

Providing Trust in Wireless Sensor Networks using a Bio-Inspired Technique

Félix Gómez Mármol and Gregorio Martínez Pérez

Departamento de Ingeniería de la Información y las Comunicaciones

University of Murcia

Facultad de Informática, Campus de Espinardo, 30.071 Murcia, Spain

{felixgm, gregorio}@um.es

Abstract

Wireless Sensor Networks (WSNs) are becoming more and more spread and both industry and academia are focusing their research efforts in order to improve their applications. And one of the first steps to follow in order to achieve that expected improvement is to assure a minimum level of security in such a restrictive environment. Even more, ensuring confidence between every pair of interacting nodes is a critical issue in this kind of networks. Under these conditions we present in this paper an adjustment of a bio-inspired algorithm, called BTRM-WSN, based on ant colony systems aiming to provide trust in WSN. Experiments and results demonstrate the accuracy and robustness of the proposed model.

1 Introduction

WSN [1] are networks based on small size nodes cooperation. Those nodes are mainly characterized by their low energy consumption, their low cost and, of course, their wireless communication. They can be used to make measurements of temperature, pressure, humidity, lightness, etc, but currently they often have certain probabilities of failure, as well as high restrictions of computing, memory and energy capabilities.

WSN are usually composed of a large number of these nodes which, together with their highly dynamic topology, may lead to some scalability problems. A number of research groups are deeply working on them since they have several interesting applications covering from military ones to environmental ones, passing through sanitary applications, domotics, Intelligent Transportation Systems (ITS), etc.

And it is just because of their multiple application scenarios and their important restrictions that they usually suffer from many security weaknesses. Hardware failures could be a source of wrong critical information spreading. But even more, nodes belonging to a WSN could misbehave when they were asked for a measurement, or some data.

Without loss of generality, we will adopt the scheme where some nodes of the network request some services (and act, therefore, as clients) and some others provide those services (thus acting as servers or services providers). In such a scenario, a node could provide a fraudulent service when this is requested, even if it actually offered the desired one.

In addition, since we have supposed one of the most restrictive cases, where every sensor is only able to communicate with its direct neighbors (that is, it can not establish a direct communication with a node more than one hop ahead), a malicious node could avoid reaching its benevolent neighbors, or leading always to other malicious nodes, forming thus a collusion.

It is therefore necessary to accurately distinguish trustworthy nodes from fraudulent ones. This trustworthy nodes identification can be achieved through a trust and reputation model. In this paper we specifically present a trust model for WSN in order to carry out the selection of the most trustworthy node through the most reputable path offering a certain service.

Our proposed model is based on a bio-inspired algorithm called ant colony system (ACS) [2, 3, 4, 5], where ants build paths fulfilling certain conditions in a graph (equally, in a network). These ants leave some pheromone traces that help next ants to find and follow those routes.

Although ACS was initially mainly designed for static networks, experiments demonstrate that the adaptations done to make it suitable for WSN lead to an accurate performance of the model. As we will see later, it allows a client to interact most of the times with a trustworthy server, rather than with a misbehaving one.

The rest of the paper is organized as follows: section 2 presents a review of a number of trust and reputation models and works oriented to WSN. In section 3 we present our trust model proposal, describing its main features and analyzing some derived security threats. Experiments and results are exposed in section 4 and, finally, section 5 shows conclusions and future work.

2 Background and Related Work

In this section we will present a review of some of the most relevant and novel trust and reputation models over Wireless Sensor Networks.

2.1 ATRM

ATRM [6] is an agent-based trust and reputation management scheme for wireless sensor networks where trust and reputation management is carried out locally with minimal overhead in terms of extra messages and time delay.

It is based on a clustered WSN with backbone, and its core is a mobile agent system. It requires a node's trust and reputation information to be stored respectively in the forms of t -instrument and r -certificate by the node itself. In addition, ATRM requires that every node locally hold a mobile agent that is in charge of administrating the trust and reputation of its hosting node.

Considering any two nodes n_i and n_j , the t -instrument issued by n_i to n_j under context C_\star is defined as:

$$TI(n_i, n_j, C_\star) = E_{AK}(D, H(D))$$

where $E_{AK}(M)$ is an encryption function using n_i 's symmetric key, $H(M)$ is a hash digest function, $D = (ID(n_i), ID(n_j), C_\star, T, t_{i,j})$, T is a time-stamp implying the time when the t -instrument is issued and $t_{i,j}$ is the trust evaluation made by n_i on n_j .

If there are k concerned contexts, for any node n_i , its r -certificate is defined as:

$$RC(n_i) = E_{AK}(R, H(R))$$

where $R = (ID(n_i), T, ((r_1, C_1), (r_2, C_2), \dots, (r_k, C_k)))$, which means that n_i 's reputation is r_1 under context C_1 , r_2 under context C_2 , \dots , r_k under context C_k at time point T .

Before starting any transaction between n_i and n_j , the former asks its local mobile agent to obtain the r -certificate of the latter by directly querying n_j 's local mobile agent. Based on n_j 's r -certificate, n_i decides whether or not to start the transaction.

After the transaction is finished, n_i makes a trust evaluation on n_j based on the quality of the service it gets, and then submits this evaluation to its local mobile agent which then accordingly generates a t -instrument for n_j and sends it to n_j 's local mobile agent.

Based on the collected t -instruments, a mobile agent periodically issues its hosting node updated r -certificates. But since mobile agents are designed to travel over the entire network and run on remote nodes, they must be lunched by trusted entities. Therefore, in ATRM it is assumed that (1) there is a trusted authority that is responsible for generating and launching mobile agents, and (2) mobile agents are resilient against the unauthorized analysis and modification of their computation logic.

2.2 RRS for P2P and MANETs

In [7] authors present an enhancement of CONFIDANT [8], which is a robust reputation system for P2P and mobile ad-hoc networks where everyone maintains a reputation rating $R_{i,j}$ and a trust rating $T_{i,j}$ about everyone else they care about. From time to time first-hand reputation information $F_{i,j}$ is exchanged with others and only second-hand reputation information that is not incompatible with the current reputation rating is accepted, using a modified Bayesian approach.

Every node i keeps a summary record of first hand information about node j in a data structure called $F_{i,j}$, having the form (α, β) . Let's assume node i makes one individual observation about j and let be $s = 1$ if this observation is qualified as misbehavior, and $s = 0$ otherwise. Then the updating of first hand information is carried out as follows

$$\begin{aligned}\alpha &= u\alpha + s \\ \beta &= u\beta + (1 - s)\end{aligned}$$

where u is a discount factor for past experiences, which serves as the fading mechanism and is defined as $u = 1 - \frac{1}{m}$, being m the order of magnitude of the number of observations over which it is believed it makes sense to assume stationary behavior.

The reputation rating $R_{i,j}$ is also defined by two numbers, say (α', β') . Initially it is set to $(1, 1)$ and it is updated on two types of events: (1) when first hand observation is updated and (2) when a first hand information $F_{k,j}$ published by some other node k is accepted and copied.

In the former case, the update is the same as for the first hand information. In the latter case, if node i considers k trustworthy, then $F_{k,j}$ is considered by node i , who modifies $R_{i,j}$ according to

$$R_{i,j} = R_{i,j} + wF_{k,j}$$

Here, w is a small positive constant. Otherwise i considers k untrustworthy and uses the results of the deviation test as follows. Let $\mathbb{E}(\text{Beta}(\alpha, \beta))$ be

the expectation of the distribution $\text{Beta}(\alpha, \beta)$ and let $F_{k,j} = (\alpha_F, \beta_F)$ and $R_{i,j} = (\alpha', \beta')$. Thus, the deviation test is

$$|\mathbb{E}(\text{Beta}(\alpha_F, \beta_F)) - \mathbb{E}(\text{Beta}(\alpha', \beta'))| \geq d$$

where d is a positive constant (deviation threshold). If the deviation test is positive, the first hand information $F_{k,j}$ is considered incompatible and is not used. Else $F_{k,j}$ is incorporated as shown before.

