Trust Management Issues for Ad Hoc and Self-organized Networks

Vassileios Tsetsos[1,3], Giannis F. Marias[2], and Sarantis Paskalis[1]

Pervasive Computing Research Group, Dept. of Informatics and Telecommunications, University of Athens, Panepistimiopolis, Ilissia, GR-15784, Athens, Greece
[1] {b.tsetsos, paskalis}@di.uoa.gr [2] marias@mm.di.uoa.gr
[3] corresponding author - tel: +302107275362 fax: +302107275601
http://p-comp.di.uoa.gr

Abstract. Self-organized and ad hoc communications have many fundamental principles in common and also face similar problems in the domains of security and Quality of Service. Trust management, although still in its first steps, seems capable of dealing with such problems. In this paper we present an integrated trust management framework for self-organized networks. In addition, starting from our experience with the presented framework, we indicate and discuss important research challenges (among them interoperability and integration issues) for the future evolution of the trust-based autonomic computing paradigm. We argue that ontologies can address many of these issues through the semantics they convey.

Keywords: trust management, ad hoc networking, ontologies, interoperability

1 Introduction

Pervasive Computing envisages computing environments with ubiquitous connectivity among the deployed devices and provision of advanced "intelligent" services. As this vision comes closer to realization, new computing and communication models are deemed as necessary for the efficient handling and performance of the participating complex systems. Autonomic Communications may be one possible solution towards the next generation of large-scale networking. A case where the autonomic paradigm can be applied, is the well-known ad hoc networking paradigm. This is a special case, since ad hoc networks impose many more challenges (in all relevant computing domains) than the infrastructure-based ones. In fact, there is strong relevance between mobile ad hoc networks (MANETs) and autonomic computing and communications (ACC) which is based on their fundamentally dynamic nature. Thus, the issues addressed in this paper concern both MANET and ACC paradigms.

Security, as well as Quality of Service (QoS), issues have been well studied in existing networked systems. Such issues also arise in ACC and MANET systems, however, their handling, in general, differs from that of current systems. The special characteristics of ACC impose new security threats and risks that the future security mechanisms should take into account. In particular, the self-optimization principle of autonomic systems, if applied in an entity/element- and not a system-basis, can lead to increased

competitiveness and, thus, reduced reliability of the overall system's behavior. For example, such self-optimization could dictate the individual communicating entities to behave maliciously or in a selfish manner. In an open autonomic environment, entities, information assets, data communications and robustness are subject to the following threats:

Attacks on the authenticity of entities, such as impersonation and *Sybil* attacks [1].

Attacks on the privacy of communication flows, such as passive eavesdropping, or even sinkhole and wormhole [2], traffic analysis, with an objective to disclose the exchanged data and the identity of communicating entities.

Attacks on entities or resources availability, such as denial of service attacks, materialized through sleep deprivation torture, flooding and active interfering, or even attacks on the network performance, through selfish nodes.

Attacks on the integrity of the information assets through unauthorized alteration of distributed IT resources and of stored or exchanged data.

As it has already been identified [3], one of the most promising, though challenging, mechanisms to address both security and QoS risks in ACC is the trust establishment, evaluation and management between the cooperating entities. One of the core processes involved in the trust evaluation process is the reputation management, where cooperating entities exchange their experiences and recommendations about third parties. Trust is a soft-security method in contrast to hard-security methods such as certificate-based authentication and Public Key Infrastructures (PKI). The main advantages of soft-security are that it requires less formal information about the cooperating entities and it does not assume any infrastructure to be available. Since these are fundamental assumptions of the ACC and MANET paradigms, it seems quite reasonable to exploit trust mechanisms in such systems.

