BP-P2P: Belief Propagation-Based Trust and Reputation Management for P2P Networks

Erman Ayday
School of Electrical and Comp. Eng.
Georgia Institute of Technology
Atlanta, GA 30332, USA
Email: eayday@gatech.edu

Faramarz Fekri
School of Electrical and Comp. Eng.
Georgia Institute of Technology
Atlanta, GA 30332, USA
Email: fekri@ece.gatech.edu

***Abstract*—In this paper, for the first time, we introduce a Belief Propagation (BP)-based distributed trust and reputation management algorithm. The proposed algorithm can be utilized in many distributed systems from Peer-to-peer (P2P) networks to social and mesh networks. In this work, we focus on P2P networks and explore the application of BP-based trust and reputation management in a decentralized environment in the presence of malicious peers. In a typical P2P trust and reputation management system, after each transaction, the client peer (who receives a service) provides its rating about the quality of the service provided by the server peer for that transaction. In such a system, we view the problem of trust and reputation management as to compute two sets of variables: 1. the reputation parameters of peers based on their quality of service, and 2. the trustworthiness parameters of peers based on the ratings they provide after each transaction. We distinguish between these two parameters as a peer might provide high quality service as a server while providing malicious ratings as a client. The proposed scheme, referred to as BP-P2P, relies on the BP algorithm in an appropriately chosen factor graph representation of the P2P network. The reputation and trustworthiness parameters are computed by a BP-based distributed message passing algorithm between the peers on the factor graph. We provide a detailed evaluation of BP-P2P via analysis and computer simulations. We show that BP-P2P is very robust in computing trustworthiness values and filtering out malicious ratings. Specifically, we prove that BP-P2P iteratively reduces the error in the reputation values of peers due to the malicious ratings with a high probability. Further, comparison of BP-P2P with some well-known and commonly used P2P reputation management techniques (e.g., EigenTrust and Bayesian Framework) indicates the superiority of the proposed scheme in terms of robustness against malicious behavior. We also show that the computational complexity of BP-P2P grows only linearly with the number of peers and the communication overhead of BP-P2P is lower than the well-known EigenTrust algorithm.**

I. Introduction

Peer-to-peer (P2P) networks [1] are commonly defined as distributed architectures in which the workload is partitioned between the peers and each peer is equally privileged. As opposed to traditional client-server networking (in which certain peers are responsible for providing resources while other peers only consume), every peer plays the role of both a client and a server in the P2P networks [2]. In other words, each peer provides access to its resources (e.g., processing power, storage) as a server without the need for a central authority. P2P networks have especially became popular as distributed file sharing systems in which peers exchange files between each other (such as Gnutella or Napster).

Due to their size and the distributed architecture, P2P systems are highly vulnerable to attacks by the malicious peers. The most common attack to P2P systems is in the form of injecting inauthentic files (or introducing viruses) to the network. Malicious behavior in P2P networks is mainly confronted by utilizing trust and reputation management systems in which peers (as clients) get to rate the peers (acting as servers) based on the quality of the transactions. A trust and reputation management mechanism is a promising method to protect the client peer by forming some foresight about the server peers before using their resources. Using a distributed trust and reputation management mechanism, reputation values of the servers and the trustworthiness values of the clients (on their ratings) can be computed by the peers without needing a central authority. As a result of this, malicious behavior can be detected and honest behavior can be encouraged in the network. However, this rating mechanism puts the server peers in a vulnerable position as the malicious peers may undermine (or boost) the reputation values of certain peers by providing incorrect ratings. Hence, building a resilient trust and reputation management system that is robust against malicious activities becomes a challenging issue. Despite recent advances in trust and reputation management in P2P networks, there is yet a need to develop reliable, scalable and dependable schemes that would also be resilient to various ways a distributed trust and reputation system can be attacked.

In this paper we introduce the first application of the Belief Propagation (BP), an iterative probabilistic algorithm, in the design and evaluation of distributed trust and reputation management systems. Belief Propagation [3], [4] is a message passing algorithm for performing inference on graphical models. It is a method for computing marginal distributions of the unobserved nodes conditioned on the observed ones. In our previous work, we developed the general theory of BP-based reputation management algorithms for centralized environments [5]. However, in a distributed infrastructure, trust and reputation management is more complicated than in the centralized solutions. Hence, using the basic theory in our previous work [5], in this paper, we focus on P2P networks and explore the application of BP-based trust and reputation management in a decentralized environment in the presence of malicious peers. We note that the proposed algorithm can also be utilized in various distributed systems such as mesh and social networks. In mesh networks, the proposed algorithm can be used to evaluate the types and behaviors of the nodes, while in social networks, it can be used to evaluate the behaviors and trustworthiness values of the peers.

We introduce the "Belief Propagation-Based Trust and Reputation Management for P2P Networks" (hereafter referred to as BP-P2P). Our objective is to compute both the reputation values of the peers as servers and their trustworthiness values as clients (based on the ratings they provide for the servers). We view this problem as an inference problem which involves computing

This material is based upon work supported partially by the National Science Foundation under Grant No. IIS-1115199, and a gift from the Cisco University Research Program Fund, an advised fund of Silicon Valley Community Foundation.

the marginal probability distributions of the reputation values from the global joint probability distribution function of many variables. This problem, however, cannot be solved directly in a large-scale system, because the number of terms grow exponentially with the number of peers in the network. The key role of the BP algorithm is that we can use it to compute those marginal distributions in the complexity that grows only linearly with the number of peers. BP-P2P describes the P2P network on a factor graph; allowing the peers to compute the reputation and trustworthiness values by distributed message passing between each other. The main contributions of our work are summarized in the following.

1) We introduce the first application of the Belief Propagation (BP) algorithm in the design and evaluation of distributed trust and reputation management systems for P2P networks. We introduce a graph-based mechanism which relies on a factor graph to compute the reputation of each peer (as a server) and its trustworthiness value (as a client) by a BP-based iterative and distributed message passing algorithm.
2) The proposed distributed algorithm enables the peers to compute the reputation values (of other peers) with a small error in the presence of attackers. Further, it also allows the peers to obtain the trustworthiness values (of other peers) by analyzing the ratings provided, which enables them detect and filter out malicious ratings effectively.
3) The computational complexity of BP-P2P is at most linear in the number of peers in the network, making it very attractive for large-scale systems. Further, its communication overhead is lower than the well-known EigenTrust algorithm which is particularly designed for P2P networks.
4) The proposed BP-P2P outperforms the existing and commonly used P2P reputation management techniques such as the EigenTrust algorithm [6] and the Bayesian Approach [7] (which is also proposed as the reputation management system of the well-known CONFIDANT protocol [8]) in the presence of attackers.

The rest of this paper is organized as follows. In the rest of this section, we summarize the related work. In Section II, we describe the proposed BP-P2P in detail. Next, in Section III, we analyze BP-P2P using a mathematical model for the peers. Further, we evaluate BP-P2P via computer simulations and compare BP-P2P with the existing and commonly used P2P reputation management schemes. Finally, in Section IV, we conclude the paper.