The trust rating $T_{i,j}$ is equal to (γ, δ) . Initially $(\gamma, \delta) = (1, 1)$ and an update is performed whenever node i receives a report by some node k on first hand information about node j . Let $s = 1$ if the deviation test succeeds, and $s = 0$ otherwise. The trust rating $T_{i,j} = (\gamma, \delta)$ is updated by

$$\begin{aligned} \gamma &= v\gamma + s \\ \delta &= v\delta + (1 - s) \end{aligned}$$

where v is a discount factor for trust, similar to u .

Finally, how node i considers other nodes as trustworthy or misbehaving is carried out as follows:

$$\begin{cases} \text{normal} & \text{if } \mathbb{E}(\text{Beta}(\alpha', \beta')) < r \\ \text{misbehaving} & \text{if } \mathbb{E}(\text{Beta}(\alpha', \beta')) \geq r \end{cases}$$

$$\begin{cases} \text{trustworthy} & \text{if } \mathbb{E}(\text{Beta}(\gamma, \delta)) < t \\ \text{untrustworthy} & \text{if } \mathbb{E}(\text{Beta}(\gamma, \delta)) \geq t \end{cases}$$

2.3 PTM

PTM [9] is a decentralized trust model for pervasive dynamic open environments where human intervention has been minimized and a recommendation protocol has been defined. Authors also implemented the model using the J2ME Personal Profile [10].

Each node has its own key pair, a list of trustworthy and untrustworthy users, behavioral information and available certificates. Trust relationships are expressed through fuzzy logic, fulfilling certain properties, such as: reflexive, non-symmetrical, conditionally transitive and dynamic [11]. These relationships can be established as direct or indirect.

- In the first case, A will trust B without intervention of third parties. For that A takes into account some available previous knowledge about B , otherwise A will use an inference engine to interpret the established rules.

- The indirect trust relationships are given by recommendations from TTPs. A TTP is a peer who has a trust value higher than a certain threshold. Such recommendations are distributed using a pervasive recommendation protocol (PRP) among close entities or using public key certificates.

Once the trust relationship is established, A calculates B 's degree of trust, T_i :

$$T_i = \begin{cases} T_{i-1} + \omega \cdot V_{a_i}(1 - T_{i-1}) & \text{if } V_{a_i} > 0 \\ T_{i-1}(1 - \omega + \omega \cdot V_{a_i}) & \text{otherwise} \end{cases}$$

Where the strictness factor ω is related to the user's disposition regarding the present and the past, and V_{a_i} is computed according to the related weight to each kind of action, W_a , which is rewarded or punished according to the past behavior, both positive, a^+ , and negative, a^- , as follows:

$$V_{a_i} = W_{a_i}^{(m)} \cdot \frac{(a^+ - a^-)((a^+ - a^-) \cdot \sigma)^{2m}}{(a^+ - a^-)((a^+ - a^-) \cdot \sigma)^{2m} + 1}$$

σ is a value in the interval $(0, 0.05]$, which can be calculated from the security level $m \geq 1$.

The calculation of action value includes classic a priori probabilities about the user behavior. From a priori probabilities authors calculate posteriori ones applying the Bayes' theorem. The density function represents the distribution of these probabilities for binary events in the interval $[0, 1]$, so:

$$\beta(a^+ + 1, a^- + 1) = \int_0^1 X^{a^+} (1 - X)^{a^-} dx = \frac{\Gamma(a^+ + 1)\Gamma(a^- + 1)}{\Gamma(a^+ + a^- + 2)}$$

$$P(X|a^+, a^-) = \frac{a^+!(a^-)!}{(a^+ + a^- + 1)!}$$

The next Beta density function can be deduced from last equation:

$$f_x(X|H_{act}) = \frac{(a^+ + a^- + 1)!}{a^+!(a^-)!} X^{a^+} (1 - X)^{a^-}$$

where H_{act} is the historical behavior. This probabilistic model could be of use for evaluating the risk of a new interaction.

2.4 ATSN

An agent-based trust model for WSN is presented in [12] using a watchdog scheme to observe the behavior of nodes and broadcast their trust ratings. The sensor nodes receive the trust ratings from the agent nodes, which are responsible for monitoring the former and computing and broadcasting those trust ratings. According to the received information, sensor nodes will make the decision about cooperate with their neighbors or not.

In ATSN the reputation space is defined as $RS = \{ \langle p, n \rangle \mid p, n \in \mathbb{N} \}$, where p is the number of positive outcomes and n is the number of negative ones. Given $\langle p, n \rangle$ the probability x of obtaining a positive outcome is computed as follows:

$$P_{\langle p, n \rangle}(x) = P(x \mid \langle p, n \rangle) = \frac{x^p(1-x)^n}{\int_0^1 x^p(1-x)^n dx}$$

Additionally, the certainty of event $\langle p, n \rangle$ is calculated with the next expression:

$$c(p, n) = \frac{1}{2} \int_0^1 \left| \frac{x^p(1-x)^n}{\int_0^1 x^p(1-x)^n dx} - 1 \right| dx$$

Moreover, the trust space is defined as $TS = \{ (pt, nt, ut) \}$, satisfying the following conditions:

$$\begin{cases} pt, nt, ut \in [0, 1] \\ pt + nt = c \\ pt + nt + ut = 1 \end{cases}$$

where pt , nt and ut refer to positive trust, negative trust and uncertainty, respectively.

Let now $T = (pt, nt, ut)$ be the transformation from reputation space to trust space, where pt , nt and ut are computed according to the next formula:

$$\begin{cases} pt = c \frac{p+1}{p+n+2} \\ nt = c \frac{n+1}{p+n+2} \\ ut = 1 - pt - nt \end{cases}$$

2.5 RFSN

RFSN [13] is a framework where sensor nodes maintain reputation for other nodes in the network. A node monitors through a watchdog mechanism the behavior of other nodes, based on which it builds up their reputation over

time. It uses this reputation to evaluate trustworthiness and in predicting their future behavior. At the time of collaboration, a node only cooperates with those nodes that it trust.

Thus, a data structure termed reputation table RT_i is defined where reputations maintained by node i are stored.

$$RT_i = \{ R_{ij} \}$$

being R_{ij} the reputation of node j maintained by node i . A node builds each of these entries in the reputation table over time through the watchdog mechanism as follows

$$R_{ij} = f(D_{ij}, R_{ij})$$

where the output of the watchdog mechanism, D_{ij} , is used to recursively update the reputation of node j at node i . D_{ij} represents the rating that is allocated to the latest action of node j by node i .

Moreover, in RFSN the reputation of a node is a made up of two sub-components, $(R_{ij})_D$ and $(R_{ij})_{ID}$, as shown next

$$R_{ij} = (R_{ij})_D + (R_{ij})_{ID}$$

Direct reputation $(R_{ij})_D$ is build up using direct observations through the watchdog mechanism and indirect reputation $(R_{ij})_{ID}$ is build up using second hand information. But node i should give more weight to the second hand information received from a highly reputed node and vice-versa. Therefore, $(R_{ij})_D$ and $(R_{ij})_{ID}$ are computed as follows

$$\begin{aligned} (R_{ij})_D &= f(D_{ij}, (R_{ij})_D), \quad \forall j \in N_i \\ (R_{ij})_{ID} &= (R_{ij})_{ID} + w_{ik} \times R_{kj}, \quad \forall k \in N_i \end{aligned}$$

where $w_{ik} = g(R_{ik})$ represents the weight that is derived based on the reputation between the two nodes i and k , R_{ik} .

