Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 8, No. 4, November 2017), Pages: 37-57
www.jacr.iausari.ac.ir

DisTriB: Distributed Trust Management Model Based on Gossip Learning and Bayesian Networks in Collaborative Computing Systems

Abdulbaghi Ghaderzadeh✉[1], Mehdi Kargahi[2], Midia Reshadi[1]
1) Department of Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
2) School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran
b.ghaderzadeh@iausdj.ac.ir; kargahii@ut.ac.ir; ce.srbiau@gmail.com

Received: 2016/10/03; Accepted: 2016/11/06

Abstract

The interactions among peers in the Peer-to-Peer systems as a distributed collaborative systems are based on asynchronous and unreliable communications. Trust is an essential and facilitating component in these interactions specially in the uncertain environments. Various attacks are possible due to large-scale nature and openness of these systems that affects the trust. Peers has not enough information about other peers thus they can use trust management to reason about future interactions with other peers. In fact any peer needs to know the exact prediction of the trustworthiness of other peers. In the proposed approach the Bayesian network based trust management model is used to infer the trust value of task processor peers based on various aspects of their behaviors. The previous behavior of peers is considered to determine meticulous trust value of these peers in completely distributed environment. Since the trust is multifaceted concept, each aspect is evaluated using a single Bayesian network to provide finer-grained inference of trust. In the proposed model, DisTriB (Distributed Trust Management Model Based on Gossip Learning and Bayesian Networks), many aspects such as network link capacity and workload of task processor peers is used to construct the Bayesian network. The optimum time window size is found to obtain better performance too. Finally, a robust, asynchronous, gossip based protocol is proposed that can withstand high churn and failure rates, and can spread the trustworthiness of peers while the processing of tasks in the proposed collaborative computing system is performed. Simulation results shows that the proposed approach outperforms previous works.

Keywords: *Trust, Gossip Learning, Bayesian Network, Time Window, Distributed P2P Computing Networks*

1. Introduction

The P2P (Peer-to-Peer) computing systems (called public-resource computing) [1] as the collaborative computing system have been more attractive in recent years due to many spectacular features like anonymity, dynamism, scalability and sharing the cost.

P2P architecture [2] provides open and unlimited environment to share resources (CPU cycles, memory etc.) and it has the ability to integrate these resources. The peers in P2P architecture could be used as building blocks to build a system with tremendous resources. P2P system is used for distributed computing to achieve great processing

power in our research. Applications such as distributed.net[3] and SETI@home[3, 4] use the idle CPU cycles of thousands of computers connected to the Internet in order to break encryption codes and find signs of intelligent life in outer space. However the special features of these systems such as heterogeneity, autonomy and openness make it the ideal environment for attackers and cause the enormous security problems in distributed computing systems. Unstructured P2P computing systems do not have any central control and credit point and they have more security threats compared to centralized systems due to their nature. Peer validation in such distributed system without any central server requires more complicated mechanisms.

The key to reach correct functionality of a P2P computing system is correct collaboration among peers. In a P2P computing system, the determined level of honesty in processor peers is expected. However increasing the number of peers in large scale systems cause the increment of malicious peers that they are responsible for many security attacks and threats. Due to decentralized nature of these systems, protecting peers from these threats could not be possible therefore it is required that peers should be protect themselves, be responsible of their behaviors and exchange the information about malicious behaviors to each other.

The most of security issues are related with trust in a distributed computing system. The task distributor peers send the tasks to the task processor peers and they should select high trusted task processor peers to assign the tasks. Trust evaluation is based on many parameters like reliability, availability, credibility and honesty of a peer. A task processor peer should satisfy one or more of these parameters. Our studies shows that trust analysis using a low number of parameters because the performance degradation of system so the more parameters to evaluate trust should be used compared to previous works. In this paper a Bayesian network based trust management model is used to infer the trust value of task processor peers based on various aspects of their behavior. The previous behavior of task processor peers is considered to determine meticulous trust value of these peers in completely distributed environment. To do this each aspect is evaluated by a single Bayesian network and compare to previous works more aspects is used such as network link capacity and workload of task processor peers. The main advantage of using Bayesian network is the easier extension of the model to involve more dimensions of trust and easier combination of Bayesian networks to form an opinion about the overall trustworthiness of a task processor peer. Each task distributor peer can evaluate their task processor peers according to some own criteria. By updating Bayesian networks the behaviors of a task processor peer is captured. Finding of optimum time window size is another important aim to obtain better performance. A robust, asynchronous, gossip based protocol is proposed that can withstand high churn and failure rates, and can spread the trustworthiness of peers with the processing of tasks simultaneously. Gossiping is used to consider the subjectivity of trust. Finally the details of implementing of our algorithm is described, and the experimental evaluations is discussed. Simulation results show that our approach outperforms similar works.

Our specific contributions include the following: (1) A pure distributed computing P2P network in peerSim is simulated. (2) A novel, efficient, distributed learning method is introduced to model the behaviors of task processor peers based on their previous activities and the malicious peers are detected efficiently. (3) A Bayesian network based model is constructed based on different aspects to evaluate trustworthiness of task processor peers and a robust, asynchronous, gossip based protocol is proposed that can withstand high churn and failure rates, and can spread the worthiness of peers in the

distributed P2P computing system. (4) Finally, the details of implementing the algorithm is described, and the experimental evaluations is discussed.

The outline of the paper is as follows. In section II we summarize related works. In Section III we describe our approach and the core algorithmic contributions of the paper, while Section IV contains an experimental analysis. Section V concludes the paper.

2. Related Works

In the recent years, P2P systems are widely used for developing large scale distributed computing applications. The distributed.net [5], SETI@home [3, 4] and Genome@home [6] are the well known examples of these systems. SETI@home is the largest effort in distributed processing in terms of participants to search for Extra-terrestrial Intelligence. SETI@home aggregate computing power of a large number of computers linked to the Internet in order to process the horrific quantity of radio signal data gathered at the Arecibo Observatory in Puerto Rico. The competitive feature of SETI@home is when users try to earn credits by produce results faster than others. Users collaborate with each other in teams to produce results faster. However some users try to send results which are not yet completely processed by modifying the client code. Here the trust management system should deploy effectively and the user behaviors has to police by SETI@home servers. BOINC, Berkeley Open Infrastructure for Network Computing [7], is a similar projects on large scale distributed computing that is a free programming tool to develop other large scale distributed processing applications. Another example for implementing such large scale distributed processing is GreenTea [8] that is a purely Java-based P2P platform. Participating users in GreenTea can share their resources such as CPU cycles, storage spaces, network link capacity and services. JNGI is another decentralized P2P computing framework for large-scale computation presented by Jerome Verbeke et al [9]. They divide the computational resources into groups according to their functionality in their framework. The groups are: the monitor group, the worker group, and the task dispatcher group. CCOF, cluster computing on the fly, presented by Virginia Lo, et al. [10] harvest the CPU cycles from desktop PCs by a wave scheduler when the large blocks of ideal time at night are available. By using a geographic based overlay network to organize nodes by time zone the system provide better quality of service. To reduce the number of redundancy and complete more tasks in distributed P2P processing, Sing Jin Choi, et al. [11] present a group based dynamic computational replication mechanism that adaptively selects the volunteers executing the replicated tasks on the basis of some volunteer properties. More discriminant grouping in P2P computing environment is presented by Dubey J. & Tokekar V. [12] by determining eight different groups. They use different parameters like peer availability, peer credibility, and peer computation time. Another study proposed by them [13] is to process of real time application's tasks, identifies the group of reliable peers from the available peers. They divide the peers into four peer groups: administrator group, query manager group, task distributor group and task processor group. This categorization is used in our paper.