A. Related Work and Their Shortcomings

Trust and reputation management systems for P2P networks and online systems received a lot of attention [6], [7], [9]–[16]. In [9] and [10], authors provide a good survey of the work on the use of trust and reputation management systems for P2P networks. Most proposed P2P trust and reputation management mechanisms utilize the idea that a peer can monitor others and obtain direct observations [11] or a peer can enquire about the reputation value of another peer (and hence, obtain indirect observations) before using the service provided by that peer [12], [13]. EigenTrust [6] is one of the most popular reputation management algorithms for P2P networks. However, the EigenTrust algorithm is constrained by the fact that trustworthiness of a peer (on its feedback) is equivalent to its reputation value. However, trusting a peer's feedback and trusting a peer's service quality are two different concepts. As we will discuss in Section III-A, a malicious peer, by providing incorrect or malicious ratings, may attack the reputation management system while providing a high quality service. In other words, a node may have high reputation but low trustworthiness value. Thus, equating the two will affect the performance of the reputation management. Further, the EigenTrust algorithm relies on the presence of pre-trusted peers in the network which is either impractical or limiting in most networks. Most importantly, the EigenTrust algorithm computes the global reputation values by a simple iterative weighted averaging mechanism which is vulnerable to collaborative attacks from the malicious peers. Use of the Bayesian Framework is also proposed in [8]. In the Bayesian Framework, each reputation value is computed independent of the other nodes' reputation values. However, the ratings provided by the nodes induce a probability distribution on the reputation values. These distributions are correlated because they are induced by the overlapping set of nodes. Therefore, ignoring these dependencies could affect the performance dramatically. The strength of BP-P2P stems from the fact that it captures this correlation in analyzing the ratings and computing the trust and reputation values. Different from the existing schemes, BP-P2P algorithm is a graph based iterative algorithm motivated by the previous success on message passing techniques and BP algorithms on various applications such as inference and decoding error correcting codes.

II. Belief Propagation-Based Trust and Reputation Management for P2P Networks

We assume two different sets in the system: i) the set of servers, $\mathbb{S}$ and ii) the set of clients, $\mathbb{U}$. We further assume that every peer in the network plays the role of both a client and a server. In other words, each peer provides access to its resources (e.g., provides files) as a server. On the other hand, each peer also uses the resources of other servers as a client. Therefore, sets $\mathbb{S}$ and $\mathbb{U}$ are not disjoint and each peer i is represented both in set $\mathbb{S}$ (as a server) and in set $\mathbb{U}$ (as a client). Transactions occur between the servers and clients, and clients provide feedbacks in the form of ratings about servers after each transaction (based on the service quality of the transaction). First, for the simplicity and clarity of the presentation, we will describe the *fundamental scheme* assuming that each peer computes its own reputation value (as a server) and trustworthiness value (as a client) via distributed message passing and report these values to other peers when they are queried. However, this fundamental scheme is not completely secure as malicious peers may report incorrect values for their own reputation and trustworthiness values upon an inquiry. Then, in Section II-A, we will describe how we make this scheme completely secure by allowing different groups of peers (referred as the score managers) to compute the reputation and trustworthiness values of individual peers.

Let G_j be the reputation value of server j ($j \in \mathbb{S}$) and T_{ij} be the rating that client i ($i \in \mathbb{U}$) reports about server j ($j \in \mathbb{S}$), whenever a transaction is completed between the two peers. Moreover, let R_i denote the trustworthiness of the peer i ($i \in \mathbb{U}$) as a client. In other words, R_i represents the amount of confidence on the correctness of any rating provided by client i. We assume there are u clients and s servers in the system (i.e., $|\mathbb{U}| = u$ and $|\mathbb{S}| = s$)[1]. Let $\mathbb{G} = \{G_j : j \in \mathbb{S}\}$ and $\mathbb{R} = \{R_i : i \in \mathbb{U}\}$ be the collection of variables representing the reputations of the servers and the trustworthiness values of the clients, respectively. Further, let $\mathbb{T}$ be the $s \times u$ server-client matrix that stores the rating values (T_{ij}), and $\mathbb{T}_i$ be the

[1]Since each peer is both a server and a client, $u = s$ in a typical P2P network.

set of ratings provided by client i. We consider slotted time throughout this discussion. At each time-slot (or epoch), BP-P2P algorithm is executed using the input parameters $\mathbb{T}$ and the present $\mathbb{R}$ to obtain the reputation parameters and the updated trustworthiness values at each peer. We note that each peer has only a part of the input parameters based on its previous transactions. More specifically, we assume that every peer i knows the ratings it previously provided as a client (i.e., $\mathbb{T}_i$) and the set of servers for whom it provided these ratings. Moreover, every peer i knows the ratings it previously received from other peers as a server and the set of clients who provided these ratings (similar to [6]). After BP-P2P completes its iterations, each peer computes its new reputation value as a server as well as its updated trustworthiness value as a client. For simplicity of presentation, we assume that the rating values are from the set $\Upsilon = \{0, 1\}$. The extension in which rating values can take any real number can be developed similarly.

The reputation management problem can be viewed as finding the marginal probability distributions of each variable in $\mathbb{G}$, given the observed data (i.e., evidence). There are s marginal probability functions, $p(G_j|\mathbb{T}, \mathbb{R})$, each of which is associated with a variable G_j; the reputation value of server j. We formulate the problem by considering the global function $p(\mathbb{G}|\mathbb{T}, \mathbb{R})$, which is the joint probability distribution function of the variables in $\mathbb{G}$ given the rating matrix and the trustworthiness values of the clients. Then, clearly, each marginal probability function $p(G_j|\mathbb{T}, \mathbb{R})$ may be obtained as follows:

$$p(G_j|\mathbb{T}, \mathbb{R}) = \sum_{\mathbb{G}\setminus\{G_j\}} p(\mathbb{G}|\mathbb{T}, \mathbb{R}), \tag{1}$$

where $\mathbb{G}\setminus\{G_j\}$ implies all variables in $\mathbb{G}$ except G_j.

Unfortunately, the number of terms in (1) grows exponentially with the number of variables, making the computation infeasible for large-scale systems. Further, (1) can only be solved in a centralized environment in which all the evidence $\mathbb{T}$ and $\mathbb{R}$ is available at a central unit. On the other hand, P2P networks are typically distributed environments, and hence, solving (1) at each peer is not feasible in such networks in which each peer has only a part of the evidence. Thus, we propose to factorize (1) to local functions f_i using a factor graph and utilize the Belief Propagation (BP) algorithm to calculate the marginal probability distributions in linear complexity and in a distributed environment. A factor graph is a bipartite graph containing two sets of nodes (corresponding to variables and factors) and edges incident between two sets. Following [4], we form a factor graph by setting a variable node for each variable G_j, a factor node for each function f_i, and an edge connecting variable node j to the factor node i if and only if G_j is an argument of f_i. We note that computing marginal probability functions is exact when the factor graph has no cycles. However, the BP algorithm still gives good approximate results for the factor graphs with cycles.

First, we arrange the collection of the clients and the servers together with their associated relations (i.e., the ratings) as a factor graph, as in Fig. 1. In this representation, each client corresponds to a factor node shown as a square and each server is represented by a variable node shown as a hexagon in the graph. Each rating is represented by an edge from the factor node to the variable node. Hence, if a client i ($i \in \mathbb{U}$) has a rating about server j ($j \in \mathbb{S}$), we place an edge with value T_{ij} from the factor node i to the variable node representing server j. We note that if any new rating arrives from client i about server j, the T_{ij} value is updated by averaging the new rating and the old value of the edge multiplied with the fading factor $\gamma_{ij}(t) = \vartheta^{t-t_{ij}}$ (where ϑ and t_{ij} are the fading parameter and the time when the last transaction between client i and server j occurred, respectively).

