TrustSets - Using Trust to Detect Deceitful Agents in a Distributed Information Collecting System

Quang Anh Nguyen Vu, Benoit Gaudou, Richard Canal, Frédéric Armetta, Salima Hassas

TrustSets - Using trust to detect deceitful agents in a distributed information collecting system

Quang Anh NGUYEN VU*†‡, Benoit GAUDOU*†, Richard CANAL*†, Salima HASSAS‡ and Frédéric ARMETTA‡
* UMI 209 UMMISCO, Institut de Recherche pour le développment (IRD), Bondy, F-93143, France
† MSI, Institut de la Francophonie pour l'Informatique (IFI), Ha Noi, Viet Nam
Emails: nguyenvu.quanganh@yahoo.com, benoit.gaudou@alumni.enseeiht.fr, richard.canal@auf.org
‡ Laboratoire LIESP, université Claude Bernard Lyon 1, France
Emails: hassas@bat710.univ-lyon1.fr, farmetta@bat710.univ-lyon1.fr

***Abstract*—This paper presents a study on how to improve a distributed information collecting system in which information is collected by a multi-agent system constituted by communicating agents, assuming the hypothesis that some agents of this system can deliberately (*liar agents*) or in good faith (*defective agents*) produce or communicate incorrect information. To ensure the coherence of the information system under these constraints, we aim to gradually limit the impact of the incoming perturbations. To reach this goal, we propose that each agent develops its own communication strategy from a TrustSet it builds using information collected by itself and information received from agents it communicates with.**

I. Introduction

We consider a distributed information collecting system in the form of a multi-agents system (MAS). Each agent of the system can search information, collect information items and communicate. It collects information either directly or indirectly via communication with other agents. We assume that some agents disturb the system by disseminating false or inaccurate information either because their perception is flawed or because their interest goes against the community's one. In this article, the information to collect is assumed invariable during the experimentation.

This article provides a methodology to deal with such a perturbed distributed information collecting system. We study ways to ensure the coherence of the system (*i.e.* adequation between the agents' environment representation and the real environment) and its robustness (*i.e.* the agents' capacity to adopt strategies allowing to obtain this coherence despite the disturbed communication system). To limit the influence of agents transmitting incorrect information, we propose to use the concepts of trust and reliability. TrustNet [1] appears to be a promising way to allow each agent to build its own evaluation of the other agents. One extension of the TrustNet model, the TrustSet, is introduced in this article. It allows keeping track of the information path. Agents start out with no knowledge about the behavior of other agents. They improve this knowledge by using direct and indirect interactions and by considering the path followed by the information from its source. While interacting, the model of trustworthiness is refined and used to judge the reliability of information in order to reject undesired communications. We show on a simple mapping example in which a patrol of possibly flawed robots maps a dangerous area that use of TrustSets can make the system more robust against deceitful agents by improving communication flow.

Section I introduces the problem. Trust and related works are detailed in Section II. The TrustSet is presented in Section III before describing how agents use TrustSets to define their communication strategy in Section IV. Section V presents the "danger mapping" application and its corresponding simulation before showing the comparative results obtained using different communication strategies. Finally Section VI presents the future research.

II. Trust

A. Why do we need trust ?

Communication is a source of enrichment and also perturbation in an information collecting system where information can be altered by unreliable agents. To limit the influence of deceitful agents, we propose to work on communication strategies in an attempt to eliminate both false information and agents that are responsible for spreading false information in the system. We give agents the ability to evaluate the reliability of other agents and to choose the agents they want to interact with. For this purpose, one of the most efficient tools is the concept of trust: in this paper, trust is used in order to identify and isolate unreliable agents. See [2] to understand the importance of trust in agents' theory.

Trust in MAS can be simply defined as the probability with which an agent believes that another agent will enter in a beneficial interaction with it. So in the sequel, trusts will have values in $[0, 1]$.