Thus, frameworks that provide self-protection of the distributed information assets, entities authenticity, and resource availability are considered essential. Since ACC specifies a self-evolving paradigm such self-protection frameworks will be situation-driven. Nodes and entities should react consistently and correctly to different situations [4] based on high level policies. In the first part of this paper, we present ATF (Ad hoc Trust Framework), a lightweight framework for trust management in self-organised networks, like MANETs. This framework is designed so as to detect selfish, malicious or unreliable behavior of communicating network nodes and provide feedback to the various services of each node on how to assess the trustworthiness of the corresponding services of other peers. ATF is lightweight in the sense that it does not perform extensive risk and behavior analysis for trustworthiness assessment, nor does it include a reasoning service capable of adapting the systems behavior with respect to other nodes' behavior and alternative strategies. Additionally, it does not involve computationally heavy security tasks, such as key generation, key agreement and cryptography. This simplicity allows its application in real systems with minor integration effort (see also Section 4). In the second part of the paper, we discuss some design aspects of trust systems with respect to the ACC vision. Based on our experience with ATF design, we indicate some challenging research areas. Among them is the interoperability between heterogeneous trust-aware ACC entities, the design of trust policies and the semantics of trust. In addition, being familiar with semantic knowledge representation and engi-

neering we foresee many applications of ontologies in these areas and we outline some possible ways for their contribution.

2 Architecture of the ATF

ATF is based on a distributed and modular architecture. Each of the modules resides in every node and performs a well-defined set of actions to evaluate the reputation of another node or to inform others about the trustworthiness of third nodes. ATF incorporates self-evidences, recommendations, subjective judgment and historical data to evaluate the trust level of other nodes. These elements are the inputs to a trust computation model. The model also consolidates, among others, user's natural behavior, through the designation of a user-oriented Trust Policy. Such policy defines the parameters that will influence the trust computation process.

The ATF architecture is depicted in Fig. 1 and consists of the following components: *Trust Sensors (TS), Trust Builder (TB), Reputation Manager (RM) and Trust Policy (TP)*. Every node implements these components. Every node also provides a number of typical *communication functions (i.e., services)* such as packet forwarding, routing, naming, etc. In general, as function can be considered any service or application provided by a node. Moreover, every node implements a special virtual service, called *Recommendation Function (RF),* which provides recommendations for specific nodes to third parties (this function is implemented by RM). *ATF* adopts the definition introduced in [5] 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 *i* is defined as $R(i,f)= \{i, f, TV_{i,f}\}$ with $TV_{i,f}$ being the *Trust Value (TV)* for the function *f* of node *i*. This value is updated through direct evidence, recommendations and subjective criteria.

Before proceeding, it is necessary to provide the nomenclature that will be used hereafter (see Fig. 2). We use the term "detector" to denote a node that directly monitors the behavior of another node's functions, called "target". A "requestor" is a node that asks for recommendations, which are issued by "recommenders". "Neighborhood" is the set of all adjacent nodes.

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 [6], this evidence monitoring is performed independently by each node through a "Monitor" [8] which logs every activity in an "Evidence Store". In [8] a Watchdog mechanism is proposed as an observation device for routing behavior of nodes participating in ad hoc networks. The proposed mechanisms are usually function-specific. For example, monitoring the packet forwarding function of an adjacent node is different, in terms of functionality and semantics, from monitoring the routing service. ATF is based on *TSs* to detect direct evidences. A *TS* operates similarly to common sensors: translates a physical phenomenon or behavior in a machine interpretable form. In our case this phenomenon is the trustworthiness of a

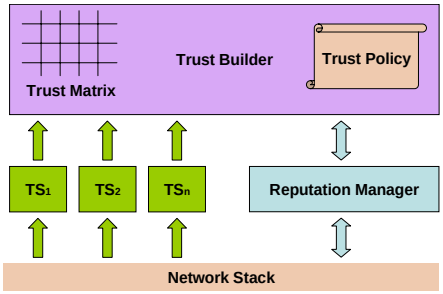

Fig. 1. ATF architecture

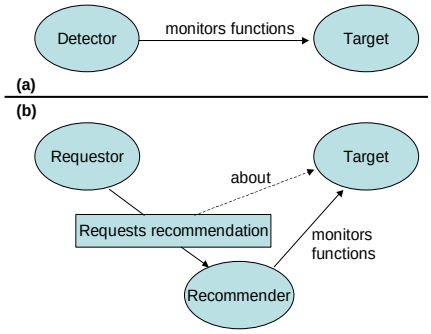

Fig. 2. Trust entity model (a) direct monitoring scenario, (b) recommendation-based scenario

node. A *TS* monitors the behavior of an adjacent node, on behalf of its housed node, and compares this behavior 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 others. The proposed generic methodology consists of the following:

Definition of a conceptual model 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 recognition and analysis, the expected functionality would be expressed in terms of acceptable patterns.

