Telecommun Syst (2011) 46: 163–180
DOI 10.1007/s11235-010-9281-7

Providing trust in wireless sensor networks using a bio-inspired technique

Félix Gómez Mármol · Gregorio Martínez Pérez

Published online: 18 February 2010

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. One of the first issues to solve 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 a bio-inspired trust and reputation model, called BTRM-WSN, based on ant colony systems aiming at providing trust and reputation in WSNs. Experiments and results demonstrate the accuracy, robustness and lightness of the proposed model in a wide set of situations.

Keywords Trust & reputation management · Wireless sensor networks · Bio-inspired algorithms

1 Introduction

WSNs [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.

WSNs 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 working on them since this kind of networks has several interesting applications ranging from military ones to environmental ones, passing through sanitary applications, domotics, Intelligent Transportation Systems (ITS) [2], etc.

However, due to their important restrictions, they usually suffer from many security weaknesses, which make them often vulnerable to certain threats. Hardware failures could be a source of wrong critical information spreading, for instance. But even more, nodes belonging to a WSN could misbehave when they are 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.

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 cannot 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 [3, 4].

F. Gómez Mármol (✉) · G. Martínez Pérez
Departamento de Ingeniería de la Información y las Comunicaciones, University of Murcia, 30.071 Murcia, Spain
e-mail: felixgm@um.es

G. Martínez Pérez
e-mail: gregorio@um.es

In this paper we specifically present a trust and reputation model for WSNs, called BTRM-WSN (Bio-inspired Trust and Reputation Model for Wireless Sensor Networks) 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) [5–8], where ants build paths fulfilling certain conditions in a graph. 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 WSNs 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: Sect. 2 presents a review of a number of trust and reputation models and works oriented to WSNs. In Sect. 3 we present our trust model proposal, describing its main features and characteristics. An interesting analysis of some derived security threats that could be applied in our model is shown in Sect. 4. Experiments and results are exposed in Sect. 5 and, finally, Sect. 6 shows our 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 [9] is an agent-based trust and reputation management scheme for WSNs 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 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 $C_\bigstar$ is defined as:

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

where $E_{AK}(M)$ is an encryption function using n_i's symmetric key, $H(M)$ is a hash digest function, $D = (ID(n_i), ID(n_j), C_\bigstar, T, t_{i,j})$, 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 $R = (ID(n_i), T, ((r_1, C_1), (r_2, C_2), \ldots, (r_k, C_k)))$, which means that n_i's reputation is r_1 under context C_1, r_2 under context C_2, …, 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 QDV

Authors of [10] present an Ant Colony Optimization approach for reputation and quality-of-service-based security in WSNs. They specifically propose a quality-based distance vector protocol known as QDV, where the more reputation a node has, the more reliable it is for communication purposes.

QDV is able to protect the network against packet injection by those malicious nodes which have been detected. This protection is made by identifying those nodes who drop the packets forwarded to them.

In this model reputation is based on pheromone content of a path for communication. Thus, a path having more deposits of pheromone, τ_{ij}, is considered more secure. On the other hand, QoS considers the distance between two communicating nodes, η_{ij}. Therefore:

$$\phi_{ij}(t) = \frac{\sum_{k=1}^{n_i} \tau_{kj}}{n_i}$$

where τ_{kj} is the pheromone trace between nodes k and j, n_i is the number of i's neighbors, and if $\phi_{ij}(t) < \tau_{\min}$, misbehavior or security violation is detected, which means node i has less forwarding capabilities.

In the same direction, QoS is defined as the percentage of exposed traffic according to:

$$\theta_{ij}(t) = \frac{\sum M_g(t) + \sum M_r(t) - \sum M_d(t)}{\sum M_g(t) + \sum M_r(t)}$$

being $\sum M_g(t)$, $\sum M_r(t)$ and $\sum M_d(t)$ the total number of generated, received and dropped packets.

Finally, the quality-of-security, QSec, depends on the two previous parameters and defines the communication and transfer between two nodes. It is the deciding factor as to which node needs to be selected as the next node in the path and is computed as the weighted sum of reputation and QoS:

$$W_n(t) = w_1\phi_{ij}(t) + w_2\theta_{ij}(t)$$

2.3 ATSN

An agent-based trust model for WSN is presented in [11] 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:

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

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 a triple $TS = \{(pt, nt, ut)\}$, 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:

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

2.4 RFSN

RFSN [12] 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 subcomponents, $(R_{ij})_D$ 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

$$(R_{ij})_D = f(D_{ij}, (R_{ij})_D), \quad \forall j \in N_i$$

$$(R_{ij})_{ID} = (R_{ij})_{ID} + w_{ik} \times R_{kj}, \quad \forall k \in N_i$$

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 {*cooperate*, *don't cooperate*} and a simple threshold based policy is used to decide the value of B_{ij}.

2.5 CORE

CORE [13] is a generic mechanism based on reputation to enforce cooperation among nodes in a MANET in order to prevent selfish behavior. All members of a community have to contribute to the community life in order to be entitled to use its resources. In CORE reputation is a measure of someone's contribution to common 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 as follows:

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

where $r^t_{s_i}(s_j|f) \in [-1, 1]$ stands for the subjective reputation value calculated at time t by subject s_i on subject s_j 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 $ir^t_{s_i}(s_j|f)$, 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:

$$r^t_{s_i}(s_j) = \sum w_k \cdot (r^t_{s_i}(s_j|f_k) + ir^t_{s_i}(s_j|f_k))$$

where w_k represents the weight associated to the functional reputation value and $r^t_{s_i}(s_j)$ 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.

2.6 DRBTS

DRBTS [14] 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, $s_1, s_2, \ldots, s_n$ and m BN, $b_1, b_2, \ldots, b_m$. 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-han 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:

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

2.7 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 [9, 13, 15, 16].

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 [11, 12]. Others take advantage of Bayes theorem [17] and a posteriori probabilities, or just use a Beta distribution [15] in order to represent ratings.

As far as we know our model is on of the first ones (together with [10]) in applying a bio-inspired technique such as ant colony system (ACS) to develop a trust and reputation model for WSN.

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 wireless sensor networks 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 [2]). 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 processing 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 nothing else about the whole topology of the network (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) [5–8].

3.2 BTRM-WSN, a bio-inspired approach

BTRM-WSN is a bio-inspired trust and reputation model for Wireless Sensor Networks 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 cannot be directly applied there. 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 cannot wait until all the launched ants came back in one iteration of the algorithm.