B. Related works

The model of trust by Marsh [3] takes into account only direct information and experiences. In a more complete model, ReGreT, by Sabater *et al.* [4], an agent computes its trust in another agent using direct experiences and reputation. Up to now, a lot of trust and reputation models have been published [5]. Let us mention on line reputation models where the reputation mechanism is based on ratings given by users after a transaction. SPORAS [6] is an improved version of such

models: only the most recent rating between two users is considered. We can also refer to the model based on Dempster-Shafer theory proposed by Yu and Singh [7] and to the Mui *et al.*'s one using a Bayesian analysis [8]. In some models, like Schillo *et al.*'s one [9], agents communicate not only factual information but also trusts they have in other agents. Our work is based on this model. Compared to the information items collected by the distributed information collecting system (*data*), all information about trust are considered as *metadata*. Our agents as Schillo's ones are able to exchange data and metadata. Agents can thus build a network of trust values called 'TrustNet'. The final trust value of an agent towards another one is an aggregate of direct experiences and testimonies.

We have chosen this last model for two reasons. First, in distributed information collecting systems, agents share all their information, information they collect themselves and information they receive from other agents. Second, as we consider a system without centralized control or information set, communication are only one-to-one. Thus a classic reputation system cannot be used. TrustNets provide a kind of local and distributed reputation. Agents can compute trusts based on direct experiences (which provide them a *direct trust* on other agents) and on experiences between other agents (*indirect trust*) not represented here by reputation but by information carried by TrustNets.

III. TrustSets

When agents use information from others agents, it is important for them to know which agent is trustworthy and which is not. For this purpose, we introduce a data structure derived from the TrustNet that we call 'TrustSet' which is a pair of TrustGraph and TrustTable. The public part (the TrustGraph) is a directed graph which contains both direct trust and indirect trust, while the private part (the TrustTable) is a simple table which contains the intrinsic trust (a trust value estimated from direct and indirect trusts).

The main difference between Schillo's model and ours is on the *data*: agents in Schillo's model transmit data related to the trustfulness of an agent, whereas our agents transmit information about the environment, from which agents have to deduce trust. We thus present algorithms for computing trusts from received information, but also algorithms for computing the information reliability from the trust values, *i.e.* from the TrustSet. Moreover in Schillo's model, agents communicate information about only one agent behaviors; in contrarily, we propose that agents exchange their whole TrustGraph. We thus have to propose methods to compute intrinsic trusts which are contained in Trust Table, to merge TrustGraphs and to deal with trust incoherence on shared paths.

A. State of the art on trust networks

Trust networks in MAS consist of transitive trust relationships between connected agents. Trust can be derived by analyzing the trust paths linking the agents together. Two operators are required to build such trust networks: *aggregation* to deal with trusts aiming the same object but coming from different sources and *propagation* to compute trust along a path. Several models have been developed according to the methods used to implement these operators. In [10], first-hand observations are exchanged between neighboring nodes and merged only if neighbors' opinions are close to its own opinion. In EigenTrust by Kamvar *et al.* [11], in order to aggregate local trust values, a node asks its neighbors for their opinions about other peers. Neighbor's opinions are weighted by the trust the node places on them. [12], [13] propose similar algorithms that evaluate trust by combining opinions from selected groups. Some approaches work on local interaction rules using algebraic graph theory such as [14].

B. TrustGraphs, TrustTables and TrustSets

Agents use **TrustGraphs** to compute their trust in other agents. A TrustGraph is a directed graph which contains both direct trusts and indirect trusts (received via communication). Nodes represent agents. The root node is the owner of the TrustGraph (A in Figure 1). Two nodes connected by an arrow mean that agents have met each other. Edges carry information about agents' trust estimation. Values assigned to arrows connecting the owner to other nodes represent the owner's trust value in agents it has already met. Each value represents the ***direct trust*** of A in another agent (i.e. B in Figure 1) computed by comparing data from A and B. It is noted DT_{AB}. Any arrow from B to C carries an ***indirect trust***, denoted IT_{BC}, the trust of B in C communicated by B to A. Each node is annotated with a value, the ***intrinsic trust***, denoted T_{AB}, which represents the trust of A in B taking into account both direct and indirect trusts. Intrinsic trust values will be stored in a table called **TrustTable**.