Definition of the *observation methods/mechanisms*. Possible realizations include the pattern analysis of logs, messages overheard through promiscuous mode of operation, return codes of remote procedures, etc. The observation method is dependent on the function that a particular *TS* is supposed to monitor.

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. The number of successes and failures eventually leads to the quantification of the actual direct evidence.

At this point we should mention a special-purposed trust sensor of ATF that evaluates the trustworthiness of a node regarding its recommendation function. *RFTS* (Recommendation Function Trust Sensor), as any other *TS*, compares the observations, i.e., recommendations received for a target node, with the direct evidence of that node. If these values differ significantly the trustworthiness of the respective recommenders regarding their *RF* is decreased. In addition, a testing mechanism can decrease the impact of lying recommenders as follows: in regular intervals a requestor requests recommendations for particular functions of target nodes for which it maintains a large number of interactions in the recent past. The requestor increases or decreases the trustworthiness of the recommenders' *RF*, according to the deviation between these two values (direct evidence and recommendation).

Trust Builder (TB). This component computes the *TV* of other nodes' functions. In particular, it computes the *TV* of the nodes with which there is established interaction or intention for cooperation. All these *TVs* lie in the *Trust Matrix*, which is consulted by applications and other system/network services, and the role of the *Trust Builder* is to maintain and update this matrix. The actual *TV* computation depends on several factors and is described in Section 3.2. Factors such as, interaction history and direct evidence may introduce high certainty for a node's behavior, whereas recommendations might have less contribution. The weighting of these factors should be defined after extensive simulations and expressed through a Trust Policy.

Reputation Manager (RM). The *RM* role is to manage the recommendations exchange procedure (i.e., to provide recommendations from other nodes to the *TB* in order the latter to compute the *TVs,* and to give recommendations about third parties). In an on-demand schema, originator requests recommendations for a target node when it has insufficient experience with it. Thereafter, *RM* selects the nodes to contact (recommenders) in order to obtain requested values. These should be as close as possible to the originator in order to minimize communications overhead, but should also be rated as "good" recommenders (i.e., the originator has a high TV for their recommendation function). When recommendations arrive, the *RM* aggregates them and returns a single value to the *TB*. When a recommender receives a request for recommendation, its *RM* contacts *TB* and obtains the *DE* (see Section 3.2) for the requested function of target node, if any. Next, the recommender returns this value to the requestor node (see Fig. 2b). *RM* could be also responsible for informing other nodes whenever the *TV* of a function of a node is rapidly changing. In this event-driven schema, *TB* triggers *RM* upon trust value changes and *RM* informs the other nodes (e.g. through flooding or multicasting).

Trust Policy (TP). As already mentioned, each node maintains a *Trust Policy* (*TP*); a set of parameter values, which can fully define the functionality of its *Reputation Manager* and *Trust Builder*. In the current version of our model such policy is quite simple, but in the future will be enriched with more advanced features that enable trust strategy definition and enforcement.

3 Trust Computation Model

3.1 The Qualitative Perspective

The majority of the trust computation approaches acknowledge that two main components should be taken into consideration: *Direct Evidence* (DE) and *Recommendations* (*RECs*) from third parties. The *DE* is calculated based on *TSs'* feedbacks 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.

The proposed scheme incorporates several user-defined 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 (recommendations) for the trust building process. Moreover, interactions can be categorized to positive and negative (according to the success/failure model incorporated by each *TS*) to enable flexible computation of trust.

Socio-cognitive approaches to trust [9] imply that trust computation should also include a *subjective component***,** since 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 defined in 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.

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 observations or recommendations in order to provide stable and smoothed TV fluctuation [10]. Others assign minor weights to evidence (direct or indirect) received in the past to allow for reputation fading [11]. Even if the first approach seems more suitable for trust modeling, the scheme proposed here can support both policies. In the following paragraphs a detailed description of how the TB manipulates all the aforementioned factors is provided.