Therefore, the algorithmic scheme presented in ACS [8, 18] has to be redefined as shown in Algorithm 1.

Algorithm 1 BTRM-WSN

```
1:  while (condition) do
2:    for k = 1 to Number_of_ants do
3:      S_k ← 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 ← S_k
10: while (timeout does not expire) and
11: Num_returned_ants < %Number_of_ants
12:
13: if (Q(Current_Best) > Q(Global_Best)) then
14:   Global_Best← Current_Best
15: Pheromone_global_updating
16:    (Global_Best, Q(Global_Best), ρ)
17:
18: return Global_Best
```

The first change we can appreciate is that the main loop 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. 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 1, 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.

3.2.1 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:

$$Q(S_k) = \frac{\overline{\tau}_k}{Length(S_k)^{PLF}} \cdot \%A_k$$

where $\overline{\tau}_k$ is the average pheromone of the path found by ant k, $PLF \in [0, 1]$ 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 to have a qualified 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 preferably select those paths which are as short as possible and which have been selected as many times as possible.

3.2.2 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, $\overline{\tau}_k \in [0, 1]$. If $\overline{\tau}_k$ is greater than a certain transition threshold, $TraTh \in [0, 1]$, then ant k stops and returns current solution with a probability equal to $\overline{\tau}_k$. Otherwise, if $\overline{\tau}_k \leq TraTh$, ant k considers sensor s not enough reputable and keeps trying to find 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 traditional transition rule defined in ACS.
   (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 arrives at a sensor which offers the requested service (and then stops and returns that new 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 cannot interact again with the same malicious server in the next transaction, so ants will not stop when they find it and consequently the 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 1 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.

3.2.3 Pheromone updating

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:

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

where $\Omega = (1 + (1 - \varphi) \cdot (1 - \tau_{s_1 s_2}) \cdot \eta_{s_1 s_2}) \cdot \tau_{s_1 s_2}$ is the convergence value of $\tau_{s_1 s_2}$ when time $t \to \infty$ (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 1 (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:

$$\tau_{rs} = (1 - \rho)\tau_{rs} + \rho(1 + \tau_{rs}\eta_{rs}Q(S_{Global\_Best}))\tau_{rs} \tag{2}$$

Therefore, the higher are the pheromone trace, the heuristic value, the quality of the path and $\rho \in [0, 1]$, 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 $IniPh \to 0$, for instance, everybody would mistrust everyone at the beginning and it would be difficult to distinguish trustworthy sensors from malicious ones. However, if $IniPh \to 1$ then everybody would trust everyone at the beginning and it would also be difficult to distinguish benevolent sensors from fraudulent ones.

3.2.4 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 (2) 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, if satisfaction is less than a certain punishment threshold $PunTh \in [0, 1]$:

$$\tau_{rs} = (\tau_{rs} - \varphi \cdot df_{rs}) \cdot Sat \cdot (1 - df_{rs}) \tag{3}$$

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:

$$df_{rs} = \sqrt{\frac{d_{rs}}{L(S_k) \cdot (L(S_k) - d_{rs} + 1)}} \tag{4}$$

being $d_{rs} \in \{1, 2, \ldots, L(S_k)\}$ the actual distance (number of hops) between sensor r and s, and $L(S_k)$ the length of the path found by ant k.

Otherwise, if $Sat \geq PunTh$ then:

$$\tau_{rs} = \tau_{rs} - \varphi \cdot (1 - Sat) \cdot 2df_{rs} \tag{5}$$

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:

$$\tau_{rs} = (\tau_{rs} - \varphi) \cdot Sat \tag{6}$$

3.3 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.

And if our WSN is composed of very restricted sensors, we could adopt a dynamic scenario where some of them switched off for awhile if they did not have any transaction along a timeout or if they were very active (providing too many requests) during another timeout.

But if we have a low-constraint WSN (equally, a high-performance WSN) and 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. Likewise, sensors will remain always awake.

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 accurate decisions, since it knows (through the pheromone traces) which servers are more reputable and which not.

3.4 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 Sect. 3.2.2, the number of ants depends on the number of sensors composing the network. Thus, we defined the number of ants launched by every client as $N_{ants} = \lceil N_s^{0.35} \rceil$, 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 small overload.

The accuracy and robustness of BTRM-WSN will be demonstrated in Sect. 5 where experiments and results will be shown.

4 Security threats

The fact that every node maintains the pheromone traces of its neighbors and it is the only one who can manage, control and modify them, can lead to some security threats.

But the only security threats related to this matter can appear if a malicious server colludes with other malicious servers, because a sensor is only able to manage the pheromone traces of its neighbors, but by the same reason it cannot control the pheromone traces that its neighbors have associated with it.

Nevertheless, it is important to notice that a collusion is only possible if the malicious sensors know each other and also know who the benevolent sensors are. And this assumption is not always feasible in every Wireless Sensor Network.

Therefore, two types of security threats may happen if a collusion among malicious sensors can be created. Malicious sensors can praise their malicious neighbors by assigning them the maximum level of pheromone. Equally they can slander their benevolent neighbors by giving them the minimum value of pheromone. We will discuss in detail both situations next.

4.1 Praising malicious sensors

A set of malicious sensors can form a collusion in order to increase their self profit and interests. Each of them manage the pheromone traces of its neighbors, so what they can do is to praise those neighbors belonging to the collusion by giving them the maximum level of pheromone. And, of course, they will not decrease those traces although a client asked for it.

In this situation the malicious node who modifies the pheromone traces of its neighbors can act as a malicious service provider or could behave properly and supply the right requested service. If the second thing occurs ants will choose it as the service provider and its collusion will have no sense.

But if it behaves in a fraudulent way as well and a client selects it to have an interaction with it, all the links falling into it will be punished as explained before and ants will not select it again (or will select it with a very low probability), so its false praising would be useless.

4.2 Slandering benevolent sensors

Another possible security threat would consist in slandering benevolent nodes. This is achieved by assigning the minimum level of pheromone to those benevolent neighbors of a malicious one.

Again the malicious node can actually provide fraudulent services or right ones. In the first case, if there are alternative paths leading to the slandered benevolent sensor, ants should be able to discover them; otherwise, ants would select another different benevolent node.

And if the malicious server acts properly and provides the right service, ants will select it and its collusion will not have sense neither.