A TrustGraph of agent X denoted as $TG_X =< \{Node_X\}, \{< Arc_X, Value_X >\} >$and its TrustTable $TT_X =< \{< Node_X, Value_X >\} >$ form a **TrustSet** denoted as $TS_X = (TG_X, TT_X)$. The TrustGraph is built thanks to collected or transmitted information. It represents the *public part* of the TrustSet and will be communicated to other agents. The TrustTable is computed thanks to algorithms that can be specific to a particular agent; it thus represents the *private part* of the TrustSet and will not be communicated to other agents.

An example of TrustSet built by A is proposed in Figure 1: it includes a TrustGraph formally represented by $TG_A =< \{A, B, C\}, \{< AB, DT_{AB} >, < BC, IT_{BC} >\} >$ and a TrustTable represented by $TT_A =< \{< A, T_{AA} >, < B, T_{AB} >, < C, T_{AC} >\} >$. DT_{AB} is the direct trust of A in B, IT_{BC} the trust of B in C communicated to A by B. T_{AA} is the intrinsic trust of A in itself, T_{AB} the intrinsic trust of A in B (calculated from the direct trust DT_{AB} and from all the indirect trusts IT_{XB} associated to arcs leading to B).

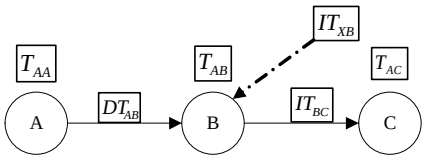

Figure 1. Example of TrustSet

C. TrustSet dynamics

1) Initializing TrustSet: Initially, each agent builds its own TrustSet. The TrustGraph is initialized with the root node and no arrow. The TrustTable contains one value, the trust of the root agent in itself, initialized at 1 because it has no reason to doubt on itself. Both TrustGraph and TrustTable will then be updated thanks to information exchanged with other agents. Note that we assume that each agent has an *a priori* trust in other agents used as initial value for the first time it communicates with another agent, that is noted in the sequel T_{init}.

2) Communicating TrustSet: Each time an agent A communicates with an agent B, it will eventually communicate to B its data but also some of its metadata. In particular, it will share its TrustGraph, which contains all public metadata, but will not share its TrustTable because it is built by personal calculation and thus contains private information. When an agent receives a TrustGraph, it integrates it in its own one. Then it uses the obtained TrustGraph to update its TrustTable.

3) Merging TrustGraphs: We consider an information exchange between agents A and B. We take the point of view of A but the process is the same for B. The update of A's TrustGraph will be computed in 3 stages:

- A calculates its trust DT_{AB} in B or updates the existing value by comparing its own data with received ones;
- A connects B 's TrustGraph to its own TrustGraph;
- A corrects every inconsistency in the shared paths.

a) Stage 1 - Computing direct trust by comparing information: To compute its trust in B, agent A compares its own direct data D_A with D_B, the ones transmitted by B. The computation follows 3 steps:

- A selects only the comparable data, which means data about the same items. Their number is denoted β.
- A computes the incoherence level between D_A and D_B. For this purpose, it uses a distance denoted δ, where $\delta(x, y)$ represents the distance between the data x from D_A and y from D_B. Intuitively, $\delta(x, y) = 0$ means that the two pieces of information are coherent (*i.e.* they carry the same information on the same item), whereas $\delta(x, y) > 0$ represents the inconsistency degree between both data. The incoherence level (denoted Inc_level) is computed following the formula:

$$Inc\_level = \frac{\sum_{i \in \{1,\beta\}} \delta(x_i, y_i)}{\beta * \delta_{Max} info}$$

  with (x_i, y_i) each pair of comparable data and $\delta_{Max} info$ the largest distance between incoherent information.
- A threshold μ represents the maximum incoherence level acceptable by an agent before decreasing its trust value in another agent. If $Inc\_level < \mu$, A will increase its trust in B by a factor τ^+; if $Inc\_level > \mu$, A will decrease its trust in B by a factor τ^-; if $Inc\_level = \mu$, A lets its trust in B unchanged.

