Performance evaluation of a self-evolving trust building framework

Giannis F. Marias, Vassileios Tsetsos, Odysseas Sekkas, Panagiotis Georgiadis
Dept. of Informatics and Telecommunications, University of Athens,
Panepistimiopolis, Ilissia, Greece, GR15784
marias@mm.di.uoa.gr, {b.tsetsos, o.sekkas, georgiad}@di.uoa.gr

Abstract

A self-evolving reputation scheme for trust establishment in distributed peer networks is presented and evaluated. The framework, called Ad-hoc Trust Framework (ATF), incorporates subjective behavior of end-users, direct observations of behaviors, recommendations, and history of evidences to assess the trustworthiness of peer entities. It considers several idiosyncrasies of the wireless self-organized networks, such as lack of computational resources. ATF is associated with a generic model for the evaluation of the trustworthiness of adjacent or distant nodes. It relies on a sophisticated reputation method, called TrustSpan, to contact only trusted peers for recommendations, and, thus, it minimizes communication costs for trust building, accelerating the trust evolution process. To evaluate the performance of the ATF framework we have deployed a large number of simulation scenarios. The performance assessment results show that ATF achieves to rapidly identify selfish nodes with high accuracy, and with relatively low communication costs.

1. Introduction

In Mobile Ad hoc and self-organised networks, (MANETs), mobile nodes are continuously associated or disassociated with each other, according to their topological arrangement. The incentive of a newcomer node that desires to join the network is to offer functionality (e.g., routing and packet forwarding) to the existing nodes, which, in their turn, reciprocate this by offering equivalent functionality to new members. This cooperation enforcement builds up the trust that should be assembled by the nodes, and which is essential for the steady-state operation of a MANET: adjacent nodes build up trust with time to enforce cooperation, and improve the security, connectivity and quality of service provided by the network. On the other hand, the value of the self-evolving trust diminishes when adjacent nodes become distant due to mobility, since several unreliable nodes might intermediate on the communication path.

Thus, the self-established trust between any two nodes for the routing and packet forwarding services might be lost with time, influencing network's performance. Moreover, nodes behave passionately rather than rationally [1]. Selfish, malicious and hacker nodes may initially accept the cooperation enforcement principles in order to be associated with a mobile ad-hoc network, but their real intentions might be guileful. A *selfish node* is disinclined to spend its resources (e.g., battery) in order to serve network's operations and to maximize social welfare (e.g., forward packets not destined for it), but it demands the execution of network tasks that maximize its own profit (e.g., asks for the delivery of packets originated from or destined for it). *Malicious nodes* have as a primary goal to damage network's operation, rather than to avoid the depletion of their resources (e.g., battery). Flooding and sleep deprivation torture [2] are techniques commonly used by malicious nodes. *Hacker nodes'* objective is the interception of the information exchanged between the network nodes. Hacker intents are materialized through sinkhole and wormhole, impersonation and Sybil attacks [3].

Self-evolving, reputation-based, schemes are considered suitable for trust establishment in spontaneous networks, such as MANETs, where key or certificate distribution centers are ephemerally present and computation resources (e.g., energy, memory) are scarce. Such schemes are based on the determination of the trustworthiness of nodes, regarding their offered functionality. A primary goal of rational nodes is to cooperate in order to avoid, or even mutually isolate, notorious nodes (i.e., selfish, malicious or hacker) from the routine network operations. Such cooperation requires the exchange of recommendations, and the identification of trusted recommenders. Additional goals include the reactive minimization of the effects introduced in normal operations by the notorious nodes, or the pro-

active cooperation enforcement of selfish nodes through credit-based mechanisms.

In this paper a generic, distributed, framework for self-evolving trust establishment, named *Ad-hoc Trust Framework (ATF)*, originally proposed in [16], is further discussed and evaluated. *ATF* incorporates self-evidences, recommendations, subjective judgment and historical evidences to continuously evaluate the trust level of peers. To capture such semantics, *ATF* is armed with a novel trust computation model. The model consolidates user's natural behavior, through a *Trust Policy*.

ATF does not use any hard trust building method, such as symmetric or public cryptography, or message authentications schemes. Thus, it avoids complex computations and the expenditure of resources (power, CPU and memory). In that sense, *ATF* is usable in different types of distributed, or peer systems (ad hoc networks, pervasive computing devices, autonomic systems), although here it is primarily exploited in the context of ad hoc networking.

The structure of the paper is as follows: First we present the proposed *ATF* architecture and its functional modules. In Section 4 we present the functionality of the trust building module. In Section 5 we describe how the trust model incorporates behavioural elements. Subsequently, in Section 6 we discuss the novel *TrustSpan* mechanism that assists *ATF* to rapidly identify and use recommendations provided only by trusted recommenders. In section 7 we define the simulation environment and provide the performance evaluation of the *ATF* framework. Finally, we discuss the open issues concerning *ATF*, and provide some directions for further research and assessment.

2. The ATF Architecture

ATF uses a layered architecture and consists of the following components: *Trust Sensors (TS), Trust Builder (TB)* and *Reputation Manager (RM)*. The proposed framework is fully distributed. Every node hosts these components and provides a number of routine functions (services), such as packet forwarding, routing, etc. Moreover, every node implements a special function, called *Recommendation Function (RF*). This is a simple service that provides recommendations to third parties, upon request.

