Applied Soft Computing 11 (2011) 345–355

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Original research paper

# Fuzzy reputation-based trust model

Ayman Tajeddine, Ayman Kayssi*, Ali Chehab, Hassan Artail

*Electrical and Computer Engineering Department, American University of Beirut, Beirut 1107 2020, Lebanon*

ARTICLE INFO

*Article history:*
Received 21 November 2007
Received in revised form
30 September 2009
Accepted 17 November 2009
Available online 22 November 2009

*Keywords:*
Trust
Reputation
Fuzzy systems
Distributed systems

ABSTRACT

We present PATROL-F (comPrehensive reputAtion-based TRust mOdeL with Fuzzy subsystems) as a comprehensive model for reputation-based trust incorporating fuzzy subsystems to protect interacting hosts in distributed systems. PATROL-F is the fuzzy version of our previous model PATROL, and aims at achieving a truly unique model incorporating various concepts that are important for the calculation of reputation values and the corresponding decisions of whether or not to trust. Among the incorporated concepts are direct experiences and reputation values, the credibility of a host to give recommendations, the decay of information with time based on a dynamic decay factor, first impressions, and a hierarchy of host systems. The model also implements the concepts of similarity, popularity, activity, and cooperation among hosts. In addition, PATROL-F's fuzzy subsystems account for humanistic and subjective concepts such as the importance of a transaction, the decision in the uncertainty region, and setting the result of interaction. We present simulations of PATROL-F and show its correctness and reliability.



## 1. Introduction

One of the most critical issues in distributed systems is security. For an entity to interact with another, it should trust the other entity to ensure the correctness and credibility of its responses. The ideal solution to this concern is to have an environment that is fully trusted by all its entities. However, because such a solution cannot be achieved, research has focused on trust and reputation to secure distributed systems.

Our proposed approach is truly unique and fully comprehensive incorporating fuzzy subsystems for subjective concepts and integrating various trust related characteristics. It requires that a host asks about the reputation of a target host that it wants to interact with. It calculates a reputation value based on its previous experiences with the target host in addition to the gathered reputation values from other hosts, and then it decides whether to interact with the target host or not. The initiator also evaluates the credibility of hosts providing reputation values by estimating the similarity, the activity, the popularity, and the cooperation of the queried host. Moreover, the initiator will use fuzzy subsystems to evaluate humanistic and subjective concepts such as the importance of a transaction, setting the result of interactions, and deciding whether to interact or not when the decision is not obvious. Hosts use different dynamic time decay factors that depend on the consistency of the interaction results with other hosts.

The rest of the paper is organized as follows: Section 2 surveys the previous work in the area of trust and reputation using fuzzy logic. Section 3 presents our previous trust model, PATROL, with all its parameters. Section 4 presents our proposed trust model, PATROL-F, with its fuzzy subsystems. The simulation results are presented in Section 5. Section 6 shows the system overhead and discusses some security issues. Finally, Section 7 presents some conclusions.

## 2. Previous work

Trust and reputation mechanisms have been proposed in various fields such as distributed computing, agent technology, grid computing, economics and evolutionary biology. In this section, we review the recent work done in reputation-based trust. Research groups have been focusing on fuzzy logic to propose effective trust models.

Ramchurn et al. [11] developed a reputation- and confidence-based trust model using fuzzy logic to assess previous interactions, where confidence is derived from direct interactions, and reputation is derived from indirect interactions or information gathered from other agents in the community. Shuqin et al. [14] proposed a method for building trust and reputation based on observations and recommendations while using fuzzy logic to deal with judgments. Castelfranchi et al. [2,8] developed a socio-cognitive trust model using fuzzy cognitive maps. Their model differentiates internal and external attributes and considers four kinds of belief sources that are direct experiences, categorization, reasoning, and reputation. Their implementation allows for changing components depending on different situations and different agent personalities.

* Corresponding author. Tel.: +961 1 374 374; fax: +961 1 744 462.
*E-mail address:* ayman@aub.edu.lb (A. Kayssi).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.11.025

**Nomenclature**

$repY/X_{before\_int}$ the value that is being calculated *now* for the reputation of $Y$ at $X$, before their interaction (Eq. (1))

$repY/X$ the last calculated reputation of $Y$ with respect to $X$, modified to account for the time interval since the last time that host $X$ was interested in finding host $Y$'s reputation (Eq. (1))

$\sum_i \alpha_i repY/X_i$ the weighted sum of reputations of $Y$ as reported by the neighbors of $X$ ($X_i$) (Eq. (1))

$\sum_j \beta_j repY/X_j$ the weighted sum of reputations of $Y$ as reported by the friends of $X$ ($X_j$) (Eq. (1))

$\sum_l \delta_l repY/Z_l$ the weighted sum of reputations of $Y$ as reported by strangers ($Z_l$) (Eq. (1))

$\alpha_i$, $\beta_j$, $\delta_l$ the weight factors which depend on the reputation of the individual neighbors, friends, and strangers, respectively (Eq. (1))

$A$, $B$, $C$, and $D$ the weight factors for the respective reputation of $Y$ with respect to neighbors of $X$, friends of $X$, and strangers in the host space. These factors are empirically determined constants which should satisfy the condition $A > B > C > D$ (Eq. (1))

initial value the reputation value at time $t = t_0$ (Eq. (2))

final value neutral value (Eq. (2))

$t_0$ last time host $Z$ computed the reputation of host $Y$ (Eq. (2))

$\tau$ the decay factor that determines how quickly reputation information becomes invalid – calculated on a per-host basis (Eq. (2))

$RI$ the result of the interaction as perceived by $X$. $RI$ is in the same range of a reputation value, determined according to different criteria (Eq. (3))

$\xi$ a parameter typically in the range 0.1–0.3, which gives more weight to the results of the interaction that just took place and less weight to previous reputation (Eq. (3))

$u_i$ the reputation value of host $i$ in initiator host reputation vector (Eq. (5))

$v_i$ the reputation value of host $i$ in target host reputation vector (Eq. (5))

$n$ the total number of hosts appearing in the reputation vectors of both the initiator and target hosts (Eq. (5))

$\sum Int\_from_X$ the sum of all the interactions done by host $X$ in the past time interval $T_I$, as reported by other hosts (Eq. (6))

$\sum Int\_from_{hosts}$ the sum of all the interactions done by all the hosts in $T_I$ (Eq. (6))

$\sum Int\_with_X$ the sum of all the interactions done with host $X$ in $T_I$ (Eq. (7))

$\sum Int\_with_{hosts}$ the sum of all the interactions done by all the hosts in $T_I$ (Eq. (7))

$Old$ the reputation values of host $Y$ before the current interaction (Eq. (8))

$New$ the reputation values of host $Y$ after the current interaction (Eq. (8))

$Att_Y$ the total number of interactions that $Y$ was asked for in last $T_I$ (Eq. (12))

Rubeira et al. [5] ascertained that trust information should be subjective and dynamic because past behaviors do not infer anything about future ones. The model incorporates probabilistic and fuzzy uncertainty. Song et al. [12] proposed a trust model based on fuzzy logic for securing grid resources by updating and propagating trust values across different sites. In addition, they developed a scheduler called SeGO to optimize computing power while assuring security under restricted budgets.

In other work, Song et al. [13] proposed a trust model that uses fuzzy logic inferences to handle uncertainties and incomplete information gathered from peers. The authors simulated their model, FuzzyTrust, over the public domain transaction data from eBay, and demonstrated that it is more effective than the EigenTrust algorithm. Ramchurn et al. [10] developed a trust model based on reputation and confidence for autonomous agents in open environments. Their model uses fuzzy sets to allow agents to evaluate previous interactions and create new contacts. In [3], Manchala described trust metrics, variables, and models and used fuzzy logic to verify transactions. He developed trust protocols based on cryptographic techniques to decrease breaches. In [15], Schlager et al. targeted trust in e-commerce federations using an attribute-based authentication and authorization infrastructure. They used Fuzzy Cognitive Maps and trust metrics to perform reputation management.