The value of $\tau^+$and τ^- are computed depending on properties that designers want to give to the system. For this, we introduce two thresholds: an upper threshold (Upp) and a lower threshold (Low). Between Upp and 1, agents are regarded as "reliable". Between Upp and Low, agents are "under observation". Between 0 and Low, agents are regarded as "unreliable". We also denote NI the estimated number of interactions each agent needs to reach its objective and ρ_{stab} the stabilization rate ($\rho_{stab} \in [0, 1]$), representing the stage of the simulation after which designers consider the classification of agents in reliable and unreliable sets to be done. Thus $\rho_{stab} * NI$ represents the number of interactions needed to pass from T_{init} to Upp or Low (which means that $T_{init} + NI * \rho_{stab} * \tau^+ = Upp$). We can thus express both factors:

$$\tau^+ = \frac{(Upp - T_{init})}{\rho_{stab} * NI} \quad \text{and} \quad \tau^- = \frac{(T_{init} - Low)}{\rho_{stab} * NI}$$

b) Stage 2 - Merging two TrustGraphs: Then A builds an intermediate TrustGraph $TG_{AB} =< \{Node_*\}, \{< (Arc_*, Value_*) >\} >$ from: $TG_A =< \{Node_A\}, \{< (Arc_A, Value_A) >\} >$ and $TG_B =< \{Node_B\}, \{< (Arc_B, Value_B) >\} >$. The new nodes set consists of all the nodes of both TrustGraphs: $Node_* = Node_A \cup Node_B$. The new arrows set includes all the arrows from both sets Arc_A and Arc_B. Moreover we add to this set the arrow AB to link both TrustGraphs and we remove any arrow coming back to A to avoid cycles. We thus have: $Arc_* = \{AB\} \cup Arc_A \cup Arc_B \setminus \{BA, ..., XA\}$. Albeit these arrows are deleted, their associated trust values are taken into account in the TrustTable update and influence the trust the agent has in itself.

We associate the value DT_{AB} to the arc between AB. For each other arc, values of the arrows are taken from their original TrustGraph.

c) Stage 3 - Managing trust incoherence on shared paths: There is inconsistency on a shared arrow when an arrow appears in both TrustGraphs with different values on it. This case can typically appear when A and B meet C at two different time points. To avoid this incoherence, a new value of the incoherent trust value is computed as follows. Let XY be an arrow common to TG_A and TG_B. XY carries the value IT_{XY_A} in TG_A and IT_{XY_B} in TG_B.

- We build the sets $Path_A$ and $Path_B$ of all paths of TG_A and TG_B including the arrow XY. To avoid cycle problems, we choose only the shortest path.
- We compute the trust value $IT_{XY_{AB}}$ of the common arrow in the TrustGraph resulting from the merge of A's and B's TrustGraph with the formula below:

$$IT_{XY_{AB}} = \frac{\sum_{X \in Path_A} T_{AX} * IT_{XY_A} + \sum_{Y \in Path_B} T_{AY} * IT_{XY_B}}{\sum_{X \in Path_A} T_{AX} + \sum_{Y \in Path_B} T_{AY}}$$

This new trust value is the average of trusts on the shared path balanced by the sum of trusts along all paths.

4) Updating the TrustTable: The intrinsic trusts must be recalculated after the update of trusts in TrustGraphs if one of the basic elements changes or if a new element enters into its calculation. Two steps are required to calculate all intrinsic trusts in the A's TrustTable. First, we calculate intrinsic trusts on all new nodes transmitted by B. Second, we calculate the intrinsic trusts of all impacted nodes except nodes computed

before.The intrinsic trust of A in an agent X is calculated by this formula:

$$T_{AX} = \frac{T_{AA} * DT_{AX} + \sum_{Y \in AGENTS}(T_{AY} * IT_{YX})}{T_{AA} + \sum_{Y \in AGENTS} T_{AY}}$$

Note that $T_{AX} = DT_{AX}$ when only one arc goes from A to X and T_{AA} is set to 1.

