A generic framework towards trust building in self-organized, peer, networks

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

Self-evolving reputation schemes for trust establishment in peer networks received increasing attention during the last few years. In this paper we present a distributed, self-evolving, trust establishment framework, which takes into account the idiosyncrasies of the wireless self-organized networks. The framework consists of modules that capture direct proofs of trust or exchange recommendations in order to compute the trustworthiness of peers. It incorporates subjective behavior of end-users, direct observations of behaviors, recommendations, and history of evidences to assess the trustworthiness of peer entities. The proposed framework is associated with a generic model for the evaluation of the reputation level of adjacent or distant nodes. It capitalizes on a new reputation method, called TrustSpan, to trigger only the trusted nodes maintaining the requested recommendations, enhancing the level of trust, minimizing the network overheads for trust building, and accelerating the trust evolvement process. TrustSpan is provided with a supplementary mechanism that evaluates the trustworthiness of the recommenders. Additionally, it incorporates smoothly the historical trust evidences to evaluate current trust levels, enabling rapid identification of actual selfish and malicious behaviors of nodes, without penalizing nodes due to sporadic misbehaviors or rational network errors.

1. Introduction

A Mobile Ad hoc Network (MANET) is a self-organized wireless network consisting of nodes responsible for its creation, operation and maintenance in an absence of a central coordination entity. Due to the locomotion of mobile nodes, the structure of the network is re-organized and, thus, the network topology varies with time. Mobile nodes are continuously associated or dissociated, according to their location and the arrangements of the other nodes. The incentive of a newcomer node that wills to associate 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 principle encapsulates the trust that should be assembled by the nodes, and which is essential for the steady-state operation of a MANET: adjacent[1] 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-evolved trust diminishes when adjacent nodes become distant due to mobility, since several unreliable nodes might intermediate on the communication path. Thus, the trust established between two nodes for routing and packet forwarding might be lost with time, influencing network's performance. Moreover, it is a utopia to assume that nodes behave rationally, rather than passionately [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 intentions might change later on. A *selfish node* is disinclined to spend its resources (battery or CPU cycles) 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). On the other hand, Denial of Service (DoS) attacks are originated from *malicious nodes*, which 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'* primary objective, on the other hand, is the interception of the information exchanged between the network nodes, thus invalidating the ad hoc collabora-

[1] Since in this paper we mostly address trust-aware networking and communications, the term "adjacency" refers to topological proximity.

tions. 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 trust establishment that involves 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 hackers) 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 proactive cooperation enforcement of selfish nodes through credit-based mechanisms.

In this paper a generic framework for self-evolving trust establishment, named *Ad-hoc Trust Framework (ATF)*, is proposed and discussed. The framework is based on a distributed architecture. Each of the modules resides in every node and performs a well-defined set of actions to evaluate the trustworthiness of specific nodes or recommend them to third parties. It incorporates self-evidences, second-hand 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 user-designated *Trust Policy*.

ATF is decoupled from traditional trust building methods, such as symmetric or public cryptography, avoiding complex computations and the expenditure of resources (power, CPU and memory). In that sense, the proposed trust computation framework is usable in different types of distributed, or peer systems (ad hoc networks, pervasive computing services, 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 3 we briefly describe the functionality of the Trust Sensors module. Subsequently, in Section 4 we present the functionality of the trust building module. In Section 5 we describe how the trust model incorporates real-life characteristics. In Section 6 we define and discuss the novel *TrustSpan* mechanism that assists *ATF* to rapidly identify and use recommendations provided only by trusted recommenders. Finally, we discuss the open issues concerning *ATF*, we provide some concluding remarks and directions for further research.

2. The ATF Architecture

ATF uses a layered architecture, which is depicted in Figure 1 and consists of the following components: *Trust Sensors (TS), Trust Builder (TB)* and *Reputation Manager (RM)*. The proposed framework is fully distributed. Every node is equipped with these components and provides a number of typical 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* adopts the definition introduced in [8] for the reputation of a node's function, which is defined through the triplet: *Reputation = {NodeId, Function, Trust Value}*. Thus, the reputation of a function *f* of node *n* is defined as $R(n,f)= \{n, f, TV_{i,f}\}$, where $TV_{n,f}$, is the *Trust Value (TV)* for the function *f* of node *n*. *TV* is updated through direct evidence, recommendations, historical evidences and subjective criteria.