In [16] Schmidt et al. proposed a fuzzy based framework to decide on the selection of the optimal business partner. Their framework integrated social information, alliances, organization services, and products in e-commerce markets. In another work [17], Schmidt et al. proposed a fuzzy based customizable trust model integrating post-interaction processes such as interaction reviews and adjustment of credibility. TRUMMAR [4] was developed for mobile agents, and will be briefly described in Section 3.

In Table 1, we summarize the different features incorporated in the models discussed above. We also show how the model proposed in this paper, PATROL-F, is comprehensive and includes all the features listed in the table. Note that the model characteristics will be explained in detail later in the paper.

## 3. Patrol

In this section, we present a very brief overview of our non-fuzzy reputation-based trust model, PATROL, on which PATROL-F is based. For a full description of PATROL, please refer to [1].

### 3.1. Trust metrics

In PATROL, we differentiate between two types of trust between hosts: the first type of trust is in the competence of a host to perform a specific task up to the expectations of the initiator host. After every interaction, the initiator host evaluates the interaction results and saves the level of competency of the target host. This saved value is used as direct experience in the host's own trust decisions, and as a feedback reputation value given to other hosts to be incorporated in their trust decisions. The second type of trust is in the confidence in a host's consistency and credibility in giving trusted advice and feedback [6]. This type is based on the traits Activity, Similarity, Popularity, and Cooperation, which will be described shortly.

### 3.2. Trummar

PATROL is based on the TRUMMAR model [4]. TRUMMAR was implemented on local networks and the PlanetLab [7] wide-area network. Consider the situation where host $X$ wants to interact with host $Y$ in order to accomplish a certain task. Host $X$ will not interact unless it is confident that host $Y$ is trustworthy. In order to find out

**Table 1**
Summary comparison of previous work with the proposed model.

| Model characteristics | [11] | [14] | [2] | [8] | [5] | [12] | [13] | [10] | [3] | [15] | [16] | [17] | [4] | [1] | PATROL-F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hierarchy of trust | | | | | | | | | | | X | | X | X | X |
| Position of member in community | | | X | X | | | | | | | | | X | X | X |
| Prior-derived reputation | X | X | X | X | X | X | X | X | | X | X | X | X | X | X |
| Trust propagation | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| First impression | | | | | | | | | | | | | X | X | X |
| Result of interaction | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Fixed decay factor | | | | | | | | | | | | | X | X | X |
| Dynamic decay factor | | | | | | | | | | | | | | X | X |
| Trusting vs. Suspicious | | | | | | | | | | | | | X | X | X |
| Similarity | | X | | | X | | | | | | | | | X | X |
| Activity | | | | | | | X | | | | | | | X | X |
| Popularity | | | | | | | | | | | | | | X | X |
| Cooperation | | | X | X | | | | | | | | | | X | X |
| Number of interactions | | | | | | | | | | | | | | X | X |
| Fuzzy techniques | X | X | X | X | X | X | X | X | X | X | X | X | | | X |

whether host $Y$ is trustworthy or not, host $X$ calculates a reputation value for host $Y$, depending on a hierarchy of trust (Self, Neighbors, Friends, and then Strangers). The reputation value is calculated as follows:

$$repY/X_{before\_int} = ArepY/X + B\frac{\sum_i \alpha_i repY/X_i}{\sum_i \alpha_i} + C\frac{\sum_j \beta_j repY/X_j}{\sum_j \beta_j} + D\frac{\sum_l \delta_l repY/X_l}{\sum_l \delta_l} \quad (1)$$

Note that reputation values are restricted to values between 0 and $k$, where $k$ is the pre-defined maximum reputation value, such that $0 \le repY/X \le k$. To achieve this condition the constraint $A+B+C+D=1$ should be satisfied.

As time progresses, if little or no interaction occurs, the reputation values of a host with respect to other hosts changes to an unknown state according to the following:

$$repY/Z(t) = final\_value + (initial\_value - final\_value)e^{(t-t_0)/\tau} \quad (2)$$

Now trust can be determined by introducing two threshold values $\theta$ and $\phi$, referred to as the absolute trust and absolute mistrust thresholds, respectively:

- If $repY/X \ge \theta \Rightarrow Y$ can be trusted.
- If $repY/X \le \varphi \Rightarrow Y$ cannot be trusted.
- If $\varphi < repY/X < \theta \Rightarrow$ a fuzzy approach will be used to decide (refer to Section 4).

After the interaction, if any, $X$ recalculates the reputation of $Y$ using:

$$repY/X(0) = \xi \times repY/X_{before\_int} + (1-\xi) \times RI \quad (3)$$

Details on the notations used are found in Nomenclature section and [4].

### 3.3. First Impression

Consider as a special case, that host $Z$ is newly added to the system. The hosts in the system will subject $Z$ to an initial test period during which they will send it nonessential test data, with known results and expected times of completion. These hosts will keep on calculating $Z$'s reputation values and check its trustworthiness until the reputation information stabilizes and the first impression ($FI$) value is set. Note that every host will have its own specific period to test the trustworthiness of host $Z$ depending on how paranoid or trusting a host is. This way host $Z$ cannot guess the duration of this test period to show good behavior for its duration, and then "misbehave" after other hosts start to trust it.

If a certain host $Q$ does not want to wait for this test period to complete, it can set a random $FI$ value for host $Z$ depending on how trusting or paranoid host $Q$ is. This aspect is consistent with Falcone's statement [9] that interaction-based models must assume an initial default reputation that may result in unfair losses to the trusting host.

### 3.4. The weighting factors

In PATROL, every reputation value obtained from another host is multiplied by the corresponding weighting factor ($\alpha_i$, $\beta_j$ or $\delta_l$). The weighting factor of a certain host represents the trust in its capability to give dependable recommendation about other hosts and depends on its similarity ($Sim$), activity ($Act$), and popularity ($Pop$) values. The value of $\alpha_i$ is calculated as follows (similar equations are used for $\beta_j$ and $\delta_l$):

$$\alpha_i = a \times Sim + b \times Act + c \times Pop \quad (4)$$

The constants $a$, $b$, and $c$ are factors chosen subjectively by each host in order to give more importance to one parameter with respect to the others. Note that $a+b+c=1$, and they have to be consistently used in all the calculations of the weighting factors.

#### 3.4.1. The similarity value

The similarity value ($Sim$) determines the similarity of two hosts in their evaluation procedures and reputation values, and it is calculated as follows:

$$Sim = 1 - \sqrt{\frac{\sum (u_i - v_i)^2}{25n}} \quad \text{for all common hosts } i \quad (5)$$

Note that the factor 25 is used to normalize $Sim$, since the values of $u_i$ and $v_i$ can be between 0 and $k$ (we use $k=5$). $Sim$ assumes values between 0 (least similar) and 1 (most similar).

#### 3.4.2. The activity value

The activity value ($Act$) reflects the level of activity of a certain host in the past interval of time $T_I$. In order to keep hosts from giving false information, we will calculate the total number of interactions of a host from the other hosts' reputation vectors. The activity of a host $X$ is therefore:

$$Act(X) = \frac{\sum Int\_from_X}{\sum Int\_from_{hosts}} \quad (6)$$

Note that $Act$ assumes values between 0 and 1, with 1 being a limit value that is approached when a host is interacting much more frequently than other hosts.