IV. How do agents use their TrustSet to improve their performance and the system robustness ?

A. Impact of trusts on communication

According to the results obtained in their TrustTable, agents modify their communication strategy. The main idea is to separate agents more precisely at each step by separating untrusted agents (with which it has already been proved useless or harmful to communicate) from other agents (those who could yet provide useful information - under observation and reliable ones). An agent decides to stop communicating (partially or totally) with another agent as soon as the trust value computed for this agent falls under the Low threshold. This way, bad information ceases to flood the information network and gradually disappears due to the information reliability computation mode described in the next section.

B. Computing the information reliability

Via communication, agents receive conflicting information. Agents have thus to determine which ones are reliable and which are not. In order to calculate the information reliability, each agent uses a probability tree to represent one information item in its memory. Each value or each range of values for one information item is associated to a node of the tree. The pair $(\theta, \{A_i...A_m\})$ is associated to each edge, where θ is the probability for the information to be true (its reliability) and $\{A_i...A_m\}$ the information sources. Each time a new data comes, the agent updates these values in the tree. A major issue for an agent is to determine the reliability of one information item when several groups of agents give different information on the same item. It needs to compare the relative weights of these groups. For this purpose, a TrustWeight TW is assigned to each group depending on individual trusts as follows.

First, the TrustWeight of each group is evaluated by splitting the group in three sets (reliable, unreliable and others) according to the Upp and Low thresholds defined above. A weight is assigned to each agents community: α for reliable agents, γ for unreliable agents, β for other agents. Then, the balanced sum of the cardinals of these sets is computed. By setting appropriate values for α, β and γ, this method aims at creating an equilibrium between the quality of agents in a group and their quantity. If we note p_i each data associated to an information item p brought by a group G_i of $\{G, ..., G_n\}$, the set of groups having transmitted different data for the same item p , the agent computes the reliability of each data p_i as follows before choosing the data that has the maximal reliability:

$$reliability(p_i) = \frac{TW(G_i)}{\sum_{k \in [1,n]} TW(G_k)}$$

For the agent, the chosen data is the value of the item at this instant and the associated reliability the item reliability.

V. An example: Danger Mapping

A. Description

One of the main examples we have chosen to test the efficiency of TrustSets to improve the detection of deceitful agents in a distributed information collecting system, that we will refer to as 'Danger Mapping' in the sequel, figures a swarm of localized mobile robots patrolling in an unknown land. The objective of each robot is to build the most complete, precise and reliable map of the land using least resources as possible. In particular, their job consists in detecting dangerous spots and evaluating their dangerousness degree. Robots can detect directly the state of a nearby zone thanks to their sensors. They can also communicate with other robots to exchange knowledge about the land. We assume that each robot has limited perception and communication ranges. Among the robots, some can collect and transmit false information due to flawed sensors.

B. Modeling and Simulation

The land with its danger zones is figured by a 75 x 75 grid. The space is toroidal - meaning that if robots (figured by agents) move off one edge of the grid, they appear on the opposite edge. An integer value is randomly given to each patch of the grid, from 0 to n, figuring the danger level of the zone. Agents can collect data in their perception range, communicate in their communication range, update their data and metadata bases and move. Unreliable agents are defined as agents unable to detect the correct danger level of a zone, as if the spectrum of their sensors was shifted out. The simulation can stop either when one agent knows the whole map or all the reliable agents know the map, as the user chooses.

1) The Goals: A danger mapping simulator[1] has been developed in order to investigate the possibilities to improve the coherence and robustness of an information collecting system based on robots, some of them deceitful ones. The key questions addressed are :

- What is the effect of unreliable robots on the community?
- How can deceitful robots be detected?
- What are the compared performances of the system when robots use several strategies:
  - robots do not communicate
  - robots communicate only data
  - robots communicate data and ReputationTables[2] .
  - robots communicate data and TrustGraphs but keep private TrustTables.
- How robust is the system to deceitful robots?