ATF follows [8] for the definition of the reputation of a node's function: *Reputation = {NodeId, Function, Trust Value}*. Thus, the reputation of a function *f* of node *n* is defined as $R(n,f)= \{n, f, TV_{n,f}\}$, where $TV_{n,f}$, is the *Trust Value (TV)* for the function *f* of node *n*.

For the rest of the paper we use the term *detector* to denote a node that *directly* monitors the behavior of another node's functions, called a *target*. In such a case the detector captures *Direct Evidence (DE)* about the trustworthiness of a particular function of the *target*. A *requestor* is a node that asks for recommendations. A *recommender* issues recommendation responses (upon request). *Adjacent* are the nodes that are in the coverage (one-hop) area of each other.

In general, a trust building mechanism could be laid out based on two diverse architectural directions. The first relies on an on-demand, and the second on an event-driven reputation system. The difference between these two approaches lies in the way that nodes are being informed for changes in *Trust Values* (*TVs*). The *ATF* architecture is capable of supporting on-demand recommendations.

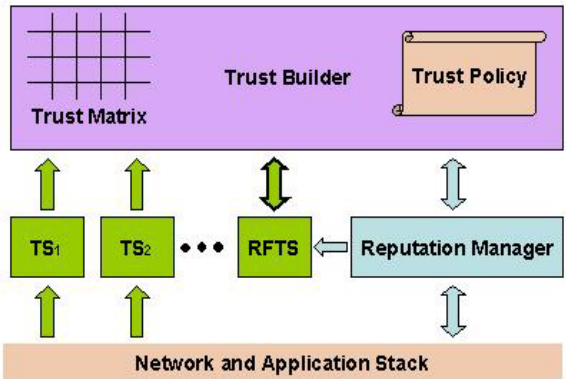

Figure 1. ATF architecture

2.1 ATF Architecture components

The architecture consists of the following modules (see Figure 1):

Trust Sensors (TS). The majority of the proposed reputation systems agree that the most significant factor for trust building is the direct experience (or direct evidence). In the SECURE project [9], this evidence monitoring is performed independently by each node through a "Monitor" [10], which logs every activity in an "Evidence Store". In [4] a Watchdog mechanism is proposed as an observation device for routing behaviour of nodes participating in ad hoc networks. In *ATF,* for every function offered by an *adjacent* node there is a *Trust Sensor* that monitors its execution/operation. A *TS* operates similarly to any other common sensor: translates a (physical) phenomenon in a machine interpretable form. In our case this phenomenon is the trustworthiness of a node with respect to its quality of service. A *TS* monitors the behaviour of another node (through real time measurements or statistical analysis of logs) and compares it to a predefined reference atti-

tude, (i.e. expected functionality). In that sense, the *ATF* scheme uses *TSs* to assist a node to define the credibility of other collaborating parties. The proposed generic mechanism requires the quantification of the difference between the observations and the expected functionality. An intuitive and easy-to-implement approach to this issue is the categorization of observations to *Successes* and *Failures* relating to the expected functionality [11]. A *TS* maps the captured evidence (i.e., observations) to a numerical value and forwards this value to the *Trust Builder* for further trust computation.

Trust Builder (TB). This component computes the *TV* of other nodes' functions. *TVs* are stored in a *Trust Matrix,* which *Trust Builder* maintains and updates, and are used when there is an intention for cooperation or interaction with other nodes. A TV value is distinct for each discrete function per node. A node, for example, may be trustworthy to perform packet forwarding, but unreliable to contribute on routing. *TV* computation depends on several factors, such as direct evidence, recommendations, historical data and subjective criteria. The weight that each factor contributes on the evaluation of TV is further explained in [16]. Each node follows its own *Trust Policy*, which specifies the user's subjectivity (e.g., distrustful, deceivable) and the weights.

Reputation Manager (RM). The *RM's* main role is to provide recommendations from third parties to the *TB* in order for the latter to compute the required *TVs*. In the ATF on-demand scheme, *TB* requests recommendations for a target node when it has inadequate information for it. Thereafter, *RM* selects the nodes to contact (recommenders) in order to obtain requested values. These should be as trusted as possible and close enough so as to limit communications overheads. For that purpose, the *RM* takes into account the trust values of the recommenders' Recommendation Function and their distance (number of hops). When a recommender receives a request for recommendation, its RM contacts *TB* and obtains the *TV* for the requested function of target node, if any. Next, the recommender returns this value to the requestor node. *TB* uses the recommendations collected by the *RM* and the latter uses *TVs* already computed in order to inform other nodes.

Recommendation Function Trust Sensor (RFTS). This special-purposed trust sensor evaluates the trustworthiness of a node regarding its recommendation function. *RFTS,* as any other *TS*, categorizes a direct observation as *Success* or *Failure*. The *RM* of a detector asks from recommenders the recommendations that correspond to a specific function of a target. A recommendation is returned back only when the recommender has adequate *DE* about the target, so as to reduce rumour spreading. These values are then passed to the *TB* and the *RFTS* of requestor. *RFTS* stores in a matrix the values that received from each recommender. Whenever the detector node completes a direct interaction with the target node, the *TS* for the specific function returns *Success* or *Failure* to the *TB* according to its observation. The *TB* then builds the *DE* for the target node and returns this value to the *RFTS* for the evaluation of the recommendation function of the recommenders. *RFTS* compares this *DE* with TV_{RF} stored for each recommender. If the two values do not have considerable deviation (expressed as a threshold in Trust Policy), then *RFTS* returns *Success* to the *TB*, else returns *Failure.* A detector node (i.e., evaluator) should be able to check the *TV* of the recommendation function of those nodes offering recommendations (recommenders). Thus, in [16] we have proposed a testing mechanism that estimates and updates the trustworthiness of recommenders.