3.2 The Quantitative Perspective

This section describes the mathematical formulae for the proposed trust computation model. The *trust time (T)*, as already mentioned, counts the directly observable interactions. We monitor the temporal evolution of every function and node, so this time scale is represented as matrix of size NxF, where *N is the number of nodes* in the network, and *F corresponds to the overall number of supported functions*.

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

Each node should have at least one time matrix. Each time the outcome of an interaction (success or failure) with a node's function is captured, the corresponding element of the detector's time matrix is increased by one unit.

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:

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

$$TV' \equiv TV'(n,f,t) = [a \cdot DE_{n,f} + (1-a) \cdot REC_{n,f}] \cdot SUB_{n,f}(t),\ t = T_{n,f}$$

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

$$DE_{n,f} \in [0,1],\ REC_{n,f} \in [0,1] \text{ and } SUB_{n,f}(t) \in [0,2]$$

Thus, $TV' \in [0,2]$, as well. In order to map the *TV* values within the [0,1] interval we use a unit step function *u(t)* (see Eq. 1) to normalize *TV'* into the final *TV***.** The range of *TV(n,f,t)* is [0,1], where 0 declares distrust for a specific function *f* of a target node *n,* and 1 declares complete trust in *n* for *f.*

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

DE_{nf} is the *DE* for a target node *n* and its function *f,* as observed by the corresponding *TS* of the detector. The elements of the DE matrix are defined as:

$$DE_{n,f}(t) = w \cdot TS(n,f) + (1-w) \cdot A_H(DE_{n,f}) \text{ and } TS(n,f) \in \{0,1\}$$

A "0" value of TS indicates Failure, while "1" denotes Success. The coefficient w adjusts the weights assigned to recent and historical DE values and the A_H is an average of the last *H* DE values.

$REC_{n,f}$ stands for the aggregated recommendations we have for the function *f* on node *n* from third parties. These recommendations are third parties' *DEs*. We also keep the history of *RECs* received. Thus, each node has a NxF matrix, with elements:

$$REC_{n,f}(t) = w \cdot NEWREC(n,f) + (1-w) \cdot A_H(REC_{n,f})$$

NEWREC is the more recent REC.

The SUB component of the TV computation formula incorporates the node's subjectivity, as discussed in the previous subsection. SUB is a 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 prevent enthusiastic nodes from endangering the network's rationality. An example SUB time-function could be defined as:

$$SUB_{n,f}(t) = u(t-20), \quad t = T_{n,f}$$

This function indicates that no matter what DEs or RECs a requestor node has for a target function, it will not trust it until twenty direct interactions have been observed. We should remind that all parameters involved in the present model (including SUB) are defined in each node's trust policy.

We have evaluated the performance and quality of the proposed framework in [25]. This evaluation showed that the on-demand recommendation requests and the corresponding responses introduce small communication overheads. Moreover, simulations showed that ATF enables peers to rapid identify the trust values for functions provided by peer nodes (e.g., forwarding). Thus, ATF provides sufficient means to fair nodes for rapidly identifying and isolating selfish nodes.

4 Integration and Interoperability Issues

Trust management for ACC and MANET systems raises, as happens with every new computing paradigm, questions regarding its seamless integration with current network protocol stacks. In particular, one should define how the introduction of trust affects the architecture and operation of current services and identify potential difficulties or problems during such integration.

In general, three types of modifications are needed for such integration (see Fig. 3):

1. **Introduction of a trust plane**. This is a vertical plane similar to the user/control and management planes of other networking specifications. It includes all the necessary components in order to assess trust of third entities: sensing mechanisms, policy-driven evolution of trust, interaction memory etc. Trust plane operates also as a broker since it disseminates to all other interested layers the observations of each specific layer. For example, it may give feedback to the networking layer (i.e., routing) about the physical layer operation of peer entities.
2. **Recommendation exchange through a trust protocol.** Such protocol could be implemented in the application layer and is responsible for the request and receipt of recommendations (i.e., in ATF it would be part of the Reputation Manager).
3. **Trust-aware versions of current protocols**. The observations collected by the trust plane or the recommendations collected through the trust protocol should somehow affect the operation of the network stack. This can be only performed if the protocols support trust-driven reconfigurability.

