A TRUst based Information Sharing Model (TRUISM) in MANET in the Presence of Uncertainty

Khalid Zaman Bijon*, Md Munirul Haque† and Ragib Hasan†
*Institute for Cyber Security & Department of Computer Science, University of Texas at San Antonio
† Department of Computer and Information Sciences, University of Alabama at Birmingham

***Abstract*—In the absence of centralized trusted authorities (CTA), security is one of the foremost concern in Mobile Ad-hoc Networks (MANET) as the network is open to attacks and unreliability in the presence of malicious nodes (devices). With increasing demand of interactions among nodes, trust based information sharing needs more stringent rules to ensure security in this pervasive computing scenario. In this paper, we present a novel multi-hop recommendation based trust management scheme (TRUISM). We adapt famous Dempster-Shafer theory that can efficiently combine recommendations from multiple devices in the presence of unreliable and malicious recommendations. A novel recommendation-routing protocol named 'buffering on-the-fly' has been introduced to reduce the number of recommendation traffic by storing trust values in intermediate nodes. TRUISM also provides a flexible behavioral model for trust computation where a node can prioritize recommendations based on its requirements. Evaluation result shows that our model not only performs well in the presence of contradictory recommendations but also ensures a faster and scalable trust based information sharing by reducing the overall packet flow in the system.**

***Keywords:*Trust, MANET, Dempster-Shafer, Recommendation**

I. Introduction

MANET is a network without centralized trusted authorities (CTA) and fixed infrastructure where mobile devices (nodes) are connected by wireless links. It is pervasive in nature as the network is self-configuring and quickly deployable and nodes in it are capable of self-directed operation without conducting a CTA. These criteria make this network a suitable choice over wired networks for dynamic environments such as rescue, battlefields, meeting rooms, etc. where temporarily established mobile applications are sufficient and fixed network infrastructure is not required or inconvenient due to several reasons including cost-effectiveness. However, this dynamic and openness nature incites unreliability and vulnerability as the nodes need to rely on neighbor-based routing to establish multi-hop communication where malicious nodes can join and leave the network frequently and dynamically.

In MANET, due to hardware limitations, nodes largely depend on each others resources. Hence, information sharing among nodes is very crucial in this network which also needs to preserve security by ensuring that a node shares a piece of information with other nodes only if they are legitimate. Conventionally, in MANET, this security is maintained by trust management in which each node implements trust as a metric to judge trustworthiness of other nodes.

Basically, this trust is a communication route where information flows from one node to another. For instance, if node A trusts node B, a route is created from A to B in which information can flow from A to B. In this network, sometimes a node (source) may need to discover multi-hop communication routes with another node (destination) in which destination lacks trust information about source. Conventionally, destination node develops this trust based on recommendations from the intermediate nodes of the multi-hop route. This unique property of MANET trust management is called neighbor-based routing where opinions/recommendations from neighbors largely influence the trust calculation process. Hence, integrity and validity of the recommendation is critically important since malicious nodes may present in a multi-hop route and any misguided recommendation from them may result insecure information flow. A node may also get contradictory recommendations from different routes for a single node. Therefore, trust management system should have proper mechanism to recognize the correct and valid one for ensuring secure information sharing.

Contributions: The contributions of the paper are as follows:
1) We incorporate a new aging (trust degradation) factor for recommendation progression named 'path reliability based' aging, that enables this model to prioritize recommendations based on the overall trustworthiness of the nodes in a specific recommendation route. This improves the verification process of the integrity and validity of a recommendation.
2) TRUISM is the very first model that uses a probabilistic approach ('Dempster-Shafer') [16], other than bayesian networks, for calculating trust from multiple contradictory recommendations. Different scenarios have been shown to elaborate the impact of introducing it in weeding out malicious/incorrect recommendations and making the correct decision.
3) TRUISM also introduces a new recommendation-routing technique called buffering on-the-fly for overall reduction of network traffic. We have shown how this simple mechanism helps to fill out the trust table maintained by each node in a faster way, eventually, improves the QoS for overall network traffic and ensures faster response to recommendation requests.
4) Finally, unlike conventional trust models, we introduce a novel property called trust willingness for the development of trust in MANET. Trust willingness resembles the human nature where one can wishfully prefer a person over others, motivated

by factors other than personal interactions or recommendations from neighbors. We believe, trust willingness is an inevitable property as it improves resistance against attacks like collusion attack by increasing ignorance to the recommendations that a node considers incorrect in a particular situation. We identifies influencing factors of trust willingness and devise a differential equation that calculates a suitable trust for the current situation.

The remainder of the paper is organized as follows. Section II discusses related work. The description of TRUISM and its components are given in section III. Section IV presents the recommendation trust model. Section V discusses Dempster-Shafer based recommendation combining process. Behavioral model is given in section VI. Finally, evaluation results of and conclusion are given in section VII and VIII respectively.

II. Related Work

There exists a very rich literature in trust management specially in dynamic and distributed network systems like pervasive computing, ad-hoc networks and also MANET. Dynamic Trust Management [4], Virtual Community Trust [1], Resurrecting Duckling model [17] and similar kind of efforts are not practical due to complex structure and properties, e.g. complex language to specify trust property, very long or hierarchical recommendation chain, etc. In pervasive computing, a number of trust models has been proposed, specially, by including recommendations in trust computation process. Recommendation trust, in general, can be identified as transitive trust proposed in Josang et al [11], [12] where they describe semantic criteria of transitive trust. However, in pervasive computing, most of the trust models suffer several limitations to handle multi-hop recommendation or combining them in the presence of malicious nodes. For instance, Pervasive Trust Management Model [2] proposes a recommendation protocol that has limitation to handle multi-hop recommendations. TrustBAC [6] has proposed a trust vector mechanism using 3 components namely experience, knowledge, and recommendation. However, they offer fix structure to combine them. REGRET [15] proposes a rating mechanism to formulate trust with time based aging.TRAVOS [18] has utilized direct and indirect trust to form overall trust. Also, trust-models [9], [20] are proposed to work in multi-hop recommendation environment. However, we show later, their only used 'distance based aging' is not sufficient to guarantee a recommendations trustworthiness in multi-hop scenarios. Also, a number of efforts has conducted for the trust management in MANET. Marti et al [14] proposes an approach to increase the throughput by detecting and avoiding the malicious nodes. A Bayesian statistics is proposed in [5] to filter recommendation from slander nodes Theodorakopoulos and Baras [19] estimate trust by probabilistic approach to combine multiple recommendation from neighbors. SORI [10] proposes an architecture for collaboration based reputation in which reputation is evaluated based on only monitored information of nodes. Jie et al [13] propose a Bayesian network developed to deal with probabilistic causality. Velloso et al [21] proposes maturity-based model to weight more recent experience over a older one.Balakrishnan et al [3] propose a subjective knowledge formulation for trust calculation.where they combine recommendation, direct trust and observed opinion in order to calculate the trust value.