2.2 Related work

Similar trust management architectures are presented in [1] and [5]. Jøsang in [1] proposes a recursive hierarchy of trust that is based on historical knowledge; the derived trust can be used for further trust derivation. The underlying trust, to a certain extent, not only reflects security (i.e., protection against malicious attacks), but also other aspects of dependability as long as these aspects contribute to increasing the final trust. In [5] the authors consider an architecture that consists of the Monitor, the Reputation System, the Path Manager, and the Trust Manager. The Monitor detects damaging behavior (intrusion, DoS, etc). The Trust Manager deals with incoming and outgoing ALARM messages, sent by a node to warn others of malicious nodes. The Reputation System manages a table consisting of entries for nodes and their rating. The rating is modified only when there is sufficient evidence of malicious behavior. The Path Manager ranks paths according to security metrics (e.g. reputation of the nodes in the path) and deletes paths containing malicious nodes.

3. Trust Computation in Trust Builder

3.1. The Qualitative Perspective

ATF incorporates several user-defined and time-dependent weights. Time-dependence is important, since it allows the modelling of temporal trust strategies, which can be followed by the participating nodes.

Additionally, the weights are defined separately for each node in its *Trust Policy (TP)*. For the *ATF* scheme, time is treated as a discrete sequence of *direct* interactions between the nodes. Thus, time elapses in a different rate for every separate node. We use only direct interactions as a time reference, since they are generally regarded more important than the indirect ones (recommendation exchanging) for the trust building process. Moreover, interactions are categorized to positive and negative (according to the *Success/Failure* model incorporated by each *TS*) to enable flexible computation of trust.

The majority of the trust computation approaches acknowledge that two main components should be taken into consideration: the **Direct Evidence** (*DE*) and the **Recommendations** (*RECs*) from third parties. The *DE* is calculated from the *TSs'* feedbacks**,** as described in previous section, and is useful for evaluation of adjacent nodes' functionality. *RECs* are communicated between the entities participating in the trust network, according to a reputation dissemination protocol, implemented in *RM*.

Many socio-cognitive approaches for trust, e.g., [12], dictate that trust computation should also include a **subjective component**. Each node has a unique, subjective, way to trust others. Here, we adopt this approach, and a separate *Subjective-factor* component (*SUB*) is introduced in the trust computation model. This component is time-dependent, as well, so as to enable time-variant trusting behaviour of nodes (e.g., a node may want to trust a newcomer node only up to a point, until it establishes a specific number of successful interactions with it). *SUB* is dictated by the *TP* of each individual node, and can model typical trust characters, such as unwary, suspicious, unbeliever, etc. This component provides flexibility in the trust strategy of a user, without imposing significant complexity in the overall trust computation.

3.2. Historical data as input for trust evaluation

History is an additional concept that has drawn attention in the trust community. Several researchers use history as an implicit component in the trust computation. Some assign specific weight to past direct observations or recommendations in order to provide smoothed *TV* fluctuation [6]. The CORE system uses a time dependent function that gives higher relevance to past observations to evaluate the direct reputation of a target node's function [14]. This function takes into account only the B recent observations, and it is represented through a Finite Impulse Response filter (FIR). Authors in [14] believe that if more relevance is given to past observation then sporadic misbehavior in recent observation will not influence the evaluation of the trust value. In [15], a similar approach is used. Older observations become less important that newer ones and observation older than a threshold T are ignored. In [13] a different approach is followed to allow for reputation fading. Through a modified Bayesian approach less weight is given to evidences received in the past. The procedure that updates the trust rating depends on m, i.e., the order of magnitude of the number of observations over which it is assumed stationary behavior. The open issue with this approach is that stationary behavior might not be observed when nodes might selfish or misbehave. *ATF* combines the two approaches. We believe that a significant weight should be assigned in the current observation. This will enable an evaluator node to rapidly identify when a target node starts to misbehave. Additionally, the history of the observations that will be considered for the evaluation of a trust value of a target node, when a new observation is produced, depends on the subjective behavior of the evaluator node. Thus a distrustful node might need long past observations, whilst a deceivable node might need only the newer ones. Thus, when the new observation (i.e., *TS(n,f)*) occurs, a distrustful node takes into account the history *h1* of the direct evidences, as seen in figure 2, whilst a deceivable node takes into account the history *h2*.

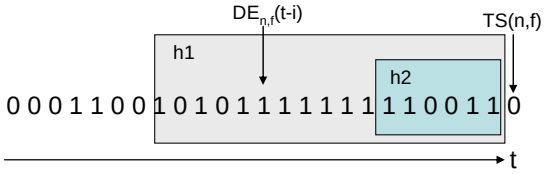

Figure 2. Different observation windows as used for the assessment of the DE trust value

Thus, for each new observation or recommendation (i.e., New Value) the following equation is used to relate historical data with current observations (**Eq.1**):

$$NewValue_{n,f}(t) = w * CurrentValue_{n,f} + (1-w)\left[\sum_{i=1}^{H} NewValue_{n,f}(t-i)\right] / H$$

For simplicity we write $NewValue_{n,f}(t)=WA_H(n,f,t)$ to denote the weighted average over the last H data and the new value recorded at time t. Equation 1 was chosen and verified after an analytic assessment that we contacted in [16] using a preconfigured scenario and different values of *w* and *H*. According to the analysis

in [16], moderate values of w (e.g., 0.2) ensure that the calculated values belong to a range that is not deviated from the real behavior. Thus, moderate values of *w* in Eq.1 ensure that the direct evidence will capture the actual behavior of the target node, without significant deviations due to sporadic misbehaviors or rational errors. Additionally, the historical average component of Eq. 1 introduces different sharpness on the estimation. When only the recent observations are required (e.g., *H=3*, corresponding to an impressionable node) then the *DE* approaches faster the minimum or maximum values, illustrating sudden fluctuations. On the other hand, when the evaluator takes into account long history (e.g., H=8 for a disbeliever node) then the *DE* approaches less rapidly the threshold values, illustrating smooth fluctuations.