It is necessary to provide here the nomenclature that will be used hereafter. 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 a particular function of the *target*. A *requestor* is a node that asks for recommendations. A *recommender* issues recommendation responses (on-demand scheme). *Adjacent* are the nodes that are in the coverage (one-hop) area of each other. A *path* is a sequence of wireless links (hops) leading from a source node to a destination node.

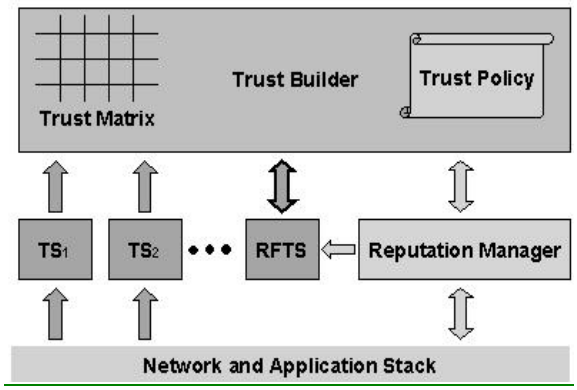

Figure 1. ATF architecture

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*). In the first case (on-demand), a recommender informs the interested node (i.e. the requestor) about the *TV* of a specific target node's function, only when the requestor explicitly requests this value. In the second approach (event-driven), whenever a detector recomputes a *TV* for a target node, it informs immediately the other nodes, through a broadcasting, multicasting or unicasting mechanism. The *ATF* architecture is capable of supporting on-demand recommendations.

2.1 Architecture components

A short description of the architecture's components and their roles is given below, and more details will be found in following sections.

2.1.1. Trust Sensors (TS).

For every function offered by an *adjacent* node there is a *Trust Sensor* that monitors its execution/operation. It actually maps the captured evidence to a numerical value and forwards this value to the *Trust Builder* for further trust computation. The mechanism that captures the direct evidence for a function of an adjacent node can be a generalization of the watchdog mechanism (described in [4]). Furthermore, direct evidence can be monitored through real time measurements and statistical analysis of logs. Nodes procure a set of *TSs* and activate a desired subset, according to their needs.

2.1.2. 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* is distinct for each discrete function per monitored node. A node, for example, may be trustworthy to perform packet forwarding, but unreliable with respect to routing. The *TV* computation depends on several factors, such as the behavior of the examined node in the past (history), recommendations from trusted third parties, and direct evidence. Of course, each factor contributes with a specific weight. History and direct evidence may introduce high certainty for a node's behavior, whereas recommendations might have less contribution. Moreover, each node follows its own *Trust Policy*. The *Trust Policy* brings in the model the user's subjectivity (e.g., distrustful, deceivable), expressed with a time-varying numerical function.

2.1.3. Reputation Manager (RM).

The *RM's* main role is to provide recommendations from third parties to the *TB* in order the latter to compute the required *TVs*. In an on-demand schema, *TB* requests a recommendation for a target node. 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 such a value exists. 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.

2.1.4. Recommendation Function Trust Sensor (RFTS).

This is special-purposed trust sensor that evaluates the trustworthiness of a node regarding its recommendation function. *RFTS,* as any other *TS*, categorizes a direct observation as *Success* or *Failure*. As previously mentioned, the *RM* of a detector receives from recommenders the recommendations that correspond to a specific function of a target. A recommendation is sent 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*. For each recommender, *RFTS* stores in a matrix the values that received from *RM*. 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 (defined 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, a testing mechanism is essential. In regular intervals the evaluator requests recommendations for particular functions of target nodes that maintains direct evidence and especially for those nodes that experienced a large number of interactions

in the recent past. When recommendations arrive, the evaluator compares each received value to the *DE* that maintains. According to the deviation of these two values (direct evidence and recommendation value) the evaluator increases or decreases the *TV* for the recommendation function of the 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 data (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 propose the CONFIDANT 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. The main difference between ATF and CONFIDANT is that the former is capable of supporting on-demand recommendations while the latter is based on ALARM messages broadcast by a node to warn others of malicious nodes. Moreover, in ATF each node is characterized through a Trust Policy that takes into account a subjective judgment (SUB factor, see section 4.1) and historical data to evaluate the trust level of functions that other nodes perform.

