TRMSim-WSN, Trust and Reputation Models Simulator for Wireless Sensor Networks

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, s/n
30.071, Murcia, Spain
{felixgm, gregorio}@um.es

***Abstract*—Trust and reputation models research and development for distributed systems such as P2P networks, Wireless Sensor Networks (WSNs) or Multi-agent systems has arisen and taken importance in the last recent years among the international research community. However it is not always easy to check the correctness and accuracy of a model and even more, to compare it against other trust and reputation models. This paper presents TRMSim-WSN, a Java-based trust and reputation models simulator aimed to provide an easy way to test a trust and/or reputation model over WSNs and to compare it against other models. It allows the user to adjust several parameters such as the percentage of malicious nodes or the possibility of forming a collusion, among many others.**

I. Introduction

Relevance and utility of distributed networks is improving everyday due to the numerous applications they have and all the research efforts that are focused on them. Specifically Wireless Sensor Networks [1] (WSNs) are widely spread and employed in multiple scenarios such as fire detection, weather measurements and even traffic management in Vehicular-to-Vehicular [2] (V2V) networks.

Nevertheless, this kind of networks has its own drawbacks. Their wireless way of communication, their battery and bandwidth constraints or their location in open environments, for example, lead them to some security threats. Recently, trust and reputation management has become a novel way of dealing with some of these important issues. Thus, several trust and/or reputation models over WSNs [3], [4], [5], [6] have been developed and studied.

But most of them provide their own and particular test set in order to demonstrate their accuracy and goodness, leading to a more difficult way of objectively comparing them with other models. Furthermore, it is usually hard to design a new trust and/or reputation model without any guideline. Therefore, in this paper we present TRMSim-WSN [7], a trust and reputation models simulator for WSNs aimed to provide a generic tool in order to test and compare trust and reputation models. We have included some of the most common experiments found in the literature for this kind of models and also developed an API which constitutes a template for easily including new trust and reputation models into our simulator.

A number of network simulators [8], [9] has been proposed and developed in the last few years. But most of them focus their attention on complex communication protocols, which many times require an expert knowledge. TRMSim-WSN is one layer above. It abstracts developers from low level communication issues and centers specifically on trust and reputation models.

The rest of the paper is organized as follows. Section II describes some simulation environments for trust and/or reputation models. In section III we explain the generic API designed to implement new trust and reputation models and how these are added to the simulator. TRMSim-WSN is presented in section IV and section V shows some conclusions and future work.

II. Related work

Lots of network simulators have been developed in order to test new communication protocols and check their correctness, robustness or accuracy. Authors of [8], for instance, present a survey of P2P network simulators, describing their main features and also their limitations.

However, while the number of those network simulators is considerable there is a lack of trust and reputation models simulators for distributed networks. One of the few ones that have been designed and published is TOSim [10]. It has been created to be highly modular and configurable, without incurring in excessive overload both in terms of memory and time. In order to simulate behavior related to trust, authors consider four threat models of malicious peers to cause insecure files to be uploaded to the system.

Another important platform for simulating reputation models is ART testbed [11] which has become in recent years a reference environment in the field of reputation models for multi-agent systems. This testbed serves, on the one hand, as a competition forum in which researchers can compare their models against objective metrics and, on the other hand, as a set of tools with flexible parameters, allowing researchers to perform customizable and easily-repeatable experiments.

Nevertheless, we have not found any simulator, competition or environment targeting trust and reputation systems for WSNs. Therefore, as far as we know, TRMSim-WSN is the first trust and reputation model tool covering this objective.

III. Generic Trust and Reputation Model

Each trust and reputation model has its own specific characteristics and particularities. However, most of them share the same abstract schema or pattern about what steps have to be given in order to complete a whole transaction in a distributed system making use of a trust and/or reputation model.

Therefore, one of the main targets followed by our work was to design and provide a trust and reputation models interface as generic as possible. So first of all, we identified the four main steps to be done in most of this kind of models [12], [13]. Figure 1 shows these steps.