3.3. The Quantitative Perspective

This section describes the mathematical formulae for the proposed trust computation model. The *trust time*, as already mentioned, counts the directly observable interactions. The proposed model defines *Positive Time* (*PT*), *Negative Time* (*NT*) and *Total Time* (*TT*). The latter simply adds the other two time scales. We monitor the temporal evolution of every function and node, so these time scales are represented as matrices of size *NxF*, where *N* is the number of nodes in the network, and *F* corresponds to the overall number of supported functions.

$$PT \equiv (PT_{n,f}) \in N,\ NT \equiv (NT_{n,f}) \in N,$$

$$TT \equiv (TT_{n,f}) \in N,\ where\ \ n = 1...N, f = 1...F$$

Each node should have at least one time matrix. Each time a corresponding interaction (*Success*, or *Failure*) with a node's function is observed, the corresponding element of the time matrix is increased by one unit. Hereafter, $T_{n,f}$ will denote $PT_{n,f}$, $NT_{n,f}$ or $TT_{n,f}$. Each detector maintains a *NxF Trust Matrix (TM)*, representing the *TV* that the node computes per monitored function of a target node. Each element $TM_{n,f}$ ($1 \le n \le N$ and $1 \le f \le F$) refers to a specific function *f* of a particular node *n*, and it varies with time. The formulae for *TM* and *TV* are:

$$TV' \equiv TV'(n,f,t) = (a * DE_{n,f} + b * REC_{n,f}) * SUB_{n,f}(t)$$

$$where\ t = T_{n,f},$$

$$TV(n,f,t) = TV' \cdot u(1 - TV') + u(TV' - 1),$$

$$TM \equiv (TM_{n,f}),\ and\ TM_{n,f} = TV(n,f,t) \in [0,1]$$

As we discuss in the next paragraphs: $DE_{n,f} \in [0,1]$, $REC_{n,f} \in [0,1]$, and $SUB_{n,f}(t) \in [0,2]$. Thus, $TV' \in [0,2]$, as well. The parameters *a* and *b* (see *TV* formulae) are step increasing and decreasing functions, as it will be clarified in section 5. In order to map the *TV* values within the [0,1] interval we use a unit step function *u(t)* (see Eq. 2) to normalize *TV'* into the final *TV***.** The range of *TV(n,f,t)* is [0,1], where 0 means that the detector distrusts a target node *n* for a specific function *f,* and 1 means that it fully trusts *n* for *f*.

$$u(t) = \begin{cases} 0, & t < 0 \\ 1/2, & t = 0 \\ 1, & t > 0 \end{cases} \qquad \textbf{(Eq. 2)}$$

DE_{nf} is the *DE* for a target node *n* and its function *f*, as observed by the corresponding *TS* of the detector. The *NxF* matrix *DE,* which is stored in every node, is defined as $DE \equiv (DE_{n,f}) \in [0,1]$, where the matrix elements $DE_{n,f}$ are computed according to eq. 1, as follows:

$$DE_{n,f}(t) = WA_H(n,f,t), \text{ and } TS(n,f) \in \{0,1\}$$

The lower value (i.e. 0) denotes *Failure*, while the higher (i.e. 1) denotes *Success*. The coefficients w_1 and w_2 adjust the weights assigned to recent and historical Direct Evidence values, respectively. $REC_{n,f}$ stands for the aggregated recommendations we have for the function *f* on node *n* from third parties. These recommendations are either third parties' *DEs* or *RECs*. One could argue that allowing a third party to include *RECs* received from others (second-hand *RECs*) in its own recommendations would enable rumor-spreading effects. However, this is necessary in cases we have no other evidence for the trustworthiness of a node. We also keep the history of *RECs* received. Thus, each node has an NxF *REC* matrix, defined as $REC \equiv (REC_{n,f}) \in [0,1]$, where the matrix elements $REC_{n,f}$ are computed according to equation 1 as follows:

$$REC_{n,f}(t) = WA_H(n,f,t) \qquad \textbf{(Eq. 3)}$$

The *SUB* component of the TV computation formula incorporates the node's subjectivity. This subjectivity is a key differentiator between the various nodes and stems from the socio-cognitive approaches to trust modelling [12]. *SUB* is an *NxF* matrix with elements in the $\{f : T \to [0,2]\}$ domain. Thus, its elements are time-functions. The range [0,2] allows the detector to distrust (i.e. value 0) the target node, trust it (i.e. value 1), be enthusiastic about the target node (i.e. value 2)

or develop any other intermediate form of subjective trust strategy. We have chosen the value 2 as an upper bound to allow enthusiastic nodes but not to such a degree that would endanger the network's rationality. In other words we want to have nodes with diverse trust strategies, but we want to restrict the deviation of this diversity. An example *SUB* time-function could be defined as:

$$SUB_{n,f}(t) = u(t-2), \quad t = PT_{n,f}$$

This function indicates that no matter what *DEs* or *RECs* a requestor node has for a function of a target node *(n,f)*, it will not trust the latter until two successful (positive) direct interactions have been observed. In case the aforementioned defined *SUB* component is used indiscriminately for all target nodes and provided functions, it will be identical for all the elements of the detector's *SUB* matrix. The set of the *SUB* functions is defined in the *TP* and it can be adjusted depending on the target node and the monitored function. However, in practice, it is highly unlikely that a node will have NxF different *SUB* functions. Instead, a detector will usually use identical *SUB* function for every target node or every function.