The researchers showed a lot of interest on trust and reputation in P2P networks and many trust management algorithms have been developed for P2P computing systems in recent years. A nice survey on different trust management problems in different kind of P2P networks presented by Xue Chen et al. [14]. In this paper the peers are classified

according to their contribution in the network in three categories: good nodes, malicious nodes, and average nodes. The good nodes try to offer service to other peers and share the authentic files. Malicious nodes are attacker and the average nodes are malicious nodes. Authors introduce the general trust function for trust evaluation.

Stefan Kraxberger et al. [15] proposed the security concepts for P2P computing and divide them in to two parts: routing security concept and group security concept. Jung-Tae Kim et.al. [16] give suggestions for enhancements on the security mechanism in P2P systems by identifying threats and various protection methods. Ajay Ravichandran et al. [17] present Eigen group trust based on Eigen Trust for trust management that any peer obtain reputation values from all peers in the network and calculate global trust value. They built their algorithm on top of a peer group infrastructure and manage trust within groups and between different groups witch construct a hybrid solution relates reputation management and peer communities. Chen Ding et.al. [18] make a survey on trust management in P2P systems and propose a computational trust management model that calculates reputation scores and shows the relationships based on trust and reputation. They studied DMRep and EigenRep reputation based trust management protocols too. Tu Chao et al. [19] proposed the random extraction algorithm, an improved trust model, to consider a value of reputation of service in their trust model to measure the quality of service and they add credibility of recommendation to measure the accuracy of the information of recommendation. Yao Wang and Julita Vassileva in [20] and Jigyasu Dubey and Vrinda Tokekar in [21] propose a trust model based on Bayesian network in P2P file sharing applications and P2P computing systems respectively. The farmer model describes a trust and reputation mechanism for agents to identify most appropriate companions from their viewpoints and propagate this information to other agents while the latter models try to measure the trustworthiness of task processor peer in unstructured P2P computing framework. The main disadvantage of Yao and Vassileva work is that they only used the aspects in Bayesian network related to P2P file sharing networks. The main disadvantage of Dubey and Tokekar work is that they did not investigate different time window size to find optimum time window size and they use limited aspects in their Bayesian network trust model. Time window is a parameter to limit the number of transactions considered in the history of peer's behaviors. Some more aspects need to be added in both Bayesian networks based models to improve performance metrics. In the proposed method more aspects is used in Bayesian network trust model related to all type of P2P networks and different time window size is checked to obtain the optimum time window size and gain better performance. Our simulations show that our trust model gain better performance and propagate the trust and reputation scores better than other methods.

3. Proposed Method

3.1 System Model

The following assumptions are made about proposed model:

3.1.1 Network Topology

A pure P2P network with a set of peers $\{P_1, P_2, \ldots, P_n\}$ and average degree γ is considered. This network is modeled as an undirected graph G = (V,E) where V is the set of nodes representing peers and E is the set of edges between neighboring peers in

the overlay network. Our protocol is built on top of this P2P overlay network. It is assumed that peers in the network are able to communicate with their direct neighbors only. Each peer also maintains a limited amount of information about its direct neighbors. Peers in P2P network are grouped. As in [22] a group is defined as a computing unit that consists of a number of individual peers interacting with each other with respect to some measures like: 1) Some level of distribution of work (i.e. responsibility), 2) Common purposes and objectives, 3) Common set of service and a security level, 4) Recognized position such as role, authority, etc., 5) Established standards and principles with reference to issues related to the group; 6) Development of accepted credits such as incentive and penalty.

'G' is assumed as a set of all the peers participating in a P2P computing network. The P2P computing system has 'n' different peer groups $g_1,\ g_2,\ldots,g_n$. The peer group 'g_i' is a subset of 'G' and all the peers in 'g_i' are governed under the set of rules 'Ri' that describe the conditions required to belong to a group and formed based on a particular interest criterion. So we have:

$$g_i = \{x \mid x \in G \text{ and } x \to R_i\} \tag{1}$$

$$R_i = \{r \mid r \text{ is a rule or criterion to form } g_i\} \tag{2}$$

$$G = g_1 \cup g_2 \cup \ldots \cup g_n \text{ and } g_i \neq \Phi \tag{3}$$

Four peer groups are considered as following: administrator group, query manager group, task distributor group and task processor group.[23]

3.1.2 Trust Formation

To be precise the following definition of trust in P2P networks is used. Peers in P2P networks, function as autonomous nodes. Each peer in P2P network could communicate, share, send and download data to/from any other peer that could be located anywhere in the geographical space.

A capability set C_s indicates the capability of the P2P network. $C_s = \{C_1, C_2, \ldots, C_m\}$ where C_i is a capability. The capability set C_{Pi} of a peer P_i is then $C_{Pi} \subset C_s$. A minimum capability set of P2P network is also defined as $C_{s(\min)}$ = minimum capability set which must be satisfied by a peer to be in P2P network. Thus for P_i and P_j to be peers $C_{Pi} \cap C_{Pj} \geq C_{s(\min)}$. General trust represents the trust that a peer has in another peer. A continuous range trust scale of $[-1,+1)$ is used, that -1 means a complete distrust, and +1 means complete trust. We define trust relations through a directed cyclic graph (DCG). As shown in Fig. 1. a peer in P2P network illustrated as a node of DCG and a directed edge between two peers P_i and P_j represents the presence of one-sided trust. We could see that the DCG is subjective and non-symmetric.

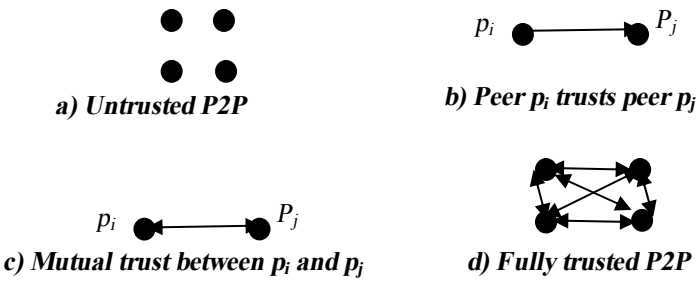


Figure 1. Representation of Trust. (a) is untrusted P2P, (b) represents one sided trust where p_i trust p_j but p_j may not trust p_i, (c) represents full trust between p_i and p_j, and (d) represents a fully trusted P2P network where a cycle exist between any pair of nodes.