Work in [1] is considered as a basis for the *ATF* architecture. The concept of knowledge in [1] is integrated as history, subjective factors and Trust Policy to our model. On-demand recommendations and direct evidences, which are stored in the Trust Matrix, support the recursive operation, as they are used for forthcoming estimations. Finally, work in [1], defines an abstract trust deviation mechanism to evaluate trust levels. Here we further elaborate on this, defining the *Trust Builder* module and a, novel, associated trust evaluation model. As stated in [6], DoS attacks might occur if negative reputations spread around. Moreover, Buchegger and Le Boudec in [7] mentioned that the usage of only positive recommendations minimizes the effect of rumor spreading phenomena. In the *ATF* we take into account direct evidence, as well as recommendations provided only by trusted nodes. Additionally, we propose an innovative testing mechanism to evaluate the trustworthiness of recommenders. We assume the existence of a protocol that propagates recommendations between the nodes. Such a protocol is described in [8].

3. The Trust Sensor Mechanism

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. The *ATF* is based on *Trust Sensors (TSs)* to detect direct evidences. 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 and compares it to a predefined reference attitude, (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 concerns the following:

- A conceptual model and definition of a node's expected functionality. This model is, generally, in close relationship with the observation methods selected (as discussed below). For example, if the observation method is pattern analysis, the expected functionality would be expressed in terms of acceptable patterns.
- *The observation methods/mechanisms*. Possible realizations include the pattern analysis of logs, promiscuous mode of operation, passive eavesdropping, return codes of remote procedures, etc. The observation method is dependent on the function that a *TS* is supposed to monitor.
- *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].

4. Trust Computation in Trust Builder

4.1. The Qualitative Perspective

In this section we describe the intuitive properties of a generic trust- and reputation-building component. The proposed scheme 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 [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.

4.2. 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 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*.

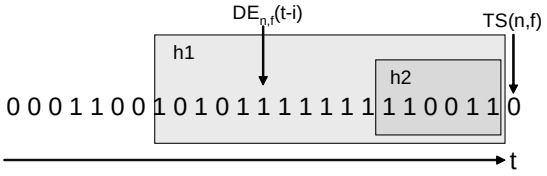

Figure 2. Different windows for the observations as used for the assessment of the DE