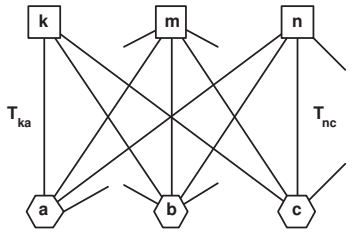

Fig. 1: Setup of the scheme.

Next, we suppose that the global function $p(\mathbb{G}|\mathbb{T}, \mathbb{R})$ factors into products of several local functions, each having a subset of variables from $\mathbb{G}$ as arguments as follows[2]:

$$p(\mathbb{G}|\mathbb{T}, \mathbb{R}) = \frac{1}{Z} \prod_{i\in\mathbb{U}} f_i(\mathcal{G}_i, \mathbb{T}_i, R_i), \tag{2}$$

where Z is the normalization constant and $\mathcal{G}_i$ is a subset of $\mathbb{G}$. Hence, in the graph representation of Fig. 1, each factor node is associated with a local function and each local function f_i represents the probability distributions of its arguments given the trustworthiness value and the existing ratings of the associated client.

We now introduce the messages between the factor and the variable nodes (i.e., between the servers and the clients) to compute the marginal distributions at each server using BP. To that end, we describe the message exchange between peer k (as a client) and peer a (as a server) in Fig. 1. We represent the set of neighbors of the variable node (server) a and the factor node (client) k as $\mathbf{N_a^s}$ and $\mathbf{N_k^c}$, respectively (neighbors of a server are the set of clients who rated the server while neighbors of a client are the servers whom it rated). Superscripts in the representation of the neighbors denote whether the neighbors of a peer are determined considering the peer as a client (c) or as a server (s). We note that neighbors of a peer as a server (or variable node) do not need to be the same as its neighbors as a client (or factor note). We further let $\Xi = \mathbf{N_a^s}\setminus\{k\}$ and $\Delta = \mathbf{N_k^c}\setminus\{a\}$. Factor and variable nodes in Fig. 1 iteratively exchange probabilistic messages following the BP algorithm, updating the degree of beliefs on the reputation values of the servers as well as the trustworthiness values of the clients at each step, until the iterations stop. Let $\mathbb{G}^{(\nu)} = \{G_j^{(\nu)} : j \in \mathbb{S}\}$ be the collection of variables representing the values of the variable nodes at the iteration ν of the algorithm. We denote the messages from the variable nodes to the factor nodes and from the factor nodes to the variable nodes as μ and λ, respectively. The message $\mu_{a\to k}^{(\nu)}(G_a^{(\nu)})$ denotes the probability of $G_a^{(\nu)} = \ell$, $\ell \in \{0, 1\}$, at the ν^{th} iteration. On the other hand, $\lambda_{k\to a}^{(\nu)}(G_a^{(\nu)})$ denotes the probability that $G_a^{(\nu)} = \ell$, for $\ell \in \{0, 1\}$, at the ν^{th} iteration given T_{ka} and R_k.

The message from the factor node k to the variable node a at the ν^{th} iteration is formed using the principles of the BP as

$$\lambda_{k\to a}^{(\nu)}(G_a^{(\nu)}) = \sum_{\mathcal{G}_k^{(\nu)}\setminus\{G_a^{(\nu)}\}} f_k(\mathcal{G}_k^{(\nu)}, \mathbb{T}_k, R_k^{(\nu-1)}) \prod_{x\in\Delta} \mu_{x\to k}^{(\nu-1)}(G_x^{(\nu)}), \tag{3}$$

where $\mathcal{G}_k$ is the set of variable nodes which are the arguments of the local function f_k at the factor node k. This message transfer is illustrated in Fig. 2. Further, $R_k^{(\nu-1)}$ (the trustworthiness of client k calculated at the end of $(\nu-1)^{th}$ iteration) is a value between zero and one and can be calculated as follows:

$$R_k^{(\nu-1)} = 1 - \frac{1}{|\mathbf{N_k^c}|} \sum_{i\in N_k^c} \sum_{x\in\{0,1\}} |T_{ki} - x| \mu_{i\to k}^{(\nu-1)}(x). \tag{4}$$

[2]It is shown that such a factorization eventually gives the marginal probability distributions via the BP algorithm [4].

The above equation can be interpreted as one minus the average inconsistency of client k calculated by using the messages it received from all its neighbors. The above computation must be performed for every neighbors of each factor nodes. This finishes the first half of the ν^{th} iteration.

During the second half, the variable nodes (servers) generate their messages (μ) and send to their neighbors. Variable node a forms $\mu^{(\nu)}_{a\to k}(G^{(\nu)}_a)$ by multiplying all information it receives from its neighbors excluding the factor node k, as shown in Fig. 3. Hence, the message from variable node a to the factor node k at the ν^{th} iteration is given by

$$\mu^{(\nu)}_{a\to k}(G^{(\nu)}_a) = \frac{1}{\sum\limits_{h\in\{0,1\}}\prod\limits_{i\in\Xi}\lambda^{(\nu)}_{i\to a}(h)} \times \prod_{i\in\Xi}\lambda^{(\nu)}_{i\to a}(G^{(\nu)}_a). \quad (5)$$

This computation is repeated for every neighbors of each variable node. The algorithm proceeds to the next iteration in the same way as the ν^{th} iteration. It is worth noting that since each peer is both a server and a client, at the first half of the iteration, each peer generates messages as a client and in the second half of the iteration, each peer generates messages as a server. Further, each peer waits for all the messages from its neighbors before it creates its new message either as a client or as a server. We note that the iterative algorithm starts its first iteration by computing $\lambda^{(1)}_{k\to a}(G^{(1)}_a)$ in (3). However, instead of calculating in (4), the trustworthiness value R_k from the previous execution of BP-P2P is used as initial values in (3).

BP-P2P stops after Ψ iterations (which is a pre-defined number and its selection will be discussed in Section III-C). At the end of each iteration, the reputations are calculated at each server. To calculate the reputation value $G^{(\nu)}_a$, each server computes $\mu^{(\nu)}_a(G^{(\nu)}_a)$ using (5) but replacing Ξ with $\mathbf{N^s_a}$, and then sets $G^{(\nu)}_a = \sum_{i=0}^{1} i\mu^{(\nu)}_a(i)$. Thus, after the last iteration (i.e., Ψ^{th} iteration), each server peer obtains its updated reputation value and each client peer obtains its updated trustworthiness value.

A. Secure BP-P2P

As we discussed before, the fundamental scheme described in Section II is not completely secure since each peer computes and reports (upon an inquiry) its own reputation and trustworthiness values. However, it is clear that a malicious peer would report its own reputation and trustworthiness values to other peers incorrectly. Therefore, we propose to use a group of randomly selected peers (referred as the *score managers*) to do the message exchange, and hence, compute the reputation and trustworthiness values on behalf of each peer as in [6]. It is important to note that the score managers are not trustworthy peers and they can also be malicious as will be discussed later. Similar to [6], to assign score managers, we use a Distributed Hash Table (DHT) [17]. DHTs use a hash function to deterministically map the unique ID of each peer into the points in a logical coordinate space. At any time, the coordinate space is partitioned among the peers in the P2P network such that every peer covers a region in the coordinate space. Hence, score manager(s) of an arbitrary peer i is determined by hashing the unique ID of peer i into a point in the coordinate space and the peer which currently covers this point as part of its DHT region is appointed as the score manager of peer i[3]. Thus, any peer can easily determine the score manager(s) of peer i from its unique ID. We assume that the DHT can cope with the dynamics of the network (e.g., score managers leaving the system) as in [6]. Since it is not the main contribution of this paper, we do not give further detail about the selection of the score managers. As we mentioned before, our main contribution is the computation of trust and reputation values at the score managers via the BP-based algorithm. Next, we show how we modify our fundamental scheme such that it can be executed by score managers.