**Table 2**
Reputation table.

| Host ID | Status | Reputation value | Weighting factors | Decay factor ($\tau_i$) | Interactions with (*Int_with*) | Interactions from (*Int_from*) | Attempts (*Att*) |
|---|---|---|---|---|---|---|---|
| 192.168.1.5 | Neighbor | 3.7 | [0.7 0.8 0.4] | 5230 | 43 | 65 | 72 |

#### 3.4.3. The popularity value

The popularity value (*Pop*) is a measure of how much a host is liked and how much its services are asked for in the system. Popularity of a host is calculated as follows:

$$Pop(X) = \frac{\sum Int\_with_X}{\sum Int\_with_{hosts}} \quad (7)$$

Note also that *Pop* can have values between 0 and 1 with 1 being a limit value.

### 3.5. Decay factor $\tau$

Each host keeps a decay factor for every other host it is interacting with. The decay factor $\tau_Y$ is a dynamic decay factor specific to a host *Y*. Initially, the decay factors are set to a nominal value and then each $\tau_Y$ will change (between pre-defined maximum and minimum values $\tau_{max}$ and $\tau_{min}$) based on the consistency of host *Y*'s reputation values. The decay factor in PATROL is calculated empirically as follows:

1. Calculate the difference $D = New - Old$ (8)
2. If $|D| \leq 1$ then $\tau_Y = \tau_Y \times round(5 - 4 \times |D|)$ (9)
3. If $D > 1$ then $\cdot\ \tau_Y = \dfrac{\tau_Y}{2 \times round|D|}$ (10)
4. If $-D > 1$ then $\cdot\ \tau_Y = \tau_Y \times round|D|$ (11)
5. Limit the new value of $\tau_Y$ to the range $[\tau_{min}, \tau_{max}]$

All the constants used depend on the values of $k$, $\tau_{min}$, and $\tau_{max}$ (5, 1000 cycles, and 10,000 cycles, respectively, in our implementation.)

### 3.6. Gathering/saving reputation values

Each host maintains a table similar to the one shown in Table 2. The table contains entries for other hosts with the following information: A unique host identifier, which may be an IP address, a URL, etc.; the status of the host being a neighbor, a friend, or a stranger; the calculated reputation value; the calculated weighting factors $\alpha_i$, $\beta_j$, $\delta_l$; the decay factor $\tau_i$; the number of interactions the host has done with every other host in the last time interval $T_l$ (this time interval is common to all hosts in the system); the number of interactions a host has done with this host in the last time interval $T_l$; the cumulative number of attempts the host tried to interact with any other host in the last time interval $T_l$.

The reputation vector is only a subset of the reputation table containing the identifier, the reputation values, and the number of interactions with and from a host during the last $T_l$. However, host *X* will not ask about the reputation vector more than once within a time interval $T_A$. In addition, host *X* will only ask the hosts that have their cooperation value above the cooperation threshold according to the flowchart in Fig. 1. If the number of attempts to host *M*, in the last time interval $T_l$, is below a certain threshold, the cooperation value will be considered inaccurate and host *M* will be queried. However, when the number of attempts exceeds the threshold, host *X* will have gathered enough information to calculate a valid cooperation value of host *M*.

Whenever asked, the hosts will decay all their reputation values based on their specific $\tau_i$ and send their reputation vectors to host *X*. Host *X* will now calculate weighting factors, reputation values, and decay factors $\tau_i$ of all queried hosts. *X* then checks the reputation value of *Y* and decides whether or not to interact with it.

#### 3.6.1. The $T_A$ time interval

Within the time interval $T_A$, *X* will not gather reputation vectors from all the hosts again because these vectors are already saved with minimal changes. $T_A$ is pre-defined at each host based on how up-to-date it needs its data to be. There is a trade-off between sending more data on the network and getting the most recent information.

#### 3.6.2. The cooperation value

The cooperation value (*Co*) reflects the willingness of a host to cooperate and give services to other hosts. Cooperation of *Y* with respect to *X* is calculated as follows:

$$Co(Y/X) = \frac{Int\_with_Y}{Att_Y} \quad (12)$$

*Co* can take values between 0 and 1, with 1 being most cooperative. *Co* will decrease the possibility of hosts ignoring inquiries about reputation vectors; thus saving the network bandwidth and the waiting time for an uncooperative host.

## 4. PATROL-F

In this section, we present PATROL-F (comPrehensive reputAtion-based TRust mOdeL with Fuzzy subsystems), as an improvement over PATROL incorporating fuzzy subsystems to incorporate subjective concepts and perform humanistic decisions. PATROL-F is a truly unique and comprehensive model that incorporates the most essential humanistic concepts to determine trust-based decisions.

### 4.1. Why the fuzzy approach?

Fuzzy logic is conceptually easy to understand, flexible and tolerant of imprecise data. In addition, fuzzy logic is based on natural language. In PATROL-F, there are three areas that benefit greatly from the introduction of fuzzy logic.

First, a fuzzy subsystem is included to set the importance factor and related decisions. To decide and choose which data is more critical or indispensable than other data, or which data is needed