It is important to have in mind that there must be at least one accessible benevolent server in the WSN and the key consists of finding it. It actually does not matter which specific sensor is selected to interact with, the important thing is to select a trustworthy one.

5 Experiments and results

Once we have shown in detail the description of our bio-inspired trust and reputation model over Wireless Sensor

Table 1 BTRM-WSN parameters

phi	0.01	**alpha**	1.0	**Nants**	0.35
rho	0.87	**beta**	1.0	**Niter**	0.59
q0	0.45	**TraTh**	0.66	**PunTh**	0.48
IniPh	0.85	**PLF**	0.71		

Networks and have described some related security threats, it is time to demonstrate its accuracy, scalability and robustness.

To do so, we have developed a whole testbed focused on three main targets. First, 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 WSNs, we gathered as well the average path length of the solutions found by our model. As we mentioned before, in a environment with a lot of restrictions like WSNs, 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 WSNs randomly generated, 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).

We have defined the main condition of our algorithm (line 1 of Algorithm 1) as a certain number of iterations. And that number is defined as $N_S^{N_{iter}}$, (similar to the number of ants definition) where N_S is the number of sensors belonging to the WSN and $N_{iter} \in [0, 1]$.

The same set of parameter values (shown in Table 1) is used for all the experiments and environments.

We have configured four different scenarios: static WSNs, dynamic WSNs, oscillating WSNs and static WSNs with collusion among malicious servers, as we will explain next.

5.1 Experiments and results over static WSNs

The first tested scenario consisted of static Wireless Sensor Networks, that is, networks where their sensors do not switch off and do not move, maintaining thus always the same topology.

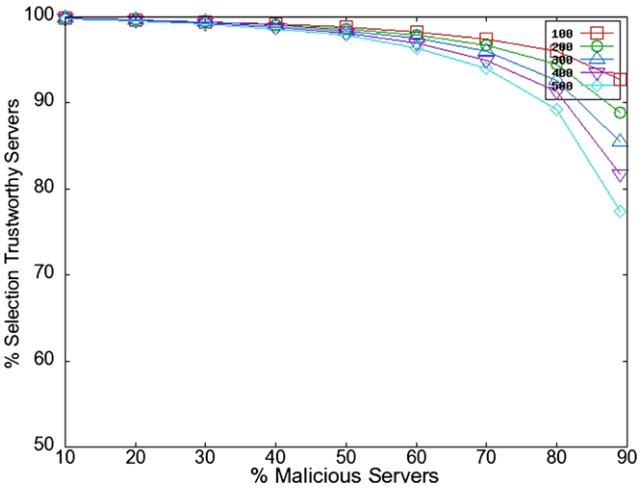

Fig. 1 Static WSNs. Selection percentage of trustworthy servers

5.1.1 *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 Fig. 1.

The very 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.

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 80%, 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 90%. Its performance gets worse when the percentage of malicious servers in the WSN increases, and the problem intensifies when the size of the WSN grows.

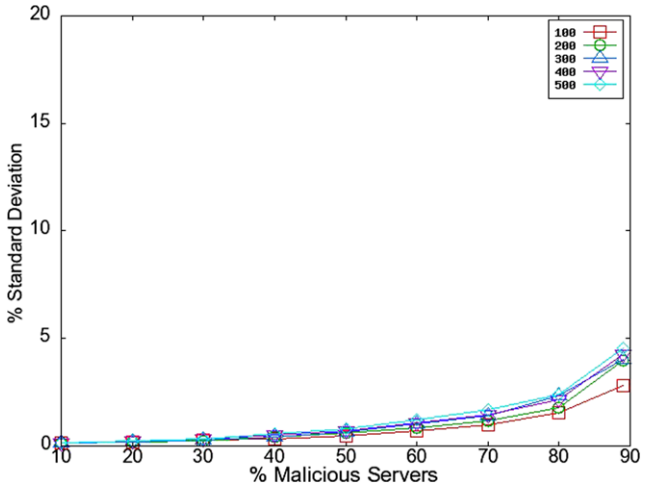

Fig. 2 Static WSNs. Standard deviation of the selection percentage of trustworthy servers

5.1.2 *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. 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 Fig. 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 90%. Furthermore, this standard deviation remains below a 5% in every case.

This means that when there are less than or equal to 90% of malicious servers in the network, regardless its size, BTRM-WSN is able to select a high percentage of trustworthy servers (as shown in Fig. 1) with a quite high accuracy, regardless the topology of the WSN.

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 500 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 77.35% of the times, with a standard deviation of 4.55%.

So if the percentage of malicious servers is high (greater than or equal to 90%, for instance), and the number of sensors composing the networks is also high, then the percentage selection of trustworthy servers is lower but, however, still accurate (i.e., BTRM-WSN is independent of the topology).

This means that if the random tested WSNs size is 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).

5.1.3 *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 Fig. 3.

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 3.844.

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 90%. Under this percentage, the average number of hops is quite low (near to 2), 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.

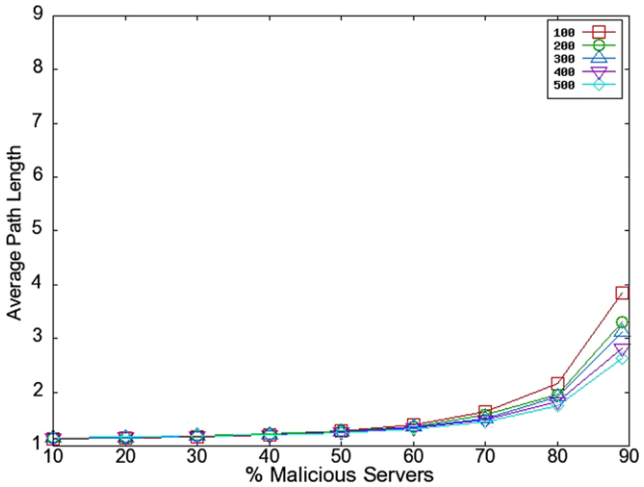

Fig. 3 Static WSNs. Average path length leading to trustworthy servers

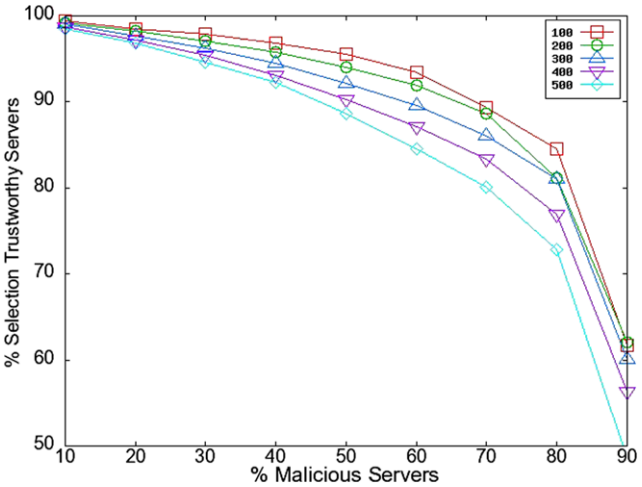

Fig. 4 Dynamic WSNs. Selection percentage of trustworthy servers

5.2 Experiments and results over dynamic WSNs

As we have already mentioned, the first of the two proposed versions of our model is aimed to deal with WSNs composed of sensors with quite high restrictions in energy consumption, bandwidth, storage capacity, etc.