To represent the distribution of trust values according to transaction history the beta probability density functions is used as in [24]. The beta distribution is a continuous distribution function indexed by the two parameters α and β. It can be expressed the beta PDF using the gamma function as follows:

$$beta\left(p|\alpha,\beta\right)=\frac{\Gamma\left(\alpha+\beta\right)}{\Gamma\left(\alpha\right)\Gamma\left(\beta\right)}p^{(\alpha-1)}\left(1-p\right)^{(\beta-1)},where\ 0\le p\le,\alpha,\beta>0. \tag{4}$$

To calculate the probability expectation value of the beta distribution we have:

$$E\left(p\right)=\frac{\alpha}{\alpha+\beta} \tag{5}$$

We can represent the posterior probabilities of binary events as beta distributions but we need to extend it to multidimensional outcomes. In our trust formation we use five tuples $\left(r_C,TP_{\bar{C}}\right),\left(r_W,TP_{\bar{W}}\right),\left(r_B,TP_{\bar{B}}\right),\left(r_{CT},TP_{\overline{CT}}\right),\left(r_D,Tp_{\bar{D}}\right)$ for five dimensions of trust of task processor peers. C means the peer credibility is considered and $\bar{C}$ means not considering peer credibility. W means the peer workload is considered and $\bar{W}$ means not considering peer workload. B means the peer bandwidth is considered and $\bar{B}$ means not considering peer bandwidth. CT means that peer computation time is considered and $\overline{CT}$ means not considering peer computation time and finally D means the peer distance is considered and $\bar{D}$ means not considering peer distance. $r_i\left(i\in\left\{C,W,B,CT,D\right\}\right)$ is the number of transactions with outcome i and TPC denotes task processor peer's overall capability as our outcome. The parameters of beta probability function are set as follows:

$$\alpha_i=r_i+1,\beta_{\bar{i}}=TPC_{\bar{i}}+1,\ (i\in\left\{C,W,B,CT,D\right\},\left(\bar{i}\in\left\{\bar{C},\bar{W},\bar{B},\overline{CT},\bar{D}\right\}\right) \tag{6}$$

$$\tau_i=\frac{\alpha_i}{\alpha_i+\beta_{\bar{i}}},\ (i\in\left\{C,W,B,CT,D\right\},\left(\bar{i}\in\left\{\bar{C},\bar{W},\bar{B},\overline{CT},\bar{D}\right\}\right) \tag{7}$$

$$\text{Since }\beta_{\bar{C}}=\alpha_W+\alpha_B+\alpha_{CT}+\alpha_D-1, \tag{8}$$

$$\text{Then}\,\tau_i = \frac{\alpha_i}{\sum_{j\in\{C,W,B,CT,D\}}\alpha_j - 1} = \frac{r_i + 1}{\sum_{j\in\{C,W,B,CT,D\}} r_j + 2}\left(i \in \{C,W,B,CT,D\}\right). \quad (9)$$

To get the trust values these are normalized:

$$P_i = \frac{\tau_i}{\sum_{j\in\{C,W,B,CT,D\}}\tau_j}\left(i \in \{C,W,B,CT,D\}\right). \quad (10)$$

Where the P_i is the probability of outcome i happening in the future. The task distributor peer's confidence factor $\gamma_i\left(i \in \{C,W,B,CT,D\}\right)$ measures the probability that in the calculated trust value P_i the actual trust value lies within an acceptable level of error ϵ and calculated as follows:

$$\gamma_i = \frac{\int_{P_i-\epsilon}^{P_i+\epsilon} K^{\alpha_i-1}\left(1-K\right)^{\beta_i-1} d_K}{\int_0^1 \rho^{\alpha_i-1}\left(1-\rho\right)^{\beta_i-1} d_\rho}\left(i \in \{C,W,B,CT,D\}\right) \quad (11)$$

Then the task distributor peer can set a threshold θ_γ to determine that if he has enough confidence. If the following equation fulfilled by the trust value of the task distributor peer about a task processor peer, then he feels confident about the prediction about the behaviour of that task processor peer.

$$\gamma_C > \theta_\gamma, \gamma_W > \theta_\gamma, \gamma_B > \theta_\gamma, \gamma_{CT} > \theta_\gamma, \gamma_D > \theta_\gamma \quad (12)$$

These tuples can be combined together since these five tuples have some relationships (Equation(11)) to save storage space. The format of confidence factors and trust values for task processor peers are $\left(P_C, P_W, P_B, P_{CT}, P_D\right)$ and $\left(\gamma_C, \gamma_W, \gamma_B, \gamma_{CT}, \gamma_D\right)$ respectively. Fading factor $\lambda_\gamma \in [0,1]$ can be used to forget old transactions since task processor peers may change their behaviour over time:

$$r_{t+1} = \lambda_\gamma r_t + r \qquad TPC_{t+1} = \lambda_\gamma TPC_t + TPC. \quad (13)$$

Where r_{t+1} and TPC_{t+1} are the trust parameters for $t+1$ transactions and the task processor peer's capability of the $t+1$ the transaction is $\left(r, TPC\right)$.

3.1.3 Gossiping Recommendations

If a task distributer peer is not confident enough in a trustworthiness of a task processor peer according to Equation (13) or wants to interact with an unknown task processor peer, he could get other task distributer peer's recommendations. This process is embedded in the heart of our protocol by gossiping. In fact task distributer peers exchange their knowledge about their corresponding task processor peers during task assignment to have the ability to find more appropriate task processor peers and to earn enough information about them. The recommendations are trust values given to task processor peers by other task distributer peers. A Bayesian network trust model is built to perform the estimation for each aspect of trust (credibility or workload or link capacity or computation time or distance). As shown in next section all of constructed Bayesian networks will have the same structure but different parameters.

3.2 Bayesian Network Based Trust Model

The Bayesian network based trust model presented in [21] is extended by adding additional aspects (e.g. peer workload and peer bandwidth) and variant time window size to improve performance metrics. Similarly task processor peers in our P2P computing network, have not similar capabilities. For example, some task processor peers may have high network link capacity, while others connect through weak network link capacity. Some may have higher speed processor compared to other task processor peers. Therefore capability of task processor peers can be represented by various characteristics, such as peer credibility [21], peer workload, peer bandwidth, peer availability, peer computation time [12] and peer distance [13] to other peers. These aspects are defined as follows:

Peer Credibility (C_P): The probability that the result produced by a peer is correct.

$$C_P = C_R / \left(E_R + C_R + I_R \right) \tag{14}$$

Where C_R denotes the number of correct results returns from individual task processor and E_R denotes the number of erroneous results and I_R denotes the number of incomplete results returns from individual task processor peer.

Peer Workload *(W_P)*: This parameter denotes the average CPU Usage in individual task processor peer and it scale in a continuous range scale of [0,1] that 0 is the idle CPU and 1 denotes the fully busy CPU.

Peer Bandwidth *(B_P)*: Different peers in the P2P network have not similar network link capacity. In our simulation the bandwidth of peers are randomly set from 256 KBps to 4096 KBps. This parameter are normalized later to scale in [0,1].

Peer Computation Time (CT_P): When a task processor peer perform the assigned task, peer computation time is calculated as follows:

$$CT_P = AT \times A_P \tag{15}$$

Where AT denotes the peers ideal time and A_P denotes peers availability. The CT_P represent the time that required by a peer when actually executes the task in the presence of peer autonomy failures.

Peer Distance ($D(T_D, T_P)$): the distance between the task distributor peer and task processor peer.

$$D\left(T_D, T_P\right) = T_C - T_S \tag{16}$$

Where T_D is task distributer peer and T_P is task processor peer and T_C is the time when T_D sends the task to T_P and T_S is the time when the response returns from T_P.

3.3 Constructing Bayesian Based Trust Model

The task processor peer's overall capability C_{TD} is calculated in task distributor peer to have the most appropriate task processor peers(T_D) each time the new task exists. If task distributer peer T_D has not been interacted so far he could acquire its capability by gossiping with other task distributor peers. To improve the performance, the information propagated by gossiping between task distributor peers in every Δ unit time.