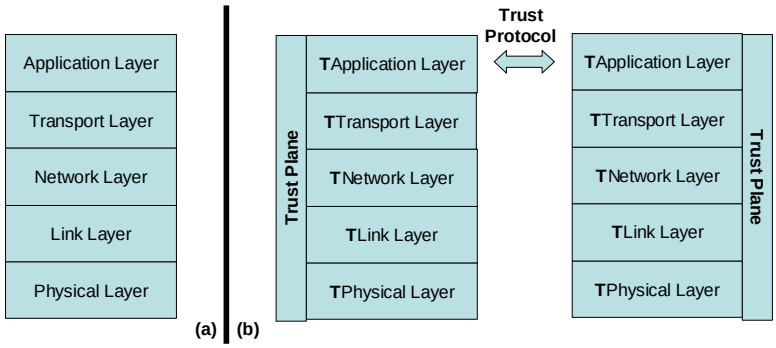

Fig. 3. (a) Traditional network stack (b) New stack elements imposed by incorporation of trust (bold text in figure)

The trust plane is a distributed entity, residing in each node and dealing with trust management issues. It is similar in nature to the knowledge plane proposed in [24]. The similarities consist of the autonomic and distributed nature of the planes, and the "subjective" approach the proposed constructs follow. The trust plane, however, imposes harder design requirements. In addition, the representation and reasoning of trust entities and relationships is strict and the operation of the whole autonomic systems network is very sensitive to any weak interpretation of trust.

A good example of a trust-aware, self-adapted and reconfigurable "protocol" is the software radio [12]. Specifically, software radio serves as a radio communication system technology that uses software for the modulation and demodulation of the signal. The use of software is not only cost-beneficial; it releases the physical layer from the tight hardware integration. That is, the interface to the physical layer is no more a fixed hardware interface, but a set of interfaces provided by the deployed software. The net result is that the physical layer can be altered to any supported protocol by simple software redeployment triggered by the trust plane. In general, the protocol adaptations may be as minor as a parameter tweaking or as major as a complete operation protocol swap in cases of fully autonomic operation [13].

One problem of such extended reconfigurability is the extensive management overhead required and the interoperability issues raised when nodes with different protocol configurations wish to communicate. In general, an autonomic network may sooner or later evolve to a collaboration domain of heterogeneous nodes. This is in contrast to the traditional networking paradigm, where all nodes adhere to strict standards (e.g., SSL, RSVP) in order to "stay connected". Such paradigm, although is typically simpler to implement, imposes a major restriction to the network: the nodes should not differentiate regarding their interfaces or protocols. This restriction aims, besides the obvious co-operation simplicity, at the *preservation of service and protocol semantics* that describe their messages, parameters and interaction sequences.

Obviously, this way of inter-networking is not suitable for flexible autonomic networks. In such networks, the nodes in order to continue collaborating should adhere to some common *interoperability rules*. This is a *soft-standardization* approach in opposition to the abovementioned *hard-standardization* one. A good enabler for such interoperability frameworks are the ontology-based knowledge management facilities. These have recently met wide acceptance by the research community as a means to introduce Artificial Intelligence techniques into practical applications. The most popular initiative in this discipline is the Semantic Web [15].

Ontology [16] is a terminology shared by all parties interested in an application domain (e.g., networking engineers, companies and forums). Apart from the taxonomy of the concept model of interest, an ontology also contains restrictions and formal axioms relevant to this model. Towards the vision for soft-standardized interoperability ontologies can be utilized as protocol hierarchies where a protocol is classified under a class if it satisfies its necessary (and sufficient) conditions. Such conditions may, for example, involve existence of protocol parameters or restrictions on parameter values. In general, every discrete protocol configuration can be mapped to an ontology class. The classification performed by reasoning engines on the ontology instances can infer compliance or not between the various protocols. Similar approaches for integration have been already exploited in other domains, such as network management [17].