Trust is obtained in RFSN by taking the statistical expectation of the probability distribution representing the reputation between those nodes, i.e., $T_{ij} = E(R_{ij})$.

Finally, when faced with the question of cooperating with a node j in the network, the behavior of node i , B_{ij} , is derived from the trust metric of the two nodes. B_{ij} is a binary variable $\{ \text{cooperate}, \text{don't cooperate} \}$ and a simple threshold based policy is used to decide the value of B_{ij} .

2.6 CORE

CORE [14] is a generic mechanism based on reputation to enforce cooperation among the nodes of a MANET in order to prevent selfish behavior. All members of a community that shares resources have to contribute to the community life in order to be entitled to use those resources. In CORE, reputation is a good measure of someone's contribution to common network operations and it is defined as compositional.

That is, the overall opinion on an entity that belongs to the community is obtained as a result of the combination of different types of evaluations. Authors of CORE define a subjective reputation, an indirect reputation and a functional reputation.

The first one is the reputation calculated directly from a subject's observation, and the general formula to calculate it is:

$$r_{s_i}^t(s_j|f) = \sum \rho(t, t_k) \cdot \sigma_k$$

where $r_{s_i}^t(s_j|f) \in [-1, 1]$ stands for the subjective reputation value calculated at time t by subject s_i on subject s_j (which is a former's neighbor) with respect to the function f ; $\rho(t, t_k)$ is a time dependent function that gives higher relevance to past values of σ_k and $\sigma_k \in [-1, 1]$ represents the rating factor given to the k -th observation.

The indirect reputation of subject s_j collected by s_i at time t for the function f is denoted as $ir_{s_i}^t(s_j|f)$, and can take only positive values, preventing thus denial of service attacks based on malicious broadcasting of negative ratings for legitimate nodes.

Finally, the functional reputation refers to the subjective and indirect reputation calculated with respect to different functions f . All these types of reputation are combined to assess a global value of a subject's reputation, using the following formula:

$$r_{s_i}^t(s_j) = \sum w_k \cdot (r_{s_i}^t(s_j|f_k) + ir_{s_i}^t(s_j|f_k))$$

where w_k represents the weight associated to the functional reputation value and $r_{s_i}^t(s_j)$ is the global reputation value that is evaluated in every node. The choice of the weights w_k used to evaluate the global reputation has to be accurate because it can affect the overall system robustness.

Each entity s_i in CORE is enriched with a set of reputation tables (RT) and a watchdog mechanism (WD). Each row in the RT consists of four entries: the unique identifier of the entity, a collection of recent subjective

observations made on that entity's behavior, a list of the recent indirect reputation values provided by other entities and the value of the reputation evaluated for a predefined function. Each network entity has one RT for each function that has to be monitored. The RT and the WD together constitute the basis of the collaborative reputation mechanism presented in this model.

2.7 DRBTS

DRBTS [15] is a distributed security protocol aimed at providing a method by which beacon nodes (nodes that assist other sensor nodes to determine their location), BN, can monitor each other and provide information so that sensor nodes, SN, can choose who to trust, based on a quorum voting approach. In order to trust a BN's information, a sensor must get votes for its trustworthiness from at least half of their common neighbors.

Let's consider a WSN consisting of n SN, s_1, s_2, \dots, s_n and m BN, b_1, b_2, \dots, b_m . If a BN reports a trust value over a SN's threshold for another BN, the sensor counts that as a positive vote from the first BN to the second.

There are two classifications of information available for the reputation system. On the one hand, the first hand information is the location information transmitted by a BN, overheard by another BN in its communication range. On the other hand, the second hand information is the reputation information gathered by a BN and published while responding to a request for location information. Both these types of information are used by the BN to update the reputation of their neighbors.

The reputation of b_i from b_k point of view, $R_{k,i}$ is updated as follows:

$$R_{k,i} = \mu_1 \times R_{k,i} + (1 - \mu) \times \tau$$

If b_k believes that the location broadcasted by b_i is truthful, $\tau = 1$, otherwise $\tau = 0$. $\mu_1 \in [0, 1]$ is a factor to weight previous experience against current information.

When a node requests location information, every beacon neighbor of the requesting node will publish its Neighbor Reputation Table (*NRT*) along with its own location. Let's assume b_k is the publishing node and b_j receives $R_{k,i}$. Before incorporating $R_{k,i}$, b_j first performs a simple deviation test as follows:

$$|R_{j,i} - R_{k,i}| \leq d$$

If the above deviation test is positive, then the information is considered compatible with b_j 's first hand experience, and is accepted. b_j then updates $R_{j,i}$ in NRT_{b_j} as follows:

$$R_{j,i} = \mu_2 \times R_{j,i} + (1 - \mu_2) \times R_{k,i}$$

However, if the deviation test is negative, then the published information is considered to deviate too much from its own first-hand experience, and is disregarded as incompatible information. In order to discourage nodes from publishing false information, the lying node's reputation is decreased as follows:

$$R_{j,k} = \mu_3 \times R_{j,k}$$

Finally, DRBTS assumes an initial state of mistrust, that is, every new BN's reputation value is 0.

2.8 Discussion and motivation

In this section we have reviewed a number of works, projects and models related to the management of trust and reputation concepts in WSN. Some of them have even become one of the most known in this field [6, 7, 8, 14].

Nevertheless, not all of them take into account the strong restrictions about processing, storage or communication capabilities, in the same way. Even more, some of them just present a formal model without showing any set of experiments demonstrating the accuracy, robustness, scalability and overload introduced by their models in such a sentient environment.

Some of them rely on a watchdog mechanism with or without using a multi-agent system [12, 13]. Others take advantage of Bayes theorem [11] and a posteriori probabilities, or just use a Beta distribution [7] in order to represent ratings.

As far as we know our model is the first one in applying a bio-inspired technique such as ant colony system (ACS) to develop a trust and reputation model for WSN. Even more, it is the first one in providing the most trustworthy path (not only the most reputable node) leading to a specific sensor.

Likewise, we have taken into consideration the important limitations found in WSN, so we have tried to design a model as much lightweight, efficient, robust and scalable as possible. In fact we present two versions of our model, depending on the features of the WSN where it is to be deployed.

If we are facing a very restrictive network, a simpler model is proposed. This simpler and less resource consuming scheme is, however, more vulnerable to some security threats as we will see later. On the other hand, if we are dealing with a WSN whose nodes are devices with more capabilities and security is a very important issue, then we bet on another more sophisticated model with a small overload on the network.

3 Bio-inspired Trust Model for WSN

3.1 Assumptions/Scenario description

Several types of WSNs can be found depending on what kind of nodes they are composed of. You can meet from a static WSN where nodes have a certain location, to a highly mobile one where nodes move everywhere (like in a VANET [16]). You can also find from a very restrictive WSN where all nodes remain most of the time asleep in an idle state, to another one comprising nodes provided with high performance features capable of process many requests per second and that are nearly always active.

Throughout this paper we will assume a scenario where a WSN is composed of nodes with relatively high sensor activity. Without loss of generality, we will consider some nodes requesting generic services and some nodes providing them. In the future these services can be specified in detail. How this definition is carried out is out of the scope of this paper.

We will also assume that every node will only know its neighbors (that is, those nodes within its wireless range), and anything else about the whole topology of the net (at least at the early stages). Additionally, this topology is considered to be relatively highly dynamic, with many nodes entering or leaving the community. If this frequent logging in and out of nodes is due to the mobility of these nodes or because they switch on and off, is out of the scope of this paper, as well.