That is the reason why we decided to develop this set of experiments. Here some nodes switch off for awhile sometimes saving thus an important amount of energy consumption.

The decision scheme of when to switch off and on is as follows: when a server receives and supplies 20 requests it automatically switches off during a certain timeout. On the other hand, if a server does not receive at least 20 requests within a time interval, it also switches off during another timeout.

Once we defined our dynamic scenario in the manner explained above, we carried out the same experiments that were done for static networks, i.e., we measured the percentage selection of trustworthy servers, the standard deviation of this selection percentage, and the average path length of the routes found leading to trustworthy servers.

5.2.1 Selection percentage of trustworthy servers

Figure 4 shows the selection percentage of trustworthy servers achieved with BTRM-WSN over WSNs composed of 100 to 500 sensors with a percentage of malicious servers from 10% to 90%.

As it is observed, the selection percentage is nearly greater than 90% when the percentage of malicious servers is less than or equal to 50%, regardless the size of the WSN. And it remains obtaining qualified outcomes (above the 70%) when the proportion of malicious servers is less than or equal to 80%.

Selection percentage get worse when the percentage of malicious servers increases and even worse if the size of the Wireless Sensor Network is greater.

Nevertheless, we can state that BTRM-WSN is resilient to a dynamic behavior of the sensors composing the WSNs it is running, if the percentage of fraudulent sensors is less than 80%. And the worsening is not too high when the number of sensors increases.

5.2.2 Standard deviation of the selection percentage of trustworthy servers

In Fig. 5 we can observe the standard deviation of the selection percentage of trustworthy servers achieved in this experiment, using dynamic Wireless Sensor Networks.

We can see that the standard deviation remains quite low and nearly undistinguishable when the size of the network is greater than or equal to 200 sensors and the percentage of malicious servers is less than or equal to 80%, which means that, in those cases BTRM-WSN is very accurate and almost always finds the same percentage of trustworthy servers.

If we are dealing with a smaller Wireless Sensor Network or the proportion of malicious servers is greater than or equal to 90%, however, this standard deviation increases remarkably, being its maximum value a 19.35%, when the tested WSN is composed of 100 sensors.

5.2.3 Average path length leading to trustworthy servers

As it can be checked in Fig. 6, the average path length obtained in a dynamic environment is greater than in a static one (see Fig. 3).

However, although the size of the network can reach high values, the average path length never exceeds 8.66 hops in any case, which is still a good outcome for Wireless Sensor Networks.

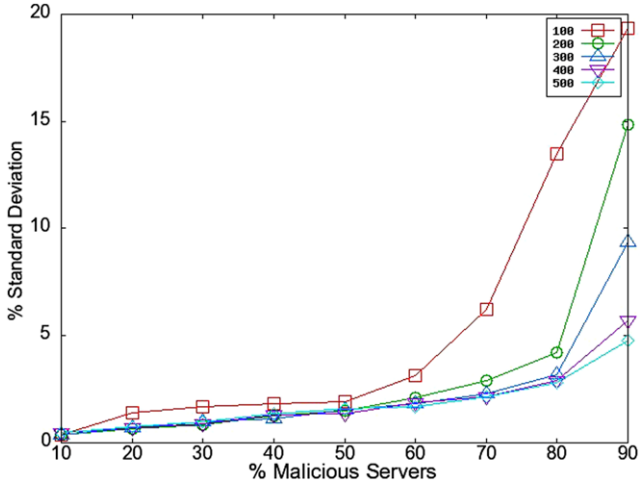

Fig. 5 Dynamic WSNs. Standard deviation of the selection percentage of trustworthy servers

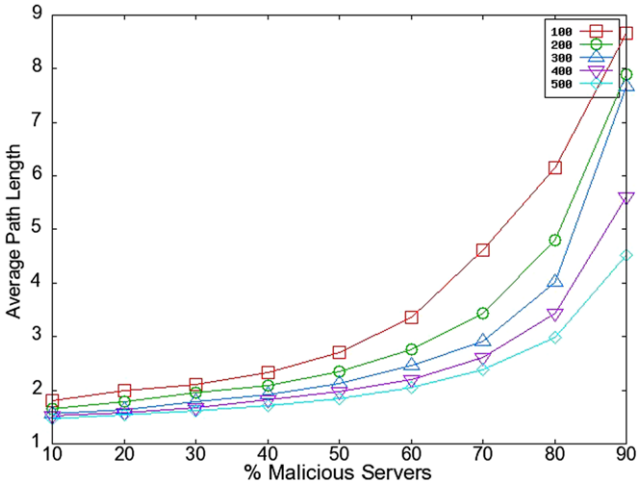

Fig. 6 Dynamic WSNs. Average path length leading to trustworthy servers

Equally to the static scenario, when the size of the network is lower and the percentage of malicious servers composing the network is greater, then the average path length also increases.

This experiment together with the selection percentage of trustworthy servers and the standard deviation of that selection percentage constitute the proof that BTRM-WSN obtains quite good and accurate outcomes over dynamic Wireless Sensor Networks, with a low influence from the size of the networks and the percentage of malicious servers.

We can state, therefore, that BTRM-WSN presents a technique to identify trustworthy servers that is suitable for dynamic Wireless Sensor Networks.

5.3 Experiments and results over oscillating WSNs

Another tested scenario developed consisted of Wireless Sensor Networks where the goodness of the servers belonging to them could change along the time.

How a sensor decides to be benevolent or malicious at each time is out of scope of this paper. We designed, therefore, our particular proposal as follows: after every 20 transactions are carried out (i.e., after every client has had 20 transactions) all the benevolent servers composing the Wireless Sensor Network become malicious.