5. The Trust Policy

As already mentioned, for each node a *Trust Policy* (*TP*) defines the functionality of its *Reputation Manager* and *Trust Builder*. A *CP* captures the conceptual subjective behavior of the entity (e.g., end-users) and incorporates the real-life attitude. The parameters of the *TP* are summarized in Table 1. The parameter *HFI* is used by the proposed RM module and its application is described in the following section.

Table 1. The parameters of the Trust Policy

Parameter	Semantics
MI	The Minimum Interactions required for being confident about the *TV* of a target
a	The impact (weight) of the *DE* on the *TV* ($0 \le a \le 1,\ a+b=1$). This is an increasing stepwise function over *MI*
b	The impact (weight) of the *REC* on the *TV* ($0 \le b \le 1,\ a+b=1$). This is an decreasing stepwise function over *MI*
w	The weight of history and the current direct or indirect evidence ($0 \le w \le 1$), as discussed in Section 4.2
TV_{RF} *threshold*	The minimum allowable TV_{RF} assessed to a node in order to be a recommender
$SUB_{n,f}(t)$	A set of time-functions that define the trust strategy of a node. *SUB(0)* is the trust strategy for newcomers, i.e., assigns an initial *TV* to a newcomer
HFI	Honorable Friend Index. The minimum number of trusted recommenders, required to be consulted by a requestor to reliably evaluate the trustworthiness of an unknown node

The parameters *a* and *b* (see *TV* formulae) are step increasing and decreasing functions on *MI*. This policy was chosen because when a detector node realizes the existence of a newcomer only the recommendations of the trusted recommenders should be used (high values of parameter *b*). Thus, in the initial phase the *RECs* are essential. On time, when the detector starts to interact directly with the newcomer, the direct evidences (*DE*) become more important, and this happens only when the *MI* value is exceeded.

6. Reputations over Trusted Nodes

As illustrated in Eq. 3 a requestor is based on recommendations to compute the initial *TV* for newcomer nodes. However, only trusted nodes (i.e., nodes illustrating a high *TV* value for the recommendation function) should provide these recommendations. This will minimize the effects of rumour spreading and avoid potential DoS attacks [7]. Moreover, the selected recommenders should be as close as possible to the requestor so as to have limited communications overhead. Buchegger and Le Boudec [5] proposed the use of a Path Manager module to rank routes according to security metrics (e.g., reputation of nodes in the path) and to delete paths containing malicious nodes. A path ranker, called Pathrater, is also proposed in [4] to mitigate the effects of routing misbehaviour.

TrustSpan is a mechanism that is used by the *Reputation Manager* (*RM*), and enables a requestor to consult only trusted nodes whenever recommendations for the evaluation of newcomer nodes' trustworthiness are required. We consider as *newcomer* the node for which the requestor has no trust-related knowledge, or has inadequate experience (i.e., too few past direct interactions to assess its reliability). When a newcomer becomes adjacent to a requestor, the latter does not know *a priori* whether it is trustworthy and, therefore, it executes this algorithm in order to collect recommendations from trusted recommenders. A requestor charac-

terizes a node as a *trusted recommender* if the *TV* for its recommendation function is high. We remind that RF is monitored and evaluated, just like any other function, via the *RFTS* sensor.

The *TrustSpan* algorithm is presented in Listing 1. The weights in the line *A*, the HFI and the TV_{RF}, are defined through the TP. The Distance Matrix (DM) includes the distances of other nodes. *HFI* denotes the minimum number of recommenders required by a requestor node in order to enable a reliable evaluation of the trustworthiness of a target node's function, and it is independent of the target node or its function.

Listing 1. TrustSpan Algorithm

```
Inputs: HFI, DM, TM, TV_RF threshold
Outputs: IDs of nearest trusted recommenders

procedure TrustSpan (){
  for (j=0;j<numberOfNodes;j++)
    TV_RF[j] = TM[j][RF];
  for (i=0; i<length(TV_RF); i++)
A:  Candidate_Recommenders[i]=0.75*TV_RF[i]+0.2*(1/DM[i]);
  int trusted_IDs[] = new int[HFI];
  for (k=0; k<HFI; k++){
    if(max(Candidate_Recommenders) >= TV_RF_threshold){
      int maxID = indexOfmax(Candidate_Recommenders);
      trusted_IDs[k] = maxID;
      Candidate_Recommenders[maxID] = 0;
     } else break;
  }
  return trusted_IDs;
}
```

According to the *TrustSpan* approach, each time a requestor asks recommendations for a function *f* of a target node *n*, considers only the trusted recommenders. Thus, it tries to minimize both the delay in the propagation of the requested recommendations and the communication overheads. *TrustSpan* is invoked (i.e., recommendations are solicited) only when a detector has inadequate number of direct interactions with a target node (newcomer). This number is defined in *TP* through the parameter *MI*. *MI* also affects which nodes reply to recommendation requests. In particular, the recommenders selected by *TrustSpan* that have less than *MI* direct interactions with a target, will not give recommendations about it.