Using the DHT, each peer k is assigned with ξ score managers from the set $\mathbb{H}_k = \{H^1_k, H^2_k, \ldots, H^\xi_k\}$. We assume that each score manager of peer k knows: i) neighbors of peer k as a server (i.e., $\mathbf{N^s_k}$), and hence, the score managers of these neighbors, ii) neighbors of peer k as a client (i.e., $\mathbf{N^c_k}$), and hence, the score managers of these neighbors, iii) ratings previously provided by peer k as a client (i.e., $\mathbb{T}_k$), iv) ratings previously received by peer k as a server, and v) trustworthiness value of peer k as a client computed at the previous execution of the algorithm. The score managers in $\mathbb{H}_k$ generate the BP messages on behalf of peer k both as a server (variable node) and as a client (factor node). Further, they compute the reputation value (as a server) and the trustworthiness value (as a client) of peer k based on the messages they receive from the score managers of peer k's neighbors.

We note that since each peer in the network plays the role of both a client and a server; each score manager also has the same property. Therefore, at the first half of an iteration, each score manager generates messages as a client (factor node) and in the second half of the iteration, the same score manager generates messages as a server (variable node) on behalf of the peer it is responsible for. From now on, for the clarity of presentation, we refer to a score manager as a "client score manager" when it generates messages as a client, and as a "server score manager" when it generates messages as a server. Thus, different from the fundamental scheme described in Section II, BP messages are now exchanged between the client score managers and the server score managers. Due to this change, the factor graph in Fig. 1 is also modified based on the score managers of the peers. As an example, we illustrate the change in the connectivity of client k in Fig. 4 assuming $\xi = 3$. As illustrated in the figure, instead of connecting client k to the servers it rated (a, b and c), we connect the score managers of peer k to the score managers of peers a, b and c in the factor graph.

Messages are exchanged between the score managers of the peers following the principles of the BP algorithm (as described in Section II) and the algorithm stops after Ψ iterations (selection of Ψ will be discussed in Section III-C). We note that every score manager waits to receive all messages from its neighbors before it generates its new message. Further, score managers keep track of the iteration numbers to both remain loosely synchronized between each other and realize when to stop the algorithm. When a client i wants to use the service of a server j, it queries the reputation value G_j from the score managers of the peer j. Similarly, the trustworthiness value of a peer (as a client) can also be queried from its score managers. Once the client i receives all the computed G_j values from ξ different score managers in $\mathbb{H}_j$, it computes the mean of the received reputation values to determine the final reputation value of server j.

There is one obvious drawback of using score managers for BP-P2P algorithm. When a malicious peer becomes the score manager of a reliable or malicious peer, it may create and send bogus messages to its neighbors. Therefore, malicious messages may propagate in the BP algorithm affecting the efficiency of the algorithm. We describe the attack strategies of such malicious score managers in Section III-A. Further, we evaluate BP-P2P under this attack both analytically and via simulations

[3]If peer i has more than one score managers, the unique ID of peer i can be concatenated with an integer before hashing.

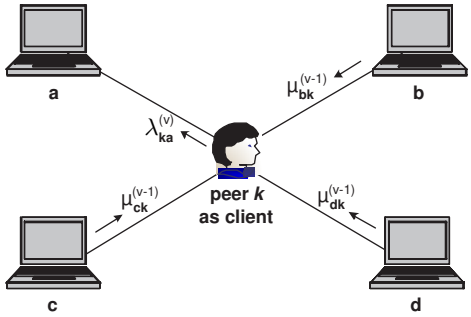

Fig. 2: Message from the factor node (client) k to the variable node (server) a at the ν^{th} iteration.

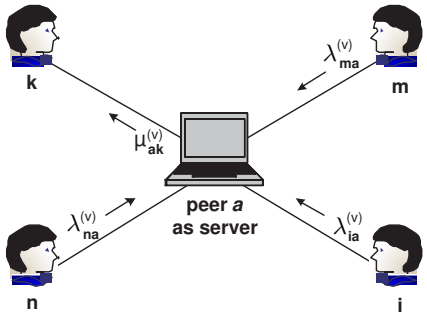

Fig. 3: Message from the variable node (server) a to the factor node (client) k at the ν^{th} iteration.

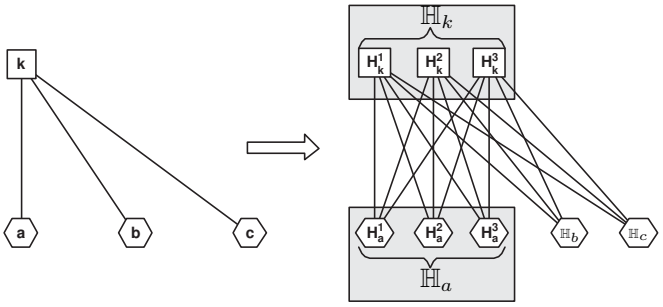

Fig. 4: Utilizing score managers in BP-P2P.

in Sections III-B and III-C, respectively.

B. Efficient BP-P2P

In this section, we provide some discussion on the computational complexity and communication overhead of BP-P2P.

1) Computational Complexity: One can show that the computational complexity of BP-P2P as $\mathbf{max}(\mathrm{O}(\xi c), \mathrm{O}(\xi v))$ per peer in the number of multiplications, where c and v are small numbers representing the average number of ratings generated by a client and the average number of ratings received by a server. Therefore, the computational complexity of BP-P2P is at most linear in the number of peers in the network, making it very attractive for large-scale systems.

2) Communication Overhead: In the fundamental scheme described in Section II, each client (and each server) sends different messages to each of its neighbors at each iteration. This introduces extra communication overhead to the scheme when multiple score managers are present for each peer. Therefore, in the following, we modify the messages in (3) and (5) between the peers (between the score managers of the peers in the secure version described in Section II-A) to reduce the communication overhead due to multiple score managers for each peer.

Before discussing these modifications in the messages between the score managers, we first approximate and simplify the message in (3) by assuming that the arguments of a local function at a factor node are independent from each other (to reduce the computational complexity at the client peers). Using this assumption, it can be shown that $\lambda^{(\nu)}_{k\to a}(G^{(\nu)}_a) \propto p(G^{(\nu)}_a|T_{ka}, R^{(\nu-1)}_k)$, where

$$p(G^{(\nu)}_a|T_{ka}, R^{(\nu-1)}_k) = \Big[(R^{(\nu-1)}_k + \frac{1-R^{(\nu-1)}_k}{2})T_{ka} + \frac{1-R^{(\nu-1)}_k}{2}(1-T_{ka})\Big]G^{(\nu)}_a + \Big[\frac{1-R^{(\nu-1)}_k}{2}T_{ka} + (R^{(\nu-1)}_k + \frac{1-R^{(\nu-1)}_k}{2})(1-T_{ka})\Big](1-G^{(\nu)}_a). \quad (6)$$

This resembles the belief/pleusability concept of the Dempster-Shafer Theory [18], [19]. Given $T_{ka} = 1$, $R^{(\nu-1)}_k$ can be viewed as the belief of the client k that $G^{(\nu)}_a$ is one (at the ν^{th} iteration). In other words, in the eyes of client k, $G^{(\nu)}_a$ is equal to one with probability $R^{(\nu-1)}_k$. Thus, $(1 - R^{(\nu-1)}_k)$ corresponds to the uncertainty in the belief of client k. In order to remove this uncertainty and express $p(G^{(\nu)}_a|T_{ka}, R^{(\nu-1)}_k)$ as the probabilities that $G^{(\nu)}_a$ is zero and one, we distribute the uncertainty uniformly between two outcomes (one and zero). Hence, in the eyes of the client k, $G^{(\nu)}_a$ is equal to one with probability $(R^{(\nu-1)}_k + (1-R^{(\nu-1)}_k)/2)$, and zero with probability $((1 - R^{(\nu-1)}_k)/2)$. We note that a similar statement holds for the case when $T_{ka} = 0$. It is worth noting that, as opposed to the Dempster-Shafer Theory, we do not combine the beliefs of the clients. Instead, we consider the belief of each client individually and calculate probabilities that $G^{(\nu)}_a$ being one and zero in the eyes of each client as in (6).