5 Other Issues

Trust Semantics

Another aspect that is more technical but closely related to interoperability is the clear definition of trust semantics. For example, autonomic systems developed in different domains may not represent their trust values using the same representation forms. As a simple scenario consider a system A that assesses trustworthiness using real numbers in the range [0,1] (i.e., what ATF does), a system B that uses integers in the range [1,12] and system C that uses fuzzy set theory in order to translate its observations to symbolic values. Apparently, unless we map each system's values to a commonly-adopted trust value reference system, these systems will not be able to communicate their trust-related information. Such mapping entails the careful specification of semantics for each involved trust value system.

This problem can be also addressed by ontology-based knowledge representation, since ontologies are the ideal candidate for playing the role of such reference system. The interacting systems should align their trust models with the conceptual trust model of the reference system. Furthermore, through such alignment, the semantics of the reference system are assigned to each specific trust model. This can be of great value, since the axioms and restrictions described in this reference trust ontology are automatically inherited by the specific models and can be exploited for advanced trust reasoning. Some interesting trust reasoning techniques utilizing semantic (web) technologies are presented in [18][23].

Trust Policies

From the initial research steps of ACC, policies have been recognized as a means for allowing self-organization of systems adhering to some predefined rules. Modern policy management has become quite formalized. Hierarchical policy architectures are proposed [21][22] that use distributed policies in hierarchical environments (grids, storage, ad hoc networks, etc.). The evolution of policy management divides the policy lifecycle into separate building blocks, which can be modified independently. This ability creates new opportunities to experiment and evaluate different policy schemes in the definition stage, in the enforcement stage or in any other in-between stage.

The introduction of multi-level policy architectures might play a significant role in the future of autonomic computing. Multi-level policies may bear a hierarchical or even a mesh structure, depending on the complexity of the facility and the specific need of the situation. In hierarchical policy architectures, a grand policy sets the basic rules and more specific policies apply to specific tasks.

After specifying the policy architecture, the policy definition will likely be a major field of innovation and experimentation. For example, advanced tools such as stochastic processes could be involved to randomize and, thus, conceal the trust-based decision making processes from potential enemies/attackers. Moreover, elements of game theory can be exploited to optimize the performance in such environments, where conflict of interest is apparent (in [14] trust management is described as a strategy game). Nevertheless, the complexity of the solutions will be a major factor to the acceptance or not of the final scheme.

For the implementation and enforcement of rule- and logic-based policies Semantic Web technologies (i.e., ontology languages) can be used. In fact many modern policy-based trust systems [19][20] have adopted such technologies due to their rich expres-

siveness, tractability (i.e., low computational complexity) and developer community adoption. However, exploiting ontologies for expressing policy rules implies that we have already established well-defined semantics for trust itself.

Finally, regarding the reasoning behind policy enforcement and trust assessment, while cognitive-evolutionary approaches may seem suitable for construct with overwhelmingly many parameters, such as the trust plane and trust policy, their relaxed reasoning could lead to security compromises in any unforeseen lapses. On the other hand, a full-knowledge, strict-reasoning approach is probably utopic given the complexity of the issue. Hence, the most likely path would be to accept the possible trust breach of a subset of the connected systems and put effort on isolating the identified malicious systems. To accomplish this self-healing task, features such as redundancy, anomalies detection and self-reconfiguration are necessary.

6 Conclusions

Trust-based communications is a key element of self-organized systems, as they enable advanced interaction models, even between unknown entities. We described a light-weight trust framework, suitable for ad hoc communications that incorporates direct evidence, recommendations, history and subjective factors, in order to evaluate the trustworthiness of peer nodes. Such model requires several modifications to the current system architectures, which in their turn affect the interoperability of nodes. We discussed such integration and interoperability issues, as well as other issues related to trust-aware self-adaptable systems. Finally, we believe that ontology-based knowledge representation can clarify the semantics of such systems and, thus, we outlined some options towards this direction. As a future work, we aim to further investigate the coupling of ontologies and trust frameworks in order to propose more specific solutions to the issues discussed in this paper.