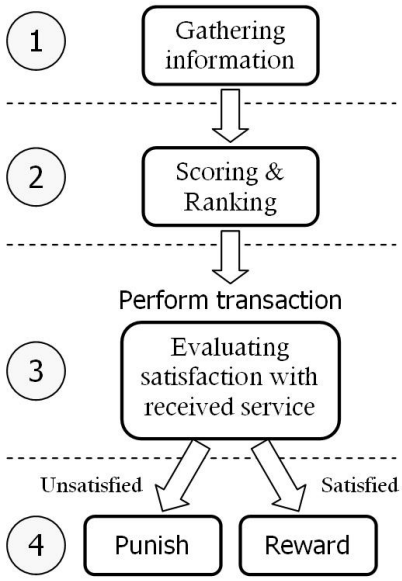

Fig. 1. Generic Trust and Reputation Model Scheme

We have developed an abstract Java class called `TRModel_WSN` containing one attribute: a set of generic parameters for trust and reputation models (abstract class `TRMParameters` containing the name of the parameters file and the parameters themselves in the form <parameter=value>).

In order to add a new trust and/or reputation model to the simulator both subclasses of `TRModel_WSN` and `TRMParameters` have to be implemented. A subclass of `Service` class could be also defined in order to specify more details or characteristics (such as associated costs or quality parameters, for instance) of a certain service.

Additionally, class `TRModel_WSN` defines the five public abstract methods shown in table I in order to accomplish the steps illustrated in figure 1.

The first method, `gatherInformation`, is responsible for collecting or gathering the necessary information from other nodes in the network (indirect experiences, recommendations, reputation values, etc.) if we are dealing with a pure reputation model, direct experiences or pre-trusted nodes, if what we have is a pure trust model, or a combination of both, which is the most common case.

Returned Value	Method Name	Arguments
`GatheredInfo`	`gatherInformation`	`Client Service`
`Vector<Sensor>`	`scoreAndRanking`	`Client GatheredInfo`
`Outcome`	`performTransaction`	`Vector<Sensor> Service`
`Outcome`	`reward`	`Vector<Sensor> Outcome`
`Outcome`	`punish`	`Vector<Sensor> Outcome`

TABLE I
TRModel_WSN ABSTRACT METHODS

Its first parameter is the `Client` who is requesting the desired service and, therefore, needs the application of the trust and reputation model in order to find the most trustworthy or reputable server offering the `Service` given as a second parameter.

It returns a `GatheredInformation` object. Currently this class only contains the paths leading to those servers which are candidates to be selected as service providers. Each model can create a subclass of this one including the specific information needed to work.

The second method, `scoreAndRanking`, receives the gathered information from the previous one and scores each path leading to a server, returning either a sorted collection of these servers (according to the score received) or the path leading directly to the most trustworthy server found.

The third abstract method belonging to class `TRModel_WSN`, called `performTransaction`, receives as a parameter the path found in the previous step, so it can actually apply for the required service to the server selected as most trustworthy or most reputable by the implemented model.

Then the server, according to its goodness, will provide exactly the same service it has been asked for, a worse one or even a better one, in some cases. Once the client receives the service, it assesses its satisfaction and returns its value in an `Outcome` object (necessary to perform some statistics in order to evaluate the accuracy of the model). Some models would store in this step that transaction satisfaction as a direct experience.

Finally, the last two methods, `reward` and `punish`, carry out the fourth step pointed out in the scheme shown in figure 1. That is, they perform the reward and punishment, respectively, to the server who has been selected to have the transaction with. Depending on the satisfaction of the client with the supplied service, one or the other will be applied.

They both receive two parameters: the path leading to the most trustworthy or reputable server found in the second step and the outcome got in the third one, containing, among other things the satisfaction or dissatisfaction of the client with the received service.

It is worth mentioning that there are, however, some trust and reputation models which do not apply any additional punishment and/or reward to those nodes the interaction has

been carried out with. Thus, these two methods may not have any particular code associated depending on the particular trust and reputation model being implemented and addapted to the TRMSim-WSN proposed architecture.

Regarding the parameters needed for the trust and reputation model, abstract class `TRMParameters` defines several protected methods, used to store and retrieve generic parameters of any of the primitive types.