We now describe how we modify the BP messages in (6) and (5) to reduce the communication overhead caused by multiple score managers per each peer. Let $\mathbb{H}_k$ be the set of score managers of peer k (in the secure version described in Section II-A). In principal, at each iteration, we let each score manager H^i_k in $\mathbb{H}_k$ broadcast a single message to all of its neighbors instead of sending different messages to each of its neighbors. Then, each neighbor of the score manager H^i_k computes the actual BP message in (6) or (5) using the broadcasted message (from H^i_k) and the information it already possesses. In the following, we discuss the details of this.

Let $\mathbb{H}_{\mathbf{N^c_k}}$ denote the set of score managers of neighbors of client k. Instead of computing (6) for all its neighbors separately, each client score manager in $\mathbb{H}_k$ (in the secure version) only computes and broadcasts the updated trustworthiness value of the client k to its neighbors in $\mathbb{H}_{\mathbf{N^c_k}}$ instead of sending different messages to each of its neighbors. Since the score managers in $\mathbb{H}_{\mathbf{N^c_k}}$ know the rating value given by client k to the servers they are responsible for, each score manager in $\mathbb{H}_{\mathbf{N^c_k}}$ computes the actual message itself. For example, since the score managers of server a know T_{ka}, they compute the message in (6) by only using the broadcasted $R^{(\nu-1)}_k$ value.

Similarly, all messages from a server score manager (H^j_a) to its neighbors (in $\mathbb{H}_{\mathbf{N^s_a}}$) may be communicated simultaneously via a single broadcast step to decrease the communication overhead. The message from H^j_a to one of its neighbors i in $\mathbb{H}_{\mathbf{N^s_a}}$ is formed by multiplying all the messages received at H^j_a excluding the one received from the score manager i (similar to (5)). Thus, H^j_a can simply broadcast the multiplication of all received messages to its neighbors, and allow i (and all other neighbors) to deduce the actual message from this broadcast.

Therefore, at each iteration, a server score manager broadcasts a single message instead of sending different messages to each of its neighbors.

We note that we used these modified message formats for the evaluation of BP-P2P (in Section III). Let Ψ be the total number of iterations required for a single execution of the algorithm and ξ be the number of score managers for each peer. Then, each score manager sends (on the average) $2\xi\Psi$ messages during the execution of the BP-P2P algorithm ($\Psi \leq 10$ as will be discussed in Section III-C). On the other hand, in EigenTrust [6] each score manager sends (on the average) $\mathbf{max}(\mathrm{O}(2\xi\Psi c), \mathrm{O}(2\xi\Psi v))$ messages during the algorithm, where c and v represent the average number of ratings generated by a client and the average number of ratings received by a server. Further, Ψ is reported to be (on the average) 10 for EigenTrust [6]. Therefore, we conclude that the proposed BP-based algorithm does not introduce a significant communication overhead to the network.

III. Security Evaluation

In order to facilitate future references, frequently used notations are listed in Table I.

$\mathbb{S}$	The set of peers acting as servers
$\mathbb{U}_M$	The set of malicious clients
$\mathbb{U}_R$	The set of non-malicious clients
$\mathbb{H}_i$	The set of score managers of peer i
r_m	Rating given by a malicious client
d	Total number of newly generated ratings, per time-slot, per a non-malicious client
b	Total number of newly generated ratings, per time-slot, per a malicious client

TABLE I: Notations and definitions.

A. Threat Model

We mainly focus on the malicious behaviors of the clients and score managers and explore their impact on the proposed trust and reputation management algorithm.

1) Malicious Clients: There are two major attacks that are common for any trust and reputation management mechanisms: i) Bad-mouthing, in which malicious clients attack the servers with the highest reputation by giving low ratings in order to undermine them, and ii) Ballot-stuffing, in which malicious clients try to increase the reputation values of servers with low reputations. Further, there are opportunities for the malicious score managers to attack specifically to the BP algorithm by creating incorrect BP messages. In the following, we describe how we modeled the adversary considering the aforementioned threats to evaluate BP-P2P in the most adverse environment.

We assumed that the malicious clients initiate bad-mouthing[4]. Further, all the malicious clients collude and attack the same subset Γ of servers in each time-slot (which represents the strongest attack), by rating those servers as $r_m = 0$ (assuming the rating values are either 0 or 1). In other words, we denote by Γ the set of size b in which every victim server has one edge from each of the malicious clients (in the factor graph in Fig. 1). The subset Γ is chosen to include those servers who have the highest reputation values but received the lowest number of ratings from the non-malicious clients (assuming that the attackers have this information[5]). We note that this attack scenario also represents the RepTrap attack in [20] which is shown to be a strong attack. To the advantage of malicious peers, we assumed that a total of T time-slots had passed since the initialization of the network and a fraction of the existing peers change behavior and become malicious after T time-slots. In other words, malicious clients behaved like non-malicious clients and increased their trustworthiness values before mounting their attacks at the $(T+1)^{th}$ time-slot. We will evaluate the performance for the time-slot $(T+1)$.

2) Malicious Score Managers: As we discussed in Section II-A score managers of an arbitrary peer i generate the BP messages on behalf of the peer i (both as server and as a client). Therefore, malicious score managers of a peer may create and send incorrect messages to their neighbors. By doing so, malicious score managers specifically attack the accuracy of the BP algorithm. When a malicious peer j is the score manager of a peer i (i.e., $j \in \mathbb{H}_i$) it creates bogus BP messages depending on the type of peer i as below:

- If i is a non-malicious peer:
  - When j creates a message as a server score manager on behalf of peer i, it reports an incorrect value for the probability of $G_i^{(\nu)} = \ell$ ($\ell \in \{0,1\}$) at every iteration ν (e.g., if normally $G_i^{(\nu)} = 1$ with high probability, j creates a message reporting that $G_i^{(\nu)} = 0$ with a probability of $1-\varrho$, where ϱ is a positive number close to zero).
  - When j creates a message as a client score manager on behalf of peer i, it reports a low trustworthiness value for peer i as a client (e.g., j reports the trustworthiness value of peer i as $R_i^{(\nu)} = \sigma$ at every iteration ν, where σ is a positive number close to zero).
- If i is a malicious peer:
  - When j creates a message as a server score manager on behalf of peer i, it reports a high value for the probability of $G_i^{(\nu)} = 1$ at every iteration ν to favor its ally (e.g., j creates a message reporting that $G_i^{(\nu)} = 1$ with a probability of $1-\varrho$, where ϱ is a positive number close to zero).
  - When j creates a message as a client score manager on behalf of peer i, it reports a high trustworthiness value for the malicious peer i as a client (e.g., j reports the trustworthiness value of peer i as $R_i^{(\nu)} = 1-\sigma$ at every iteration ν, where σ is a positive number close to zero).