Our model is aimed to help a node requesting a certain service to the network to find the most trustworthy route leading to a node providing the right requested service. A node (equally a path) can be considered untrustworthy either because it intentionally provides a fraudulent service or because it provides a wrong one due to hardware failures or performance deterioration. As we mentioned above, we are considering dynamic topologies, so we needed to use a technic capable of dealing efficiently with this issue. And in our opinion, one mechanism that fulfills quite well this matter is the ant colony system (ACS).

3.2 ACS, a bio-inspired approach

Ant colony system (ACS) [2, 3, 4, 5] is a bio-inspired algorithm mainly used in optimization problems such as the travelling salesman problem [17] or the quadratic assignment problem [18]. It is based on the behavior of an ant colony in the nature.

Ants, when exploring the land in order to find the optimum path from the anthill to a certain target (said, some food), leave pheromone traces which are used by following ants to guide them through the same way. Furthermore, if an obstacle is present throughout the route, ants are able to avoid it finding again the new optimum path.

This algorithm is applied in those problems which can be modelled as graphs. Thus, a set of ants is launched and they start building paths fulfilling certain conditions (depending on the problem definition). There are several concepts to be addressed here, such as how the pheromone traces are updated, how an ant decides which next node to transit to or how to measure the quality of each path found in order to keep the best one.

```

1 for  $It = 1$  to Number_of_iterations do
2   for  $k = 1$  to Number_of_ants do
3      $S_k \leftarrow$  initial node
4
5   for  $i = 2$  to Number_of_nodes do
6     for  $k = 1$  to Number_of_ants do
7        $S_k \leftarrow S_k \cup$  Transition_Rule( $S_k, \tau, \eta, \alpha, \beta, q_0$ )
8       Pheromone_local_updating( $S_k, \varphi, \tau_0$ )
9
10    for  $k = 1$  to Number_of_ants do
11      if ( $Q(S_k) > Q(Current\_Best)$ ) then
12         $Current\_Best \leftarrow S_k$ 
13
14    if ( $Q(Current\_Best) > Q(Global\_Best)$ ) then
15       $Global\_Best \leftarrow Current\_Best$ 
16
17    for  $i = 1$  to Number_of_nodes do
18      Pheromone_global_updating( $Global\_Best, Q(Global\_Best), \rho$ )
19
20 return  $Global\_Best$ 

```

Algorithm 1: ACS, Ant Colony System

Algorithm 1 shows the basic general structure of the ant colony system algorithm, where S_k is the solution built by ant k , and $Q(S_k)$ is the quality of that path. τ stores the pheromone traces and η represents a heuristic values associated with each edge of the graph.

The transition rule for ants is defined as follows. Let ant k be at node r at a certain moment. The probability of moving towards node $s \in J_k(r)$, where $J_k(r)$ is the set of r 's neighbors not visited yet by ant k , is computed as:

$$p_k(r, s) = \begin{cases} \frac{[\tau_{rs}]^\alpha \cdot [\eta_{rs}]^\beta}{\sum_{u \in J_k(r)} [\tau_{ru}]^\alpha \cdot [\eta_{ru}]^\beta} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

However, it was demonstrated that using this transition rule led to a very slow convergence of the algorithm, so an additional transition rule was developed, as shown next:

$$s = \begin{cases} \arg \max_{u \in J_k(r)} [\tau_{ru}]^\alpha \cdot [\eta_{ru}]^\beta & \text{if } q \leq q_0 \\ \text{formulae (1)} & \text{otherwise} \end{cases} \quad (2)$$

where $q_0 \in [0, 1]$ is a constant and $q \sim [0, 1]$ is a random number within the interval $[0, 1]$. α and β are two values used to determine the balance between the pheromone traces and the heuristic information. So if $q \leq q_0$ is true, the most promising node is selected as the next step of ant k ; otherwise, that node is chosen using expression (1).

About pheromone updating, as it can be seen in algorithm 1, there are two kind of updates: a local and a global one. The pheromone local updating is carried out every time an ant moves from one node to another and it has the next appearance:

$$\tau_{rs}(t) = (1 - \varphi) \cdot \tau_{rs}(t - 1) + \varphi \cdot \tau_0 \quad (3)$$

where $\tau_{rs}(t)$ is the pheromone trace of edge e_{rs} at time t , $\varphi \in [0, 1]$ is a constant used to moderate the local updating and τ_0 is the initial value of pheromone when the algorithm is launched.

On the other hand, the pheromone global updating is only applied on those edges belonging to the best path found at the moment by all ants, and it is carried out through the following expression:

$$\tau_{cs}(t) = (1 - \rho) \cdot \tau_{cs}(t - 1) + \rho \cdot Q(Global_Best) \quad (4)$$

where $\rho \in [0, 1]$ is a constant used to define the pheromone global updating.

Finally, how to assess the quality of a solution depends entirely on the definition of the problem we are dealing with. In the travelling salesman problem, for instance, it would be defined as the total distance of the circuit composed by all the cities (nodes) of the map (graph).

In our particular case, we will need to redefine many topics since the default definition given above is not enough to accomplish the resolutions exposed in section 3.1.

3.3 BTRM-WSN

BTRM-WSN is a bio-inspired trust and reputation model for WSNs aimed to achieve to most trustworthy path leading to the most reputable node in a WSN offering a certain service. It is based on the bio-inspired algorithm of ant colony system but, due to the specific restrictions and limitations found in WSNs, the ACS can not be directly applied. Some adaptations, therefore, have to be made.

In our model, for instance, every node maintains a pheromone trace for each of its neighbors. This pheromone traces $\tau \in [0, 1]$ will determine the probability of ants choosing a certain route or another, and can be seen as the amount of trust given by a node to other one. The heuristic values $\eta \in [0, 1]$, however, are defined as the inverse of the delay transmission time between two nodes (or the inverse of the distance between them).

The fact that every node controls its own pheromone traces and heuristic values, and no one else but it can modify them can become an important security threat.

Other issue that avoids the direct application of the ACS in this environment is the fact that while an ant is searching for the most reputable server providing a requested service, it could happen that some of the nodes that form the path followed by that ant become inaccessible (either because they switch off or because they move out of the range of their previous sensor in the path). In that situation, the ant would be unable to come back to the client and it would get lost. In other words, when a client launches a set of ants, it has no guarantee at all that all of them are going to return and, of course, it can not wait until all the launched ants came back in one iteration of the algorithm.

Therefore, the algorithmic scheme presented before has to be redefined as shown in algorithm 2.

```

1 while (condition) do
2   for  $k=1$  to Number_of_ants do
3      $S_k \leftarrow$  initial sensor (client)
4     Launch ant  $k$ 
5
6   do
7     for every returned ant  $k$  do
8       if ( $Q(S_k) > Q(Current\_Best)$ ) then
9          $Current\_Best \leftarrow S_k$ 
10    while (timeout does not expire) and
11      (Number_returned_ants < %Number_of_ants)
12
13    if ( $Q(Current\_Best) > Q(Global\_Best)$ ) then
14       $Global\_Best \leftarrow Current\_Best$ 
15
16    Pheromone_global_updating( $Global\_Best, Q(Global\_Best), \rho$ )
17
18 return  $Global\_Best$ 

```

Algorithm 2: BTRM-WSN

The first change we can appreciate is that the main bucle is now defined by a generic condition, which may be a certain number of iterations (like in the original algorithm) or it can even be a certain timeout. This definition will depend on the specific WSN this model is going to be applied to.

On the other hand, this algorithm consists of the following steps:

1. Every ant adds the first sensor to its solution, which is always the client they are departing from. Then each ant decides which next sensor to move to according to the transition rule and it is sent there (lines 2-4).
2. Once every ant has left the client, this one waits until they come back. For every returned ant, the client compares its solution and keeps the best one. As explained before, in a WSN the client has no guarantee that all the ants that were launched are going to come back, so it just waits until a timeout expires or a certain percentage of all the ants has returned (lines 6-11).
3. The best solution found by all or some of the ants issued in the current iteration is compared with the global best solution and swapped if it is appropriate (lines 13-14).

