ITERATIVE ALGORITHMS FOR TRUST AND REPUTATION MANAGEMENT AND RECOMMENDER SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

Erman Ayday

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2011

ITERATIVE ALGORITHMS FOR TRUST AND REPUTATION MANAGEMENT AND RECOMMENDER SYSTEMS

To my lovely wife, my parents,

and in loving memory of my grandfather

ACKNOWLEDGEMENTS

First and foremost, I would like to express my appreciation to my advisor, Prof. Faramarz Fekri, whose knowledge, wisdom, experience, and caring paved the way for success in my academic career. Dr. Fekri has provided me the tools, the will, the power, and the strive to reach success. His support will always be remembered.

I would like to extend my special thanks to Professor Ian F. Akyildiz. I got the chance to learn about wireless sensor networks from him, a pioneer in this field, and to be inspired by his knowledge and personality. I am also indebted to Dr. Akyildiz for creating a family atmosphere for me during his house parties and soccer games. I thank him deeply for being a great mentor and support to me.

I also would like to thank my other thesis committee members: Professor Douglas M. Blough and Professor Ling Liu, whom I had the opportunity to learn a lot from, and Professor Steven W. McLaughlin, whom I had also the opportunity to take a class and to benefit from his valuable insight for my career. I thank all of these outstanding professors for their inspiration, support, and guidance on my thesis.

I have so many friends and colleagues whom I would also like to thank. I would like to thank all my friends at the Georgia Tech Information Processing, Communications and Security (IPCAS) Research Lab for their support. I would like to extend my gratitude to the students, faculty, and staff at the Center for Signal and Image Processing (CSIP), who provided a great, thriving, and friendly work environment.

I would like to thank my great and dear family. My dear mom, Selmin Ayday, who thought me all the languages I can speak, and my dear dad, Can Ayday, who taught me to love research and more importantly to be a good human being. They taught me to be strong and determined. Thank you very much for all your love and support, and dedication and sacrifices that you made in your lives to make my life a better and more fruitful one. Hearing your voices from miles away has been a great source of energy for me to keep going. I also would like to thank my dear grandmothers Sevim Ayday, Ayten Colpan, and my dear grandfather Zuhtu Colpan for their loving care, support and encouragement. Last but not least, I thank my best friend and dear wife, Meltem. We started this journey 4 years ago together, and her presence has made a significant difference in both my professional and personal life. Thank you for always being there for me.

I would like to dedicate this thesis in the loving memories of Sevgi Soydan, Mustafa Remzi Soydan and my unforgettable grandfather Hayrettin Ayday. Without his support, I wouldn't even think of achieving this.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis investigates both theoretical and practical aspects of the design and analysis of iterative algorithms for trust and reputation management and recommender systems. It also studies the application of iterative trust and reputation management mechanisms in ad-hoc networks and P2P systems.

First, an algebraic and iterative trust and reputation management scheme (ITRM) is proposed. The proposed ITRM can be applied to centralized schemes, in which a central authority collects the reports and forms the reputations of the service providers (sellers) as well as report/rating trustworthiness of the (service) consumers (buyers). It is shown that ITRM is robust in filtering out the peers who provide unreliable ratings. Next, the first application of *Belief Propagation* algorithm, a fully iterative probabilistic algorithm, on trust and reputation management (BP-ITRM) is proposed. In BP-ITRM, the reputation management problem is formulated as an inference problem, and it is described as computing marginal likelihood distributions from complicated global functions of many variables. However, it is observed that computing the marginal probability functions is computationally prohibitive for large scale reputation systems. Therefore, the belief propagation algorithm is utilized to efficiently (in linear complexity) compute these marginal probability distributions. In BP-ITRM, the reputation system is modeled by using a factor graph and reputation values of the service providers (sellers) are computed by iterative probabilistic message passing between the factor and variable nodes on the graph. It is shown that BP-ITRM is reliable in filtering out malicious/unreliable reports. It is proven that BP-ITRM iteratively reduces the error in the reputation values of service providers due to the malicious raters with a high probability. Further, comparison of BP-ITRM with some well-known and commonly used reputation management techniques (e.g., Averaging Scheme, Bayesian Approach and Cluster Filtering) indicates the superiority of the proposed scheme both in terms of robustness against attacks and efficiency.