Our model differs from these models in several accounts. Our proposed 'path reliability based' aging performs more realistically than those models only use 'distance based' and 'time based' aging.We also introduce 'Dempster-Shafer' based probabilistic model to combine recommendations which is a generalization of Bayesian model. Ensuring QoS of data availability is also crucial in this network [7] since nodes dependence with each-other is very high. In a trust based MANET system, besides data exchange nodes also exchange recommendation packets. Unlike other models, we analyze traffic situation for recommendation flows and minimize it using 'buffering on-the-fly' technique. Again, to the best of our knowledge, TRUISM is the first model that initiates recommendation packet by the service requestor (n_{sr}) instead of the service provider (n_{sp}). It follows the general idea that the node who wants a service should bear the load of it.

III. TRUISM

Trust is an attribute of a node that is a quantified representation of dependability/satisfaction metrics on other nodes in the network, and is generally developed based on interaction experiences with them at a particular time. We propose TRUISM in which trust is dynamically calculated before an interaction and also based on the outcome of the interaction the direct trust or trust confidence of a node on another node is updated. Similar to human network, our model also incorporates trust willingness property, that is a node's wishfulness to trust a node for sharing a particular information.

Fig 1 shows a typical trust network in MANET. Nodes are represented as circles and two nodes are connected by an arrow if there is a trust between them and direction of the arrow represents the trust of a node on another one. For example, the arrow from node A to B represents the trust of A on B. Hence, there is a communication route from A to B. The basic properties, sets, and functions of TRUISM are described below.

A. Basic Properties

TRUISM has the following basic properties of a trust model:

- **Context-dependent:** In TRUISM, trust is context dependent. Trust (T) of a node n_{sp} on another node n_{sr} for an information sharing in service context c_i, at time *t*, is not necessarily the same for another service context c_j.
- **Time based aging:** Without any further interaction, trust value of n_{sp} for n_{sr} decreases over time.
- **Distance based aging:** Recommendations from closer nodes are given more importance than the recommendations from the nodes which are further away.

Also, we introduce the following properties for TRUISM:

- **Path reliability based aging:** Recommendation from higher trusted neighbors should get more priority than lower trusted neighbors. According to this property, recommendation trust suffers from a higher rate of decrement in a path consisting of less reliable trust values compared

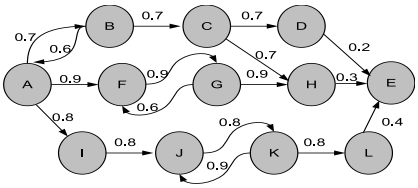


Fig. 1: *A typical trust network in pervasive environment*

to a path with higher trust values of same distance. In Fig 1, recommendation for H to A from path G→F→A gets more priority than recommendation from C→B→A.

- **Direct Trusted Neighbor (*DTN*)**: If any node n_{sp} trusts a node n_{sr}, n_{sr} is n_{sp}'s Direct Trusted Neighbor (*DTN*). Fig 1 shows that B, F, and I are *DTN*s of A.
- **Connected Neighbors (*CN*)**: If n_{sp} trusts n_{sr}, n_{sp} is n_{sr}'s Connected Neighbor (*CN*). Hence, *CN* is basically reverse of *DTN*. In Fig 1, A is a *CN* of nodes B, F, and I.

B. Basic Sets and Functions

This model has following basic sets and functions:

- N: A finite set of existing nodes in the network. A node in our model means any type of mobile devices such as PDAs, cell phones, etc.
- CService: A finite set of all existing service contexts.
- RService: N $\rightarrow 2^{\text{CService}}$. This function maps each node to a subset of CService that it received from other nodes.
- For each $n_{sp} \in N$,
  $\text{RSFrom}_{n_{sp}}$: CService $\rightarrow$ N where $\text{RSFrom}_{n_{sp}}$(service) = $\{n_i \mid n_i \in \text{N} \wedge \text{service} \in \text{RService}(n_{sp})\}$. This function gives the node from which n_{sp} received the service.
- For each $n_i \in$ N,
  T_{n_i}: N $\times \mathbb{R}_{\geq 0} \rightarrow$ [0.0, 1.0]. This gives the trust value of n_i for another node n_j at a particular time $\mathbb{R}_{\geq 0}$. The value is between 0.0 to 1.0, where a higher value implies more trust. In Fig 1, T_A(B,t) = 0.7 and T_A(F,t) = 0.9 which means A considers F more trustworthy than B. Also, 0.0 means complete distrust. If n_i does not have any interaction records with n_j, the trust for n_j is considered as 0.5, which indicates n_j is neither trusted nor distrusted. Also, existing nodes, in network, consider a newly joined node as neutral and assign a trust value 0.5 for it.
- dtnMapping: N $\rightarrow 2^{\text{N}}$; where dtnMapping(n_i) = $\{\text{n} \mid \text{n} \in \text{N} \wedge \text{T}_{n_i}(\text{n}) > 0.0\}$. This function maps each node n_i to a set of nodes for whom n_i has trust values. In other words, this function gives the *DTN*s of a node.
- cnMapping: N $\rightarrow 2^{\text{N}}$; where cnMapping(n_i) = $\{\text{n} \mid \text{n} \in \text{N} \wedge \text{T}_n(n_i) > 0.0\}$. This function maps each node n_i to a set of nodes having trust values for n_i. In other words, this function gives the *CN*s of a node.
- For each $n_{sp} \in$N and $n_i \in$dtnMapping(n_{sp}) there is a function, $Rec_{\text{n}_{sp},\text{n}_i}$: N $\rightarrow$ [0.0, 1.0]. This gives recommendation trust value, also referred as *Rec*, that n_{sp} gets from n_i for n_{sr}. Its calculation process is given in section IV.
- For each $n_{sp} \in$N, there is a function,
  $Rec_{\text{n}_{sp}}$: N $\rightarrow$ [0.0, 1.0]. This function gives combined *Rec*s that n_{sp} gets from its *DTN*s for n_{sr}.