Finally, figure 2 shows a brief class diagram including the main classes and their relationships involved in the design of our generic trust and reputation models interface.

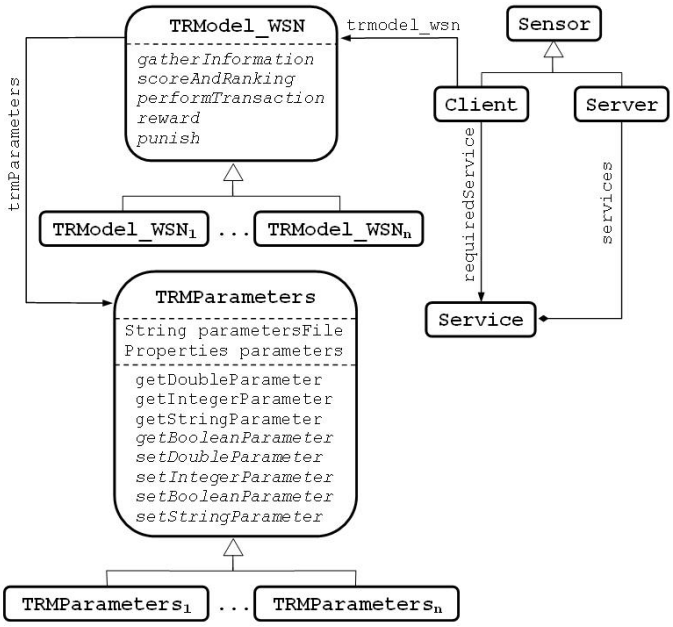

Fig. 2. Class diagram of main classes of the generic trust and reputation models interface

As it can be observed, each `Client` has its required service as an attribute, while each `Server` has a collection of offered services. Both are subclasses of `Sensor` class. Therefore, every client uses its trust and reputation model in order to: i) search the most trustworthy and/or reputable server offering the desired service, ii) apply for that certain service to the selected server and assess its satisfaction with the actually received service, and iii) punish or reward the service provider according to that satisfaction.

A further and more detailed explanation on how to add successfully a new trust and/or reputation model to the TRMSim-WSN simulator can be found in [7].

IV. TRMSim-WSN

In this section we will formally present and describe our proposal of Trust and Reputation Models Simulator for Wireless Sensor Networks, called TRMSim-WSN [7]. A screenshot of the main window of TRMSim-WSN can be observed in figure 3.

A. Network settings

The very first step to be carried out when using our simulator is to create a new WSN. To do that, there are two fields where we can establish the maximum and the minimum number of sensors we want our networks to have, as well as a slide bar to set the wireless range of every sensor. Those three parameters will determine the links density of the network (i.e., the neighborhood of every node).

Additionally, we can select which percentage of the nodes we want to act as clients requiring a default service. The rest of them will act, therefore, as servers. We can also say which percentage of those servers will not offer the required service and will then only act as relay nodes. Finally, regarding the servers who actually offer the desired service it is possible to determine the percentage of them who will be malicious ones, that is, they will not provide the service the are actually offering, but a worse one or even any service.

Once we have set all those parameters according to our needs, a new random WSN can be created just by pushing the bottom labelled "New WSN". It is also possible to load a WSN from a XML file by pushing "Load WSN" button, and to save the current one into a XML file through the "Save WSN" button.

If we want to evaluate the WSN we currently have, but with different links density, we can change the wireless range parameter and push "Reset WSN" button.

B. Simulation settings

The next thing to configure are the simulation settings. First we can determine the number of executions we want for our simulations, that is, the number of times every client in the network will ask for its default service, making use of the selected trust and reputation model. We can set the number of different random WSNs we want as well, according to the settings described in the previous subsection.

We can take some decisions regarding the visual or graphic presentation of the networks to be tested in our simulations. For instance, we can decide whether we want the wireless ranges to be shown or not, as well as the links connecting sensors or the identifier of each one of them.

TRMSim-WSN is initially released with two trust and reputation models: BTRM-WSN [3] and PeerTrust [14]. Furthermore, the parameters panel allows us to set the input parameters file, or to manually specify the value of each parameter needed by the current selected trust and reputation model.