The Bayesian network based trust model is applied based on proposed work in [21] to calculate overall capability of task processor peers in unstructured P2P computing network and uses several characteristics of a task processor peer as shown in Fig. 2.

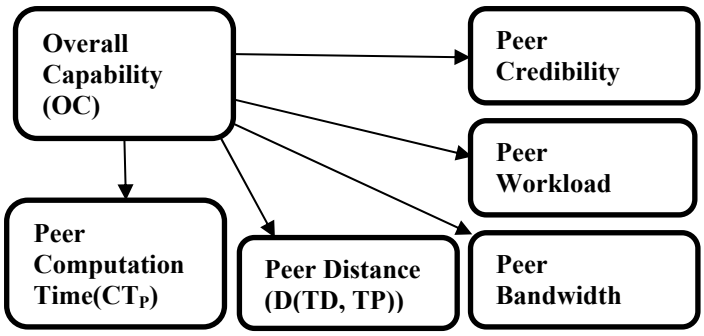


Figure 2.Bayesian Network Trust characteristics

A task distributor peers evaluate the task processor peers after each transaction and they calculate their overall capability shown in the root of the tree shown in Fig. 2. The task distributor peers construct a naïve Bayesian network for each task processor peers that they interacted with. So it is a flexible way to represent the trust between a task distributor peer and a task processor peer. Every Bayesian network has two values for the root node 'OC' "satisfied" and "unsatisfied", denoted by 1 and 0 respectively.

To consider the result of most recent 'K' transactions interacting the distributor peer with task processor peers in the transaction table we use "Time Window" of length 'K'. In despite of [21] that used fixed 'K' we evaluate different value for 'K' from 1 to all previous transactions and see that if more historical data to develop Bayesian network is considered higher precision in determining overall capability is achieved until the optimum time window size is reached.

Conditional probability table (CPT) for the node 'OC' is shown in Table I witch 'a' represent total number of satisfying transactions and 'b' represent total number of unsatisfying transactions occur within time window size = K. The 'p(OC=1)' is percentage of transactions that are satisfying and represent the value of overall trust of task distributor peer about individual task processor peer. Different characteristics of the task processor are shown in leaves in Fig. 2. For any characteristic the conditional probability table (CPT) should be defined. For example we illustrate peer workload characteristic CPT table in Table II. The node denoted by peer workload, represents how much a CPU of a peer is busy. The values are mapped to four fuzzy concepts "High", "good", "Medium" and "Low'. Each column follows one constraint, which corresponds to one value of the root node. The sum of values of each column is equal to 1. Here m1, m2, m3 and m4 represent the numbers of transactions satisfied when value for peer workload are "high", "good", "medium", and "low" respectively. Similarly n1, n2, n3, and n4 denote the numbers of transactions unsatisfied with these four values.

Table 1 Conditional Probability Table For Node Overal Compatibility

Outcome	***The Number of Transactions given the outcome in time window 'K'***	***Probabilities***
OC = 1	a	p (OC = 1) = a / (a + b)
OC = 0	b	p (OC = 0) = b / (a + b)

Table 2 Conditional Probability Table For Node Peer Workload

	The no. of transaction given OC=1, in Time Window 'K'	***Probabilities where OC=1 in Time Window 'K'***	***The no. of transaction given OC=0, in Time Window 'K'***	***Probabilities where OC=0 in Time Window 'K'***
High	m1	p(workload = "high" | OC = 1) = m1 / a	n1	p(workload = "high" | OC = 0) = n1 / b
Good	m2	p(workload = "good" | OC = 1) = m2 / a	n2	p(workload = "good" | OC = 0) = n2 / b
Medium	m3	p(workload = "medium" | OC = 1) = m3 / a	n3	p(workload = "medium" | OC = 0) = n3 / b
Low	m4	p(workload = "low" | OC = 1) = m4 / a	n4	p(workload = "low" | OC = 0) = n4 / b

To calculate, p (workload = "good" | OC = o) for example we evaluate the conditional probability with the condition that a transaction is unsatisfied when workload of processor peer is 'good'. We illustrate the derivation to calculate this probability as below:

$$P\left(workload = \text{"good"}, OC = 0\right) = \frac{\text{The no. of trans. where woarkload = "good" and OC=0}}{Total\ Number\ of\ Transactions} \tag{17}$$

$$P\left(workload = \text{"good"} \mid OC = 0\right) = \frac{P\left(workload = \text{"good"}, OC = 0\right)}{P\left(OC = 0\right)} \tag{18}$$

$$P\left(workload = \text{"good"}, OC = 0\right) = \frac{n2}{\left(a+b\right)} \tag{19}$$

$$P\left(workload = \text{"good"} \mid OC = 0\right) = \frac{\frac{n2}{\left(a+b\right)}}{\frac{b}{\left(a+b\right)}} = \frac{n2}{b} \tag{20}$$

Different Fuzzy concept categorization could be deploying simply and we can calculate other probabilities in table II in similar way.

If we have some aspects A_i according to Table 2. and a task distributer peer has not enough confident about some task processor peers the gossip learning approach can be used to learn probabilities in CPTs from other peer's recommendations (called cases) by using the expectation maximization algorithm. Let the recommendation of a specific capability for a task processor peer TP_j given by an aspect a_i at time 0 represented as $P^j_{i(0)}$, then the cases in table. 3. can be obtained. The element in the first column is trust values given by local task distributer peer that he has sufficient confidence according to equation(13).

Table 3: Cases for CPT Learning

$\mathbf{A_1}$	$\mathbf{A_2}$	…	$\mathbf{A_i}$
$P^1_{1(0)}$	$P^1_{2(0)}$	…	$P^1_{i(0)}$
$P^2_{1(0)}$	$P^2_{2(0)}$	…	$P^2_{i(0)}$
…	…	…	…
$P^1_{1(1)}$	$P^1_{2(1)}$	…	$P^1_{i(1)}$

A task distributer peer should update his Bayesian networks regularly to reflect the dynamic characteristic of trust and to do this the task distributer peer fades old probabilities before taking new cases into consideration. The following equations are used to update CPTs by adopting the method from Netica [25]:

$$t_0 = 1; t_{n+1} = \frac{1}{\sum_{j=0}^{\frac{1}{\theta_P^1}-1} \left(P_n \left(A_i \in I_j | A_1 \in I_k \right) t_n \lambda_\gamma + 1 - \lambda_\gamma \right)} \tag{21}$$

$$P_0 \left(A_i \in I_j | A_i \in I_k \right) = \frac{1}{\theta_P^1}; \tag{22}$$

$$P_{n+1} \left(A_i \in I_j | A_i \in I_k \right) = \frac{1}{t_{n+1} \left(P_n \left(A_i \in I_j | A_1 \in I_k \right) t_n \lambda_\gamma + 1 - \lambda_\gamma \right)} \tag{23}$$

Where λ_γ is the fading factor and $P_i (i \geq 0)$ are the probabilities in time i and $t_i (i \geq 0)$ is the normalization constant at time i.