IV. Recommendation Trust Model

Recommendation trust (*Rec*) is the suggested trust value that a node receives from its *DTN*s for another node that is willing to interact. Note that, unlike direct trust, calculation of *Rec* does not generally depends on one's direct interaction experiences with others rather it is some level of dependence on its neighbors. In a very dynamic environment like MANET, a node's behavior towards others might change very quickly. Thus, only personal interaction experiences may fall short to estimate actual trustworthiness. For instance, a node n_{sp} had positive trust for another node n_{sr} at time t. Afterwards, n_{sp} did not have further interactions with n_{sr} and, let say, at time $t+\Delta t$, n_{sp} receives a request from n_{sr}. Suppose, during this time span, n_{sr} starts behaving maliciously, e.g. by some malwares in device, which n_{sp}'s previous experience can not detect. However, if some neighbors of n_{sp} have some bad interaction experiences with n_{sr} during Δt, recommendations from them could help n_{sp} to identify n_{sr}'s actual trustworthiness. There are also other scenarios where recommendation should influence trust calculation, e.g. a very first interaction with a node without any previous interaction history in which n_{sp} could collect recommendations from its *DTN*s having interactions with n_{sr}.

In TRUISM, a *Rec* should flow from n_{sr} to n_{sp} and it might traverse multiple hops (nodes) to reach n_{sp}. A long multi-hop distance involves larger number of intermediate nodes that forwards recommendation towards n_{sp}. Lets say, in Fig 1, E needs to interact with A. In this case, the *Rec* for E traverses intermediate nodes B, C, D, F, G, H, I, J, K, and L in order to reach A. This higher number of recommendation flow can seriously disrupt the overall communication process of the network. In Section 4.3, we show that our proposed buffering on-the-fly minimizes this flow considerably in compare to other existing recommendation models in literature. Again, a n_{sp} might receive *Rec*s from its multiple *DTN*s and they may differ in opinions. Thus, uncertainty exists since any wrong and misleading recommendation, if considered, disrupts the overall security by flowing information to unauthorized and distrusted nodes. Dempster-Shapher theory [16] is a well-established process for combining evidence in the presence of uncertainty. We explore 'Dempster's rule of combination' in this situation and our extensive and diverse examples prove its appropriateness in combining multiple recommendations.

A. Recommendation Trust Progression Process

In TRUISM, when n_{sr} needs an information from n_{sp}, it simply multicasts the request to its *CN*s. Then these *CN*s forward the request along with their own recommendation trusts for n_{sr} to their *CN*s. This process is continued until the destination is reached or the request travels maximum hop-distance. Section VII shows a process to calculate the value of maximum hop-distance. In TRUISM a *DTN* is the inverse of a *CN*, thus, a node always gets recommendation from its *DTN*s. Consequently, this process avoids malicious recommendation from an unknown node by accepting recommendations only from *DTN*s. This is an unsolvable issue when recommendation propagates in opposite direction (from n_{sp} to n_{sr}) which

has been proposed in existing multi-hop recommendation models [8], [9]. In their propagation direction, n_{sp} receives the recommendation about n_{sr} from a node which is a *CN* of the n_{sr} but may not be a *DTN* of n_{sp}. Let say, in Fig 1, E needs an information from A. In TRUISM recommendations are initiated by *CN*s of E, i.e. H, L, and D, and it flows to A where A receives the recommendations from its *DTN*s B, F, and I. However, in other models, after getting the request, A requests recommendation for E from its *DTN*s B, F, and I. These *DTN*s then forwards the requests to their *DTN*s and, finally, A gets the recommendation from D, H, and L those are the *CN*s of E but not *DTN*s of A. Hence, A has to consider recommendations from unknown nodes.

We generate a general equation for the calculation of the recommendation trust. If a service provider n_{sp} gets a recommendation *Rec* from *DTN* n_i about a service requestor n_{sr} at time *t*, the recommendation trust $Rec_{\mathrm{n}_{sp},\mathrm{n}_i}(n_{sr})$ that n_{sp} actually considers for n_{sr} is estimated by the equation 1:

$$Rec_{n_{sp},n_i}(n_{sr}) = Rec \times \delta \quad (1)$$

Where,

- *Rec* is the recommendation trust value from n_i for n_{sr}
- $\delta = 1 - \frac{(1-T_{n_{sp}}(n_i,t))*\Psi}{10}$, is the aging factor (path reliability based)
- Ψ is the hop length from n_{sp} to n_{sr}

δ is the weighted factor that satisfies the 'path reliability based aging' property. Here, $T_{n_{sp}}(n_i, t)$ is the trust value of n_{sp} for n_i at time *t*. The value of δ ensures that the recommendation values from closer nodes will have more weight than distant nodes. Here closer means nodes with shorter distance and higher trust values. Section 4.3 shows that this factor improves the reliability of a long multi-hop recommendation.

B. Recommendation Trust Progression Algorithm

In this section, we discuss our developed algorithm for recommendation progression. This algorithm implements 'path reliability based aging' and buffering on-the-fly properties. In TRUISM, recommendation trust propagates as *RReq* packet and after receiving *RReq* the service provider node (n_{sp}) replies *RRes* to service requestor (n_{sr}) which is the status whether n_{sp} accepts the request or not. More specifically, n_{sr} first multicasts *RReq* packet and that propagates through the intermediate nodes and ultimately received by the n_{sp} and n_{sp} replies *RRes* to n_{sr}. We define the format of *RReq* and *RRes* as follows:

RReq = (ReqID, SP, SR, RecID, RT)

RRes = (ReqID, SP, SR, Status)

Here, *SP* = Service Provider ID, *SR* = Service Requestor ID, *ReqID* = Request ID, *RecID* = Recommender ID, *RT* = Recommendation Trust Value, *Status* = Accept or Reject.

