H. Al-Bahadili, and T. Zobai. A Novel Neighbor Weight-Based Trust Determination Model for Wireless Ad Hoc Networks. Proceedings of the International Conferenceon Theoretical and Mathematical Foundationof Computer Science (TMFCS-10), No. 10, pp. 116–126, Jul. 2010.

[10] A. Abu Romman. Evaluating the Performance of the Novel Neighbor Weight-Based Trust Determination Algorithm in Wireless Ad Hoc Networks. Master Thesis, Princes Sumaya University of Science and Technology, Faculty of Information Technology. Amman, Jordan, 2013.

[11] H. Al-Bahadili. Trust Determination in Wireless Ad Hoc Networks. Book Chapter in Handbook of Research on Threat Detection and Countermeasures in Network Security (Editors: A. H. Al-Hamami and Gh. M. W. al-Saadoon), Chapter 18, pp. 330-348, 2015.

[12] H. Al-Bahadili. On the Use of Discrete-Event Simulation in Computer Networks Analysis and Design.In Handbook of Research on Discrete-Event Simulation Environments: Technologies and Applications (Eds.: Evon M. O. Abu-Taieh and Asim A. El-Sheikh). Information Science Reference, Chapter 19, pp. 418-442, 2010.

[13] S. Song, K. Hwang, R. Zhou, and Y.-K. Kwok. Trusted P2P Transactions with Fuzzy Reputation Aggregation. IEEE Internet Computing, Vol. 9, No. 6, pp. 24–34, 2005.

[14] J. Sabater and C. Sierra. Reputation and Social Network Analysis in Multi-Agent Systems. Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02): Part 1, pp. 475–482, 2002.
[15] S. D. Ramchurn, D. Huynh, and N. R. Jennings. Trust in Multi-Agent Systems. The Knowledge Engineering Review, Vol. 19, No. 1, pp. 1-25, 2005.
[16] A. A. Pirzada and C. McDonald. Establishment in Pure Ad-Hoc Networks. Wireless Personal Communications, Vol. 37, No. 1-2, pp. 139–168, 2006.
[17] Y. L. Sun, S. Member, Z. Han, and K. J. R. Liu. Information Theoretic Framework of Trust Modeling and Evaluation for Ad Hoc Networks. IEEE Journal on Selected Area in Communications, Vol. 24, pp. 305–317, 2006.
[18] K. Haldar, N. Narayan and B. K. Mishra. Mathematical Model on Selfishness and Malicious Behavior in Trust based Cooperative Wireless Networks. International Journal of Computer Network and Information Security, Vol. 10, pp. 15-22, 2015.
[19] S.-S. Park, J.-H. Lee, and T.-M. Chung. Cluster-Based Trust Model against Attacks in Ad-Hoc Networks.Proceedings of the 3rdInternational Conference on Convergence and Hybrid Information Technology (ICCIT '08),Vol. 1, pp. 526–532, 2008.
[20] R. Saini and R. K. Gautam. Establishment of Dynamic Trust among Nodes in Mobile Ad-Hoc Network. Proceedings of 2011 International Conference on Computational Intelligence and Communication Networks (CICN), pp. 346–349, 2011.
[21] H. Xia, Z. Jia, X. Li, L. Ju, and E. Sha. Trust Prediction and Trust-Based Source Routing in Mobile Ad Hoc Networks. Journal of Ad Hoc Network, Vol. 11, Issue 7, pp. 2096-2114, 2013.
[22] S. R. Gowda and P. S. Hiremath. Secure Routing Schema for MANET with Probabilistic Node-to-NodeForwarding. International Journal of Computer Science Issues (IJCSI), Vol. 10, No. 3, pp. 51-58, 2013.
[23] K. Patil, P. Bhanodia, and S. Joshi. A Novel Paradigm RIP (Reputation Index Protocol) for MANET.International Journal of Engineering Research & Technology (IJERT), Vol. 2, No. 1, pp. 1-5, 2013.
[24] J. Zhang, R. Shankaran, M. A. Orgun, V. Varadharajan, and A. Sattar. A Dynamic Trust Establishment and Management Framework for Wireless Sensor Networks. Proceedings of the 2010 IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC), pp. 484–491, 2010.
[25] J. Guo, A. Marshall, and B. Zhou. A New Trust Management Framework for Detecting Malicious and Selfish Behaviour for Mobile Ad Hoc Networks. Proceddings of 2011 IEEE 10thInternational Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 142–149, 2011.

AUTHORS

Ali Abu Romman is a Digital Forensics Investigator at Jordan Anticorruption Commission since 2014. He received his MSc. degree in Information Systems Security and Digital Criminology from Princess Sumaya University for Technology (PSUT) in 2014. He received his BSc. degree in Software Engineering from University of Petra in 2008. His professional experience includes Networking, Telecommunications, ISP and IPTV. He holds several international certificates in Security and Networking including CEHv9, CCFP, CHFIv8, OSForensics, FTK, EnCase, CCNA, and CCNP and ToT. His research interests include network security and digital forensics.

Hussein Al-Bahadili is an associate professor at Faculty of Information Technology, Universityof Petra, Jordan. He received his B.Sc. in Engineering from the University of Baghdad in 1986.He received his M.Sc. and PhD degrees from University of London (Queen Mary College) in 1988 and 1991. His research interests include computer networks, routing protocols optimizations, parallel and distributed computing, cryptography and network security, and data compression.