After the *TrustSpan* algorithm has returned the identified IDs corresponding to the trusted recommenders, the *Reputation Manager* (*RM)* requests the recommendations (through unicasting) from the corresponding *RMs* of the trusted recommenders. A request includes the target node ID and the specific function, which the requestor wants to evaluate. When the required recommendations arrive or a respective timeout expires, the requestor's *RM* forwards the incoming information to the *TB* where the actual trust computation takes place.

7. ATF evaluation

7.1 Simulation environment

We performed our evaluations for the *ATF* framework using the J-SIM wireless package simulator [17]. We used the 802.11 MAC layer and CBR sources to generate traffic. The IP routing protocol that was used was the AODV. The field configuration was 50 nodes over a 300m x 300m terrain. The radio transmission range of each node was set equal to 50m. Initially, the nodes were distributed randomly on the terrain grid. Nodes mobility was simulated according to the random waypoint mobility model, in which each node moves to a randomly selected location at a configured speed (i.e., 2 m/s) and then pauses for a configured pause time (i.e., 5 secs), before choosing another random location. During the simulations, the *ATF* framework estimates the *Trust Value (TV)* of the *packet forwarding* function. Thus, *DEs*, *RECs* and reputations were used or exchanged for the *forwarding function* of each of the 50 nodes in the terrain. The number of selfish nodes was set to 10, and each of them was configured to drop a percentage (30%) of the packets that received but not forwarded. For data traffic, we used 20 CBR sessions. Each session generates packets at rate of 4 pkts/secs. The duration of each session is equal to the duration of simulation (i.e. A sec). Table 2 summarizes the parameters of the simulation environment.

Table 2. Simulation environment parameters

Number of nodes	50
Number of selfish nodes	10
Maximum speed	2 m/sec
Pause Time	5sec
Terrain dimensions	300m x 300m
Communication Type	CBR
Source-Destination pairs	20
Data Packet Rate	4 pkts/sec
Radio transmission range	30m

During the simulations we have measured:

- The communication overhead introduced by the *ATF* on-demand recommendation requests, and the corresponding responses.
- The time required by each node to identify the actual behavior of a peer (i.e., how rapidly *ATF* en-

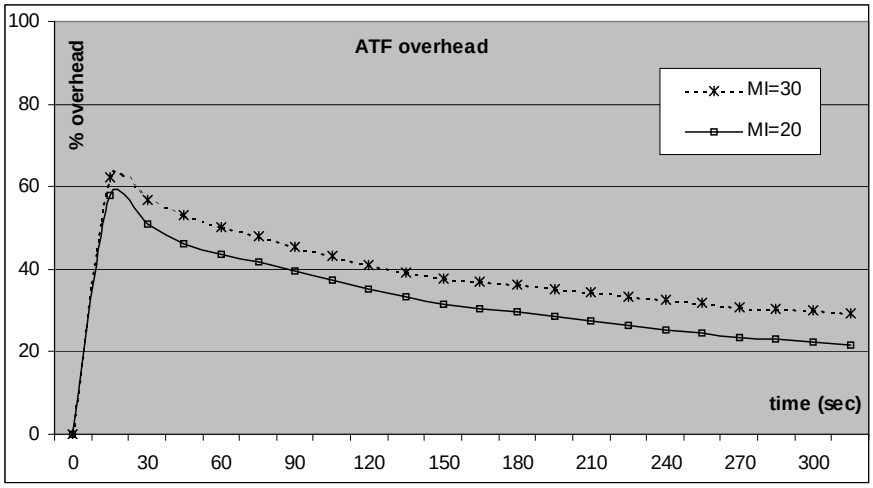

Figure 3. *ATF* overhead due to recommendations

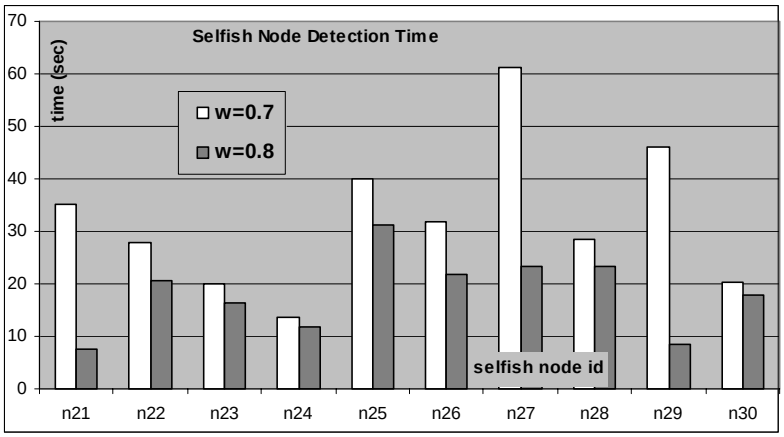

Figure 4. *ATF* convergence rate

ables peers to identify the real *TV* that corresponds to a selfish node's function)

- The accuracy of the *ATF*, i.e., the ability of peers to correctly identify the behavior of peers (in terms of success or failure).

Even if the parameters *a* and *b* (see *TV* formulae) are step increasing and decreasing functions on *MI,* respectively, in the simulations we used predefined static values.