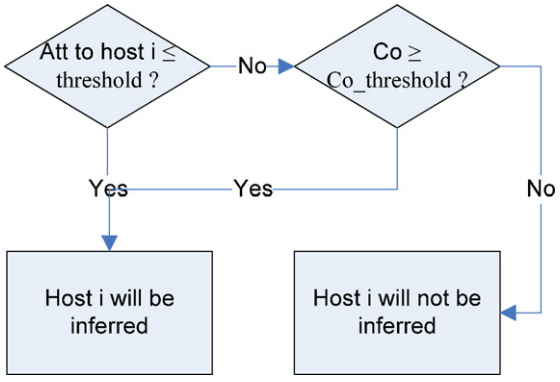


**Fig. 1.** Hosts inferred based on their cooperation.

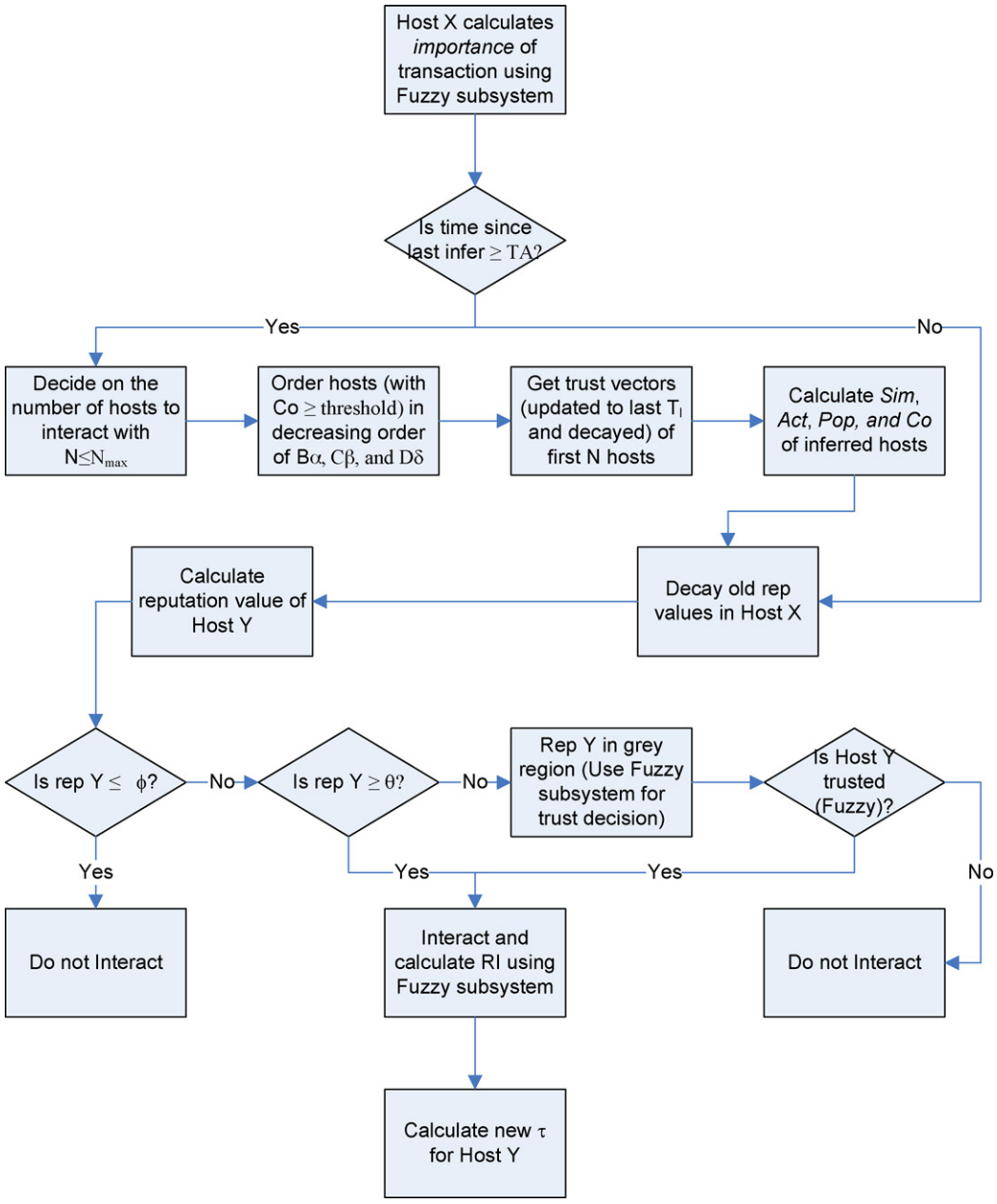


**Fig. 2.** Flowchart of PATROL-F.

more quickly, is a highly humanistic concept that fuzzy logic is able to model. Second, there is the grey region of uncertainty where the decision to trust is not straightforward. In this region, the reputation value is between the thresholds of absolute trust and absolute mistrust; thus, for trust decisions in this region, the fuzzy techniques will be effectively applied. Finally, for the *RI* value, fuzzy logic is used to capture the subjective and humanistic concept of a "good" or "better", and "bad" or "worse" interaction. *RI* becomes the result of several concepts combined together to produce a more realistic and representative value.

### 4.2. The full model flow

Fig. 2 illustrates the flow of events in PATROL-F. When host *X* wants to interact with host *Y*, *X* will first calculate the importance of the interaction with *Y* using the `Importance` subsystem. Then, *X* will calculate the time since its last inquiry of reputation vectors. If this time is found to be greater than the pre-defined time interval $T_A$, *X* will decide on the number of hosts to query about reputation values. This number is upper-bounded in order not to overflow the network by sending queries to a large number of hosts.

Afterwards, host *X* will query other hosts, whose cooperation value is greater than the cooperation threshold, about their reputation vectors. The queried hosts will decay their saved reputation values and send them along with their reputation vectors to host *X*, which will in turn, calculate the Similarity (*Sim*) values, the Activity (*Act*) values, the Popularity (*Pop*) values, and the Cooperation (*Co*) values of the queried hosts. Host *X* will then decay its old reputation values and incorporate all this information to calculate the reputation value of host *Y*. Note that if the time calculated at the beginning by host *X* does not exceed $T_A$, *X* will directly decay its old reputation values, skipping all the steps in between.

After the reputation of *Y* is calculated, a choice has to be made by host *X*: if the reputation of *Y* is less than the absolute mistrust level ($\phi$), *Y* will not be trusted and no interaction will take place; otherwise, if the reputation of *Y* is greater than the absolute trust

level ($\theta$), host *X* will trust and interact with host *Y*. However, if the reputation of host *Y* is between $\phi$ and$\theta$, host *X* uses the `FDecision` fuzzy subsystem to decide whether or not to trust host *Y*.

After the interaction, *X* calculates the result of interaction (*RI*), which is an indicator about how good or bad host *Y* performed in his interaction. This *RI* value is calculated based on the third fuzzy subsystem (the `RI` subsystem). Finally, host *X* will calculate the new decay factor ($\tau$) for host *Y* based on the difference in host *Y* result of interactions between successive interactions.

### 4.3. Importance

In PATROL-F, we set a maximum limit on the number of queried hosts. As this number increases, the waiting time to collect reputation information increases and the size of the exchanged inquiries and reputation information also increases. Thus, as this threshold increases, the total time of an interaction and the bandwidth consumption are affected negatively.

#### 4.3.1. The Importance concept

For every interaction, there is an Importance level that specifies how critical or essential the data or the interaction is. This concept determines other thresholds in the model as follows.

If the interaction is very critical, the initiator host will have to spend more time to make sure that the target host is trusted. Therefore, it will ask a relatively large number of hosts by inquiring about the reputation of that target, and it will be more paranoid, and will have smaller first impression (*FI*) values.

On the other hand, when an interaction is not critical yet it is needed quickly, the host will be more trusting, and thus will increase the *FI* values, and decrease the number of hosts to ask about the target's reputation.

This means that there is a security-time trade-off that the initiator host will have to make based on the importance of the interaction. When the host decides to ask only a portion of the hosts, the question of which hosts to ask comes up. The initiator host in this case compares the products $B\alpha$, $C\beta$, and $D\delta$ of its neighbors, its friends, and the strangers respectively. Then it will interact with the hosts with the highest products. This allows for a dynamic hierarchy of hosts, i.e. a friend host may be more trusted to give advice than a neighbor host if it has a higher alpha (or beta or gamma) value. This way, we will account for the case when a stranger is better than a friend or a neighbor, or when a friend is better than a neighbor, without losing the initial fixed hierarchy that gave higher multiplicative factors to neighbors, then friends, and finally strangers.

#### 4.3.2. The Importance logic

The Importance concept is affected by three main factors. The first factor is the monetary value of the transaction. This factor will affect the Importance value in the following manner: When the monetary value is high, the transaction is considered to be crucial and important and thus its results should not be altered. However, when the monetary value is low, the transaction is less important from the value point of view.

The second factor is the criticality of the results of the transaction. As the criticality level is raised, the results are more crucial and essential; thus less risk can be tolerated, and a higher Importance factor is given to the interaction. With low criticality, on the other hand, a lower Importance level is given to the transaction.

The third and final factor affecting the Importance level of a transaction is the time in which the results are needed. The total interaction time is the primary issue, and thus it will negatively affect the Importance level. If the results are needed very quickly, fewer hosts can be queried about the reputation value of the target host, and thus the transaction will get a lower Importance value. On the other hand, when the time is not critical and the results are not hurriedly needed, a higher Importance value may be given based on the other factors.

In the `Importance` fuzzy subsystem, the monetary value and the criticality of the result factors are divided into three intervals: low, medium, and high; whereas the time factor is divided into two: quickly and unhurriedly. The Importance level is also divided into three levels: low, medium, and high.

### 4.4. Grey region decision

The second fuzzy subsystem is needed for the decision of whether or not to trust in the grey region, where the reputation value is between the absolute trust level and the absolute mistrust level. To address this issue, we introduce a fuzzy subsystem to decide when to interact and when not to interact. In this case, the decision of interaction is a very subjective, since one host may decide to interact with a target host with a certain reputation value, while another may decide not to, based on several issues that are described below.

The fuzzy decision in the grey region is affected by three factors. The first factor is the personality of the source host. As the source host becomes more trusting, it will be more biased towards interacting with the target host even if it was a little bit suspicious. However, as the source host becomes more paranoid, it tends not to interact unless it is a guaranteed good interaction and the target host is ensured to have good intentions.