4. Finally, a pheromone global updating is performed over the links belonging to the global best path (line 16).

As explained before, the definition of the condition shown in line 1 of algorithm 2, as well as the ones defined in lines 10 and 11, depend directly on the specific features (bandwidth, transmission delay, etc.) of the sensors that compose the WSN we are dealing with.

Next we will describe in detail some features of our trust and reputation model for WSN, such as how to measure the quality of a path, how an ant decides which next sensor to travel towards, or when it should stop and return the current path. We will also explain how the pheromone updating is carried out while ants are building their routes as well as how a punishment is performed (in terms of pheromone evaporation) when the client interacts with a fraudulent server.

Additionally, the differences between the two proposed versions of our model will be explained and some final remarks about the scalability and lightness of BTRM-WSN will be shown.

3.3.1 Path quality

Each time a launched ant returns to its client carrying a solution with it, that client has to assess the quality of that solution. Specifically the ant keeps a list of all the sensors belonging to the selected path, together with the pheromone traces of the links that join them.

According to this, the path quality computation can be done in the following way:

$$Q(S_k) = \frac{\bar{\tau}_k}{\sqrt{Length(S_k)}} \cdot \%A_k$$

where $\bar{\tau}_k$ is the average pheromone of the path found by ant k and $\%A_k$ represents the percentage of ants that have selected the same solution as ant k .

On one hand, the amount of ants that in one iteration has selected the same path as ant k , and the reputation of that path, represented by its average pheromone, contribute both to have a greater quality solution. On the other hand, on equal conditions, a shorter path is preferred.

With a definition like this we achieve that our model tends to select those paths which are as short as possible and which have been selected as many times as possible.

3.3.2 Ants transition and stop condition

When an ant is travelling along the WSN searching for the most trustworthy route leading to the most reputable server it has to decide at each sensor which of its neighbors it has to move to. Every ant has also to decide whether to stop when it finds a server offering the requested service or if it should keep trying to find a more reputable one.

So let ant k be at sensor s in a certain moment of its searching. Several options can happen:

1. Sensor s offers the requested service.
 - (a) Sensor s has more neighbors not visited yet by ant k .
The average pheromone of the path followed by ant k from the client until the sensor s is computed, $\bar{\tau}_k \in [0, 1]$. If $\bar{\tau}_k$ is greater than a certain transition threshold, $TraTh \in [0, 1]$, then ant k stops and returns current solution with a probability defined by $\bar{\tau}_k$. Otherwise, if $\bar{\tau}_k \leq TraTh$, ant k considers sensor s not enough reputable and keeps trying to search a better one.
 - (b) Sensor s has no more neighbors or all of them have been already visited by ant k .
Ant k stops and returns current path.
2. Sensor s does not offer the requested service.
 - (a) Sensor s has more neighbors not visited yet by ant k .
Ant k decides which next sensor to move to according to the expression shown in equation (2).
 - (b) Sensor s has no more neighbors or all of them have been already visited by ant k .
In this situation ant k has reached a dead end and has no more options than backtracking. That is, it has to follow the inverse route it has currently built until it is at a sensor which offers the requested service (and then stops and returns that path) or until it reaches a sensor not offering the requested service but with more alternative paths not explored yet by ant k (and then keeps trying those routes). It could even happen that, while backtracking, ant k reached the client it belonged to. In that situation the whole WSN would have been explored but any server offering the requested service would have been found.

However, in order to prevent some security threats a client can not interact again with the same malicious server in the next transaction, so ants will not stop when they find it and consequently they will not choose it.

Another important issue to take care about is the number of launched ants, $N_{ants} \in \mathbb{N}$, which depends on the number of sensors that form the WSN and the dynamism of the WSN itself. It is sensible to think that the greater and the more dynamic a WSN is, the greater has to be the number of launched ants (because some of them can be lost), and vice versa.

But if the number of ants is relatively high, maybe the condition defined in line 1 of algorithm 2 should not lead to a too big number of iterations or a too large timeout. Otherwise, each execution of BTRM-WSN would require an amount of time and resources consumption that may not be acceptable in certain WSNs. Therefore, an accurate balance between the number of iterations (or timeout) and the number of ants is necessary in order to achieve reasonably good outcomes.

3.3.3 Pheromone updating

As explained before, while ants are travelling across the WSN searching the most reputable server, they modify the pheromone traces they find. This modification helps next ants to decide which path is better to follow.

Actually, there are two kind of updatings: a local and a global one. The pheromone local updating is carried out by every ant each time it decides to move from one sensor to the next. Let ant k be at sensor s_1 . Then, applying the transition scheme explained in the previous section, it decides to move towards sensor s_2 (which is a s_1 's neighbor). So, before being actually transmitted, it indicates sensor s_1 that it has to modify its pheromone trace associated with sensor s_2 in the following way:

$$\tau_{s_1 s_2} = (1 - \varphi) \cdot \tau_{s_1 s_2} + \varphi \cdot \Omega \quad (5)$$

where $\Omega = (1 + (1 - \varphi) \cdot (1 - \tau_{s_1 s_2} \eta_{s_1 s_2})) \cdot \tau_{s_1 s_2}$ is the convergence value of $\tau_{s_1 s_2}$ when $t \rightarrow \infty$ (given that $\tau, \eta, \varphi \in [0, 1]$), that is, is the pheromone trace value that would have that link after a lot of time if no other modification was carried out over it (notice that $\Omega \in [\tau_{s_1 s_2}, 2 \cdot \tau_{s_1 s_2}]$).

On the other hand, a pheromone global updating is performed over the best path found by all ants in each iteration of algorithm 2 (see line 16). This is done by sending an extra ant just to modify the pheromone traces of that route. And that modification is carried out using the next expression:

$$\tau_{rs} = (1 - \rho) \tau_{rs} + \rho (1 + \tau_{rs} \eta_{rs} Q(S_{Global_Best})) \tau_{rs} \quad (6)$$

Therefore, the higher are the pheromone trace, the heuristic value, and the quality of the path, the higher is the additional pheromone contribution over the best route.

Finally, it is worth to mention how to initialize the pheromone traces. Their initial value $IniPh \in [0, 1]$ will condition some aspects of the model. Thus, if $IniPh \rightarrow 0$, for instance, everybody would mistrust everyone at the beginning and it would be difficult to distinguish trustworthy sensors from malicious ones. However, if $IniPh \rightarrow 1$ then everybody would trust everyone at the beginning and it would also be difficult to distinguish benevolent sensors from fraudulent ones. Therefore we decided that a good initialization value could be a random value close to $IniPh$, with $IniPh \rightarrow 0.5$.

3.3.4 Punish & reward

Once BTRM-WSN has selected what it thinks is the most trustworthy path leading to the most reputable server, the client actually requests the desired service to that server. Then, depending on the goodness of the server, it will provide the same service it was offering, or another worse.

In this first stage we will consider only two possibilities. The server can be totally benevolent and provide the same service it was offering (so the client is fully satisfied), or it can be totally fraudulent and provide a completely different service than the one that was offered (having thus a fully unsatisfied client).

If the client is satisfied, a reward by means of additional pheromone contribution is done all along the selected path. The same expression used for pheromone global updating (equation (6)) can be applied here as well.