We define algorithm 1 for *RReq* packet propagation. When an intermediate node which is not the service provider receives *RReq* packet, it calculates the recommendation trust using algorithm 1. In algorithm 1, line no. 5-6 are applied for the *CN*s of n_{sr} in which these *CN*s multicast their recommendations for n_{sr} to their *CN*s (line no. 15-16). Line no. 8-14 apply for nodes receiving *RReq*, but they are not *CN*s of n_{sr}. Line no. 8 calculates the recommendation using equation 1 which applies 'path reliability based' aging. Each nodes might get multiple such recommendations from different route. A timer (*t*) is being used because every node needs to wait for all possible useful recommendations that may come from different *DTN*s. During this time period, it combines all the received recommendation trusts and stores the combined value. Recommendation trust combining process is described in section V. This storing process, basically, implements the 'buffering on-the-fly' property. Finally, it sends the recommendation to all its *CN*s. Algorithm 2 shows the steps n_{sp} follows after receiving a service request. When n_{sp} receives a *RReq* packet, it starts a timer, similar to algorithm 1, for getting all possible recommendations from its *DTN*s. It also combines these recommendations and stores the combined value. Finally, n_{sp} replies *RRes* to n_{sr} with the status 'true' to notify that the request has been received.

Algorithm 1 *RReq Progression*

```
1: procedure UpdateAndForwardRReq
2:     RReq Packet = ReceivePacket()
3:     if Packet.SP is not CurrentNode then
4:         Rec = 0
5:         if Packet.SR ∈ dtnMapping(CurrentNode) then
6:             Rec = T_CurrentNode(Packet.SR)
7:         else
8:             Rec_CurrentNode,Packet.RecID(Packet.SR)
                 = Packet.RT × δ
9:             /* δ is calculated accordingly*/
10:            t = startTimer()
11:            while t do
12:                RReq newPacket = ReceivePacket()
13:                Rec_CurrentNode,newPacket.RecID(Packet.SR)
                     = newPacket.RT × δ
14:            end while
15:            Rec = Combine all recommendations
16:            Rec_CurrentNode(Packet.SR) = Rec
17:            for all nodes ∈ cnMapping(CurrentNode) do
18:                UNICAST(Packet.ReqID, Packet.SP,
                       Packet.SR, CurrentNode, Rec))
19:            end for
20:        end if
21:    end if
22: end procedure
```

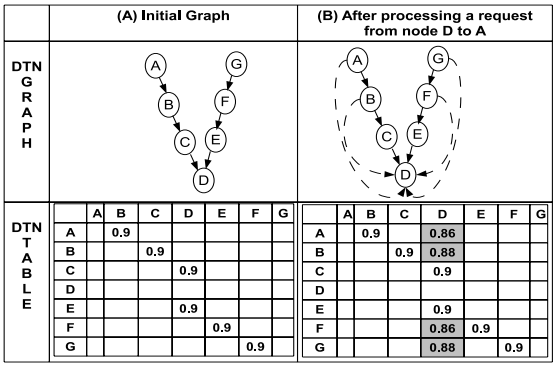
Algorithm 2 *RRes Progression*

```
1: procedure ReplyRReq
2:     RReq Packet = ReceivePacket()
3:     status = false
4:     if Packet.SP is CurrentNode then
5:         Rec = 0
6:         /*Same Code from line no. 9-14 of Algorithm 1*/
7:         Rec_CurrentNode(Packet.SR) = Rec
8:         status = true
9:         UNICAST(Packet.ReqID, Packet.SP,
               Packet.SR, status)
10:    end if
11: end procedure
```

C. Critical Issues

In this section, we explain several advantages of our proposed recommendation trust progression process over other existing multi-hop recommendation processes in literature.

1) Buffering on-the-fly: In TRUISM recommendation trust progression is a multicast process that originates from the *CN*s of the service requestor node n_{sr} and it progresses in multiple paths. During this progression, all intermediate nodes belongs to these paths receive recommendation trust and forward its recommendation to their *CN*s. The recommendation



	A	B	C	D	E	F	G
A		0.9					
B			0.9				
C				0.9			
D							
E				0.9			
F					0.9		
G						0.9	

	A	B	C	D	E	F	G
A		0.9		0.86			
B			0.9	0.88			
C				0.9			
D							
E				0.9			
F				0.86	0.9		
G				0.88		0.9	

Fig. 2: *Table fields update during recommendation trust progression*

trust value that every intermediate node receives is basically the recommendation value that it would receive when it gets a service request from n_{sr}. Therefore, each node updates the dtnMapping table and stores the trust value on-the-fly for future interaction and notify the n_{sr} so that it can update its cnMapping table accordingly. We call this process buffering on-the-fly. In Fig 2, a simple scenario is presented in which Fig 2(A) represents initial dtnMapping graph and a 2-dimensional table showing trust value of each *DTN*, e.g. node B is a *DTN* of node A where A has trust value 0.9 for B. Note that, in practice, such a single 2D table does not exist, rather, each row of this table separately belongs to the corresponding node. For instance, row 1 should belong to node A, row 2 to node B, etc. For convenience, we represent as one single 2D table. Now, if D needs a service from A, it sends the request to its connected neighbors C and E. As C or E is not the destination node, they further forward the request along with their recommendation trust for D to their *CN*s B and F. Then, B and F calculate their recommendations for D using equation 1 and forwards it to their *CN*s A and G respectively. Note that, B and F do not have any previous interaction with D. Therefore, besides forwarding to their *CN*s, they also store their calculated recommendation values for D. Consequently, in future, if D needs any service from them, it will directly send them the request. In this scenario, the service provider node A receives the recommendation for D, from its *DTN* B. By the time A receives the recommendation, B has already listed D as its *DTN* by updating its dtnMapping table, hence, D has become only two hops away from A (A→B→D), although the distance was 3 before the protocol was initiated. Thus, buffering on-the-fly ensures that the n_{sr} always becomes two hops away from n_{sp} regardless their initial distance. Therefore, the value of Ψ is always 2. In Fig 2(B), shaded fields of the table are updated during this recommendation trust progression.

2) Path Reliability Based Aging: In present literature, distance based aging is a known concept for multi-hop recommendation trust progression process [9] in which distance means hop length between nodes. A node that stays in shorter hop distance is more reliable than the longer ones and nodes in same hop distance is equally reliable. However, A node might not equally trust two nodes with same hop distance if their trust values are different. Suppose, A trusts C by 0.7 and B by 0.8 and both of them are from 1 hop distance from A.