The second factor is the previously defined fuzzy concept of Importance. The Importance concept will act here as a factor affecting the fuzzy decision. As the interaction is more important, i.e. the data is more critical or the monetary value is high, the decision will be biased towards not interacting before guaranteeing the good results. However, as the importance of the interaction is decreased, i.e. the results are needed very quickly or the criticality and monetary value of the transaction are low, the decision will tend to be to interact without much guarantee.

The final factor affecting the decision is the reputation value of the target host. This factor is maybe the most important because as the reputation value approaches $\theta$, the target host is more likely to be a good host than a host with a reputation value approaching $\phi$. The decision to trust or not to trust is therefore biased by how close the reputation value is to $\theta$ or $\varphi$, respectively.

These three factors combine to evaluate the fuzzy decision whether to interact or not. In the `FDecision` subsystem, personality of the host is considered as either trusting or paranoid, while the Importance and the fuzzified reputation values are considered as high, medium, or low.

### 4.5. RI calculation

The final fuzzy subsystem in PATROL-F is introduced to set the value of the result of interaction (*RI*). This value is also subjective: when do we consider an interaction to be good or bad and to what degree is it satisfactory to the initiator host? However, the decision of one host will affect the whole system, as it will affect the reputation value of the target host. Thus, a fuzzy subsystem is introduced here to unify the subjective criteria upon which an interaction is considered good or bad, while keeping the relative reasoning of each host.

The *RI* value is dependent on three factors. The first factor is the correctness of the result. This factor is decided by the source host by evaluating how correct the results are compared to its expectations. When this correctness value is high, the result is considered to be good and the *RI* value will tend to be high. However, when an interaction is considered to have given a bad result, a low *RI* value is assigned.

**Table 3**
Constant values in the simulation.

| Parameter | $A$ | $B$ | $\theta$ | $\phi$ | $FI$ | $\zeta$ | $a$ | $b$ | $c$ | $\tau_0$ | $\tau_{min}$ | $\tau_{max}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value | 0.55 | 0.45 | 4 | 1 | 2.5 | 0.2 | 0.4 | 0.4 | 0.2 | 5000 | 1000 | 10,000 |

The second factor is the time of the interaction as compared to the expected time by the source host. If the interaction takes more time than expected, then this host is not that good at this job, i.e. a lower *RI* value will be given. However when the interaction time tends to be less or equal to the expected time, the *RI* value will tend to be higher.

The final factor affecting the *RI* value is the monetary value of the interaction. This value is the same factor that affected the Importance concept. Interactions that have a high monetary value are considered as valuable services, and thus have a higher *RI*. However, when the monetary value is low, the *RI* value is usually lower.

In the `RI` subsystem, the result can be good or bad, the interaction time can be fast, equal, or slow as compared to the expected time, and the monetary value, as previously defined, can be high, medium, or low.

## 5. Simulation verification

### 5.1. Simulation setup

In order to evaluate our trust model and prove its effectiveness and reliability in distributed systems, we have simulated a network of ten randomly interacting hosts using Microsoft Visual C++. Only one interaction can occur during any one simulation cycle. The reputation value can range from 0 to $k = 5$, with values below $\phi = 1$ considered as bad interactions and values above $\theta = 4$ considered as good interactions. For reputation values falling in the grey region between $\phi$ and $\theta$, the choice is based on the `FDecision` subsystem explained below. Note that in the no-fuzzy simulation, the choice will be based on a probabilistic approach where the decision to trust or not to trust is biased on how close the reputation value is to $\theta$ or $\phi$ respectively. In the simulation, hosts *X* and *Y* are set as bad hosts, thus giving results of interactions randomly distributed between 0 and $\phi$, while all other hosts are good hosts giving good results of interactions between $\theta$ and *k*. The constants of our model are shown in Table 3. In our simulation, all hosts are considered as neighbors under the same administrative domain; thus there is no need for weighting factors *C* and *D*. We use a neutral first impression value indicating no knowledge about the other hosts ($FI = 2.5$) and an initial decay factor $\tau_0 = 5000$ cycles for all hosts. Also, we set $a = b = 0.4$ and $c = 0.2$ giving more weight for similarity and activity at the expense of popularity.

The fuzzy subsystems `Importance`, `FDecision`, and `RI` were designed and implemented using the fuzzy toolbox in MATLAB, and incorporated in the C++ code. We compare the behavior of the reputation of a good host and that of a bad host, and the percentage of interactions with a good host and with a bad host. In addition, we compare the number of hosts queried for reputation values by a specific host.

### 5.2. No-fuzzy simulation

In order to assess the improvement introduced by the fuzzy subsystems, we simulate the network first without any fuzzy subsystem. We then introduce the fuzzy subsystems and compare the results to the "no-fuzzy" case.

The simulation results show that the reputation value of a good host *O* in host *M* directly increases and settles in the absolute trust region, with only one attempt considered as not trustworthy. This first attempt occurs in the grey region with a reputation value of 3.8, based upon which *M* decides probabilistically not to interact with *O*. Thus, *M* considers a good host *O* as trustworthy 34 times, and as not trustworthy only once; thus interacting with the good host *O* 97.14% of the total attempts.

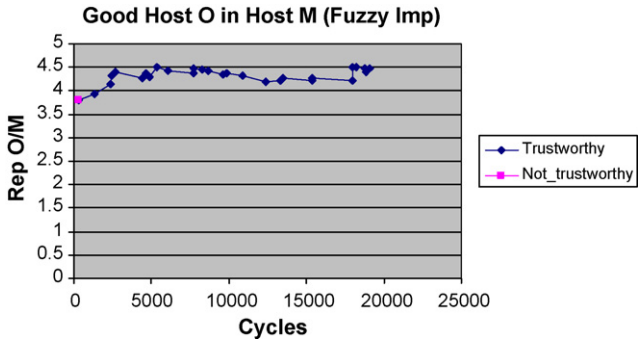


**Fig. 3.** Behavior of a good host's reputation.

The results show that the reputation value of a bad host *X* in host *M* decreases to the absolute mistrust region. In this simulation, *M* considers *X* as not trustworthy 35 times and does not interact with it. However, on 7 occasions, as the reputation value of host *X* decays to the grey region, *M* decides to give *X* a chance and to interact with it. As a result, *M* interacts 16.67% with a bad host *X*. In general, host *M* interacts 20% (18 times out of 72) with all of the bad hosts, which are *X* and *Y* in the simulated system.

In the simulation, host *M* had 364 attempts to interact with other hosts. Before each interaction it queries all the other 9 hosts about reputation values, thus sending 3276 inquiries about reputation values throughout the simulation.

### 5.3. The Importance effect

In this section, we show the results of adding the `Importance` fuzzy subsystem on the simulation results. In the code, the three inputs to the fuzzy subsystem, namely the Monetary Value, the Results (Criticality), and the Time needed are randomly generated. As shown in Figs. 3 and 4, we notice similar results to the "no-fuzzy" case. Even the percentages of host *M* interacting with good and bad hosts are similar. Here host *M* interacts 96.4% (27 times out of 28) with a good host *O*, 24% (10 out of 41) with a bad host *X*, and 19.5% (16 times out of 82) with bad hosts *X* and *Y*.

However, by introducing the Importance concept, we save in network bandwidth. In this simulation host *M* attempted 367 times to interact with other hosts, resulting in only 1431 inquiries about reputation values. Therefore, on average host *M* is asking around 4 hosts every time it attempts to interact with any other host. This indicates that with the introduction of the `Importance` fuzzy subsystem, we save almost 56% on the network bandwidth as compared to the "no-fuzzy" case.