Nonetheless, if the client is not satisfied, a punishment, i.e., an evaporation of pheromone traces of the links belonging to the selected path, is carried out. And this punishment uses the following expression:

$$\tau_{rs} = (\tau_{rs} - \varphi \cdot df_{rs}) \cdot \frac{Sat}{df_{rs}} \quad (7)$$

where $Sat \in [0, 1]$ represents the satisfaction of the client with the received service and $df_{rs} \in (0, 1]$ is a distance factor of link e_{rs} computed as follows:

$$df_{rs} = \sqrt{\frac{d_{rs}}{L(S_k) \cdot (L(S_k) - d_{rs} + 1)}}, \quad d_{rs} \in \{1, 2, \dots, L(S_k)\} \quad (8)$$

being d_{rs} the actual distance (number of hops) between sensor r and s , and $L(S_k)$ the length of the path found by ant k .

As it can be checked, having a punishing scheme like this, those edges which are closer to the client have a slighter pheromone evaporation, and vice versa.

Furthermore, all the links that fall into the malicious server are also punished. Otherwise ants could select it again through an alternative path, thinking it has become a benevolent sensor (which may not happen most of the times). Therefore, those edges have to be punished according to the next formula:

$$\tau_{rs} = (\tau_{rs} - \varphi) \cdot Sat \quad (9)$$

3.4 Two proposed models

As we have mentioned before, we have actually developed two versions of our model BTRM-WSN. The first one is the one we have been showing until now, where pheromone traces are shared for every service offered by a sensor. This allows us to achieve a lighter model (very low overload is added to the network); however, it also has some drawbacks. For instance, with a model like this, a client could not distinguish a sensor which is very good (benevolent) when supplying a certain service, but very bad (fraudulent) providing another one. It will consider that sensor as very trustworthy or untrustworthy for all the services provided.

If we have a WSN where we are only interested on monitoring the behavior of sensors about just one service (or even if the WSN only provides one service), we could use this model without the problem of distinguishing a sensor's particular behavior for each provided service.

But if we need a more resilient model, capable of dealing with multiple services, we could adopt the second version of BTRM-WSN. In this one, every sensor has a pheromone trace for each one of its neighbors, and for each one of the services provided by the WSN.

Let be m the number of services available in the WSN, and let be n_s the number of neighbors of sensor s . Then, s should manage and store $m \times n_s$ different pheromone traces. Obviously, this decision implies a bigger amount of stored information on each sensor but, on the other hand, it provides a more resilient trust and reputation model, since this is now able to distinguish each sensor as trustworthy or not, for each one of the services it offers.

If we are dealing with a WSN with high-resources sensors and where the security is a critical issue when applying for a service, we could make use of this second version of the model.

Additionally, in this second version the client gathers all the paths found by all the ants that visit it (not only its own ants) and join them in order to achieve a local view of the topology of the network (which will probably be an instantaneous view, due to the high dynamism that this kind of networks can reach). This local view can help the client to take more intelligent decisions, since it knows (through the pheromone traces) which servers are more reputable and which not.

3.5 Scalability and lightness

One of the strong points of our trust and reputation model is its scalability. In this kind of networks, whose size can vary from a handful of nodes until thousands of them, developing a scalable model is a critical issue.

Since in our model every sensor manages and controls its own pheromone traces and there is not any central entity (like a watchdog) gathering ratings or supervising all or a subset of the sensors, we can state that BTRM-WSN is scalable.

Even more, if needed, every ant could be provided with a TTL (Time-To-Live), i.e., a maximum number of hops it is able to travel. Notice that this TTL would also limit the maximum length of any solution. Even so, if a client launched a set of ants with a TTL which did not allow them to reach any server (or all the reached servers were malicious), the client could increase that TTL and launch a new set of ants.

About the lightness of the model, we have seen in the previous section that we have two versions of the model. But even the second one, where every sensor s has $m \times n_s$ pheromone traces, does not add too much overload to the network. Moreover, each transmitted ant carries a list of sensors' identifications (which can be just a number) with their corresponding pheromone traces. And since the solutions average length rarely exceeds 5 or 6 hops, the information transmitted with every ant does not involve a big overload.

Of course, the overload introduced will also depend on the number of ants travelling through the WSN. As we explained in section 3.3.2, the number of ants depends on the number of sensors composing the network. Thus, we defined the number of ants as $N_{ants} = \lceil N_s^{0.4} \rceil$, where N_s is the number of sensors belonging to the same WSN. With a definition like this we achieve quite good outcomes with a minimum overload.

The accuracy and robustness of BTRM-WSN will be demonstrated in section 4 where experiments and results will be shown.

4 Experiments and results over static WSNs

Once we have shown in detail the description of our trust and reputation model over WSN, it is time to demonstrate its accuracy, scalability and robustness.

To do so, we have developed a whole testbed focused on three main targets. We are interested in finding out how many times our model is able to select the right benevolent server to interact with. In other words, we would like to know the selection percentage of trustworthy servers.

Since our model has a strong basis on random or probabilistic decisions, we considered that it would be also quite interesting to take care about the standard deviation of that selection percentage of trustworthy servers.

Finally, as a possible measure of the adaptability of our model specifically to WSN, we gathered as well the average path length of the solutions found by our model. As we mentioned before, in an environment with so many restrictions like WSN, the shorter path is always preferred since it supposes less consumption of sensors' resources.

The experiments we carried out had the following structure. We launched our model 100 times (i.e. each client applied for a service 100 times) over 200 random WSNs, each one composed of 100 sensors. On each network, the percentage of sensors acting as clients was always a 15%. The 85% left were, therefore, sensors acting as servers.

We tried with 200 random WSNs having a 10% (over the 85% left) of malicious servers. 200 with 20%, other 200 with 30%, and so on until a 90% of malicious servers (the worst simulated situation).

But even more, we repeated those experiments over WSNs composed of 200, 300, 400 and 500 sensors (with the same percentages of clients, servers and malicious servers). In this first stage all the tested WSNs were static.

4.1 Selection percentage of trustworthy servers

So the first and main focus was to evaluate the selection percentage of trustworthy servers achieved with BTRM-WSN. The outcomes corresponding to this experiment are shown in figure 1.

The first appreciation that can be done is the similarity of the selection percentages regardless the size of the network, which constitutes a demonstration of the scalability of our model. Outcomes slightly differ from one set of random WSNs to another when we fix the percentage of malicious servers and vary the number of sensors belonging to the same WSN.

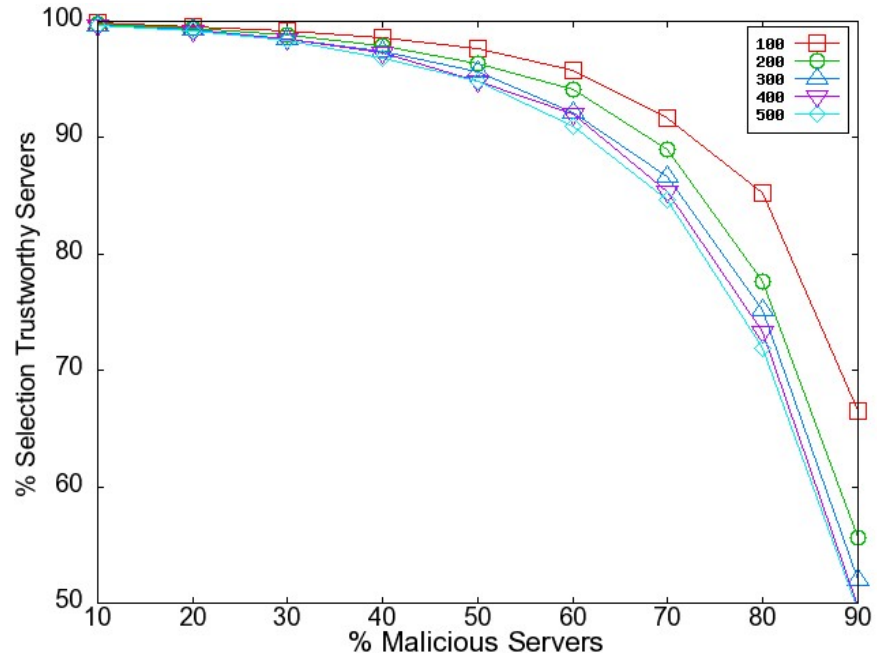


Figure 1: Selection percentage of trustworthy servers

Another conclusion that can be obtained is that the selection percentage is quite high (above the 90%) when the percentage of malicious servers is less than or equal to 50%, in every case.