7.2 Simulation Results

The communication overhead that the *ATF* framework introduces due to the on-demand recommendations is depicted in Figure 3. For this scenario *H*=4, *w*=0.8, *HFI*=3, and *a*=0,9. The overhead is measured for two different *MI* indexes (20 and 30 direct evidences). As shown in figure 3, the overhead is decreased with time, and eventually reaches the 20% of the actual IP traffic. This happens because the direct evidences that the pairs of the nodes obtain are increased with time, and thus fewer recommendations are requested form trusted peers. Additionally, for higher *MI* values the overhead is higher, since for higher *MI* the minimum direct evidences required for being confident about the TV of a target is higher, and thus more recommendations are requested form trusted peers.

Figure 4 illustrates the convergence speed of *ATF*, i.e., the time needed for the fair nodes (i.e. 40 nodes) to detect a selfish node (i.e. the 10 unreliable nodes). That is, the time needed for a node to calculate a *TV* value for the forwarding function of selfish nodes (n_{21}, n_{22}, …, n_{30}) that is equal to 0.7, since we assumed that each selfish node drops 30% of the received IP packets. In the worst-case scenario, it requires 60 seconds to conclude that a node behaves selfishly. This worst case occurs when selfish nodes (i.e., n_{27}, n_{29} and n_{25}) do not generate enough direct interactions with fair nodes, due to their mobility pattern. Thus, even if *MI* was set to 20 interactions, selfish nodes n_{27}, n_{29} and n_{25} did not generate a large number of direct interactions during

the initial simulation period. As figure 4 illustrates higher values of parameter *w* enables for faster convergence to the real *Trust Values* (i.e., 0.7). This is because low values of *w* in Eq.1 ensure that the direct evidence will capture the actual behaviour of the target node, without significant deviations due to sporadic misbehaviours or rational errors. Results in figure 4 were measured for *H*=4, *MI*=20, *HFI*=3, and *a*=0,9.

Figure 5 illustrates the accuracy of the *ATF* framework. It depicts the average *TV* of the forwarding function as measured at the end of the simulation, and only by fair nodes which have experienced *MI* interaction with each other node. Here we used *H*=4, *MI*={20,40}, *HFI*=3, *w*=0.3, and *a*={0.9, 0.8}.

For the results depicted in figure 5 we averaged the *TV* values that three pairs of CBR source-destination nodes calculate for the forwarding function of 30 peers. From these 30 peers, the first 20 were configured to be reliable and the last 10 (i.e., n_{21}, n_{22}, …, n_{30}) were selfish. These selfish nodes were set to drop 30% of IP data packets. If one of the estimators (one source or destination of the three CBR traffic pairs) did not obtain at least *MI* interactions with one of the 30 nodes (n_i, $i \leq 30$), then the averaged *TV* was not calculated. Thus, for node n_0 we did not obtain any *TV* results, whilst for node n_5 we have *TV* estimation only when we recorded twenty direct evidences (i.e., *MI*=20). Figure 5 shows that *ATF* is accurate, since it manages to predict the actual behavior of nodes. For the forwarding function of the fair nodes that there were sufficient interactions (i.e., nodes n_2, n_3, n_4, n_5, n_8, n_9, n_{12}, and n_{16}) the *TV* value was estimated near to one, which was the real value. For the forwarding function of the selfish nodes that there were enough interactions (i.e., nodes n_{22}, n_{24}, n_{25}, n_{26}, $n2_8$, n_{30}) the *TV* value was estimated near to 0.7, an estimation that approximates the exact behavior (since each selfish node dropped 30% of packet).

8. Conclusions

In this paper we discuss and evaluate the *ATF* framework for trust and reputation management in self-configured networks. A key difference from similar proposal is the materialization of a subjective trust factor, which imports natural behaviour of end-users and models different trust strategies. Additionally, *ATF* introduces the concept of trusted recommenders. The specialised *RFTS* trust sensors tests, monitors, and evaluates the trustworthiness of peers in respect to the recommendation function, and thus it contributes to the avoidance of avoid rumour spreading, phenomena. Furthermore, the *TrustSpan* mechanism is introduced to minimize communication costs and to accelerate the trust evolution process, since only trusted nodes are conducted for recommendations. Finally, the historical values are incorporated smoothly to the trust computation model, enabling rapid identification of actual misbehaving nodes, without penalizing those nodes that sporadic misbehave due to network errors that occur.

Performance assessment of the *ATF* framework through various simulation scenarios shows that the on-demand recommendation requests and the corre-