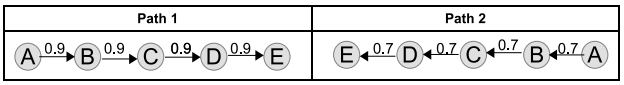


Fig. 3: *Paths with different trust values*

TABLE I: Estimated Recommendation Trust

	Without Aging	FTM	TRUISM
Path 1 (high reliable)	0.9	0.77	0.84
Path 2 (low reliable)	0.7	0.60	0.58

In this case, recommendation from B should be more trusted than C due to higher trust value of B although they are in same hop distance from A. Similarly, a recommendation from a path with equal hop length and higher trust value (more reliable) should be more trusted than a path with lower trust value (less reliable). Hence, aging factor should be such that decreases recommendation trust value at a higher rate in a less reliable path compared to a higher reliable path of equal length. In Fig 3, we show two simple *DTN* graphs with different trust values. One of them has trust value 0.9 in each link where as the other path has 0.7. Suppose, in both graphs, node E needs a service from node A, therefore, it sends the request through its *CN* (D). After traversing the path, the recommendation trust value is finally reached to A. Table I shows the calculated recommendation trust values for path 1 and path 2 using equation 1 that uses path reliability based aging and the values are 0.84 and 0.58 respectively. Table I also shows recommendation values for both paths if no aging factor is used. Also, it shows the calculated recommendation trust values using proposed approach in FTM [9] that uses only distance based aging. In FTM, the estimated values are 0.77 and 0.60 for path 1 and path 2 respectively.

V. Combining Recommendations in the Presence of Uncertainty

In this section, we describe our proposed Dempster-Shafer based formalism for combining multiple recommendations.

A. Dempster-Shafer Rule of Combination

The Dempster-Shafer theory [16] is viewed as a formula for reasoning under uncertainty. The part of the formula relevant to our work is the 'Dempster's rule of combination'. The basic functions of Dempster-Shafer formalism are follows:

1) **Frame of discernment (*H*):** If θ is a set of all possible mutually exclusive hypothesis about some problem domain, frame of discernment is the power set of θ.
   Formally: $H = 2^{\theta}$
2) **Basic probability assignment (*m*):** Basic probability assignment (m) defines a mapping of the frame of discernment(H) to an interval between 0 and 1. m of null set is 0 and sum of the elements in H is 1.
   Formally: m: $H \rightarrow [0,1]$, $m(\phi) = 0$, $\sum_{x \subseteq H} m(x) = 1$.
3) **Dempster's rule of combination:** This function combines multiple *bpa*s of the elements from set H. Lets, w is a subset of H and w is a combination of two different pairs x and y. Here, the combination of $m_1(x)$ and $m_2(y)$ is calculated using Dempster's rule which is $m_{12}(w)$,
   $$m_{12}(w) = \frac{\sum_{x \bigcap y = w} m1(x) \times m2(y)}{1 - \sum_{x \bigcap y = \emptyset} m1(x) \times m2(y)}$$

B. Combining Multiple Recommendations

We explain our methodology to combine multiple recommendation trusts (*Rec*) a node might get from its *DTN*s for a requestor node. A recommendation could be positive or negative where positive indicates that the requestor is reliable. We already mentioned that a recommendation is a fractional number from 0.0 to 1.0 and, for simplicity, we assume that a *Rec* greater or equal to 0.5 is a positive recommendation and less than 0.5 is a negative recommendation. Again, the type (positive or negative) of *Recs* that a node might receive might be all positive or all negative or a combination of both. For instance, the service provider n_{sp} gets two similar (both positive or negative) recommendations $Rec_{\mathrm{n}_{sp},\mathrm{n}_i}(n_{sr})$ and $Rec_{\mathrm{n}_{sp},\mathrm{n}_j}(n_{sr})$ for the service requestor n_{sr} from *DTN*s n_i and n_j respectively at time *t*. The probability that n_{sr} is trusted should be sufficient to identify that any of the *Recs* is correct. Let say, n_{sp} has trust value $\mathrm{T}_{n_{sp}}(n_i,\mathrm{t})$ and $\mathrm{T}_{n_{sp}}(n_j,\mathrm{t})$ for n_i and n_j. Equation 2 calculates the probability that at least any of them is correct,

$$P_{correctness} = 1 - ((1 - T_{n_{sp}}(n_i,t)) \times (1 - T_{n_{sp}}(n_j,t))) \quad (2)$$

Here, if $\mathrm{T}_{n_{sp}}(n_i,\mathrm{t})$ and $\mathrm{T}_{n_{sp}}(n_j,\mathrm{t})$ are both 0, i.e., both n_i and n_j are untrustworthy, $P_{correctness}$ becomes 0 which indicates none of the *Recs* is correct. While, $P_{correctness}$ becomes 1 when $\mathrm{T}_{n_{sp}}(n_i,\mathrm{t})$ is 0 but $\mathrm{T}_{n_{sp}}(n_j,\mathrm{t})$ is 1 since one of them is correct. Now, the combined recommendation trust for n_{sr} is,

$$Rec_{\mathrm{n}_{sp}}(n_{sr}) = \frac{Rec_{\mathrm{n}_{sp},\mathrm{n}_i}(n_{sr}) + Rec_{\mathrm{n}_{sp},\mathrm{n}_j}(n_{sr})}{2} \times P_{correctness} \quad (3)$$

Both equations could be generalized as follows for combining n number of similar type of recommendation trusts,

$$P_{correctness} = 1 - ((1 - T_{n_{sp}}(n_i,t)) \times (1 - T_{n_{sp}}(n_j,t)) \times \ldots \times (1 - T_{n_{sp}}(n_n,t))) \quad (4)$$

$$Rec_{\mathrm{n}_{sp}}(n_{sr}) = \frac{\sum_{1 \le i \le n} Rec_{\mathrm{n}_{sp},\mathrm{n}_i}(n_{sr})}{n} \times P_{correctness} \quad (5)$$

Now, let us consider a scenario where $Rec_{\mathrm{n}_{sp},\mathrm{n}_i}(n_{sr})$ and $Rec_{\mathrm{n}_{sp},\mathrm{n}_j}(n_{sr})$ are positive and negative respectively. According to Dempster-Shafer formalization we calculate,

$$m1(\{positive\}) = T_{n_{sp}}(n_i,t), \quad m1(H) = 1 - T_{n_{sp}}(n_i,t) \quad (6)$$

$$m2(\{negative\}) = T_{n_{sp}}(n_j,t), \quad m2(H) = 1 - T_{n_{sp}}(n_j,t) \quad (7)$$