Now, in order to preserve the same percentage of malicious servers, the number of previous benevolent servers, let say n_b, is kept. Then n_b random servers are selected (note that all of them will be malicious) and their goodnesses are swapped so they become benevolent and the percentage of malicious servers remains equal to the stage previous the oscillation.

With an oscillation scheme like this a benevolent server could maintain its positive goodness since it could be randomly selected to become benevolent when it indeed previously was benevolent.

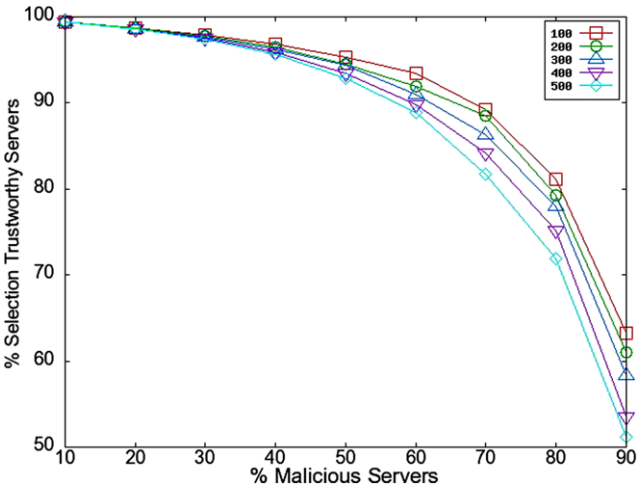

Fig. 7 Oscillating WSNs. Selection percentage of trustworthy servers

5.3.1 *Selection percentage of trustworthy servers*

As we can see in Fig. 7, outcomes got here are quite similar to those obtained in the previous experiment with dynamic WSNs.

It can be checked that the selection percentage of trustworthy servers is greater than 90% if the percentage of malicious servers is approximately less than or equal to 60%, regardless the size of the Wireless Sensor Network.

Moreover, reasonably good outcomes (those with a selection percentage above the 70%) are obtained always the proportion of fraudulent servers is less than or equal to 80%.

5.3.2 *Standard deviation of the selection percentage of trustworthy servers*

Again, similar outcomes to the ones shown for static WSNs about the standard deviation of the selection percentage

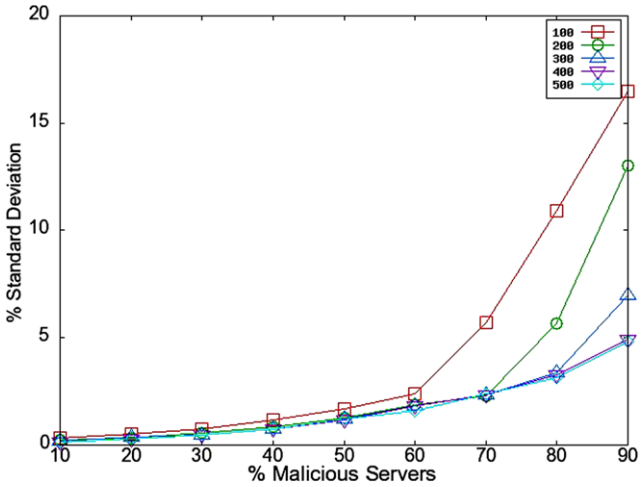

Fig. 8 Oscillating WSNs. Standard deviation of the selection percentage of trustworthy servers

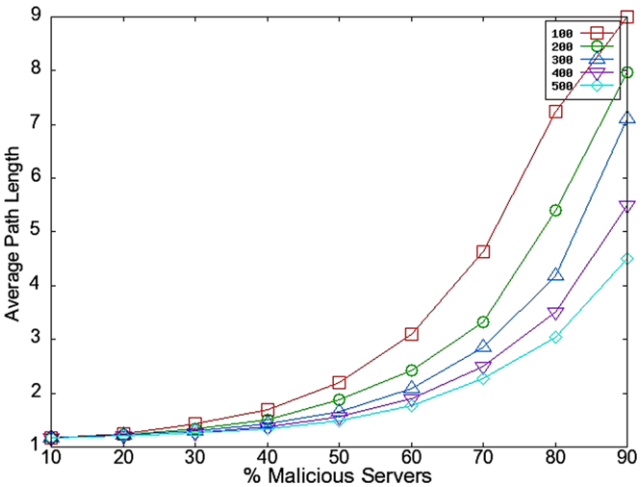

Fig. 9 Oscillating WSNs. Average path length leading to trustworthy servers

of trustworthy servers are achieved here, with oscillating WSNs. Figure 8 depicts these results.

One more time the more variable behavior of the model happens when the size of the tested WSNs is less than 200 sensors. In such situation a maximum standard deviation value of 16.5% is reached when the proportion of fraudulent servers is 90%.

Nonetheless, if the tested WSNs are composed of 200 sensors or more, and the percentage of malicious servers is less than or equal to 70%, then the standard deviation is undistinguishable and less than 2.4%.

5.3.3 Average path length leading to trustworthy servers

Figure 9 shows the outcomes about the average path length of those routes found by BTRM-WSN leading to a trustworthy server over oscillating WSNs.

It can be checked that these results are very similar to the ones shown in Fig. 6, so the same conclusions can be obtained.

The three last experiments demonstrate that BTRM-WSN is also a feasible technique in order to find the most trustworthy server over oscillating WSNs.

5.4 Experiments and results over static WSNs with collusion

The last tested scenario consisted of static Wireless Sensor Networks where a collusion among all malicious servers composing the network was built. As explained in Sect. 4, since in BTRM-WSN every sensor is the only one who can manage the pheromone traces associated with its neighbors, malicious servers could collude an falsely praise themselves or slander benevolent servers.

We chose the worst situation, where both things occurred, that is, every malicious server always had the maximum pheromone value for those of its neighbors who were also malicious, and the minimum pheromone value for those neighbors who were benevolent.

How every sensor knows if its neighbors are malicious or benevolent is out of the scope of this paper.