We estimate reputation of task processor peer by using Bayes rule. To calculate the probability that recommendation of aspect A_i lies in $I_{i1}$with the assumption that the recommendation given to a task processor peer by $A_i$lies in I_i is as follows:

$$P \left(A_1 \in I_{i1} | A_2 \in I_{i2}, \ldots, A_k \in I_{ik} \right) = \frac{P \left(A_2 \in I_{i2}, \ldots, A_k \in I_{ik} | A_1 \in I_{i1} \right) P \left(A_1 \in I_{i1} \right)}{P \left(A_2 \in I_{i2}, \ldots, A_k \in I_{ik} \right)} \tag{24}$$

The following equation can be deduced since all aspects A_i are conditionally independent given A_i:

$$P \left(A_1 \in I_{i1} | A_2 \in I_{i2}, \ldots, A_k \in I_{ik} \right) = \frac{P \left(A_2 \in I_{i2} | A_1 \in I_{i1} \right) \ldots P \left(A_k \in I_{ik} | A_1 \in I_{i1} \right) P \left(A_1 \in I_{i1} \right)}{P \left(A_2 \in I_{i2} \right) \ldots \left(A_k \in I_{ik} \right)} \tag{25}$$

Next the value of A_i can be estimated as the expectation value of its states as follows:

$$P = \sum_{i=0}^{i=\frac{1}{\theta_P^1}-1} \frac{(2i+1)\theta_P^1}{2} P \left(A_1 \in I_i \right) \tag{26}$$

In order to reduce the computational overhead and to increase the accuracy of estimates we select the most reliable task distributer peer at first to give recommendations and use the recommendation of these peers only to infer the reputation of task processor peers

3.4 Updating the Bayesian Network

The task distributor peer builds its trust values about corresponding task processor peers over time. The experience made after new task assignments should be added after

each transaction by updating of task distributor peer's Bayesian network. In the proposed approach the task distributor peers propagate their knowledge with each other via gossiping to have more choices of task processor peers to assign new tasks. This process limits malicious peers to have new chance in P2P network to have malicious behaviours. However the information in any pair of task distributor peers should be updated.

To motivate a peer to contribute continuously and to give second chance to malicious peers we use time windows to consider peers present behaviours rather than past behaviours as in [21], but the different sizes of time window is used to enhance precision. The aim is to personalize the time window size according to the specific requirements of the task distributor peers. By time window, the last k previous transactions is considered rather than all transactions. Several aspects can be used to evaluate transactions such as peer credibility [21], peer workload, peer bandwidth, peer availability, peer computation time [12], and peer distance [13]. The task distributor peer calculates the overall evaluation of a transaction by combining these criteria. Depending on requirements of task distributor peer this combination can be personalized. For example a task distributor may prefer high speed connection of processing peers due to high volume data exchanged or prefer fast processing or high accuracy. After calculating overall evaluation, the result is satisfied or not satisfied that is used to update trust value for any corresponding task processor peers as follows:

$$OS^{(TPi)} = w_{cr} \times S_{C_P}^{(TPi)} + w_{wl} \times S_{W_P}^{(TPi)} + w_{bw} \times S_{B_P}^{(TPi)} + w_d \times S_{D_P}^{(TPi)} + w_{ct} \times S_{CT_P}^{(TPi)} \quad (27)$$

Here, $OS^{(TPi)}$ represents the overall satisfaction of task distributor peer in last transaction with task processor peer (TPi). w_{cr}, w_{wl}, w_{bw}, w_d and w_{ct} are weights that denotes the importance of different aspects such as credibility, peer workload, peer bandwidth, peer computation time and peer distance respectively. The $S_{C_P}^{(TPi)}$, $S_{W_P}^{(TPi)}$, $S_{B_P}^{(TPi)}$, $S_{D_P}^{(TPi)}$ and $S_{CT_P}^{(TPi)}$ are the normalized score of peer credibility, workload, bandwidth, distance and computation time respectively. To normalize the scores the following equation is used:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (28)$$

Since $S_{D_P}^{(TPi)}$ has negative effect on the utility of $OS^{(TPi)}$, the following equation is used to normalize it:

$$x' = \frac{\max(x) - x}{\max(x) - \min(x)} \quad (29)$$

If the overall satisfaction score $OS^{(TPi)}$ becomes less than a predefined threshold ξ, the transaction is unsatisfied otherwise the transaction is satisfied.

3.5 Task Assignment Algorithm

Some design requirements to design the algorithms in distributed computing environment should be considered as following: 1) The algorithm should be robust and scalable and it should maintain a reasonable performance even in extreme failure scenarios. 2) We should have the possibility of proper decision at any time in a local

manner and all task distributor peers should be able to perform proper selection without any extra communication. 3) The algorithm should be having a low communication complexity that means not high number of sent messages should be existed.

As shown in Algorithm.1. proposed algorithm is based on event driven gossiping algorithm [26]. This algorithm is run at each task distributor peers in the P2P network simultaneously. The algorithm consists of an active loop with Δ period (line 3) and a method to handle incoming messages from other task distributor peers (line 19) in a gossiping manner. The task distributer peer TDi send its knowledge about corresponding task processor peers (the list that represents the most trusted task processor peers) to other task distributer peer TDj (line 4) randomly to propagate its aggregated knowledge. When respond messages from task processor peers arrives, the Bayesian network is updated based on transaction and then overall capability calculated based on combined aspects to extend local knowledge about task processor peers and to make better decisions based on it. To simplify the algorithm we assumed that the length of the period of the loop Δ is the same at all nodes and we have not any assumptions about the synchronization of the loops at the different task distributor peers.

Table 4 Task Assignment Algorithm

Input:

$\boldsymbol{p^{(TDi)}}$: The peerID of the task distributor peer i

$\boldsymbol{p^{(TPi)}}$: The peerID of the task processor peer i

$\boldsymbol{p}$: Candidate task processor peer set selected by $p^{(TDi)}$ with best selection criteria

Δ: The length of the loop

$\boldsymbol{Subtask_k^{(TDi)}}$: The k'th subtask that sent by $p^{(TDi)}$ to some $p^{(TPj)}$

$\boldsymbol{OS_j^{(TPi)}}$: The parameter to compute aggregated task processor peer's satisfaction, if it exceed ξ the peer is not malicious.

$\boldsymbol{S_{C_P}^{(TPi)}}$: The normalized score of peer credibility.

$\boldsymbol{S_{W_P}^{(TPi)}}$: The normalized score of peer workload.

$\boldsymbol{S_{B_P}^{(TPi)}}$: The normalized score of peer bandwidth.

$\boldsymbol{S_{D_P}^{(TPi)}}$: The normalized score of peer distance.

$\boldsymbol{S_{CT_P}^{(TPi)}}$: The normalized score of peer computation time.