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

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

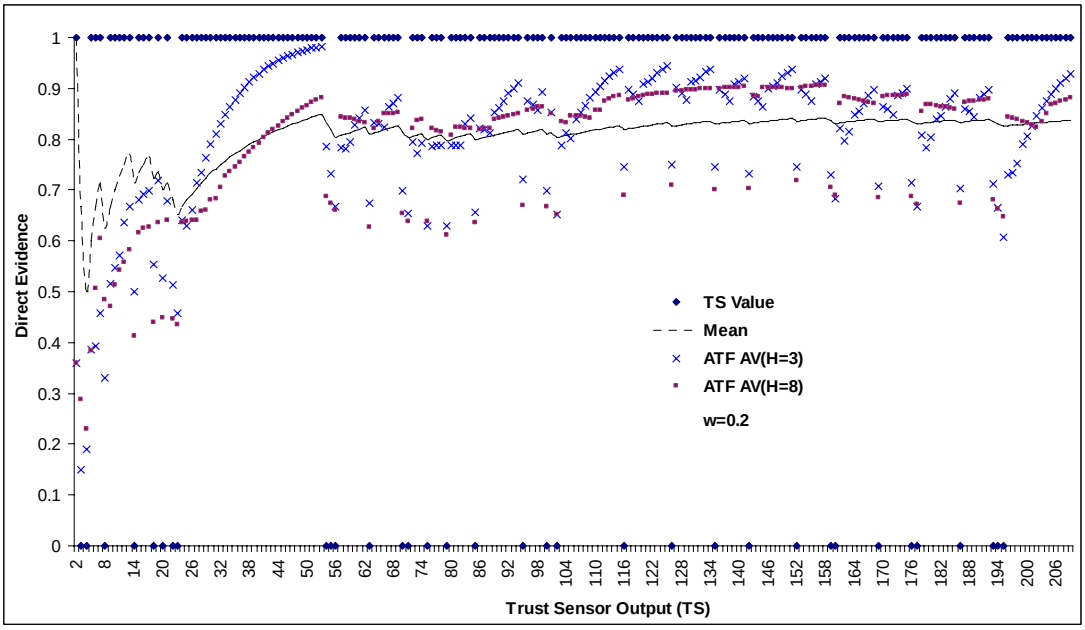

Figure 3. DE fluctuation for two different historical windows (H is equal to 3 and 8) and w=0.2 as weight of the current observation

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. For the assessment of our approach we used a preconfigured scenario, where only the *DE* parameter was taken into account. According to this scenario a target node *n* behaves rationally and forwards (function *f*) the 83% of the packets received from the evaluator node. The evaluator node (i.e., detector node *e*) promiscuously monitors the function *f* of node *n* through the *TS(n,f)*. The remaining packets (i.e., 17%) are lost due to buffer overflows or packet drops because of the noise errors. In this scenario, node *e* monitors the behavior of *n*, using the corresponding *Trust Sensor* (*TS*) for approximately N=200 packets that *n* should forward on behalf of *e*. For each successful relay that e captures, TS_f returns 1 to the *TB*, otherwise a 0 is returned. In this scenario, node *e* estimates an arithmetic average that is equal to 0.83 on the steady state. Figures 3 and 4 illustrate the fluctuations of the $DE_{n,f}(t)$ over time, when the weight w is set to 0.2 and 0.6, respectively. Each figure depicts the arithmetic Mean ($\frac{1}{N}\sum_{i=1}^{N} TS$), the *TS* responses on each observation (i.e., 0 or 1), and two discrete $DE_{n,f}(t)$ which take into account different historical windows, i.e., h_1=3 (fig.3) and h_2=8 (fig. 4). As illustrated in these figures, moderate values of w (e.g., 0.2) ensure that the calculated new values (e.g., of *DE*) belong to a range that is not deviated from the original hypothesis (i.e., node *n* is fair on function *f*). Thus, taking into account only the values after the initial training period (i.e., values after the 30th observation), the new values (e.g., of *DE*) in figure 3 (*w*=0.2) do not fall below 0.6, whilst in figure 4 (*w*=0.6) *DE's* values approximate 0.3. 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 *DE* 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.

4.3. 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.