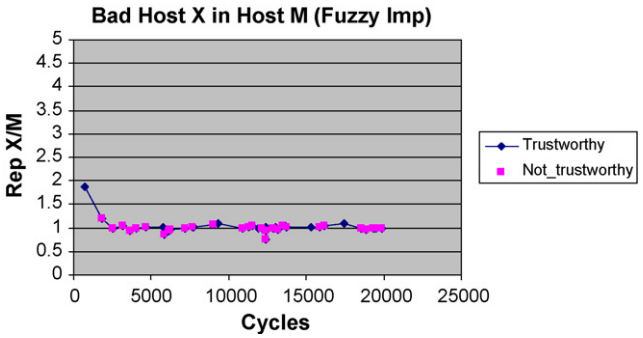


**Fig. 4.** Behavior of a bad host's reputation.

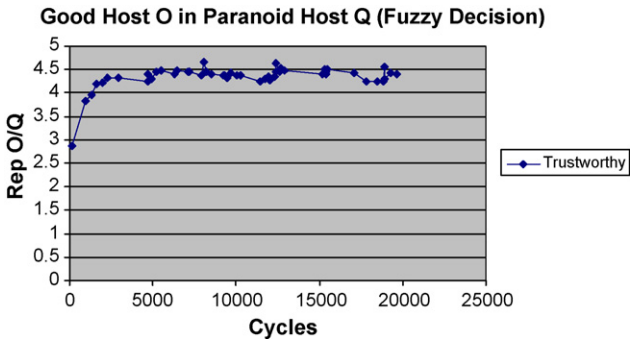


**Fig. 5.** Behavior of a good host's reputation in a paranoid host.

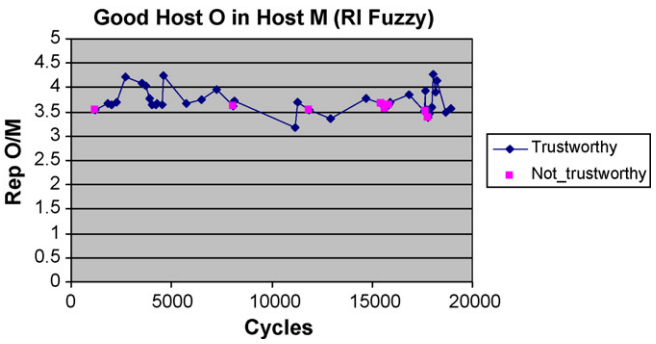


**Fig. 7.** Behavior of a good host's reputation.

### 5.4. The grey region decision effect

Now we repeat the previous simulation with only the `FDecision` subsystem incorporated. For this simulation, we have to categorize hosts as trusting or paranoid: every host is given a trusting level between 0 and 10, with 10 being most trusting. As for the other two inputs of the `FDecision` subsystem, the Importance value is generated randomly for every interaction and the reputation value is calculated using Eq. (1). We study the behavior of the reputation values of good and bad hosts at a trusting host *M* and at a paranoid host *Q*.

For a trusting host *M*, the reputation value of a good host *O* directly increases to settle in the absolute trust region with interaction on every attempt (33 out of 33 times). This is due to the fact that host *O* has always maintained a good reputation and due to the trusting nature of host *M*. For the bad host *X*, the reputation value at *M* decreases to the absolute mistrust region. However, due to the trusting nature of host *M*, *M* interacts 35.7% (15 out of 42) of the total attempts with host *X* and 27% (24 out of 87) of the attempts with bad hosts *X* and *Y*. Of course the total number of inquiries asked by host *M* throughout the simulation is similar to the "no-fuzzy" case, i.e. inquiring all 9 hosts before every one of its 367 attempts to interact with other hosts.

At the paranoid host *Q*, the reputation values of a good host *O* also directly increase to the absolute trust region, as shown in Fig. 5. Similar to *M*, *Q* interacts with *O* on every attempt (50 out of 50 times). This is due to the fact that host *O* has always maintained an excellent reputation that is considered trustworthy even for a paranoid host. Concerning the first attempt, host *O* has a reputation value of around 2.9 but the importance of this interaction happens to be very low. So, even if the reputation value can be considered as somehow on the edge of not trusting for a paranoid host, *Q* decided to give *O* a chance in an unimportant interaction to see how it will perform. O is good and maintains good reputation values keeping *Q* interacting with it all the time.

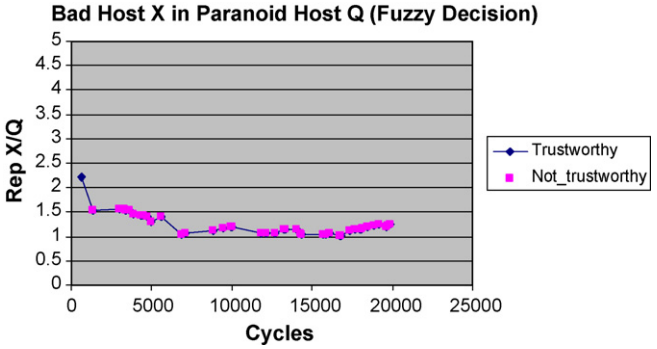


**Fig. 6.** Behavior of a bad host's reputation in a paranoid host.

Fig. 6 shows the reputation of a bad host *X* at the paranoid host *Q*. Host *Q* decides to give *X* a chance only for the first time where the reputation value is 2.2 but the importance of the transaction is very low. Afterwards, even when the reputation of *X* is in the grey region, *Q* is too afraid to interact with it. In this simulation, *Q* considers *X* as trustworthy only 2% of the time (1 out of 44) and considers all the bad hosts *X* and *Y* as trustworthy only 2.7% of the attempts (2 out of 71). This indicates that *Q* almost did not interact with the bad hosts in the system. The total number of inquiries asked by host *Q* throughout the simulation is similar to the "no-fuzzy" case, i.e. inquiring all 9 hosts before every one of its 385 attempts to interact with other hosts.

### 5.5. The RI calculation effect

In this section, we study the effect of the `RI` fuzzy subsystem alone. For the inputs of this fuzzy subsystem, the Result (Correctness) is calculated after every interaction, and the other two inputs, namely the Time (compared to Expected Time) and the Monetary Value are randomly generated for every interaction.

In Fig. 7, we notice a change in the reputation behavior of a good host *O*, for it increases but remains mostly in the grey region. We have to take into consideration in this case the new concepts that the `RI` subsystem introduces. The good host O is always giving good interactions; however this will account for the Result (Correctness) input of the fuzzy subsystem. Since the other two inputs are randomly generated, the *RI* values of *O* are not above 4 anymore. Host *O* is no longer considered a very good host, because it is deficient in the interaction times and in the importance of the interactions, or services, it is offering. Consequently, we see the reputation of host *O* with correct results, i.e. "good", staying in the grey region most of the simulation run, and consequently, host *M* interacts with it only 78% (32 out of 41) of the total attempts. As for the bad host *X*, the reputation value of *X* behaves as before, i.e. it decreases and settles in the absolute mistrust region. This common result shows that the `RI` subsystem gives a higher weight to the Correctness of the Result. It is enough, therefore, to produce an incorrect result to consider an interaction as being a bad one. To summarize, host *M* considers *X* as trustworthy 21.3% (10 out of 47) of the total attempts, and the bad hosts *X* and *Y* are considered as trustworthy 22.3% (21 out of 94) of the total attempts.

In this simulation, the total number of inquiries about reputation vectors is similar to the "no-fuzzy" case. Host *M* attempted to interact 382 times, and sent 3438 inquiries asking all other 9 hosts each time. The simulation results show that the reputation of the all-good host *O* remains from start to end in the absolute trust region, and *M* trusts *O* 100% (36 out of 36) of the attempts. Whereas for the all-bad host *X*, the reputation value at *M* decreases and settles in the absolute mistrust region; *M* trusts *X* 6.7% (3 out of 42) of the attempts.