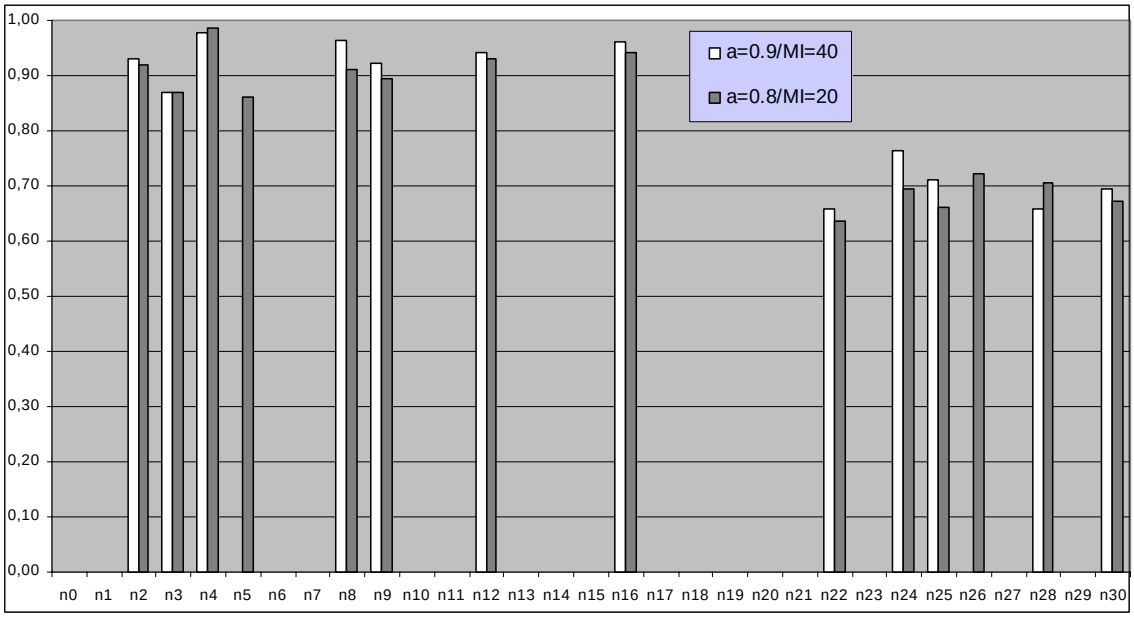

Figure 5. *ATF* accuracy

sponding responses introduce small communication overheads, especially when the MANET initiates its routine operations. This is because recommendations are required only from trusted peers. Additionally, with time the pairs of nodes start to maintain direct interactions and evidences, and thus the recommendations from peers are not essential. Moreover, simulations shown that the *ATF* enables peers to rapid identify the trust values (i.e., trustworthiness) of functions that provided by peer nodes (e.g., forwarding). Thus, *ATF* provides sufficient means to fair nodes for rapid identifying and isolating selfish or malicious nodes. Finally, *ATF* ensures high success rate on predicting nodes' behaviour, since each reliable node identifies a selfish peer with probability equal to one.

Currently, we are running scenarios using a different network setup, i.e., less dense network, various mobility model's parameters and percentages of selfish nodes, to identify how the *ATF* performs under different simulation scenarios. Finally, simulations that are under deployment will aim to measure the effect of the *RFTS* sensor (see section 2.1.4), to evaluate the communication costs of the *RTFS*, to estimate the level at which the *ATF* can increase the network throughput, and to measure energy consumption costs.

9. References

[1]. A. Jøsang, "The right type of trust for distributed systems", in Proc. 1996 Workshop New Security Paradigms, California, USA, Sept. 1996

[2]. F. Stajano, and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad hoc Wireless Networks," in Proc. 7th International Workshop on Security Protocols, 1999, pp. 172-194

[3]. J. Douceur, "The Sybil Attack," in Proc. 1st International Workshop on Peer-to-Peer Systems (IPTPS02), March 2002, Cambridge, MA

[4]. S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehaviour in mobile ad hoc networks", in Proc. of Mobicom2000, Boston, USA, Aug. 2000

[5]. S. Buchegger and J.-Y Le Boudec. "Performance analysis of the CONFIDANT protocol", in Proc. 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing, MOBIHOC2002, Lausanne, Switzerland, June 2002

[6]. P. Michiardi and R. Molva, "CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks", In Proc. 6th IFIP Communications and Multimedia Security Conference, Portorosz, Slovenia, September 2002

[7]. S. Buchegger and J-Y Le Boudec, "The Effect of Rumour Spreading in Reputation Systems for Mobile Ad-hoc Networks", in Proc. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, WiOpt03, 2003

[8]. A. Abdul-Rahman and S. Hailes, "A Distributed Trust Model", in Proc. New Security Paradigms Workshop 1997, ACM, 1997

[9]. V. Cahill et. al., "Using Trust for Secure Collaboration in Uncertain Environments," IEEE Pervasive Computing, July 2003

[10]. C. English, S. Terzis, and P. Nixon, "Towards Self-Protecting Ubiquitous Systems: Monitoring Trust-based Interactions", Journal of Personal and Ubiquitous Computing, Sept. 2005, to appear

[11]. A. Pirzada and C. McDonald, "Establishing Trust In Pure Ad-hoc Networks," in Proc. 27th Australian Computer Science Conference, 2004

[12]. C. Castelfranchi and R. Falcone, "Trust is much more than subjective probability. Mental components and sources of trust", in Proc. HICSS33, Hawaii, 2000

[13]. S. Buchegger and J-Y Le Boudec, "A Robust Reputation System for P2P and Mobile Ad-hoc Networks", in Proc. Second Workshop on the Economics of Peer-to-Peer Systems, 2004

[14]. P. Michiardi, "Cooperation enforcement and network security mechanisms for mobile ad hoc networks" PhD Thesis, Ecole Nationale Superieure de Telecommunications, Dec. 2004

[15]. Y. Wang, and J. Vassileva, "Bayesian Network Trust Model in Peer-to-Peer Networks", in Proc. Agents and Peer-to-Peer Computing 2nd Intl Workshop, AP2PC 2003, July 2003, pp., 23-34

[16]. G. F. Marias, V. Tsetsos, O. Sekkas, and P. Georgiadis "A generic framework towards trust building in self-organized, peer, networks", 1st International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, Santorini, Greece, July 2005, to appear

[17]. Available at http://www.j-sim.org/