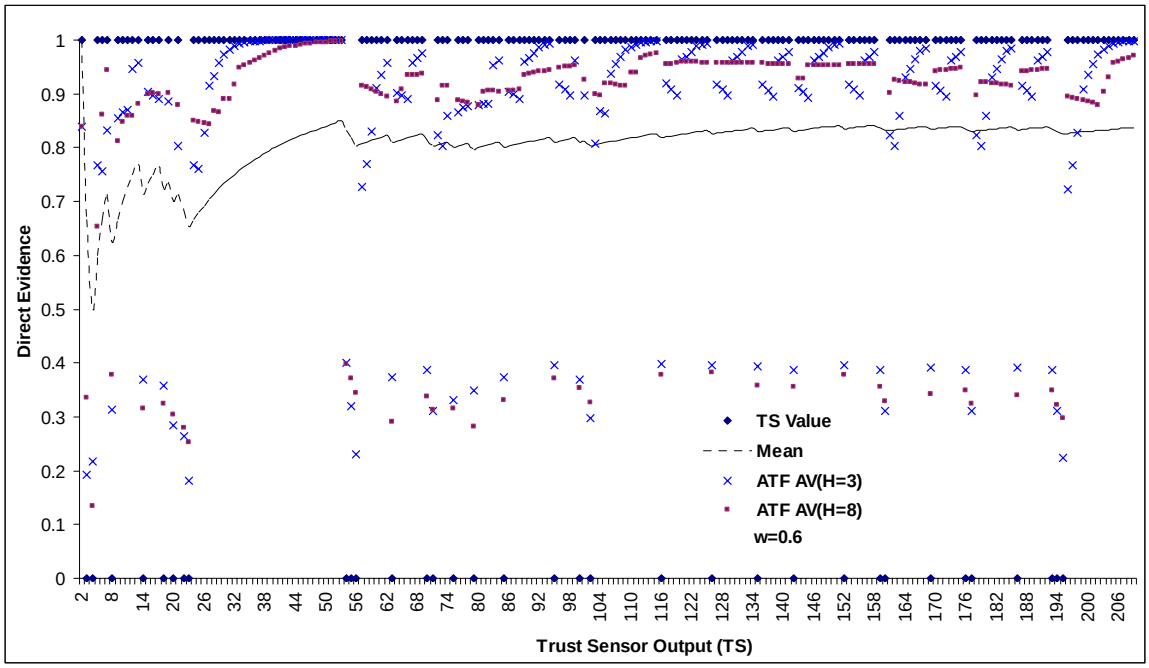

Figure 4. DE fluctuation for two different historical windows (H is equal to 3 and 8) and w=0.6 as weight of the current observation

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

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

$$TT \equiv (TT_{n,f}) \in N$$

$$where \quad n = 1...N, f = 1...F$$

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 (*CurrentValue* equals to *TS*), 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:

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

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 $\{f : T \rightarrow [0,2]\}$ 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*) is defined. This provides a set of parameter values, which can fully define the functionality of its *Reputation Manager* and *Trust Builder*. A *TP* 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 table also describes their semantics (range, necessary conditions, special values, etc.). The parameter *HFI* is used by the proposed RM module and its application is described in the following section. Other, real-life, factors or specific parameters can be included in the *TP* specification, such as the initial *DE* and *REC* for newcomers.

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. Pathrater is assisted by a watchdog mechanism that identifies misbehaving nodes that do not forward packets. The node uses this knowledge to choose the path that is most likely to deliver packets.

In this section, an innovative and generic method, called *TrustSpan* is introduced. This mechanism 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 characterizes a node as a *trusted recommender* if the *TV* for its recommendation function is high. We remind here that the 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 to be consulted 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.

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.

```
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]=0.7*TV_RF[i]+0.3*(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;
}
```

Listing 1. TrustSpan Algorithm

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. Remarks and Future Work

In this paper we introduced a framework for trust and reputation management in ad hoc networks. This framework was materialized through a novel, generic and distributed architecture comprised of components for trust building, communication of recommendations and sensing of nodes' behaviour. We also proposed a lightweight mechanism for recommendations exchange, which takes into account only the valid recommenders, without flooding the network. One main concern was to design a generic model that incorporates many open parameters, defined by the users through a policy. This enables possible extensions and alternatives during the deployment process and is in accordance with the vision for pervasive computing systems.

A key difference from similar work 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.

Concerning collusion, a possible threat that ATF can handle (to some extent) is the spreading of false recommendations orchestrated by malicious nodes. The trust building method of ATF can give considerably less weight to recommendations than to direct evidence, if the parameter *a* is set appropriately. Thus, even when the received recommendations provide false trust value about a *target* to a *detector*, the value will be quickly adjusted upon reception of the first direct evidences concerning the *target*. In addition, when a *detector* has adequate number of direct interactions with a *target* it stops requesting recommendations, and furthermore, penalized the malicious recommenders to avoid them for future reference. However, it is always possible for a node to be isolated (i.e., surrounded) by malicious nodes. In such a case, little can be done irrespective of the used trust framework.

Currently, a large set of simulation scenarios are under development for the evaluation of the model's efficiency. We are planning to estimate *ATF* against the speediness for identifying an unreliable, selfish, node. We are evaluating the communication costs, and estimate the level at which the *ATF* can increase the network throughput, as well.

8. 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