[1]Our mapping simulation is written in the simulation platform GAMA[15].

[2]In this strategy, a simple reputation table is used to store trust value on others agents. These values are computed as follows: when agent A meets agent B, firstly it calculates its trust on B (DT_{AB}) as in stage 1 of the TrustGraphs merging process. Then A updates its own trust values in all the agents met by B ($Agts_B$) via the formula:

$$T_{AX} = \frac{T_{AX}^{old} * T_{AA} + DT_{AB} * T_{BX}}{T_{AA} + DT_{AB}}$$

with $X \in Agts_B$ and T_{AX}^{old} represents the old A' trust on X.

2) Parameters: The simulation parameters are sorted into three categories: * *Environment*: we can fix the size of the grid, the number of reliable agents, the number of unreliable agents and the number of dangerous places; * *Agent*: we can choose the perception range of the agents, their communication range and one among four representative communication strategies; * *Trust*: here we fix the reliability/unreliability thresholds for the agents (and information), the stabilization rate (default: 0.5), the level of contradictory information (default: 0.05).

3) Evolution of the simulation:

1) Initialization : A number of agents, some reliable, some unreliable, are randomly created.
2) Main loop : Each agent executes the following actions in sequence until the end of the simulation.
   a) Collect accessible land data in the perception domain
   b) Communicate with reliable or under observation agents (*as estimated at the current step*) in the communication domain
   c) Update data bases (land data bases & metadata bases - Trust Sets) each time a new information is added
   d) Move
3) End : The simulation can stop either when one agent knows the total map or when all the reliable agents know the map, as the user chooses.

4) Results: On the 75 x 75 grid, we create 65 dangerous zones and 20 agents with appropriate ranges so that agents can meet a sufficient number of different agents along the experimentation to build significant TrustSets. To test the performance of the proposed approach, we tested the influence of different parameters and different strategies of communication: (A) no communication, (B) communication without trust support, (C) communication with Reputation Table, (D) communication with TrustSets.

a) Computation of the reliability threshold: We have conducted some experiments with communication strategies (C) and (D) that show that if the reliability threshold is greater than 0.6, the distance between the map obtained by the agents and the real map is equal to zero. For instance if the chosen threshold is 0.55, the average number of erroneous danger zones kept by the reliable agents at the end of the simulation is 6.6 for strategy (C) and 0.1 for strategy (D). We thus use the value of 0.6 for the reliability threshold all along the experimentations.

b) Performances of the system vs. 4 communication strategies: Figure 2 shows the exploration time (simulation steps) necessary for the reliable agents to get a map identical to the real map (where all dangerous places are correctly situated and evaluated) with the four communication strategies above depending on the ratio of unreliable agents.

We note that when the proportion of unreliable agents in a system is smaller than 0.7, using the TrustSet strategy to detect deceitful agents is more effective than other strategies, *i.e.* the time in order that all the agents in the system get the real map is the best one due to benefits acquired from communication with other reliable agents. This result seems logical : the larger

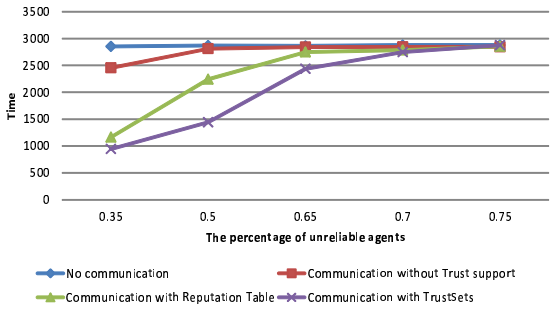

Figure 2. Number of steps to get a real map with 4 communication strategies

the number of reliable agents in the system, the higher the number of useful communication. On the other side, when the system counts many untrustful agents, the performances of the four communication strategies are identical because most of the information received are not reliable, so that agents must verify themselves the whole map. This number of 0.7 is the robustness limit of the system. This number is very interesting because it describes the proportion of unreliable agents a system can bear, in other words, its robustness. With a greater value, it is better for reliable agents not to communicate. We intend to show this point in a later publication by using the percolation theory [16] in a complex system approach of this problem.