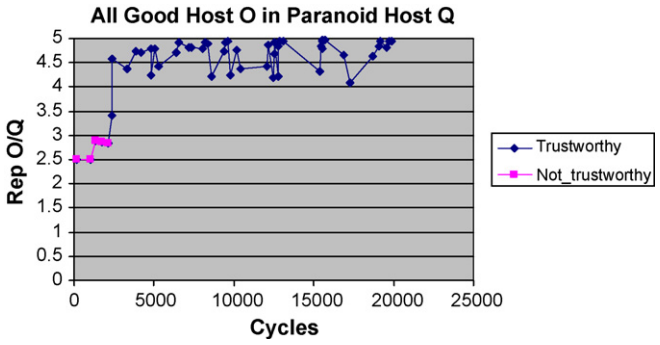


**Fig. 8.** Behavior of an all good host's reputation in a paranoid host.

### 5.6. A full comprehensive case

In this section, we show the results of simulation for the system with all three fuzzy subsystems incorporated. Each host has a certain degree as to how trusting it is. We randomly generate the Monetary Value, Results (Critical), and Time needed for the Importance subsystem. The inputs of the `FDecision` subsystem are available, i.e. Trusting/Paranoid is a host characteristic, the Importance is taken from the first subsystem, and the reputation is calculated by the model. As for the `RI` subsystem, the Result is calculated after every interaction and the Time (with respect to Expected Time) is randomly generated. The Monetary Value has the same value as that generated for the Importance of the interaction. In addition, we set the $T_A$ value to 1000 and the cooperation value *Co* to 2. In other words, we want the hosts in the system not to query more than once every 1000 cycles and not to ask hosts with a cooperation value under two. We also let host *O* be an all-good host and host *X* be an all-bad host.

For a trusting host *M*, we notice that the reputation of an all-good host *O* rises directly and settles in the absolute trust region. Host *M* interacts with host *O* 100% (28 out of 28) of the attempts. For the all-bad host *X*, its reputation decreases to the absolute mistrust region. Host *M* considers *X* as trustworthy only 17.5% (7 out of 40) of its attempts to interact with it. We notice here the very low number of inquiries sent by host *M* throughout the simulation. In 354 attempts to interact, host *M* sent only 91 inquiries, which is equivalent to around 0.25 hosts asked per interaction attempt.

For a paranoid host *Q*, we notice, from Fig. 8, that the reputation of an all-good host *O* increases after 5 interactions to the trust region. *Q* considers *O* as trustworthy 90% (46 out of 51) of the attempts. Concerning the reputation behavior of the all-bad host *X*, the reputation value decreases directly to the absolute mistrust region. Host *Q* considers *X* as trusted only 2% (1 out of 47) of the attempts to interact with it. Note that this one trusted time, the reputation of *X* is 2.5 and the importance of the interaction is low. Again, we notice that the number of inquiries for host *Q* is 77 from 392 attempts to interact with other hosts, i.e. almost 0.2 hosts are queried per interaction attempt.

## 6. System overhead and security issues

In this section we estimate the network overhead of PATROL-F by evaluating the size of the data that travels over the network to exchange the reputation information, and perform a stability test to check the effects of dishonest hosts on the system in order to estimate the limit at which PATROL-F fails.

**Table 4**
Effect of $T_A$ variations.

| $T_A$ | Num | Attempts | Hosts/attempt |
|---|---|---|---|
| 0 | 15,616 | 3953 | 3.95 |
| 25 | 10,508 | 3947 | 2.66 |
| 50 | 7896 | 3953 | 2 |
| 100 | 5204 | 3949 | 1.32 |
| 200 | 3154 | 3955 | 0.8 |
| 500 | 1445 | 3945 | 0.37 |
| 1000 | 782 | 3962 | 0.2 |
| 2000 | 389 | 3946 | 0.1 |
| 3000 | 272 | 3947 | 0.07 |
| 4000 | 210 | 3942 | 0.05 |
| 5000 | 153 | 3948 | 0.04 |

### 6.1. System overhead and data acquisition

In order to estimate the communication overhead of the system, we evaluate the number of inquiries asked by all the hosts in the system, to know the amount of the data that travels over the network to exchange the reputation information.

In PATROL, each host asks all the other hosts before every attempt to interact. Each host will therefore ask 9 hosts before every interaction. In PATROL-F, the `Importance` subsystem limits the number of hosts to be queried based on the importance of the interaction. In the simulation, the three inputs of this subsystem are randomly generated per interaction, producing uniformly distributed Importance values, and thus uniformly distributed number of hosts to be queried.

We repeat the simulation with different time intervals $T_A$ and evaluate the number of hosts queried before every attempt to interact. As shown in Table 4, with $T_A = 0$, each host asks almost 4 hosts (45% of the other hosts) before every attempt to interact. As $T_A$ increases, the number of hosts per attempt decreases until each host is basically working alone without any queries about reputation vectors. We notice that this decrease is exponential as shown in Fig. 9.

As explained in Section 3, the reputation vector contains the host identifier (IP address), the reputation values, and the number of interactions with and from a host during the last $T_I$ interval, summing up to nine bytes. However, with the overhead of packet headers, a total vector size of around 64 bytes is needed. Considering that we are in a system where the interactions are of random importance, that is each host will inquire about 45% of the other hosts on average, and that every host wants to query about all the other reputation vectors every $T_A$ seconds, which is a worst case scenario, the minimum allowed $T_A$ for an overhead bandwidth utilization of 100 kbps and 1 Mbps are as shown in Table 5. Table 6 shows the minimum allowed $T_A$ for an overhead bandwidth utilization of 100 kbps and 1 Mbps without the fuzzy importance.

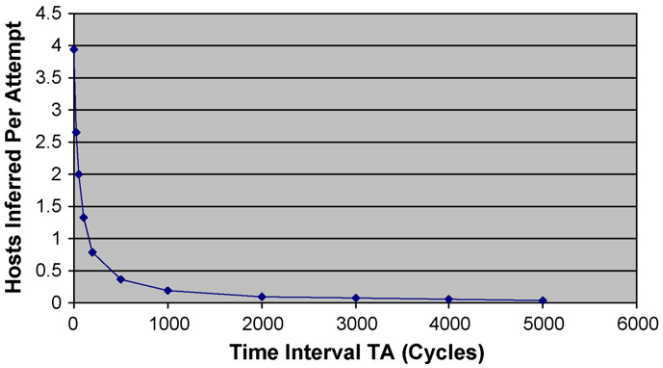


**Fig. 9.** Number of hosts queried per attempt.

**Table 5**
Allowed $T_A$ for a given overhead.

| Number of hosts | $T_A$ for an overhead of 100 kbps | $T_A$ for an overhead of 1 Mbps |
|---|---|---|
| 10 | 0.21 s | 0.02 s |
| 20 | 0.88 s | 0.1 s |
| 50 | 5.6 s | 0.56 s |
| 100 | 22.8 s | 2.28 s |

**Table 6**
Allowed $T_A$ for a given overhead (without Importance).

| Number of hosts | $T_A$ for an overhead of 100 kbps | $T_A$ for an overhead of 1 Mbps |
|---|---|---|
| 10 | 0.46 s | 0.05 s |
| 20 | 1.9 s | 0.2 s |
| 50 | 12.5 s | 1.3 s |
| 100 | 50.7 s | 5.1 s |

Comparing Tables 5 and 6, we notice that $T_A$ is decreased to less than half, and thus the system is able to keep more up-to-date reputation information without affecting the network bandwidth. These gains change when the average importance of interactions in the network changes: if the importance of the interactions is always very high, the $T_A$ values will remain as in Table 6. However, with very unimportant interactions, we will get even lower $T_A$ values than those of Table 5.