We note that since the score managers of the peers are assigned via a DHT, we assume that malicious score managers do not collaborate. We considered the above threat models for both our analytical evaluation and simulations.

B. Analytical Evaluation

We adopted the following models for various peers involved in the P2P trust and reputation management system. We acknowledge that although the models are not inclusive of every scenario, they are good illustrations to present our results. We assumed that the service quality of each server remains unchanged during our evaluation. Moreover, the rating values are either 0 or 1 where 1 represents a good service quality (e.g., providing authentic files). Ratings generated by the non-malicious clients are distributed uniformly among the servers (i.e., their ratings/edges in the graph representation are distributed uniformly among the servers). We wish to evaluate the performance for the time-slot $(T+1)$ at which malicious peers change behavior and initiate their attacks as discussed in Section III-A.

[4]Even though we use the bad-mouthing attack, similar counterpart results hold for ballot-stuffing and combinations of bad-mouthing and ballot-stuffing.

[5]Although it may appear unrealistic for some applications, availability of such information for the malicious clients would imply the worst case scenario.

The performance of a reputation management mechanism is determined by its accuracy of estimating the reputation values of the servers. Therefore, we evaluate BP-P2P in terms of the Mean Absolute Error (MAE) ($|G_j - \hat{G}_j|$) computed at each non-malicious score manager of every server j, where $\hat{G}_j$ is the actual value of the reputation of server j. We require two conditions to be satisfied: 1) the scheme should iteratively reduce the impact of malicious peers and decrease the error in the reputation values of the servers (computed at the non-malicious score managers) until the iterations stop, and 2) the error on the G_j value of each server j (computed at the non-malicious score managers) should be less than or equal to ϵ (where ϵ should be a small value) after the last iteration (i.e., Ψ^{th} iteration). In the following, we obtained the conditions and probabilities to arrive at such a scheme. We note that although the discussions of the analysis are based on RepTrap attack via bad-mouthing (as described in Section III-A), the system designed using these criteria will be robust against ballot-stuffing and combinations of bad-mouthing and ballot-stuffing as well.

The bad-mouthing attack is aimed to reduce the reputation values of the victim servers. Hence, G_a value of a victim server a (computed at the non-malicious score managers in set $\mathbb{H}_a$) should be a non-decreasing function of iterations. This leads to the below lemma.

Lemma 1: The error in the reputation values of the servers decreases with each successive iterations (until the iterations stop) if $G_a^{(2)} > G_a^{(1)}$ is satisfied with high probability at the non-malicious score managers of peer a ($\mathbb{H}_a \cap \mathbb{U}_R$) for every server a ($a \in \mathbb{S}$) with $\hat{G}_a = 1$[6].

Proof: Let $G_a^{(\omega)}$ and $G_a^{(\omega+1)}$ be the reputation value of an arbitrary server a with $\hat{G}_a = 1$ calculated at the $(\omega)^{th}$ and $(\omega+1)^{th}$ iterations at the non-malicious score managers of peer a ($\mathbb{H}_a \cap \mathbb{U}_R$), respectively. Further, let $\mathbb{H}^R_{\mathbf{N_a^s}}$ denote the set of score managers of non-malicious neighbors of server a (i.e., $\mathbf{N_a^s} \cap \mathbb{U}_R$) and $\mathbb{H}^M_{\mathbf{N_a^s}}$ denote the set of score managers of malicious neighbors of server a (i.e., $\mathbf{N_a^s} \cap \mathbb{U}_M$). $G_a^{(\omega+1)} > G_a^{(\omega)}$ if the following is satisfied at the $(\omega+1)^{th}$ iteration.

$$\prod_{j \in \mathbb{H}^R_{\mathbf{N_a^s}} \cap \mathbb{U}_R} \frac{R_j^{(w+1)}+1}{1-R_j^{(w+1)}} \prod_{j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_R} \frac{1-\hat{R}_j^{(w+1)}}{1+\hat{R}_j^{(w+1)}} > \prod_{j \in \mathbb{H}^R_{\mathbf{N_a^s}} \cap \mathbb{U}_R} \frac{R_j^{(w)}+1}{1-R_j^{(w)}} \prod_{j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_R} \frac{1-\hat{R}_j^{(w)}}{1+\hat{R}_j^{(w)}}, \quad (7)$$

where $R_j^{(w)}$ and $\hat{R}_j^{(w)}$ are the trustworthiness values of a reliable and malicious client calculated at a non-malicious score manager (as in (4)) at the w^{th} iteration, respectively.

Given $G_a^{(\omega)} > G_a^{(\omega-1)}$ holds at the ω^{th} iteration, we would get $\hat{R}_j^{(w)} > \hat{R}_j^{(w+1)}$ for $j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_R$ and $R_j^{(w+1)} \geq R_j^{(w)}$ for $j \in \mathbb{H}^R_{\mathbf{N_a^s}} \cap \mathbb{U}_R$. Thus, (7) would hold for the $(w+1)^{th}$ iteration. On the other hand, if $G_a^{(\omega)} < G_a^{(\omega-1)}$, we get $\hat{R}_j^{(w)} < \hat{R}_j^{(w+1)}$ for $j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_R$ and $R_j^{(w+1)} < R_j^{(w)}$ for $j \in \mathbb{H}^R_{\mathbf{N_a^s}} \cap \mathbb{U}_R$. Hence, (7) is not satisfied at the $(w+1)^{th}$ iteration. Therefore, if $G_a^{(\omega)} > G_a^{(\omega-1)}$ holds for some iteration ω at the peers in $\mathbb{H}_a \cap \mathbb{U}_R$, then the BP-P2P algorithm reduces the error on the reputation value (G_a) until the iterations stop, and hence, it is sufficient to satisfy $G_a^{(2)} > G_a^{(1)}$ with high probability at the non-malicious score managers of every server a with $\hat{G}_a = 1$ (the set of servers from which the victims are taken) to guarantee that BP-P2P iteratively reduces the impact of malicious clients at the non-malicious score managers until the iterations stop. ■

Although because of the Lemma 1, the error in the reputation values of the servers decrease with successive iterations, it is unclear what would be the eventual impact of the malicious peers. Once the condition in Lemma 1 is met and assuming Ψ be the total number of iterations required for a single execution of the BP-P2P algorithm, the probability (P) that BP-P2P provides an MAE that is less than ϵ for each server at every non-malicious score managers can be obtained as in (8). In (8), $R_j^{(\Psi+1)}$ and $\hat{R}_j^{(\Psi+1)}$ are the trustworthiness values of a reliable and malicious client calculated at a non-malicious score manager, respectively. Further, $\tilde{R}_j^{(\Psi+1)}$ is the trustworthiness value of a malicious client calculated at a malicious score manager.