Figure 3 illustrates the average time necessary for all reliable agents to detect correctly the map under all communication strategies. To get this result, we ran a great number of simulations for which the number of deceitful agents ranged from 1 to 15 with a total number of agents fixed. It shows that we can use the TrustSet policy to get good performances as long as we are under the system limit of 0.7. And more generally, when agents have no idea about the deceitful agents proportion, it shows that the best communication strategy is the communication with TrustSets.

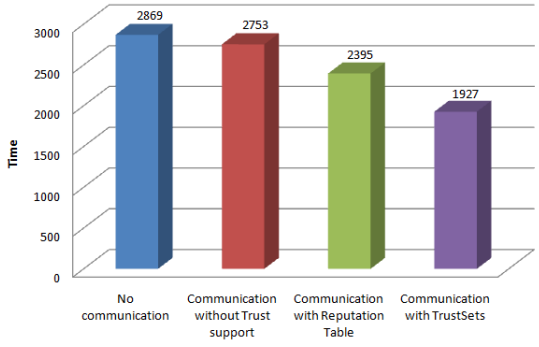

Figure 3. Average number of steps (by varying unreliable agents from 1 to 15 among 20 agents) to get a real map with 4 communication strategies

c) A basic communication strategy: The TrustSet strategy was also introduced to impact the communication system. As we said in Section V-A, when an agent knows that other agents send bad information, why would it persist in communicating with them ? It is better for this agent to stop communicating with this kind of agents as soon as it is certain that they are unreliable. The strategy adopted by agents in our simulation is as follows:

1) when an agent A meets an agent B, it can send information (its trust in B is above the Low threshold) or not (its trust in B is below the Low threshold)

2) it can receive information (B's trust in A is above Low) or not (B's trust in A is below Low)

3) it can consider the received information (its trust in B is above Low) or not (its trust in B below Low).

To demonstrate the benefits of such a communication strategy, we computed the number of meetings, the number of sent, received and handled messages. It is obvious that this basic communication strategy has an impact on the volume of communication. In Figure 4, we compare the number of meetings (which is also the number of communication when agents do not use trust) and the number of sent messages without impairing the performance of the system. We note that the volume of communications is much reduced (about 900 information exchanges for 1600 meetings). As it can be seen in Figure 4, agents reject communication from untrustful agents since the simulation step number 150. This number can change according to the value the user chooses at the beginning of the simulation for the stabilization rate. The lesser this value, the sooner the deceitful agents are dismissed from the communication system but the greater the possibility of mistakes.

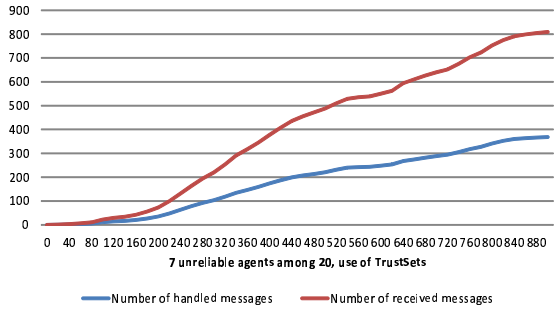

Figure 4. Comparison between the number of sent messages (red) and the number of meetings (blue)

All these results significantly confirm our choices. In particular, we show the circumstances (proportion of untrustful agents in a MAS) and limits (robustness of the system) under which a communication policy based on TrustSets can improve the comportment of a community made up of reliable and unreliable members. We show under these conditions that a TrustSet strategy gives better results than a ReputationTable strategy.