In order to consider a trust and reputation model as acceptable (with a minimum quality level), in our opinion, the selection percentage of trustworthy servers should be greater or at least equal to 70%. A smaller percentage would result in a model with certain security deficiencies. And what is clear is that a selection percentage below the 50% means that the model is not useful at all.

Our experiments have shown that BTRM-WSN remains resilient to a high percentage of malicious servers when this percentage is less than or equal to 80% (which is, however, a high percentage). Its performance gets worse when there are 90% or more malicious servers in the WSN, and the problem intensifies when the size of the WSN grows. So, in smaller WSNs our model would work properly despite of a high percentage of malicious servers.

4.2 Standard deviation of the selection percentage of trustworthy servers

It is also important to realize that by testing our model against a number of random WSNs, each of them has a random topology, so it could happen that our model was tested against networks where benevolent servers were very near to the clients (maybe one hop forward and, consequently, very easy to solve) or quite the contrary, that is, WSNs where benevolent sensors were quite far from the clients.

Figure 1 actually depicts the average selection percentage of trustworthy servers in BTRM-WSN. But an average selection percentage of 80%, for instance, could be reached because the model always found a trustworthy server the 80% of the times, or just because it found it the 100% of the times in half the tested wireless sensor networks and the 60%, in the other half, for example.

That is the reason why we decided to measure and show the standard deviation related to that average as well. And the outcomes can be checked in figure 2.

Again, the first observation that can be done has to do with the similarity between the five graphics corresponding to the five tested sizes for WSNs. And here the standard deviation also remains quite low and nearly undistinguishable among the five tested sizes where the percentage of malicious servers is less than or equal to 50%.

This means that when there are less than or equal to 50% of malicious servers in the network, regardless its size, BTRM-WSN is able to select a high percentage of trustworthy servers (as shown in figure 1) with a quite high accuracy, regardless the topology of the WSN.

The standard deviation obtained in each set of wireless sensor networks of the same size begins to diverge when the percentage of malicious servers composing them is greater than or equal to 60%. And it can be also checked that this standard deviation is greater when the size of the network is smaller and vice versa. It also increases as the percentage of malicious servers also increases.

In fact the highest value among all the experiments carried out is obtained when we tried our model over 200 random WSNs (100 times on each one), composed of 100 sensors, with a 15% of clients and a 90% of malicious servers (a 90% of the 85% left). In that experiment our model was able to reach a trustworthy server in the 66.54% of the times, with a standard deviation of 16.33%.

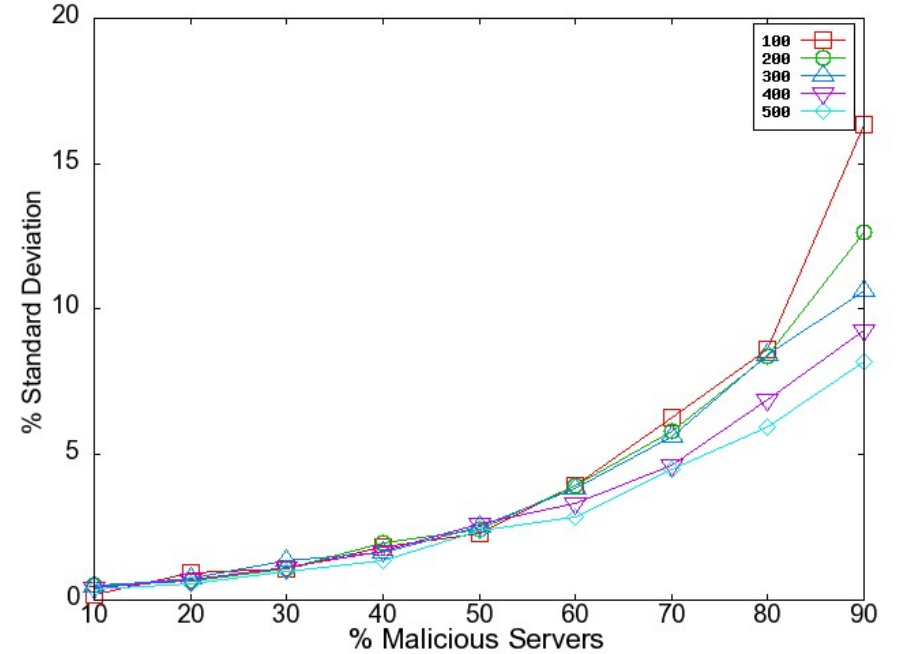


Figure 2: Standard deviation of the selection percentage of trustworthy servers

So if the percentage of malicious servers is high (greater than or equal to 60%, for instance), and the number of sensors composing the networks is also high, then the percentage selection of trustworthy servers is lower but, however, more accurate (i.e., BTRM-WSN is more independent of the topology), and vice versa.

This means that if the random tested WSNs size is not too high, those networks topology can vary from ones where BTRM-WSN works quite fine to others where it is hardly able to find the most trustworthy server. Nevertheless, if the size of the random tested networks is high, their topologies drive the model behaving most of the times in the same way (most of the times well, or most of the times not).

4.3 Average path length leading to trustworthy servers

Finally, the last developed experiment consisted of measuring the length (number of hops) of those paths found by BTRM-WSN leading to trustworthy servers. That is, when the model fails and selects an untrustworthy server, that path is discarded and not taken into account.

Doing this way we are able to estimate the average path length of those paths found by our model when it successfully reaches a benevolent server.

Our model is aimed to find the closest benevolent servers to the client requesting the service. On the one hand we think that the lesser number of intermediaries present in a transaction, the more secure and robust it can be performed. On the other hand, due to the specific restrictions related to wireless sensor networks, the resources consumption saving is a critical issue. Therefore, a shorter path leading to the final trustworthy server implies less involved sensors and, consequently, less global utilization of resources such as energy or bandwidth.

The outcomes of this experiment are presented in figure 3.

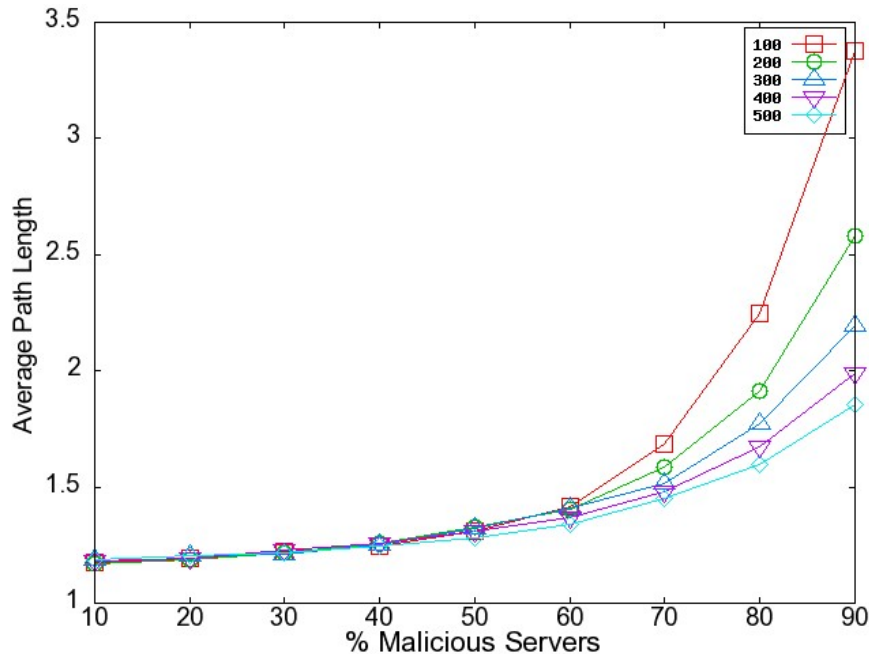


Figure 3: Average path length leading to trustworthy servers

As it can be observed, any trustworthy server is never reached (on average terms) at more than 4 hops. In fact, the highest average path length is achieved with 100 sensors WSNs with a 90% of malicious servers. In that situation, the average path length takes the value 3.3766.