Since one of the main characteristics of WSNs are their constraints about battery and energy consumption, a dynamic WSN can also be simulated, where some sensors swap into an idle state for awhile if they do not receive any request within a certain period of time. A sensor in an idle state does not receive nor transmit any message or packet. After a certain timeout they wake up again.

Once we have established all the previous settings, we are ready to start our simulations. If we want to run a simulation only over current network, we should press "Run WSN"

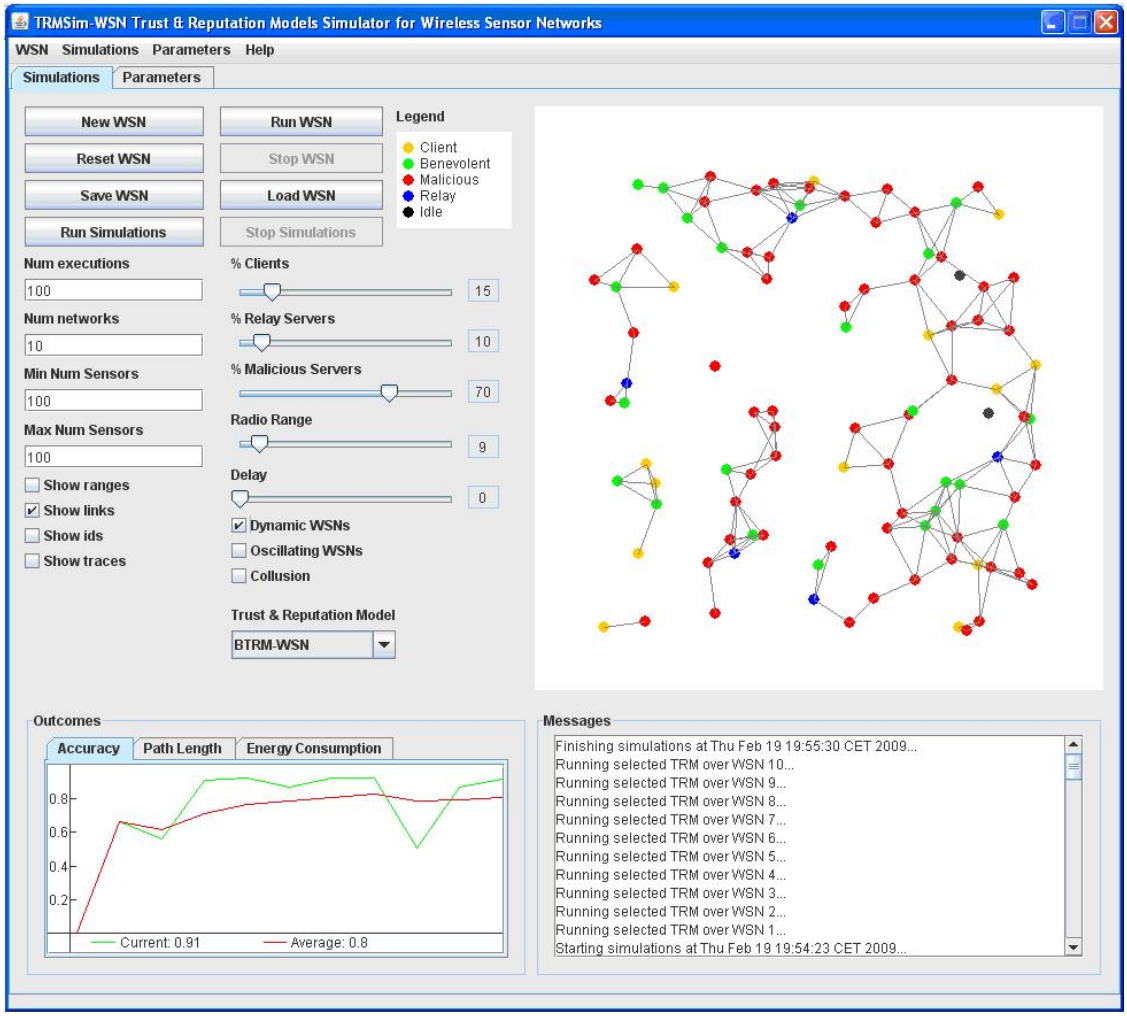

Fig. 3. TRMSim-WSN. Trust and Reputation Models Simulator for Wireless Sensor Networks

button. "Stop WSN" button allows us to force the finishing of that simulation.