The following equation calculates the probability ($P_{positive}$) that the positive recommendation is correct,

$$P_{positive} = \frac{m1(\{positive\}) \times m2(\{H\})}{1 - m1(\{positive\}) \times m2(\{negative\})} \quad (8)$$

Similarly for negative recommendation,

$$P_{negative} = \frac{m1(\{H\}) \times m2(\{negative\})}{1 - m1(\{positive\}) \times m2(\{negative\})} \quad (9)$$

Finally, the positive recommendation is considered if $\mathrm{P}_{positive}$ is greater than $\mathrm{P}_{negative}$ and the recommendation value is,

$$Rec_{\mathrm{n}_{sp}}(n_{sr}) = Rec_{\mathrm{n}_{sp},\mathrm{n}_i}(n_{sr}) * m1(\{positive\}) \quad (10)$$

Otherwise, the negative recommendation is considered,

$$Rec_{\mathrm{n}_{sp}}(n_{sr}) = Rec_{\mathrm{n}_{sp},\mathrm{n}_j}(n_{sr}) * m2(\{negative\}) \quad (11)$$

Note that, these two equations ensure that only positive or negative recommendations are considered for recommendation trust calculation if n_{sp} believes, by comparing $\mathrm{P}_{positive}$ and $\mathrm{P}_{negative}$, that the recommendation is positive or negative respectively. In case of multiple positive and negative recommendations, equation 4 is used to calculate $P_{correctness}$*(positive)* and $P_{correctness}$*(negative)* and equation 5 for separately combining positive and negative recommendations. Then, $\mathrm{T}_{n_{sp}}(n_i,\mathrm{t})$ and $\mathrm{T}_{n_{sp}}(n_j,\mathrm{t})$ are replaced by $P_{correctness}$*(positive)* and $P_{correctness}$*(negative)* in equation 6 and 7. Finally, equation 10 or 11 calculates overall recommendation trust value.

TABLE II: Different Scenarios of Recommendation Trust

Node \ scenario	1	2	3	4	5	6	7
Node1(0.55)	0.95	0.45	0.85	N/A	0.4	0.92	0.9
Node2(0.60)	0.95	0.45	0.85	N/A	0.45	0.7	0.4
Node3(0.65)	0.95	0.45	0.85	N/A	0.3	0.6	0.7
Node4(0.70)	0.95	0.45	N/A	0.85	0.8	0.4	0.35
Node5(0.75)	0.95	0.45	N/A	0.85	0.7	0.45	0.6
Node6(0.80)	0.95	0.45	N/A	0.85	0.6	0.3	0.48

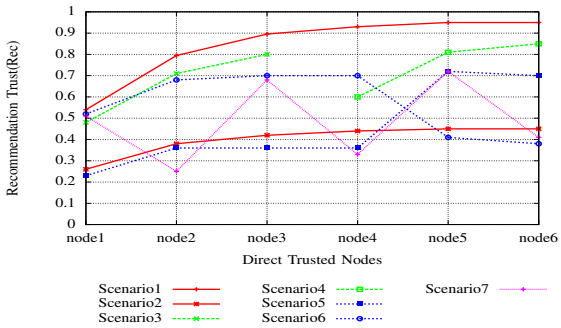


Fig. 4: *Different scenarios of recommendation trusts combination*

C. Performance Evaluation

We analyze this process performs in different scenarios. Let's say n_{sp} has 6 different *DTN*s (*node1-6*). In Table II, the n_{sp}'s trust value for each of them (column 1) and the recommendations from each of them in different interaction scenarios for a particular n_{sr} (columns 2-8).

In scenario 1 and 2, all *DTN*s give same opinion that is positive and negative respectively and the recommendation values are also equal for each scenarios. In Fig 4 we show how recommendation combination process gradually combines recommendations. In scenario 1, the recommendation value that n_{sp} gets from *node 1* is 0.5415 (using equation 1). Subsequently, combining recommendation of *node 2*, using equation 2 and 3, it becomes 0.79477 and finally combining all six recommendations, using equation 4 and 5, it is 0.9498. Similarly, in scenario 2, when only *node1* is considered the value of overall recommendation is 0.2565 and after combining all six recommendation it becomes 0.4499. It should be noted that similar kind of recommendations develop a strong opinion, thus, combined recommendation is close to their average value.

In scenario 3 and 4, same opinions from higher trusted *DTN*s create stronger opinion and higher combined recommendation trust value than lower trusted *DTN*s. In scenario 3, after combining 3 recommendations, using equation 4 and 5, from *node 1, 2 and 3* the value of *Rec* develops to 0.8013. While, in scenario 4, the the value of *Rec* is 0.8480. That is because *node 4, 5 and 6* are more trusted than *1, 2 and 3*.

In scenario 5 and 6, we show that it can identify and discard weak opinions if there is other strong opposite opinions. In scenario 5, *node 1-3* give negative recommendations with combined value of *Rec* is 0.3614. Then, a positive recommendation from *node 4* does not change the overall opinion as the formula identifies that combined value of *node 1-3* is stronger than *node 4* although individually *node 4* is more trusted. Eventually, positive recommendations from *node 5 and 6* change the situation as they are combinedly more trusted than *node 1-3* and combined recommendation is considered as positive. Similar situation is happened in scenario 6.

Finally, **scenario 7** shows how the combined recommendation trust changes when different node gives different opinion.

VI. The Behavioral Model

We present our proposed behavioral model. This is a generalized behavioral model that identifies many useful properties that influence trust computation in pervasive environment and provides a flexible mechanism by which a node can compute trust prioritizing these properties based on its requirements.