$\mathbf{m}^{(i)}$: The message sent from peer $p^{(TDi)}$ to peer $p^{(TDj)}$

$\boldsymbol{Candid_l^{(TPi)}}$: Represent $p^{(TDi)}$ knowledge about corresponding task processor peers and represent the most trusted ones

$\boldsymbol{Candid_l^{(i)}.peerSet()}$: Return the array of candidate peers

$\boldsymbol{fr_{TPj}^{(TDi)}}$:The flag that indicate if $p^{(TPj)}$ is malicious peer in $p^{(TDi)'s}$ viewpoint

$\boldsymbol{TP(p^{(TDi)})}$: Return peerID array of task processor peers corresponding to $p^{(TDi)}$

Procedure:

1: initModel()
2: **loop**
3: wait(Δ)
//Selecting most promising task processor peers to assign the task
4: Send($\mathrm{Candid_l^{(i)}}$) to Random $\mathrm{p^{(TDj)}}$
5: p ← selectPeer()
6:send $\mathrm{Subtask_k^{(TDi)}}$to $p^{(TPj)}$
7: **For** any $p^{(TPj)}$ in p **do**
//The corresponding task processor peers may not responding to our request due to
//free-riding behavior or not availability and the returned respond may be incorrect
8: **If** (IsLive($p^{(TPj)}$) && !CorrectResponse($p^{(TPj)}$)) **then**

9: Compute($S_{C_P}^{(TPi)}$)
10: $fr_{TPj}^{(TDi)}$ = True
11:**Else**
12: Update Bayesian Network for $p^{(TPj)}$ based on Table. II.
13: Compute($S_{W_P}^{(TPi)}$)
14: Compute($S_{D_P}^{(TPi)}$)
15: Compute($S_{CT_P}^{(TPi)}$)
16: $OS_j^{(TPi)}$ = CombineOverallCapability($p^{(TPj)}$) by Equation (27)
16: **End If**
17: **End For**
18: **End** loop
//Update local knowledge and getting Random $p^{(TDj)}$ knowledge about its corresponding task // processor peers
19: **Procedure** OnReceive ($Candid_l^{(TPj)}$)
20: **For** any $p^{(TPk)} \in (Candid_l^{(TPi)} \cap Candid_l^{(TPj)})$ **do**
21: **If** Reliable($p^{(TPk)}$) **then**
22: Update($OS_k^{(TPi)}$) by Equation (30)
23: **If** ($OS_k^{(TPi)} \geq \xi$) **then**
24: $fr_{TPj}^{(TDi)}$ = False
25: **End If**
26: **End If**
27:**End For**
28: **End Procedure**

When a task distributor peer wants to assign the new task to the task processor peers the list of task processing peers are sorted according to their trust values and returned in select peer procedure (line 5). Now the task distributor peer selects the highly trusted task processor peers and distribute the task among them (line 6). If the task distributor peer has no previous transaction with a task processor peer, it obtain the desired information from other task distributor peers in gossiping manner. When a task distributor peer receive responses from task processing peers the Bayesian network of corresponding peer will be updated according to number of transactions in time window. The corresponding task processor peers may not responding to the assigned tasks due to free-riding behaviour or not availability and the returned respond may be incorrect. In this case the task processor peers marks as malicious peer and its normalized score of peer credibility is updated (line 8-11). If the task processor peers perform the assigned task then they send some additional information with the results such as computation time, available bandwidth, etc. The task distributer peer updates the Bayesian network, based on these information and compute the overall satisfaction scores for each task processor peers (line 12-18).

The OnReceive method handles incoming messages from other task distributor peers in a passive thread (line 19). For any task processor peer $p^{(TPk)}$ exists in both the candidate list of peer TPi and TPj if it is reliable (that means it is not malicious peer) its overall satisfaction score updated by the following equation:

$$OS_k^{(TPi)} = \propto_1 \times OS_k^{(TPi)} + \propto_2 \times OS_k^{(TPj)} \quad (30)$$

Where the $\propto_1$ and $\propto_2$are the effectiveness parameters and $\propto_1 + \propto_2 = 1$. If $\propto_1 = \propto_2 = 0.5$ then the equation becomes the distributed averaging function. If the overall satisfaction score of a task processor peer TP_k exceeds the threshold ξ then the TP_k is marked as regular peer rather than a malicious peer (lines 23-25). The reason is to give another chance to task processor peers that change their behaviors to non-malicious peer.

4. Experimental Results and Evaluation

We used peerSim[27] for our experiments that run on Eclipse IDE[28] to simulate the behavior of task distributor peers and task processor peers. The aim is to evaluate the effectiveness of proposed approach in estimating reputation scores and trust values in different scenarios. PeerSim is an open source simulator that composed of different pluggable building blocks implemented in Java. Through these building blocks it is easier to prototype a protocol. There are two simulation models supported by PeerSim: cycle-based and event-based model [29]. We use the event-based model to have the transport layer and to have the system more like to systems in the wild such as message transport latencies. simulation-based approach has two advantages compared with the measurement-based approach. The greater flexibility in controlling the various configuration parameters of our protocol mechanisms is the first advantage. Another advantage is that it allows us to study the impact of variations in a particular mechanism while keeping the rest fixed, which is very difficult to achieve in measurement-based experiments [30].

4.1 Experimental Settings

The aim is to evaluate that if the proposed method could help the task distributor peers to select best task processor peers and most trusted peers that match their preferences. The task distributor peer is interested in correct and fast result from not busy task processor peers that have min computation time and maximum bandwidth. In the network initialization we divide task processor peers to three categories based on good nodes, malicious nodes, and average nodes categorization. We call them trusted, mid-trusted and untrusted peers. The trusted nodes are good nodes that try to offer service to other peers perfectly. The untrusted peers are malicious nodes that are attackers and return incorrect results. The mid-trusted peers are the average nodes. Mid-trust peers have variant behaviour in the network and perform malicious behaviour for some time. The population of task processor peers were 1000 and the number of transactions was 15. The bandwidth is assigned to peers in the range of [640KBps, 4096KBps] randomly. The task distributor peers announce new task every 200ms and they select the most trusted peers to assign subtasks. The satisfaction score is in the range of [0, 1) and it is 0 at the initialization phase of simulation and it is changed during simulation. The threshold ξ is set to 0.5.

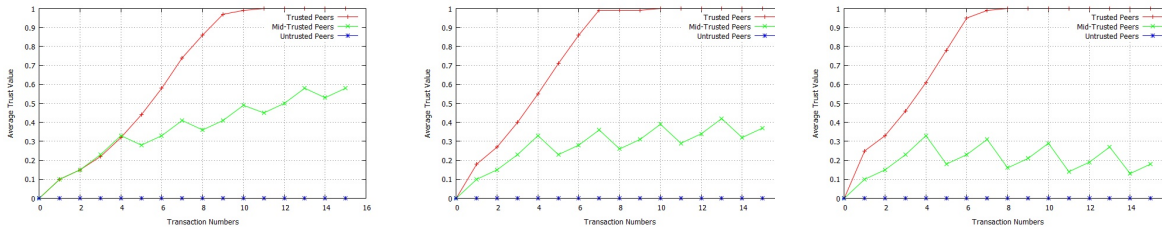
Table 5 Simulation parameters .