Otherwise, if what we want is to run a simulation over a given number of random WSNs, then we have to push button labelled "Run Simulations". We can stop that simulation whenever we want by pressing "Stop Simulations" button, and current outcomes will be shown.

Finally we can also add some delay between each simulated network, if we need to check the topology of every tested WSN. The maximum value corresponds to one second.

C. Oscillating behavior and collusion

In order to test the accuracy of every simulated trust and reputation model we have included two security threats [15] to our simulator. First one has to do with the oscillating behavior of the servers offering the requested service.

Therefore, if that option is selected, after every 20 executions (i.e. transactions or interactions), each malicious server becomes benevolent. Then the same percentage of previous malicious servers are randomly chosen to be now malicious (note that with a scheme like this a malicious server could remain malicious after 20 executions).

The second security threat introduced consists of the possibility for the malicious servers to form a collusion among themselves. That implies that every malicious sensor will give the maximum rating for every other malicious sensor, and the minimum rating for every benevolent one.

A good trust and reputation model should quickly react against these behavioral changes and collusions and readapt itself in order to prevent selecting a malicious node as the most trustworthy or reputable one.

D. Outcomes and messages

Finally, two panels help us to know what has happened or is currently happening in the simulator, and which are the results of the last simulation.

In the messages panel, for instance, several messages are shown containing useful information like the instant when the last simulation started or finished, or which is the current WSN being tested. Moreover, every action such as creating a

new WSN, loading or saving current one or showing ranges, identifiers and links, among others, are also recorded and shown there.

On the other hand, the outcomes panel lets us know the results of the current simulated network, or the average outcomes for a whole simulation. Three important values can be observed here: the accuracy of the model, the average length of all the paths found by every client of every simulated network, and the energy consumed by the model (for future work). Additional panels can be easily added if required in order to show more details about the experiments.

The average satisfaction is computed collecting the satisfaction of every client belonging to each one of the tested WSNs. However, clients who can not reach any benevolent server are not taken into account for computing these outcomes (since any trust and reputation model is useful in that situation).

In figure 3 we can observe that a simulation over 10 random dynamic WSNs (with 100 sensors each one) has been carried out using BTRM-WSN model. There were a 15% of clients, an 8.5% ($85\% \cdot 10\%$) of relay sensors, a 53.55% ($85\% \cdot 90\% \cdot 70\%$) of malicious servers and a 22.95% ($85\% \cdot 90\% \cdot 30\%$) of benevolent ones. The average number of hops needed to reach the most trustworthy server was 6.04 and the average percentage of times that the model selected a benevolent server as the most trustworthy one was 80%.

V. Conclusions and Future Work

A number of network simulators can be found nowadays, allowing us to test low level communication protocols. Nevertheless, there is a lack of simulators aimed to check the correctness and accuracy of trust and reputation models for distributed systems and, specifically, for WSNs.

In this paper we have presented TRMSim-WSN, a novel trust and reputation models simulator for wireless sensor networks. As far as we know, this is the first simulator of these characteristics for WSNs. We have shown the generic trust and reputation models interface we have designed and developed and explained how a new trust and reputation model can be easily added to the simulator.

We have also described the main features and possibilities that TRMSim-WSN offers, and how to configure it in order to carry out customized simulations.

Nonetheless, several improvements and enhancements could be applied to TRMSim-WSN. For instance, we are planning to add an energy consumption module to determine the overhead introduced by each simulated model. Another interesting option would be the possibility of selecting a specific sensor and changing its properties (services offered, goodness, etc.). Mobile wireless sensor networks, where nodes can move and change their position along the time is also a new feature we are planning to incorporate to the TRMSim-WSN simulator.