In our model, a node (n_{sp}) might trust another node (n_{sr}) with a trust value $T_{n_{sp}}(n_{sr},t)$ which is gained from the experiences of their direct interactions, e.g. request or provide a service, those happened till time *t*. $T_{n_{sp}}(n_{sr},t)$ is also called direct trust of n_{sp} on n_{sr} as it is estimated solely from their direct interactions. In TRUISM, we call it trust confidence. Let say, a node n_{sp} gets a service request from another node n_{sr} at time $t+\Delta t$. n_{sp} decides how much it relies on its own confidence that it gained by direct interaction experiences with n_{sr} till time *t*. We introduce a confidence factor β whose value ranges from 0.0 to 1.0 where higher value means increased reliability on gained confidence from own previous experiences. Equation 12 updates the trust confidence of n_{sp} on n_{sr}:

$$T_{n_{sp}}(n_{sr}, t+\Delta t) = T_{n_{sp}}(n_{sr}, t) + ((1-\beta) \times T_{n_{sp},W}(n_{sr}, t+\Delta t)) * \Delta t \quad (12)$$

Here, $(1-\beta) \times T_{n_{sp},W}(n_{sr}, t+\Delta t)$ determines the amount of confidence that should change at time $t+\Delta t$ from *t*. Note that, higher value of β allows small changes of $T_{n_{sp}}(n_{sr},t)$ at $t+\Delta t$. In equation 12, $T_{n_{sp},W}(n_{sr},t+\Delta t)$ is trust-willingness of n_{sp} on n_{sr} at time $t+\Delta t$. We define trust-willingness as a node's personalized attribute that determines how much a node wants to trust another node at a particular time based on several factors other than trust confidence. These factors do not help in building confidence on other nodes rather they influence a node's willingness to trust these nodes.We identify the following factors that should impact trust willingness:

Context Dependent Relative Trust: When n_{sp} gets a service request from n_{sr} in a particular context, the context dependent relative trust is a comparison of the trust confidence on n_{sr} with respect to other nodes with whom n_{sp} already has interactions in same service context. The equation for it is as follows,

$$T_{n_{sp},R}(n_{sr}, t) = \frac{T_{n_{sp}}(n_{sr}, t)}{\sum T_{n_{sp}}(n_i, t)/\#\ of\ nodes} \quad (13)$$

Here the denominator is the average trust confidence of the nodes already having direct interaction with n_{sp} on same context. We intend that a node with greater than the average should create positive willingness or negative otherwise. To this end, $T_{n_{sp},R}(n_{sr},t)$ - 1 is used in equation 15.

Recommendation from Neighbors: We believe positive recommendation from *DTN*s should influence trust-willingness positively. Sec V showed the process of calculating recommendation trust $Rec_{n_{sp}}(n_{sr})$ for a service requestor node n_{sr}. In equation 15, we use $Rec_{n_{sp}}(n_{sr})$ in which a positive recommendation should positively influence the trust-willingness and negative recommendation negatively. Here, a $Rec_{n_{sp}}(n_{sr})$ greater than or equal to 0.5 is a positive recommendation.

Dependence on Service Requestor: Similar to the human-interaction, a node's high dependency on another node might significantly increase trust-willingness disregarding other factors involve in information sharing in pervasive environment. For instance, a subordinate employee might highly depend for resource on supervisor's device. Thus, a request from that device gets higher priority. We call this property "dependence on service Requestor" and devise the following equation,

$$T_{n_{sp},D}(n_{sr}, t) = \frac{\sum_{service \in RService(n_{sp}) \wedge RSFrom_{n_{sp}}(service)=n_{sr}} sensitivity(service)}{\sum_{service \in RService(n_{sp})} sensitivity(service)} \quad (14)$$

In this equation, $T_{n_{sp},D}(n_{sr},t)$ is measured based on the relative weight of received services from Requestor node n_{sr} over the weights of all services the service provider previously received at time *t*. A higher value of $T_{n_{sp},D}(n_{sr},t)$ means more dependence on the requestor n_{sr}.

These three properties are combined together in following equation in order to calculate trust-willingness,

$$T_{n_{sp},W}(n_{sr}, t) = Max(1, \alpha \times (T_{n_{sp},R}(n_{sr}, t) - 1) + \gamma \times (Rec_{n_{sp}}(n_{sr}) - 1/2) + \phi \times T_{n_{sp},D}(n_{sr}, t)) \quad (15)$$

Here, α, γ and ϕ are weighted factors with value from 0.0 to 1.0 and the highest value of $T_{n_{sp},W}(n_{sr},t)$ can be 1.

Finally, time based aging property is applied by *Trust Decay*(λ) which is the decreasing rate of the confidence of a node on another when there is an interaction gap. λ varies from 0.0 to 1.0 and is applied by the following equation,

$$T_{n_{sp}}(n_{sr}, t+\Delta t) = T_{n_{sp}}(n_{sr}, t) - \lambda \times T_{n_{sp}}(n_{sr}, t) * \Delta t \quad (16)$$

By combining equation 12 and 16 we get equation 17:

$$T_{n_{sp}}(n_{sr}, t+\Delta t) = T_{n_{sp}}(n_{sr}, t) + ((1-\beta) \times T_{n_{sp},W}(n_{sr}, t+\Delta t) - \lambda \times T_{n_{sp}}(n_{sr}, t)) * \Delta t \quad (17)$$

Equation 18 reforms it to differential equation that estimates the variation-rate of trust confidence $t+\Delta t$ from *t*,

$$\frac{dT_i(t)}{dt} = (1-\beta) \times T_{n_{sp},W}(n_{sr}, t+\Delta t) - \lambda \times T_{n_{sp}}(n_{sr}, t) \quad (18)$$

VII. Simulation and Analysis

We simulated TRUISM in OMNet++ simulator to evaluate several issues regarding performance in pervasive environment.

A. Buffering on-the-fly: For demonstrating the impact of 'buffering on-the-fly', we have simulated an environment with increasing number of nodes (10 to 30 with a spacing of 5 nodes). This is a completely random scenario where nodes are interacting with one another. We have recorded the total number of table entries of each node filled by running FTM and TRUISM separately for 1 min. Since even a recommendation