Task processor peers	1000
Task distributer peers	4
Node bandwidth	*640* Kbps, 1, 2, 4 Mbps
Number of transactions	*15*
Trusted peer distribution	34 %
Mid-trusted peer distribution	33 %
malicious peer distribution	33 %
Maximum growth	20%
Duplicated requests	1
Transport layer protocol	Uniform random transport
Minimum delay	10 ms
Maximum delay	400 ms
Add nodes	5
Remove nodes	5
ξ	0.5
$\propto_1$	0.5
$\propto_2$	0.5

The task processor population is set to 1000 nodes. Although this network sizes may not look realistic, the objective of the simulation is to evaluate the ability of proposed mechanism in prohibiting untrusted peers and in this sense, the results obtained with a population of 1000 peers can be extrapolated to larger populations. In addition, experiments with larger populations are conducted and found no significant changes and similar trends could be seen. Indeed, some other related work uses similar populations. [31, 32] Table IV. shows the most important simulation parameters. The trusted peer distribution is the percentage of task processor peers that complete assigned tasks properly and is set to 34%. The mid-trusted peer distribution are those whom perform task with sometimes incorrect results. malicious peer distribution is the percentage of untrusted task processor peers. Studies have shown that malicious nodes can be found in two forms (a) those which do not perform tasks, and (b) those which respond with irrelevant content or incorrect results. Maximum growth defines how much the network can grow with respect to the network size when network dynamics is used and its value is 20% in the simulations. Duplicated requests are the number of requests for the same block sent to different peers. Duplicated requests are not considered in the simulations. The transport layer implemented in the simulation to achieve a system that is more similar to the systems in the wild. In the transport layer, the UniformRandomTransport protocol is used, which provides a way to reliably deliver messages with a random delay. The minimum and maximum delays are 10 and 400 ms respectively. Adding nodes and removing nodes parameters change dynamically the size of the network (called churn). The number of added or removed nodes in network dynamic is 5 nodes at any time.

4.2 Experimental Settings

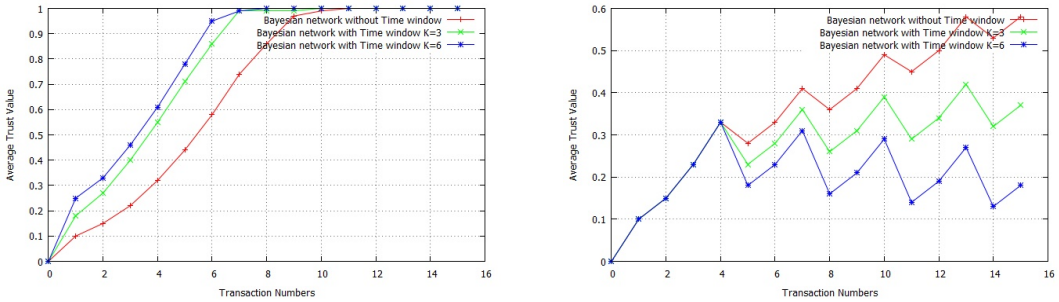
We compare the average trust values calculated by a task distributer peer using his own observations without Bayesian network and the average trust values estimated using Bayesian networks to evaluate the effectiveness of Bayesian network in trust estimation.



(a) **(b)** **(c)**

Figure 3.(a) Comparison of Average Trust Values for Bayesian Network without Time Window (b) Comparison of Average Trust Values for Bayesian Network with Time Window K= 3 & (c) Comparison of Average Trust Values for Bayesian Network with Time Window K= 6

As can be seen from Fig. 3.a, using the extended Bayesian network has caused the speedup of discouraging the mid-trusted peers. The trusted peers usually return correct results and the mid-trusted peers usually return two correct results after an incorrect result in the simulation scenario. In the different scenarios the behaviour of mid-trusted peers may change accordingly but its important to note that the trust value of these peers are gradually improved due to their positive activities. Untrusted peers are the malicious nodes and discouraged completely to improve their trust values. The different time window size is considered to evaluate the effect of changing in the number of previous transactions on the trust values. In Fig. 3.a, no time window is used and its better to say that the time widow K = 1. The Fig. 3.b and Fig. 3.c show the average trust values for Bayesian network with time window K=3 and 6 respectively. Comparison between Fig. 3.b and Fig. 3.c shows that using time window K=6 has better performance than using time window K=3. When the time window K=6 is used to calculate the average trust values, more incorrect results are considered to evaluate the trust value of a task processor peer therefore the increment ratio of average trust value is diminished for mid-trusted peers. On the contrary, the increment ratio of average trust value for trusted peers in time window K=6 is better than the average trust value for trusted peers in time window K=3. The reason is including more success experience of trusted peers in time window K=6 than time window K=3. Similar trends for different time window sizes can be seen.



(a) **(b)**

Figure 4. (a) Effect of Time Window size in Average Trust Values of Trusted Peers & (b) Effect of Time Window size in Average Trust of Mid-Trusted Peers

The effect of time window size on trusted peers and mid-trusted peers is evaluated. As can be seen from Fig 4.a the increment in widow size improves the average trust value of trusted values. However there is a trade-off between increment of time window size and the cost of memory and processing time so based on the trade-off the optimal time window size shall be found. Fig 4.b illustrates the effect of time window on average

trust value for mid-trusted peers. As can be seen from Fig 4.b using the larger time window size diminished the average trust value in the unstable behavior of mid-trusted peers in the pre mentioned simulated scenario.

Next the effect of malicious behaviours on peer's trust value is evaluated. As can be seen form Fig. 5, if task processors TP3 and TP5 perform malicious behaviors for some time in simulation process, with time window K=3 the trust values fall 10% and 5% more than the basic Bayesian network based trust model presented in [21] for TP3 and TP5 respectively after first malicious transaction, and it is almost 15% and 10% more in the successive malicious transaction. This result shows that the trust value of malicious peers reduced faster in proposed Bayesian trust model than the basic Bayesian trust model. Hence the proposed model identifies malicious peers better and earlier than basic Bayesian trust model in a P2P computing system.

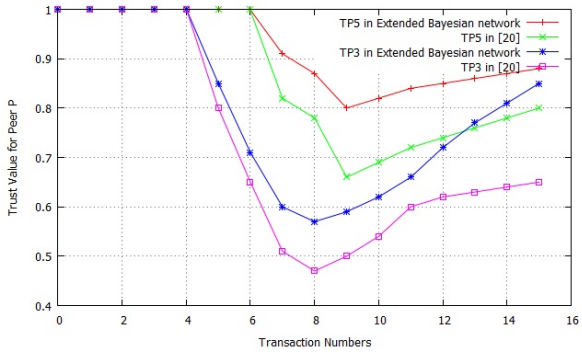


Figure 5. Comparison of Malicious Task Processor Peers TP3 and TP5 in our extended Bayesian network and Bayesian network presented in [20]

Next the optimum time window size is evaluated. As can be seen form Fig. 6. the trust value get the maximum value for both trusted and mid-trusted peers TP4 and TP6 in the time window size K=9 then the optimal time window size for trusted task processor peer TP4 is K=9. Similar trends can be seen for other trusted and mid-rusted task processor peers so the memory and time complexity could be limited to the optimal time window size. It should be mentioned that the K may change for another trusted and mid-trusted task processor peers, but its is important for any peer to identify its optimum time window size.

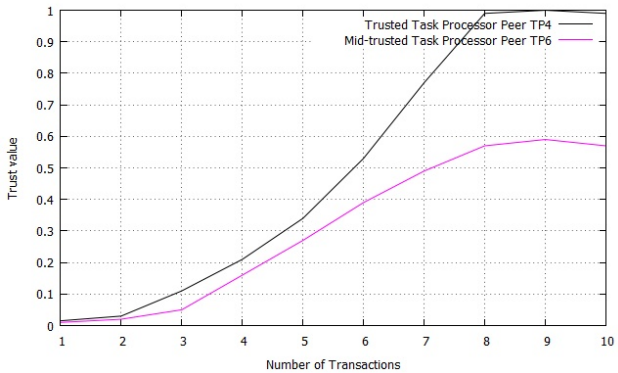


Figure 6. Optimum Time Window Size for Trusted and Mid-trusted Task Processor Peer TP4 and TP6

5. Results and Conclusion

In this paper we use the extended Bayesian networks for estimating reputation and trust values of task processor peers in pure computing P2P network. By combining different dimensions of trust, more aspects of trust value such as network link capacity and workload of task processor peers in extended Bayesian network are added. Gossiping is used to consider the subjectivity of trust. In the presented gossip learning based approach the task distributer peers exchange their knowledge about their corresponding task processor peers during task assignment to have the ability to find more appropriate task processor peers and to earn enough information about them. The optimum time window size is found to reduce memory and time complexity. The Bayesian network based approach is implemented on peerSim simulator and the results show that the presented approach completely discovered and punished the malicious peers. However, it is still an unresolved problem how to generate more complex learning models to cover other trust aspects successfully and it was left to our future works.