In the following example, we illustrate the results of our analytical evaluation. The parameters we used are $|\mathbb{U}_M| + |\mathbb{U}_R| = 100$, $|\mathbb{S}| = 100$, $\vartheta = 0.9$, $T = 20$, $b = 10$, $\varrho = \sigma = 0.1$, $\xi = 3$, and $\Psi = 10$ (selection of Ψ will be discussed in Section III-C). Further, we assumed that d is a random variable with Yule-Simon distribution, which resembles the power-law distribution used in modeling P2P and online systems [21], with the probability mass function $f_d(d;\rho) = \rho B(d, \rho+1)$ (with $\rho = 1$), where B is the Beta function. Finally, we assumed the threat model described in Section III-A. We note that we also evaluated BP-P2P with different parameters and obtained similar results. In Fig. 5, we illustrated the probability of BP-P2P providing a MAE that is less than ϵ (at each non-malicious score manager) versus fraction of malicious peers for two different ϵ values. We observed that for an acceptable value of ϵ ($\epsilon = 0.1$), BP-P2P satisfies MAE $< \epsilon$ with high probability for up to 30% malicious peers. Moreover, Fig. 6 illustrates the average MAE values provided by BP-P2P (at each non-malicious score manager) with high probability for different fractions of malicious peers. We observed that BP-P2P provides significantly small error values for up to 30% malicious peers. We note that these analytical results are also consistent with our simulation results that are illustrated in the next section.

C. Simulations

We evaluated the performance of BP-P2P via computer simulations (via MATLAB) and compared BP-P2P with the Bayesian reputation management framework in [7] (which is also proposed as the reputation management system of the well-known CONFIDANT protocol [8]) and the EigenTrust algorithm [6] in a distributed P2P network environment.

We assumed that d is a random variable with Yule-Simon distribution (with $\rho = 1$) as discussed in Section III-B. We set $T = 20$, $b = 10$, $\rho = 1$, $|\mathbb{U}| = 100$, $|\mathbb{S}| = 100$, $\varrho = \sigma = 0.1$, $\xi = 3$, and the fading parameter as $\vartheta = 0.9$[7]. Further, we assumed that rating values are from the set $\Upsilon = \{0, 1\}$. Finally, we assumed the threat model described in Section III-A in which there are both malicious clients and malicious score managers. Let $\hat{G}_j$ be the actual reputation value of server j. We obtained the performance of BP-P2P, at each time-slot, as the Mean Absolute Error (MAE) $|G_j - \hat{G}_j|$, averaged over the reputation values of all victim servers (i.e., the servers that are under attack) computed at their non-malicious score managers.

[6]The opposite must hold for any server with $\hat{G}_a = 0$.

[7]We note that for the EigenTrust and the Bayesian framework we used the same fading mechanism as BP-P2P and set the fading parameter as $\vartheta = 0.9$.

$$P = \prod_{a \in \mathbb{S}} Pr\Big\{\epsilon \geq 1 - \frac{\prod_{j \in \mathbb{H}^R_{\mathbf{N_a^s}} \cap \mathbb{U}_R} (R_j^{(\Psi+1)} + 1) \prod_{j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_R} (1 - \hat{R}_j^{(\Psi+1)}) \prod_{j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_M} (1 - \tilde{R}_j^{(\Psi+1)})}{\prod_{j \in \mathbb{H}^R_{\mathbf{N_a^s}} \cap \mathbb{U}_R} (R_j^{(\Psi+1)} + 1) \prod_{j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_R} (1 - \hat{R}_j^{(\Psi+1)}) \prod_{j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_M} (1 - \tilde{R}_j^{(\Psi+1)}) + \prod_{j \in \mathbb{H}^R_{\mathbf{N_a^s}} \cap \mathbb{U}_R} (1 - R_j^{(\Psi+1)}) \prod_{j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_R} (1 + \hat{R}_j^{(\Psi+1)}) \prod_{j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_M} (1 + \tilde{R}_j^{(\Psi+1)})}\Big\} \tag{8}$$

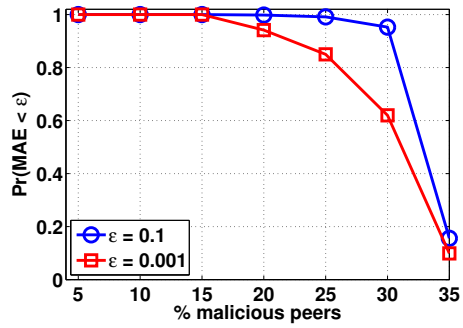

Fig. 5: Probability of MAE being less than ϵ versus fraction of malicious peers.

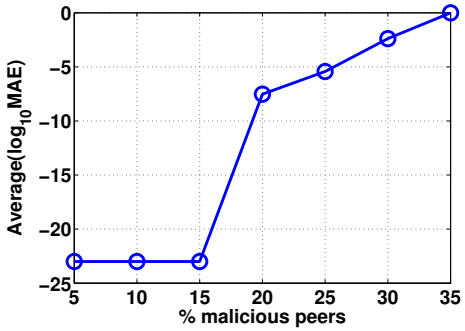

Fig. 6: The average MAE versus fraction of malicious peers.

For the Bayesian framework [7], we used the parameters from the original work [7] (deviation threshold $d = 0.5$ and trustworthiness threshold $t = 0.75$). Further, in favor of the Bayesian framework, we assumed that each peer have access to the server-client matrix $\mathbb{T}$. Therefore, we observed the reputation values computed at all non-malicious peers and we averaged the MAE over the reputation values of the victim servers. For the EigenTrust, we implemented the distributed algorithm described in [6] (with $\xi = 3$ score managers for each peer as in BP-P2P) and observed the reputation values computed at the non-malicious score managers as we did for BP-P2P. We note that we did not assume the existence of the pre-trusted peers for any schemes.

First, we determined the total number of iterations (Ψ) required for the BP-P2P algorithm. Thus, in Fig. 7[8], we observed the average number of required iterations for BP-P2P to converge at each peer (i.e., computed reputation values stop changing) for different fractions of malicious peers ($W = \frac{|\mathbb{U}_M|}{|\mathbb{U}_M|+|\mathbb{U}_R|}$), at different time-slots (measured since the attack is applied). We conclude that the average number of iterations for convergence is always less than 10 and it decreases with time and decreasing fraction of malicious peers. Thus, we used $\Psi = 10$ for the remaining of this section. Then, we evaluated the MAE performance of BP-P2P for different fractions of malicious peers (W), at different time-slots in Fig. 8. We observed that BP-P2P provides significantly low errors for up to about $W = 25\%$ malicious peers. Next, we observed the change in the average trustworthiness (R_i values) of malicious clients computed at the non-malicious score managers. Figure 9 illustrates the drop in the trustworthiness of the malicious clients with time. We conclude that the R_i values of the malicious clients (computed at non-malicious score managers) decrease over time, and hence, the impact of their malicious ratings is neutralized over time. Finally, we compared the MAE performance of BP-P2P with the Bayesian Framework and the EigenTrust algorithm. Figure 10 illustrates the comparison of BP-P2P with these schemes for different fractions of malicious peers at the first time-slot the attack is applied. It is clear that BP-P2P outperforms the Bayesian Framework and EigenTrust significantly. We note that at later time-slots, BP-P2P still keeps its superiority over the other schemes. From this comparison, we conclude that in EigenTrust, even the non-malicious score managers compute the reputation values with a large MAE in the presence of the attackers. Further, when the malicious nodes collaboratively attack, Bayesian Framework results in a high MAE in the reputation values of the servers. Finally, we observed the impact of ξ (the number of score managers for each peer) to the performance of BP-P2P under the same attack scenario (in which there are both malicious clients and malicious score managers as described in Section III-A). In Fig. 11, we illustrated MAE performance of BP-P2P for different values of ξ and for different fractions of malicious peers at the first time-slot the attack is applied. As expected, for small values of ξ (i.e., $\xi = 1$ and $\xi = 2$), BP-P2P provides higher MAE values since the probability that all score managers of a victim client being malicious increases with decreasing values of ξ.