Acknowledgement

This work was performed in the context of the project entitled "PYTHAGORAS: Support of Universities' research groups" co-funded by the "Operational Programme for Education and Initial Vocational Training" (O.P. "Education") and the European Social Fund.

References

1. Douceur, J.: The Sybil Attack. In Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA (2002)
2. Perrig, A., Hu, Y-C., Johnson, D. B.: Wormhole Protection in Wireless Ad Hoc Networks, Technical Report TR01-384. Dep. Of Computer Science, Rice University (2001)

3. EC Directorate F, IST Framework, Future and Emerging Technologies, Situated and Autonomic Communications (COMS) - Communication Paradigms for 2020, available at http://www.cordis.lu/ist/fet/comms.htm
4. Chess, D. M., Palmer, C. C., and White, S. R.: Security in an autonomic computing environment. IBM Systems Journal, Vol. 42, No 1 (2003)
5. Abdul-Rahman, A., Hailes, S.: A Distributed Trust Model. in Proc. New Security Paradigms Workshop, ACM (1997)
6. Marti, S., Giuli, T. J., Lai, K., Baker, M.: Mitigating routing misbehaviour in mobile ad hoc networks. in Proc. of Mobicom2000, Boston, USA (2000)
7. Cahill, V. et. al.: Using Trust for Secure Collaboration in Uncertain Environments. IEEE Pervasive Computing, 2(3) (2003) 52-61
8. English, C., Terzis, S., Nixon, P.: Towards Self-Protecting Ubiquitous Systems: Monitoring Trust-based Interactions. Journal of Personal and Ubiquitous Computing, (2005), to appear
9. Castelfranchi, C., Falcone, R.: Trust is much more than subjective probability. Mental components and sources of trust. HICSS33, Hawaii (2000)
10. Michiardi, P., Molva, R.: 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 (2002)
11. Buchegger, S., Le Boudec, J-Y.: A Robust Reputation System for P2P and Mobile Ad-hoc Networks. in Proc. Second Workshop on the Economics of Peer-to-Peer Systems (2004)
12. Software-defined radio forum, http://www.sdrforum.org/
13. IST End-to-End Reconfigurability Project E2R, http://e2r.motlabs.com/
14. Jøsang, A.: The right type of trust for distributed systems, in Proc. of Workshop on New Security Paradigms, California, USA (1996)
15. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web, Scientific American (2001)
16. Gomez-Perez, A., Corcho, O., Fernandez-Lopez, M.: Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web, Springer (2004)
17. López de Vergara, J. E., et al.: Semantic management: Application of ontologies for the integration of management information models, IM 2003 - 9th IFIP/IEEE International Symposium on Integrated Network Management, vol. 9, no. 1, March 2003, pp. 131 - 134
18. Golbeck, J., Parsia, B., Hendler, J.: Trust networks on the semantic web, Proceedings of Cooperative Intelligent Agents (CIA), Helsinki, Finland (2003) 238-249
19. Kagal, L., Finin, T., Joshi, A.: A Policy Based Approach to Security for the Semantic Web, 2nd International Semantic Web Conference (ISWC2003), Florida, USA (2003)
20. Tonti, G., et al.: Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder. International Semantic Web Conference (ISWC 03), Florida, USA, 2003
21. Verma, D. C.: Policy-Based Networking–Architecture and Algorithms, New Riders (2000)
22. Neisse, R.: An Hierarchical Policy-Based Architecture for Integrated Management of Grids and Networks. Fifth IEEE POLICY Workshop, New York, USA (2004)
23. Semantic Web Trust and Security Resource Guide, www.wiwiss.fu-berlin.de/suhl/bizer/SWTSGuide/
24. Clark, D., Partridge, C., Ramming, J.C., Wroclawski, J.: A Knowledge Plane for the Internet. SIGCOMM '03, Karlsruhe, Germany (2003)
25. Marias, G., Tsetsos, V., Sekkas, O., Georgiadis, P.: Performance evaluation of a self-evolving trust building framework. IEEE SECOVAL Workshop, Athens, Greece (2005)