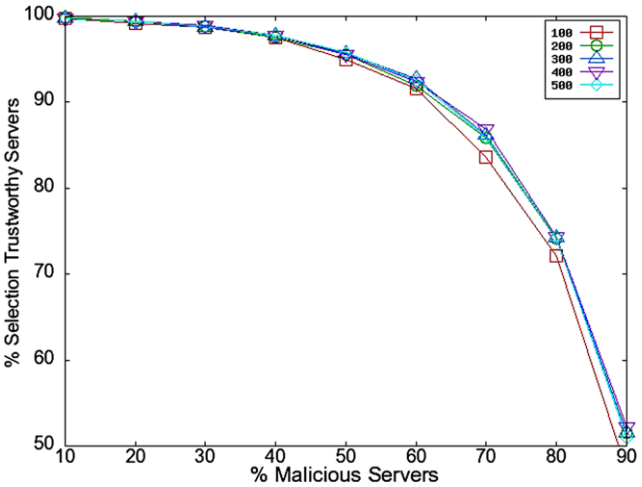

Fig. 10 Static WSNs with collusion. Selection percentage of trustworthy servers

5.4.1 Selection percentage of trustworthy servers

The worst outcomes about the selection percentage of trustworthy servers among all the tested experiments are reached here and can be seen in Fig. 10.

Again, the first appreciation that can be done is the high similarity of the five graphics corresponding to the five WSN sizes tested. This means that BTRM-WSN is highly scalable in a collusion scenario.

As it can be checked the selection percentage of trustworthy servers remains greater than a 90% when the percent-

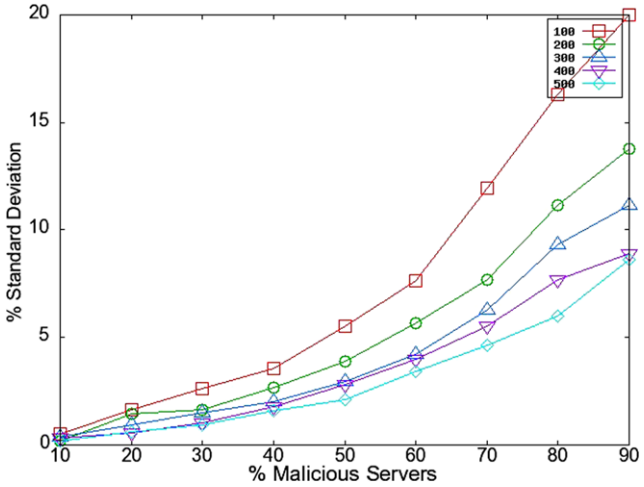

Fig. 11 Static WSNs with collusion. Standard deviation of the selection percentage of trustworthy servers

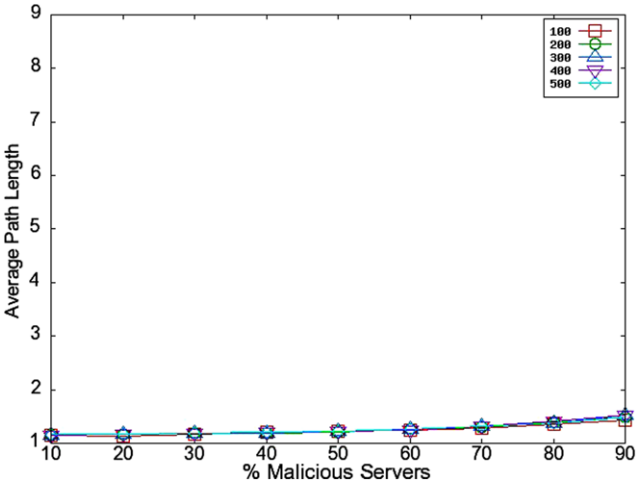

Fig. 12 Static WSNs with collusion. Average path length leading to trustworthy servers

age of malicious servers ir less than or equal to 60%. And it produces qualified outcomes (above the 70% of selection percentage) when the proportion of malicious servers is less than or equal to 80%.

But if this percentage increases, however, then our model is quite near to the limit of being useful in any way. Notice that if the selection percentage is under the 50%, then the model is completely useless.

5.4.2 Standard deviation of the selection percentage of trustworthy servers

In Fig. 11 the standard deviations of the selection percentage of trustworthy servers obtained by BTRM-WSN over static WSNs with a collusion, are shown.

An interesting behavior happens here. In the previous experiments the standard deviation remained very similar when the size of the network was high and the proportion of malicious servers was low.

However, here the standard deviation grows when the percentage of fraudulent servers also increases and the size of the tested Wireless Sensor Networks decreases.

This means that BTRM-WSN is less independent of the number of sensors composing the network when a collusion is formed.

5.4.3 Average path length leading to trustworthy servers

On the other hand, the best outcomes regarding the average length of the routes leading to trustworthy servers found by BTRM-WSN over all the tested experiments are obtained when a collusion takes place, as it can be observed in Fig. 12.

This average path length never exceeds 1.55 hops, which is a very low value. This means that most of the trustworthy servers found are very near to the client. It can also mean that in such an adverse situation like this one (static WSNs with collusion), BTRM-WSN is unable to find benevolent servers which are too far from the clients.

And it makes sense getting these values. If the proportion of malicious servers is low, it will be probable that some benevolent servers stay near the clients. And if that percentage is high, then malicious colluding servers will avoid clients' ants to travel quite far in order to find benevolent servers.

5.5 Energy consumption

Energy consumption is a critical issue when dealing with wireless sensor networks, since these ones are commonly composed by resource-constrained devices with limited features in terms of processing, memory and communicating capabilities.

Therefore, we could not ignore this topic in our trust and reputation model, so we developed a last experiment aimed to measure the average energy consumption needed by our approach.

As pointed out by [19, 20] the power required by a sensor in a WSN can be seen as a function of the distance. Different energy models can be used to estimate the energy required by a sensor s to send a message far enough to reach another sensor placed at distance d. In the most commonly used model, the energy consumption for transmitting a message at distance d is:

$$E(d) = d^{\alpha} + C$$

where $\alpha \in [2, 6]$ represents the media attenuation factor and C is a constant denoting the power used to process the radio signal (note it is a dimensionless measurement).