One more time, differences between the several sizes tested for WSNs become distinguishable when the percentage of malicious servers is greater than or equal to 60%. Under this percentage, the average number of hops is quite low (near to 1), as it can be checked in the figure.

Therefore, our model is able to reach nearby trustworthy servers regardless the size of the network and the percentage of malicious servers. Although the smaller is the former and the greater is the latter, a larger path is found.

5 Conclusions and Future Work

Managing trust and reputation in Wireless Sensor Networks (WSN) in an efficient, accurate and robust way has not been completely solved yet.

In this paper we have proposed a Bio-inspired Trust and Reputation Model for WSNs, called BTRM-WSN. It is based on the Ant Colony System (ACS) and a complete description of its main features has been shown. We have seen how the pheromone traces deposited by ants help following ants to find the most trustworthy server through the most reputable path all over the network.

Specifically we have explained how the pheromone updating is carried out, as well as how to measure the quality of a path or how to punish or reward a server depending on its behavior. We have described the ants transition and stop condition scheme followed in our model, too.

A set of experiments over static WSNs (not changing its topology along the time) have been carried out. The outcomes achieved in the three developed experiments demonstrate that BTRM-WSN fulfills quite well the initially stated expectations about security, scalability and lightness in WSNs.

This paper opens, however, many future ways. For instance, a detailed description of some security threats that could be applied here can be an interesting issue. By managing each sensor its own pheromone traces, a malicious one could always assign the maximum value to other malicious neighbors or, equally, the minimum value to other benevolent ones.

We are also considering testing our model against dynamic WSNs where sensors can enter and leave the network (maybe because they switch on and off, or because they move out of the range of their neighbors), and

against oscillating networks, where the goodness of a server can vary along the time.

We would also like to configure a collusion scenario where a set of malicious servers work altogether in order to thwart the system and check the performance of BTRM-WSN in such a situation.

A visual simulator is currently being developed in order to carry out all the proposed experiments. Our intention is to make it as much generic as possible, so it can be easily used in order to test any other trust and reputation model over WSNs.

Acknowledgements

This work has been partially funded by the MISTRAL (Middleware de gestión de Identidades de Seguridad en TRansacciones electrónicas basado en código Libre, TIC-INF 07/01-0003) project. It has been also partially funded by SWIFT (Secure Widespread Identities for Federated Telecommunications, FP7-ICT-2007-1, Grant No.: 215832) EU IST project, and by a Séneca Foundation grant within the Human Resources Researching Training Program 2007 (06826/FPI/07). Thanks also to the Funding Program for Research Groups of Excellence with code 04552/GERM/06 granted by the Séneca Foundation.

References

- [1] K. Römer, F. Mattern, The Design Space of Wireless Sensor Networks, *IEEE Wireless Communications* 11 (6) (2004) 54–61.
- [2] M. Dorigo, L. Gambardella, Ant colony system: A cooperative learning approach in the traveling salesman problem, *IEEE Transaction on Evolutionary Computing* 1 (1) (1997) 53–66.
- [3] M. Dorigo, L. Gambardella, M. Birattari, A. Martinoli, R. Poli, T. Stützle, Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006, Vol. 4150 of LNCS, Springer, Brussels, Belgium, 2006.
- [4] O. Cordón, F. Herrera, T. Stützle, A review on the ant colony optimization metaheuristic: Basis, models and new trends, *Mathware and Soft Computing* 9 (2-3) (2002) 141–175.
- [5] M. Dorigo, T. Stützle, Ant Colony Optimization, Bradford Book, 2004.
- [6] A. Boukerche, L. Xu, K. El-Khatib, Trust-based security for wireless ad hoc and sensor networks, *Computer Communications* 30 (11-12) (2007) 2413–2427.
- [7] S. Buchegger, J. Y. Le Boudec, A Robust Reputation System for P2P and Mobile Ad-hoc Networks, in: *Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems*, Cambridge MA, USA, 2004.
- [8] S. Buchegger, J.-Y. L. Boudec, Performance analysis of the CONFIDANT protocol: Cooperation of nodes, in: *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, IEEE, Lausanne, CH, 2002.
- [9] F. Almenárez, A. Marín, D. Díaz, J. Sánchez, Developing a model for trust management in pervasive devices, in: *PERCOMW '06: Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*, IEEE Computer Society, Washington, DC, USA, 2006, p. 267.
- [10] D. Wilding-Mcbride, *Java Development on PDAs: Building Applications for Pocket PC and Palm Devices*, Addison-Wesley Longman Publishing Co., Inc., 2003.
- [11] F. Almenárez, A. Marín, C. Campo, C. García, PTM: A pervasive trust management model for dynamic open environments, in: *Privacy and Trust, First Workshop on Pervasive Security and Trust*, Boston, USA, 2004.
- [12] H. Chen, H. Wu, X. Zhou, C. Gao, Agent-based Trust Model in Wireless Sensor Networks, *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, SNPD 03 (2007) 119–124.
- [13] S. Ganeriwal, M. B. Srivastava, Reputation-based framework for high integrity sensor networks, in: *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, ACM, New York, NY, USA, 2004, pp. 66–77.
- [14] P. Michiardi, R. Molva, CORE: A Collaborative Reputation Mechanism to enforce node cooperation in Mobile Ad hoc Networks, in:

Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security, Kluwer, B.V., Deventer, The Netherlands, 2002, pp. 107–121.

- [15] A. Srinivasan, J. Teitelbaum, J. Wu, DRBTS: Distributed Reputation-based Beacon Trust System, in: DASC '06: Proceedings of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing, IEEE Computer Society, Washington, DC, USA, 2006, pp. 277–283.
- [16] F. Li, Y. Wang, Routing in vehicular ad hoc networks: A survey, *Vehicular Technology Magazine, IEEE* 2 (2) (2007) 12–22.
- [17] L. M. Gambardella, M. Dorigo, Solving symmetric and asymmetric TSPs by ant colonies, in: *International Conference on Evolutionary Computation*, 1996, pp. 622–627.
- [18] L. Gambardella, E. Taillard, M. Dorigo, Ant Colonies for the QAP, *Journal of the Operational Research Society* 50 (1999) 167–176.
- [19] S. P. Marsh, Formalising trust as a computational concept, Ph.D. thesis, Department of Computing Science and Mathematics, University of Stirling (apr 1994).
- [20] S. Marti, H. Garcia-Molina, Taxonomy of trust: Categorizing P2P reputation systems, *Computer Networks* 50 (4) (2006) 472–484.

About the Authors

Félix Gómez Mármol is a PhD student in the Department of Information and Communications Engineering of the University of Murcia. His research interests include authorization, authentication and trust management in distributed and heterogeneous systems, security management in mobile devices and design and implementation of security solutions for mobile and heterogeneous environments. He received an MSc in computer engineering from the University of Murcia. Contact him at felixgm@um.es

Gregorio Martínez Pérez is an associate professor in the Department of Information and Communications Engineering of the University of Murcia. His research interests include security and management of IPv4/IPv6 communication networks. He received an MSc and PhD in computer engineering from the University of Murcia. Contact him at gregorio@um.es