VI. Conclusion and Future Works

This paper addresses the problem of disturbed communities where information collection is altered by unreliable members. By associating a reliability to information and a trust to members of the community, each member improves its perception of the world. Considering that agents can work on direct (collected from the environment) and indirect (received from other agents) information, keeping this duality not only in stored, but also in transmitted data, gives to agents the ability to build and update better tuned information for data and more accurate TrustSets for metadata. It is one of the originalities of our approach. Actually, the computing of one agent's confidence in another agent takes into account at the same time the quality of the metadata transmitted by the other agent and the distance between the data gathered by both agents. This strategy finally helps an agent to enforce its communication with trusted agents and to reject communication from untrusted ones. The extension of the notion of TrustGraph to TrustSet, in which we distinguish public and private values, and the new developed fusion algorithms between TrustSets allow us to get better results on time than traditional ReputationTable strategies. So we show that sometimes in a perturbed community, it is better not to give all the information we get to other members when these members are perhaps not reliable. Our future work will focus on the community self-organization about communication management, on the structuring of sub communities according to their reliability and on the limits of perturbation a disturbed system can support by using a complex system approach.

References

[1] M. Schillo, P. Funk, and M. Rovatsos, "Who can you trust: Dealing with deception," in *Proceedings of Autonomous Agents '99 Workshop on "Deception, Fraud, and Trust in Agent Societies"*, Seattle, USA, May 1999, pp. 81–94.

[2] R. Falcone and C. Castelfranchi, "Social trust: cognitive anatomy, social importance, quantification and dynamics," in *Autonomous Agents '98 Workshop on "Deception, Fraud and Trust in Agent Societes"*. Minneapolis, USA, 1998, pp. 35–49.

[3] S. Marsh, "Formalising trust as a computational concept," Ph.D. dissertation, University of Sterling, 1994.

[4] J. Sabater-Mir, "Trust and reputation for agent societies," Ph.D. dissertation, Universitat Autonoma de Barcelona, July 2002.

[5] J. Sabater and C. Sierra, "Review on computational trust and reputation models," *Artificial Intelligence Review*, vol. 24, no. 1, pp. 33–60, September 2005.

[6] G. Zacharia and P. Maes, "Trust management through reputation mechanisms," *Applied Artificial Intelligence*, vol. 14, pp. 881–907, 2000.

[7] B. Yu and M. P. Singh, "A social mechanism of reputation management in electronic communities," in *Proc. of CIA '00*. Springer-Verlag, 2000, pp. 154–165.

[8] L. Mui, M. Mohtashemi, and A. Halberstadt, "Notions of reputation in multi-agent systems: A review," in *Proceedings of AAMAS'02*. Bologna, Italy: ACM, July 2002, pp. 280–287.

[9] M. Schillo, P. Funk, and M. Rovatsos, "Using trust for detecting deceitful agents in artificial societies," *Applied Artificial Intelligence, Special Issue on Trust, Deception and Fraud in Agent Societies*, vol. 14, no. 8, pp. 825–848, September 2000.

[10] S. Buchegger and J. Le Boudec, "The effect of rumor spreading in reputation systems for mobile ad-hoc networks," in *Proc. of WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[11] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," in *WWW '03: Proc. of the 12th Int. Conference on World Wide Web*. New York, NY, USA: ACM, 2003, pp. 640–651.

[12] S. Marti and H. Garcia-Molina, "Limited reputation sharing in p2p systems," in *EC'04: Proceedings of the 5th ACM conference on Electronic commerce*. New York, NY, USA: ACM Press, 2004, pp. 91–101.

[13] L. Xiong and L. Liu, "PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 843–857, 2004.

[14] T. Jiang and J. S. Baras, "Trust evaluation in anarchy: A case study on autonomous networks," in *INFOCOM*, 2006.

[15] E. Amouroux, C. Quang, A. Boucher, and A. Drogoul, "GAMA: an environment for implementing and running spatially explicit multi-agent simulations," in *10th Pacific Rim International Workshop on Multi-Agents (PRIMA)*, Thailand, 2007.

[16] D. Stauffer and A. Aharony, *Introduction to percolation theory, 2nd edition*. London: Taylor & Francis Ltd., 1994.