### 6.2. Stability test

In this section, we perform a stability test to check the limit on the number of dishonest hosts that PATROL-F can tolerate and keep producing correct results. We repeat the simulation of ten hosts, where *Y* is an all-bad host. The reputation of host *Y* with respect to host *M* is calculated as the number of dishonest hosts increases. A dishonest host is one that gives a perfect reputation value for (the all-bad) host *Y*.

As shown in Fig. 10, *repY/M* remains in the absolute mistrust region and host *M* remains aware of the bad nature of host *Y*, up until we have four dishonest hosts in the system. However, as the number of dishonest hosts exceeds four, host *M* will have the wrong impression and *repY/M* exceeds the nominal value of 2.5 and reaches the absolute trust region when we have 7 or 8 dishonest hosts.

From Table 7, we notice that when the number of dishonest hosts is greater than or equal to five, host *M* interacts with *Y* more than 95% of the attempts. However with four or less dishonest host, we notice that *M* interacts with *Y* less than 23% of the attempts, most of which are of low importance. We also notice here that with the addition of the fuzzy subsystems, an improvement is achieved over

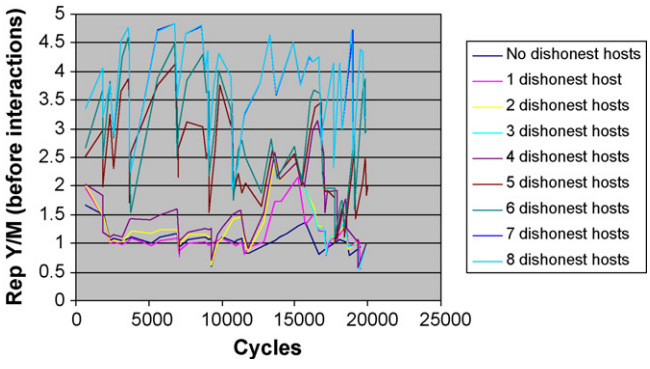


**Fig. 10.** System failure.

**Table 7**
Percentage of interactions from *M* to *Y*.

| Dishonest hosts | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Attempts | 39 | 35 | 37 | 37 | 37 | 2 | 0 | 0 | 0 |
| Interactions | 6 | 10 | 8 | 11 | 11 | 45 | 48 | 48 | 48 |
| Percent interactions | 13.3 | 22.2 | 17.78 | 22.9 | 22.9 | 95.75 | 100 | 100 | 100 |

**Table 8**
Percentage of interactions from *M* to *Y*.

| Dishonest hosts | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Attempts | 37 | 34 | 29 | 30 | 26 | 18 | 14 | 11 | 10 |
| Interactions | 11 | 15 | 17 | 18 | 22 | 30 | 35 | 38 | 38 |
| Percent interactions | 22.92 | 30.61 | 36.96 | 37.5 | 45.83 | 62.5 | 71.43 | 77.55 | 79.17 |

the results without the fuzzy subsystems. In Table 8, which shows the percentages without fuzzy subsystems, *M* interacted with *Y* almost 23% of its attempts even when there were no dishonest hosts, whereas in Table 7, *M*'s interactions with *Y* remain below 23% up until there are 4 dishonest hosts in the system.

Consequently, the system can tolerate half of the other hosts being dishonest, and is able to keep the percentage of interactions with a bad host at less than 23% of the total attempts. When more than half of the other hosts are dishonest, the system fails.

## 7. Conclusions

We presented in this paper PATROL-F, a new comprehensive model for reputation-based trust incorporating fuzzy subsystems that can be used in any distributed system. The model is unique in integrating various concepts important to the calculation of reputation values and the corresponding decisions of whether to trust or not. In addition, PATROL-F incorporates fuzzy subsystems to account for subjective concepts, namely, the importance of a transaction, the decision subsystem that decides whether to interact or not, and the result of an interaction value.

PATROL-F has been simulated using the C++ programming language with the use of the Fuzzy Toolbox in MATLAB. The results prove the correctness of the model and its ability to cope with dynamic system conditions. The different aspects of the model were verified and the roles of the different parameters were evaluated.

## References

[1] A. Tajeddine, A. Kayssi, A. Chehab, H. Artail, PATROL: a comprehensive reputation-based trust model, International Journal of Internet Technology and Secured Transactions 1 (2007) 108–131.
[2] C. Castelfranchi, R. Falcone, G. Pezzulo, Trust in information sources as a source for trust: a fuzzy approach, in: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, 2003, pp. 89–96.
[3] D. Manchala, Trust metrics, models and protocols for electronic commerce transactions, in: Proceedings of the 18th International Conference on Distributed Computing Systems, 1998, pp. 312–321.
[4] G. Derbas, A. Kayssi, H. Artail, A. Chehab, TRUMMAR—a trust model for mobile agent systems based on reputation, in: IEEE/ACS International Conference on Pervasive Services, 2004. ICPS 2004, 2004, pp. 113–120.
[5] J. Rubiera, J. Lopez, J. Muro, A fuzzy model of reputation in multi-agent systems, in: Proceedings of the Fifth International Conference on Autonomous Agents, 2001, pp. 25–26.
[6] P. Dewan, Peer-to-peer reputations, in: 18th International Parallel and Distributed Processing Symposium, Santa Fe, New Mexico, April 26–30, 2004, pp. 128–130.
[7] PlanetLab Consortium, http://www.planet-lab.org, accessed September 20, 2009.
[8] R. Falcone, G. Pezzulo, C. Castelfranchi, Quantifying belief credibility for trust-based decision, in: Proceedings of the Workshop on Deception, Fraud, and Trust in Agent Societies at AAMAS-02, Bologna, Italy, 2002, pp. 41–48.
[9] R. Falcone, K. Barber, L. Korba, M. Singh, Challenges for trust, fraud, and deception research in multi-agent systems, Trust, Reputation, and Security: Theories and Practice. Lecture Notes in Artificial Intelligence (2003) 8–14.
[10] S.D. Ramchurn, C. Sierra, L. Godo, N.R. Jennings, A computational trust model for multi-agent interactions based on confidence and reputation, in: Proceedings

of 6th International Workshop of Deception, Fraud and Trust in Agent Societies, 2003, pp. 69–75.
[11] S.D. Ramchurn, N.R. Jennings, C. Sierra, L. Godo, Devising a trust model for multi-agent interactions using confidence and reputation, Applied Artificial Intelligence 18 (2004) 833–852.
[12] S. Song, K. Hwang, M. Macwan, Fuzzy trust integration for security enforcement in grid computing, Lecture Notes in Computer Science (2004) 9–21.
[13] S. Song, K. Hwang, R. Zhou, Y.K. Kwok, Trusted P2P transactions with fuzzy reputation aggregation, IEEE Internet Computing 9 (2005) 24–34.
[14] Z. Shuqin, L. Dongxin, Y. Yongtian, A fuzzy set based trust and reputation model in P2P networks, Lecture Notes in Computer Science (2004) 211–217.
[15] C. Schlager, G. Pernul, Trust modelling in E-commerce through fuzzy cognitive maps, in: 2008 3rd International Conference on Availability, Reliability and Security (ARES'08), 2008, pp. 344–351.
[16] S. Schmidt, E. Chang, T. Dillon, R. Steele, Fuzzy decision support for service selection in E-Business environments, in: IEEE Symposium on Computational Intelligence in Multicriteria Decision Making, 2007, pp. 374–381.
[17] S. Schmidt, R. Steele, T.S. Dillon, E. Chang, Fuzzy trust evaluation and credibility development in multi-agent systems, Applied Soft Computing Journal 7 (2007) 492–505.