Acknowledgements

The authors would like to thank the anonymous reviewers for their time, feedback, and a detailed reading of the paper. Their comments were highly insightful and enabled us to greatly improve the quality of our manuscript. The authors are grateful to the constructive suggestions of the anonymous researchers, which have improved the content and the presentation.

References

[1] G. Chmaj and K. Walkowiak, "Decision Strategies for a P2P Computing System," *j-jucs,* vol. 18, pp. 599-622, mars 2012.

[2] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, *et al.*, "Peer-to-peer computing," ed: Technical Report HPL-2002-57, HP Labs, 2002.

[3] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky, "SETI@home-massively distributed computing for SETI," *Computing in Science & Engineering,* vol. 3, pp. 78-83, 2001.

[4] D. Anderson, J. Cobb, E. Korpela ,M. Lebofsky, and D. Werthimer, "SETI@home: An Experiment in public-resource computing," in *Communications of the ACM,* 2002, pp. 45:56–61.

[5] M. Bakri Bashir, M. S. B. A. Latiff, Y. Coulibaly, and A. Yousif, "A survey of grid-based searching techniques for large scale distributed data," *Journal of Network and Computer Applications,* vol. 60, pp. 170-179, 1// 2016.

[6] S.Larson, C. Snow, and V. Pande, "Folding@home and genome@home: Using distributed computing to tackle previously intractable problems in computational biology," *Computational Genomics Journal,* 2003.

[7] *BOINC* (2009).. Available: http://boinc.berkeley.edu

[8] (2009) .*GreenTea Technologies Inc.* Available: http:http://www.greenteatech.com

[9] J. Verbeke, N. Nadgir, G. Ruetsch, and I. Sharapov" ,Framework for peer-to-peer distributed computing in a heterogeneous, decentralized environment," in *Grid*, 2002, pp. 1-12.

[10] V. Lo, D. Zappala, D. Zhou, Y. Liu, and S. Zhao, "Cluster computing on the fly: P2P scheduling of idle cycles in the internet ",in *Peer-to-Peer Systems III*, ed: Springer, 2004, pp. 227-236.

[11] S. Choi, M. Baik, J. Gil, C. Park, S. Jung, and C. Hwang, "Group-based dynamic computational replication mechanism in peer-to-peer grid computing," in *Cluster Computing and the Grid, 2006 .CCGRID 06. Sixth IEEE International Symposium on*, 2006, pp. 8 pp.-7.

[12] J. Dubey and V. Tokekar, "Identification of Reliable Peer Groups in Peer-to-Peer Computing Systems," in *Advances in Communication, Network, and Computing*, ed: Springer, 2012, pp . 233-237.

[13] J. Dubey and V. Tokekar, "Identification of efficient peers in P2P computing system for real time applications," *arXiv preprint arXiv:1212.3074*, 2012.

[14] X. Chen, G. Chen, J. Liu, X. Luo, X. Li, and B. Li, "Trust factors in P2P networks ",in *Semantic Computing and Systems, 2008. WSCS'08. IEEE International Workshop on*, 2008, pp. 49-54.

[15] S. Kraxberger and U. Payer, "Security concept for peer-to-peer systems," in *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, 2009, pp. 931-936.

[16] J.-T. Kim, H.-K. Park, and E.-H. Paik, "Security issues in peer-to-peer systems," in *Advanced Communication Technology, 2005, ICACT 2005. The 7th International Conference on* ,2005 ,pp. 1059-1063.

[17] A. Ravichandran and J. Yoon, "Trust management with delegation in grouped peer-to-peer communities," in *Proceedings of the eleventh ACM symposium on Access control models and technologies*, 2006, pp. 71-80.

[18] C. Ding, C. Yueguo ,and C. Weiwei, "A survey study on trust management in P2P systems," *Department of Computer Science, School of Computing"-National University of Singapore,* 2004.

[19] T. Chao, H. Chaoji, and L. Shengli, "An Improved Trust Model Based on Reputation in P2P Networks," in *Wearable Computing Systems (APWCS), 2010 Asia-Pacific Conference on*, 2010, pp. 255-258.

[20] Y. Wang and J. Vassileva, "Bayesian network-based trust model," in *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, 2003, pp. 372-378.

[21] J. Dubey and V. Tokekar, "Bayesian network based trust model with time window for Pure P2P computing systems," in *Wireless Computing and Networking (GCWCN), 2014 IEEE Global Conference on*, 2014, pp. 219-223.

[22] J. Dubey and V .Tokekar, "Investigation of Peer Grouping Methods in Peer-to-Peer Computing Networks," *International Journal of Computer Applications,* vol. 83, 2013.

[23] J. Dubey and V. Tokekar, "A framework for pure Peer-to-Peer computing system," in *Wireless and Optical Communications Networks (WOCN), 2012 Ninth International Conference on*, 2012, pp. 1-5.

[24] Y. Wang, V. Cahill, E. Gray, C. Harris, and L. Liao, "Bayesian network based trust management," in *Autonomic and Trusted Computing*, ed: Springer, 2006, pp. 246-257.

[25] (2016) .*Netica, The world's most widely used Bayesian network development software*. Available: http://www.norsys.com/

[26] R. Ormándi, I. Hegedűs, and M. Jelasity, "Gossip learning with linear models on fully distributed data," *Concurrency and Computation: Practice and Experience,* vol. 25, pp. 556-571, 2013.

[27] M. Jelasity, A. Montresor, G. Paolo, and S. Voulgaris. (2009). *PeerSim: A Peer -to- Peer Simulator*. Available: http://peersim.sourceforge.net.

[28] Eclipse-Foundation. (2016). *Eclipse IDE for Java Developers*. Available: https://www.eclipse.org/

[29] I. Kazmi and S. F. Y. Bukhari, "PeerSim: An Efficient & Scalable Testbed for Heterogeneous Cluster-based P2P Network Protocols," in *Computer Modelling and Simulation (UKSim), 2011 UkSim 13th International Conference on*, 2011, pp. 420-425.

[30] R. L. Xia and J. K. Muppala, "A Survey of BitTorrent Performance," *Communications Surveys & Tutorials, IEEE,* vol. 12, pp. 140-158, 2010.

[31] M.-V. Belmonte, M. Díaz, J.-L. Pérez-de-la-Cruz, and A. Reyna, "COINS: COalitions and INcentiveS for effective Peer-to-Peer downloads," *Journal of Network and Computer Applications,* vol. 36, pp. 484-497, 2013.

[32] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson, "One hop reputations for peer to peer file sharing workloads," presented at the Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, San Francisco, California, 2008.