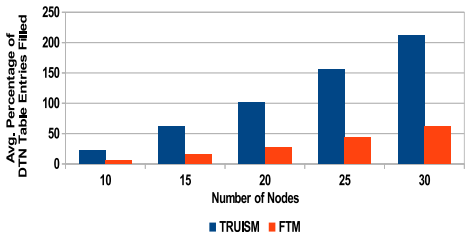


Fig. 5: *Percentage of table entries filled with varying number of nodes*

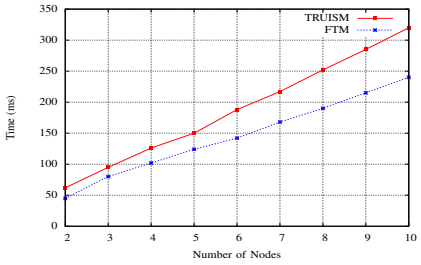


Fig. 6: *Recommendation protocols completion time for varying hop lengths*

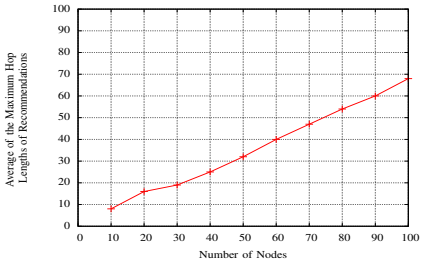


Fig. 7: *Average of recommendation hop lengths for varying nodes*

packet of path length 10 requires only 330 ms, we believed 1 min should be enough to capture reasonable amount of interactions. We have run each simulated environment for 10 times to find the average percentage of dtnMapping table entries filled in both the protocols. As shown in Fig 5, our model has a significantly higher performance in filling the trust table.

B. Timing Requirement: One of our objectives was to measure and compare the timing overhead occurring due to the introduction of buffering on-the-fly technique. For simulation, we first put 3 nodes: service provider (A), service requestor (B), and one intermediate node (C) with A does not have any direct trust value for B. When B requests A for a service, it creates recommendation path of length 2 (B→C→A) . We gradually increased the number of intermediate nodes to create path lengths up to 10. For each path length, the simulation is executed 10 times for both TRUISM and FTM and the average time required for getting recommendations along these paths is recorded. As expected, TRUISM requires little extra time compared to FTM (Fig 6). But, considering the advantages provided by the buffering on-the-fly technique, this little trade off in time can be considered as a huge gain. For example, consider the path of hop length 10 (B→C→D→E→F→G→H→I→J→K→A). Now let's say B requests A to D for a service sequentially. In FTM, the total time required for this whole process is, 240+220+190+170+140+120+100+ 80+50 = 1310 ms. This timing requirements is calculated based on results shown in Fig 6. Here, the path length from B to A is 10 and the average time for that is 240 ms. But for TRUISM the total time is only, 240+30+30+30+30+30+30+ 30+30 = 480 ms. Here, 30 ms denotes the communication time between any 2 directly connected nodes. Note that, when B makes the request to A, TRUISM updates dtnMapping table of intermediate nodes (here C-K) with the recommendation for B. TRUISM takes more time than FTM for a single recommendation, while outperforming FTM over the long run.

VIII. Conclusion

In this paper, a trust model is proposed for secure information sharing in MANET. To the best of our knowledge, this is the first model that provides a feature of multi-hop recommendation where every valid recommendation comes only from *DTN*s. This model also reduces traffic in a great volume by updating multiple trust fields of intermediate nodes. It also introduces the concept of path reliability based aging. It also successfully reflects uncertainty while combining multiple recommendations by adapting Dempster-Shafer theory. Finally, it adapts human like behavioral model by introducing 'trust-willingness' which is a node's wishfulness to trust another node in the system.

References

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proc. of the 33rd Annual Hawaii Int. Conf. on System Sciences*, 2000.

[2] F. Almenárez et al. PTM: A pervasive trust management model for dynamic open environments. In *Proc. of the 1st Workshop on Pervasive Security, Privacy and Trust*, pages 1–8, 2004.

[3] V. Balakrishnan, V. Varadharajan, and U. Tupakula. Subjective logic based trust model for mobile ad hoc networks. In *Proc. of SecureComm*, pages 30:1–30:11. ACM, 2008.

[4] M. Blaze et al. Decentralized trust management. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 164–173, 1996.

[5] S. Buchegger et al. The effect of rumor spreading in reputation systems for mobile ad-hoc networks. In *WiOpt'03*, 2003.

[6] S. Chakraborty and I. Ray. TrustBAC: Integrating trust relationships into the rbac model for access control in open systems. In *Proc. of the ACM Symposium on Access Control Models and Technologies*, 2006.

[7] J.-H. Cho et al. A survey on trust management for mobile ad hoc networks. *Communications Surveys Tutorials, IEEE*, 13(4), 2011.

[8] R. Falcone and C. Castelfranchi. Trust dynamics: How trust is influenced by direct experiences and by trust itself. In *Proc. of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[9] M. Haque and S. I. Ahamed. An omnipresent formal trust model (FTM) for pervasive computing environment. In *Proc. of the 31st Annual International Computer Software and Applications Conference*, 2007.

[10] Q. He, D. Wu, and P. Khosla. Sori: a secure and objective reputation-based incentive scheme for ad-hoc networks. In *Wireless Communications and Networking Conference*, volume 2, pages 825–830 Vol.2, 2004.

[11] A. Jøsang et al. Trust network analysis with subjective logic. In *Proc. of the 29th Australasian Computer Science Conference*, pages 85–94, 2006.

[12] A. Jøsang and S. Pope. Semantic constraints for trust transitivity. In *Prog. of the 2nd Asia-Pacific Conf. on Conceptual modelling*, 2005.

[13] J. Li et al. Future trust management framework for mobile ad hoc networks. *Communications Magazine, IEEE*, 46(4):108–114, 2008.

[14] S. Marti et al. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. of the 6th Annual Int. Conf. on Mobile computing and networking*, volume 6, pages 255–265, 2000.

[15] J. Sabater and C. Sierra. REGRET: reputation in gregarious societies. In *Proc. of the 5th Int. Conf. on Autonomous Agents*, pages 194–195, 2001.

[16] G. Shafer. A mathematical theory of evidence. Princeton University Press, 1976.

[17] F. Stajano and R. Anderson. The resurrecting duckling: security issues for ubiquitous computing. *IEEE Computer*, 35(4):22 –26, 2002.

[18] W. Teacy et al. Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model. In *Proc. of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, 2005.

[19] G. Theodorakopoulos and J. Baras. On trust models and trust evaluation metrics for ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):318–328, 2006.

[20] G. Theodorakopoulos and J. S. Baras. On trust models and trust evaluation metrics for ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):318–328, 2006.

[21] P. B. Velloso et al. Trust management in mobile ad hoc networks using a scalable maturity-based model. *IEEE Transactions on Network and Service Management*, 7(3):172–185, 2010.