Next, we simulated the same attack scenario when ratings are integers from the set $\Upsilon = \{1, \ldots, 5\}$ instead of binary values[9]. Malicious clients choose the victim servers from Γ and rate them as $r_m = 4$. The malicious clients do not deviate very much from the actual $\hat{G}_j = 5$ values to remain undercover as many time-slots as possible. We also tried higher deviations from the $\hat{G}_j$ value and observed that the malicious clients were easily detected by BP-P2P. We compared the MAE performance of BP-P2P with the other schemes at the first time-slot the attack is applied in Fig. 12 and observed that BP-P2P outperforms all the other techniques. We observed that BP-P2P provides significantly low MAE for up to $W = 25\%$ malicious clients. We further observed that the Bayesian Framework performs better than EigenTrust for this scenario.

IV. Conclusion

In this paper, we introduced the first application of the Belief Propagation algorithm in the design and evaluation of distributed trust and reputation management systems for

[8]The plots in Figs. 7, 8 and 9 are shown from the time-slot the adversary introduced its attack.

[9]For the attack against the BP algorithm, we assumed a similar scenario to the binary case as discussed in Section III-A.

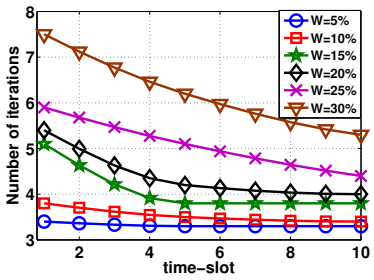

Fig. 7: The average number of iterations versus time for BP-P2P to converge when W of the existing peers become malicious.

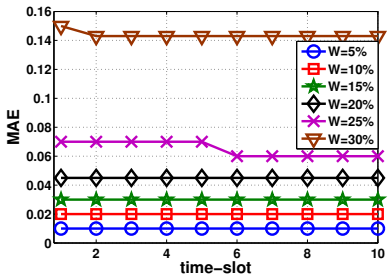

Fig. 8: MAE performance of BP-P2P versus time when W of the existing peers become malicious.

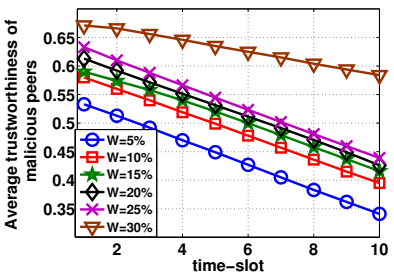

Fig. 9: Change in average trustworthiness of malicious clients versus time for BP-P2P when W of the existing peers become malicious.

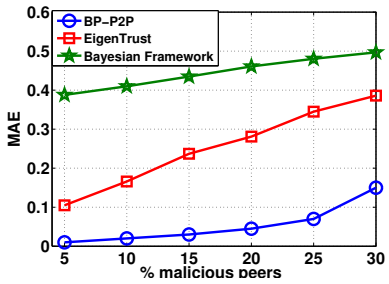

Fig. 10: MAE performance of various schemes when different fractions of the existing peers become malicious at the first time-slot the attack is applied.

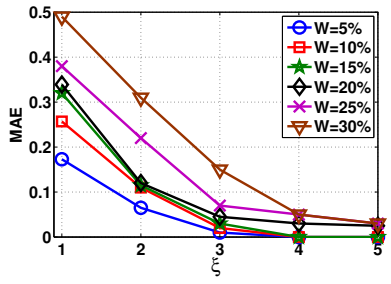

Fig. 11: MAE performance of BP-P2P for different values of ξ and for different fractions of malicious peers at the first time-slot the attack is applied.

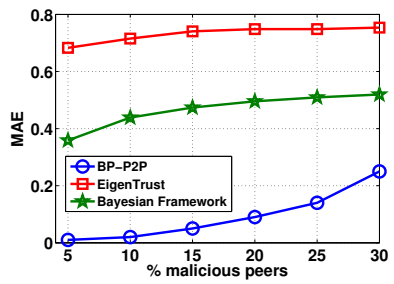

Fig. 12: MAE performance of various schemes when the rating values are from $\Upsilon = \{1, \ldots, 5\}$.

P2P networks. We presented the general protocol for Belief Propagation-Based Trust and Reputation Management for P2P Networks (BP-P2P). BP-P2P is a graph-based system in which the reputation and trustworthiness value of each peer is computed by distributed message passing among the peers in the graph. We studied BP-P2P in a detailed analysis and computer simulations. We showed that proposed BP-P2P is a robust mechanism to evaluate the reputation values of the peers from the received ratings. Moreover, it effectively evaluates the trustworthiness of the peers (in the reliability of their ratings). We also compared BP-P2P with some well-known P2P reputation management schemes and showed the superiority of the proposed scheme in terms of its robustness against malicious behavior. Finally, we showed that the computational complexity of the proposed scheme grows only linearly with the number of peers in the network while its communication overhead is lower that the well-known EigenTrust algorithm.

REFERENCES

[1] B. Fan, J. C. S. Lui, and D.-M. Chiu, "The design trade-offs of bittorrent-like file sharing protocols," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 365–376, April 2009.

[2] M. Adler, R. Kumar, K. Ross, D. Rubenstein, T. Suel, and D. D. Yao, "Optimal peer selection for p2p downloading and streaming," *In Proceedings of IEEE INFOCOM'05*, 2005.

[3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.

[4] F. Kschischang, B. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb. 2001.

[5] E. Ayday and F. Fekri, "BP-ITRM: belief propagation for iterative trust and reputation management," *to appear in ISIT '11: IEEE International Symposium on Information Theory*, 2011.

[6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in P2P networks," *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pp. 640–651, 2003.

[7] S. Buchegger and J. Boudec, "A robust reputation system for p2p and mobile ad-hoc networks," *Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems*, 2004.

[8] ——, "Performance analysis of CONFIDANT protocol (coorperation of nodes: Fairness in dynamic ad-hoc networks)," *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Jun. 2002.

[9] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara, "Reputation systems: facilitating trust in internet interactions," *Communications of the ACM*, vol. 43, no. 12, pp. 45–48, 2000.

[10] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.

[11] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," *CIKM '01: Proceedings of the 10th International Conference on Information and knowledge management*, pp. 310–317, 2001.

[12] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, "Choosing reputable servents in a P2P network," *WWW '02: Proceedings of the 11th International Conference on World Wide Web*, pp. 376–386, 2002.

[13] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," *CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 207–216, 2002.

[14] C. Dellarocas, "Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior," *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pp. 150–157, 2000.

[15] G. Wang and J. Wu, "Flowtrust: trust inference with network flows," *Frontiers of Computer Science in China*, vol. 5, no. 2, pp. 181–194, 2011.

[16] Y. Zhang and Y. Fang, "A fine-grained reputation system for reliable service selection in peer-to-peer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 8, pp. 1134–1145, 2007.

[17] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM '01: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 149–160, 2001.

[18] G. Shafer, *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, N.J., 1976.

[19] ——, "The dempster-shafer theory," *Encyclopedia of Artificial Intelligence*, 1992.

[20] Y. Yang, Q. Feng, Y. L. Sun, and Y. Dai, "RepTrap: a novel attack on feedback-based reputation systems," *SecureComm '08: Proceedings of the 4th International Conference on Security and Privacy in Communication Networks*, pp. 1–11, 2008.

[21] F. Slanina and Y. C. Zhang, "Referee networks and their spectral properties," *Acta Physica Polonica B*, vol. 36, pp. 2797–+, Sep. 2005.