Acknowledgment

This work has been partially funded by project *Middleware de gestión de Identidades de Seguridad en TRansacciones electrónicAs basado en código Libre* (MISTRAL) with code TIC-INF 07/01-0003), and by project *Infraestructura de Servicios Ubicuos y de Comunicaciones en Redes Vehiculares* with code TIN2008-06441-C02-02. It has been also funded by a Séneca Foundation grant within the Human Resources Research Training Program 2007. 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] K. Römer and F. Mattern, "The Design Space of Wireless Sensor Networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54–61, dec 2004.

[2] F. Li and Y. Wang, "Routing in vehicular ad hoc networks: A survey," *Vehicular Technology Magazine, IEEE*, vol. 2, no. 2, pp. 12–22, jun 2007.

[3] F. Gómez Mármol and G. Martínez Pérez, "Providing Trust in Wireless Sensor Networks using a Bio-inspired Technique," in *Proceedings of the Networking and Electronic Commerce Research Conference, NAEC'08*, 2008.

[4] A. Boukerche, L. Xu, and K. El-Khatib, "Trust-based security for wireless ad hoc and sensor networks," *Computer Communications*, vol. 30, no. 11-12, pp. 2413–2427, 2007.

[5] S. Buchegger and 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, jun 2004.

[6] F. Almenárez, A. Marín, C. Campo, and C. García, "PTM: A pervasive trust management model for dynamic open environments," in *Privacy and Trust*. Boston, USA: First Workshop on Pervasive Security and Trust, aug 2004.

[7] Félix Gómez Mármol, "TRMSim-WSN, a Trust & Reputation Models Simulator for Wireless Sensor Networks," http://ants.dif.um.es/~felixgm/research/trmsim-wsn.

[8] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai, "A survey of peer-to-peer network simulators," *Proceedings of the 7th Annual Postgraduate Symposium (PGNet '06)*, 2006.

[9] B. L. Titzer, D. K. Lee, and J. Palsberg, "Avrora: scalable sensor network simulation with precise timing," 2005, pp. 477–482.

[10] Y. Zhang, W. Wang, and S. Lü, "Simulating trust overlay in p2p networks," in *International Conference on Computational Science (1)*, 2007, pp. 632–639.

[11] K. K. Fullam, T. Klos, G. Muller, J. Sabater-Mir, K. Barber, and L. Vercouter, "The Agent Reputation and Trust (ART) Testbed," in *Trust Management*, ser. LNCS, no. 3986, Fourth International Conference, iTrust 2006. Pisa, Italy: Springer, may 2006, pp. 439–442.

[12] S. Marti and H. Garcia-Molina, "Taxonomy of trust: Categorizing P2P reputation systems," *Computer Networks*, vol. 50, no. 4, pp. 472–484, mar 2006.

[13] Y. Sun and Y. Yang, "Trust Establishment in Distributed Networks: Analysis and Modeling," in *Proceedings of the IEEE International Conference on Communications (IEEE ICC 2007), Communication and Information Systems Security Symposium*, Glasgow, Scotland, jun 2007.

[14] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust in Peer-to-Peer Communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004.

[15] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," in *Proc. of the International World Wide Web Conference (WWW)*, Budapest, Hungary, may 2003.

[16] E. Aivaloglou, S. Gritzalis, and C. Skianis, "Trust establishment in sensor networks: behaviour-based, certificate-based and a combinational approach," *International Journal of System of Systems Engineering*, vol. 1, no. 1-2, pp. 128–148, 2008.

[17] A. Mitseva, E. Aivaloglou, M. Marchitti, N. R. Prasad, C. Skianis, S. Gritzalis, A. Waller, T. Baug, and S. Pennington, "Towards adaptive security for convergent wireless sensor networks in beyond 3g environments," *Wireless Communications and Mobile Computing*, 2008.

[18] E. Aivaloglou, S. Gritzalis, and C. Skianis, "Towards a flexible trust establishment framework for sensor networks," *Telecommunication Systems: Modeling, Analysis, Design and Management*, vol. 35, no. 3-4, pp. 207–213, 2007.