Table 2 Static WSNs. Average energy consumption per sensor

	Number of sensors				
	100	200	300	400	500
10%	1.2×10^{14}	1.1×10^{16}	1.0×10^{16}	6.8×10^{17}	3.6×10^{18}
20%	1.3×10^{14}	1.0×10^{16}	1.1×10^{16}	7.0×10^{17}	3.7×10^{18}
30%	1.6×10^{14}	1.2×10^{16}	1.1×10^{16}	7.5×10^{17}	3.8×10^{18}
40%	2.0×10^{14}	1.3×10^{16}	1.2×10^{16}	7.8×10^{17}	4.0×10^{18}
50%	1.9×10^{14}	1.4×10^{16}	1.4×10^{16}	8.5×10^{17}	4.4×10^{18}
60%	2.8×10^{14}	1.4×10^{16}	1.5×10^{16}	1.0×10^{18}	5.0×10^{18}
70%	2.7×10^{14}	2.2×10^{16}	1.9×10^{16}	1.2×10^{18}	6.2×10^{18}
80%	5.0×10^{14}	3.0×10^{16}	3.1×10^{16}	1.8×10^{18}	8.6×10^{18}
90%	9.3×10^{14}	0.9×10^{17}	0.7×10^{18}	6.8×10^{18}	2.6×10^{19}

Following these authors' direction, we have chosen a value of $\alpha = 4$ and $C = 10^8$. Additionally, the sensors belonging to our generated networks are spread along a 10000 m^2 area, and each of them has a radio range of 10 meters.

We have collected the energy measurements from the experiments developed over static wireless sensor networks, which constitutes the worst scenario, since every sensor is permanently awake. The outcomes can be observed in Table 2.

Two direct consequences can be deducted from the table. On the one hand, the bigger the number of sensors is, the higher is the energy consumption. And on the other hand, the greater the percentage of malicious sensors is, the higher is the power consumption as well.

The increase of energy needed as the size of the network grows is explained because it implies an increase in the density of the network too. So the average number of links increases rapidly and, therefore, a higher number of messages are sent.

Regarding the percentage of malicious sensors, in a network where this kind of sensors are in majority it is more difficult to find a benevolent one and, consequently, more messages need to be sent as well.

5.6 TRMSim-WSN. Trust & reputation models simulator for WSNs

In order to carry out all the explained experiments we have developed a Java-based Trust & Reputation Models Simulator for WSNs, called TRMSim-WSN [21].

It allows a user to test BTRM-WSN over all the scenarios described in this paper (static, dynamic, oscillating and collusion), and even combinations of them, deciding the number and size of WSNs and the number of transactions or executions of the model carried out by every client. It also allows to set the percentage of clients, relay servers (those not providing the service requested by the clients), and malicious servers.

The wireless range of every sensor can be set as well, determining thus the topology of the network by means of determining every sensor's neighbors.

Currently it only implements BTRM-WSN and PeerTrust [22] models and allows to tune their parameters, but we are planning to implement additional trust and reputation models for WSNs in order to make a comparison among them.

Figure 13 shows a snapshot of TRMSim-WSN, which can be downloaded from [21], where a more complete documentation of the simulator can be found.

6 Conclusions and future work

Managing trust and reputation in Wireless Sensor Networks (WSNs) in an efficient, accurate and robust way has not been completely solved yet. Providing this management would notably increase the security in such a sentient environment, supporting thus its development and deployment.

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 next 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 reasonably well the initially stated expectations about security, scalability and lightness in WSNs.

We have also tested our model against dynamic Wireless Sensor Networks, where some nodes switched off for awhile if certain conditions occurred (changing thus the topology of the network), against oscillating WSNs, where the goodness of the servers changed along the time, and against static networks with a collusion scenario among malicious servers. It has been demonstrated that BTRM-WSN obtains accurate, robust and scalable outcomes in most of the situations.

Specifically, if the percentage of malicious servers is below the 60%, the selection percentage of trustworthy servers remains above 90% in most of the cases with a standard deviation never greater than 7.62%, regardless the size of the

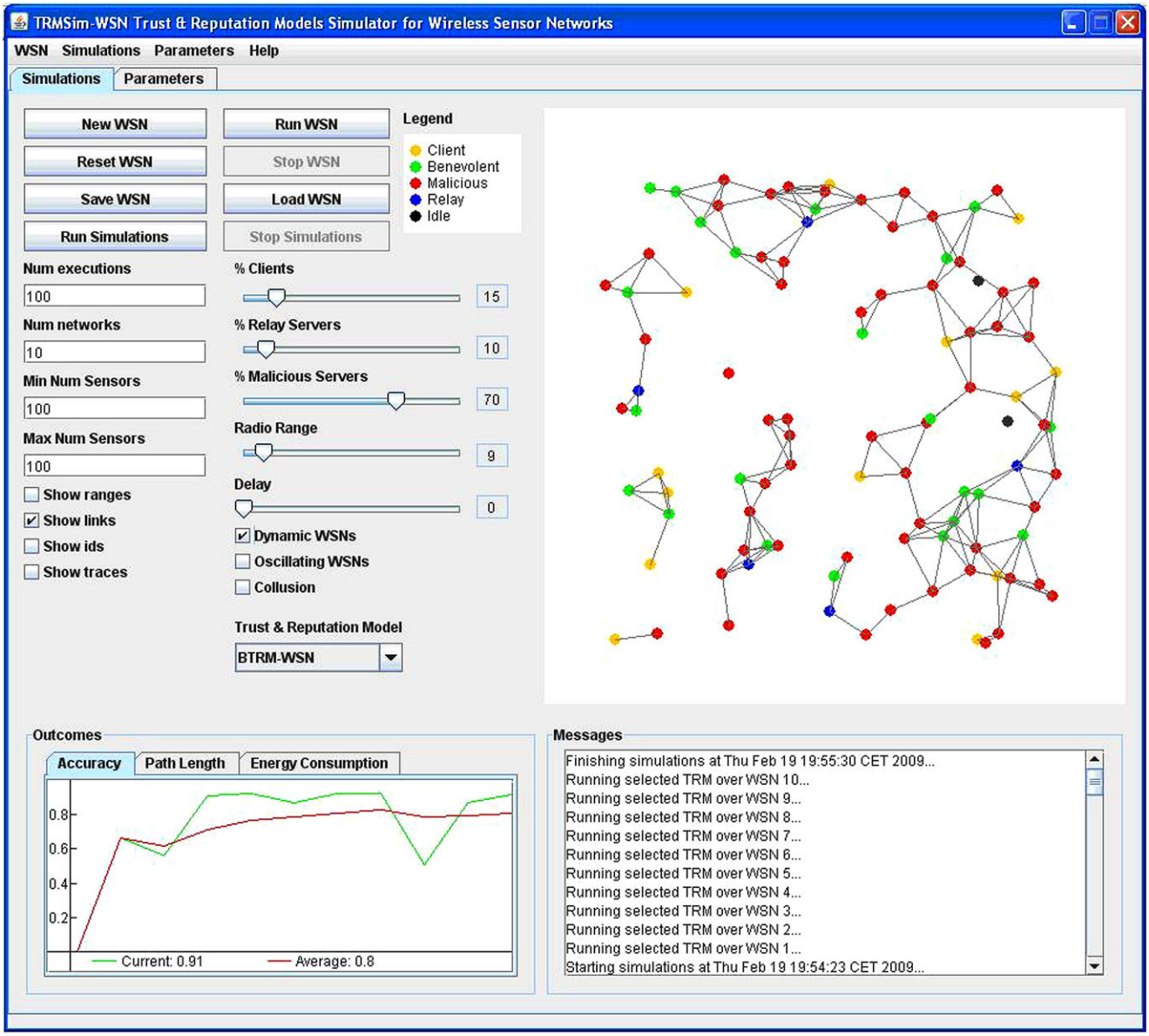

Fig. 13 Snapshot of TRMSim-WSN, a trust & reputation models simulator for WSNs

network. And if the proportion of fraudulent servers is under a 80%, we can reach a selection percentage of trustworthy servers up to 70%.

It has therefore been proved that BTRM-WSN is highly scalable, accurate, light and robust. Its main deficiencies come when the percentage of malicious servers is greater than or equal to 90%. So the key factor that makes our model failing when searching the most trustworthy server through the most reputable path is that proportion of fraudulent servers.

We have proposed two versions of our model, depending on the capabilities of the WSN we are dealing with and on the security restrictions we would like to apply. Thus, the first version is lighter and more scalable while the second is more resilient and accurate.

Regarding security, we have identified and described some security threats that could be applied in our model and other similar trust and reputation models for WSNs.

This paper opens, however, several future ways. For instance, we have used the same parameters of our model in every case. We think it would be better if each client could decide the values of its parameters and, even more, auto-adjust them along the time, in order to get a better performance.

We also need to improve the outcomes got when the proportion of malicious servers is equal to 90%, and specifically, when a collusion is formed.

Further theoretical explanations about the performance variation between the scenarios in Sect. 5 will be considered as an extension of our work.

Finally, we are planning to add more functionality to our visual simulator TRMSim-WSN. 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 research work has been 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 granted as well by the Séneca Foundation with code 04552/GERM/06.

References

1. Römer, K., & Mattern, F. (2004). The design space of wireless sensor networks. *IEEE Wireless Communications*, *11*(6), 54–61.
2. Li, F., & Wang, Y. (2007). Routing in vehicular ad hoc networks: a survey. *Vehicular Technology Magazine IEEE*, *2*(2), 12–22.
3. Marsh, S. P. (1994). *Formalising trust as a computational concept*. PhD thesis, Department of Computing Science and Mathematics, University of Stirling.
4. Marti, S., & Garcia-Molina, H. (2006). Taxonomy of trust: categorizing P2P reputation systems. *Computer Networks*, *50*(4), 472–484.
5. Dorigo, M., & Gambardella, L. (1997). Ant colony system: a cooperative learning approach in the traveling salesman problem. *IEEE Transaction on Evolutionary Computing*, *1*(1), 53–66.
6. Dorigo, M., Gambardella, L., Birattari, M., Martinoli, A., Poli, R., & Stützle, T. (2006). Ant colony optimization and swarm intelligence. In *LNCS: Vol. 4150. 5th international workshop, ANTS 2006*. Brussels: Springer.
7. Cordón, O., Herrera, F., & Stützle, T. (2002). A review on the ant colony optimization metaheuristic: basis, models and new trends. *Mathware and Soft Computing*, *9*(2–3), 141–175.
8. Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Bradford Book
9. Boukerche, A., Xu, L., & El-Khatib, K. (2007). Trust-based security for wireless ad hoc and sensor networks. *Computer Communications*, *30*(11–12), 2413–2427.
10. Dhurandher, S. K., Misra, S., Obaidat, M. S., & Gupta, N. (2009). An ant colony optimization approach for reputation and quality-of-service-based security in wireless sensor networks. *Security and Communication Networks*, *2*(2), 215–224.
11. Chen, H., Wu, H., Zhou, X., & Gao, C. (2007). Agent-based trust model in wireless sensor networks. In *Eighth ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing*, *SNPD'03* (pp. 119–124).
12. Ganeriwal, S., & Srivastava, M. B. (2004). 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* (pp. 66–77). New York: ACM.
13. Michiardi, P., & Molva, R. (2002). 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* (pp. 107–121). Deventer: Kluwer, B.V.
14. Srinivasan, A., Teitelbaum, J., & Wu, J. (2006). DRBTS: distributed reputation-based beacon trust system. In *DASC'06: Proceedings of the 2nd IEEE international symposium on dependable, autonomic and secure computing* (pp. 277–283). Washington: IEEE Computer Society.
15. Buchegger, S., & Le Boudec, J. Y. (2004). 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*.
16. Buchegger, S., & Boudec, J.-Y. L. (2002). Performance analysis of the CONFIDANT protocol: cooperation of nodes. In *Proceedings of IEEE/ACM symposium on mobile Ad hoc networking and computing (MobiHOC)*. Lausanne: IEEE.
17. Almenárez, F., Marín, A., Campo, C., & García, C. (2004). PTM: a pervasive trust management model for dynamic open environments. In *Privacy and trust, first workshop on pervasive security and trust, Boston, USA*.
18. Glover, F. W., & Kochenberger, G. A. (2003). *Handbook of metaheuristics (International series in operations research & management science)*. Berlin: Springer.
19. Li, L., & Halpern, J. Y. (2001). Minimum-energy, mobile wireless networks revisited. In *IEEE International Conference on Communications*, *ICC 2001* (Vol. 1, pp. 278–283).
20. Sánchez, J.A., & Ruiz, P.M. (2006). Improving delivery ratio and power efficiency in unicast geographic routing with a realistic physical layer for wireless sensor networks. In *Proc. 9th Euromicro conference on digital system design (DSD'06)* (pp. 591–597).
21. Gómez, M.F. TRMSim-WSN, a trust & reputation models simulator for wireless sensor networks, http://ants.dif.um.es/~felixgm/research/trmsim-wsn.
22. Xiong, L., & Liu, L. (2004). PeerTrust: supporting reputation-based trust in peer-to-peer communities. *IEEE Transactions on Knowledge and Data Engineering*, *16*(7), 843–857.

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