The introduction of the belief propagation and iterative message passing methods onto trust and reputation management has opened up several research directions. Thus, next, the first application of the belief propagation algorithm in the design of recommender systems (BPRS) is proposed. In BPRS, recommendations (predicted ratings) for each active user are iteratively computed by probabilistic message passing between variable and factor nodes in a factor graph. It is shown that as opposed to the previous recommender algorithms, BPRS does not require solving the recommendation problem for all users if it wishes to update the recommendations for only a single active user using the most recent data (ratings). Further, BPRS computes the recommendations for each user with linear complexity, without requiring a training period while it remains comparable to the state of art methods such as Correlation-based neighborhood model (CorNgbr) and Singular Value Decomposition (SVD) in terms of rating and precision accuracy.

This work also explores fundamental research problems related to application of iterative and probabilistic reputation management systems in various fields (such as ad-hoc networks and P2P systems). A distributed malicious node detection mechanism is proposed for delay tolerant networks (DTNs) using ITRM which enables every node to evaluate other nodes based on their past behavior, without requiring a central authority. Further, for the first time. the belief propagation algorithm is utilized in the design and evaluation of distributed trust and reputation management systems for P2P networks. Several schemes are extensively simulated and are compared to demonstrate the effectiveness of the iterative algorithms and belief propagation on these applications.

CHAPTER I

INTRODUCTION

Trust and Reputation are crucial requirements for most environments wherein entities participate in various transactions and protocols among each other. In most online service systems, the consumer of the service (e.g., the buyer) has no choice but to rely on the reputation of the service provider (e.g., the seller) based on the latter's prior performance. A reputation management mechanism is a promising method to protect the users against deceitful service providers. This mechanism lets a user to have some foresight about the service providers before using (or purchasing) their services. By using a reputation management scheme, an individual peer's reputation can be formed by the combination of received reports (ratings). Hence, after each transaction, a party who receives the service (referred to as the rater) provides (to the central authority) its report about the quality of the service provided for that transaction. The central authority collects the reports and updates the reputations of the service providers. The reputation mechanism, however, opens up new vulnerabilities as the raters may provide unreliable or malicious reports, demonizing the reputations of the service providers unfairly. Therefore, the main goal of a reputation mechanism is to determine the service qualities of the service providers and the trustworthiness of the raters based on their reports about the service qualities. Hence, the success of a reputation scheme depends on the robustness of the mechanism to accurately evaluate the reputations of the service providers and the trustworthiness of the raters.

Trust and reputation mechanisms have various application areas from online services [3–7] to Mobile Ad-hoc Networks (MANETs) [23, 24, 39, 96]. Most well-known commercial web sites such as eBay, Amazon, Netflix and Google use some types of reputation mechanisms. Another interesting potential application of trust and reputation management schemes is the peer review process of the journals/conferences to protect good papers against unfair reviews. Although generally considered essential to academic quality, peer review has been criticized as ineffective and sometimes unfair. A reputation mechanism can be implemented to prevent such issues.

Online auction and shopping web site, eBay, stresses the importance of reputation systems by saying that “feedback is an essential part of what makes eBay a successful community” [2]. As eBay spokesperson Lisa Malyon puts it: “As eBay has evolved, our feedback system has evolved. By relying on the feedback of other users, our buyers are able to make purchases based on confidence and trust. Sellers are no longer rated on the number of transactions; it is the service they offer for each individual transaction”. On the other hand, buyers and sellers seem to be engaged in a war of attrition where negative feedback is one of the main weapons, and now eBay has announced that sellers will no longer be able to leave negative feedback on buyers, hoping that this will help to rebalance things. Randy Farmer, author of the forthcoming book Building Web Reputation Systems (with Bryce Glass) states that a user-generated, negative public rating is simply too problematic to be left to the crowd. On the other hand, a reputation system cannot give accurate results without negative feedbacks. Hence, eBay admits that there is still a crucial need to improve the existing reputation systems. Indeed, eBay announced that it is considering changes to its feedback system that will be used soon [1].

According to a research conducted by the Pew Internet and American Life Project, A quarter of online Americans have taken advantage of one of the Internet’s true powers: the ability to let users collectively decide whether to trust a product, service or individual.“The more voices that are in the mix, the better off everybody is” said Lee Rainie, director of the Pew group [8]. Thus, an increasing number of commercial sites are following the lead long since set by Amazon.com, eBay and others that allow users to weigh in on the value or reputations of products or other users of the service. It is foreseeable that the social web is going to be driven by these reputation systems.

As in every system, trust and reputation management systems are also subject to malicious behaviors. Malicious raters may attack particular service providers in order to undermine their reputations while they help other service providers by boosting their reputations. Similarly, malicious service providers may provide good service qualities for certain customers in order to keep their reputations high while cheating the other customers. Moreover, malicious raters (or service providers) may collaboratively come up with sophisticated attacking strategies by exploiting their prior knowledge about the reputation mechanism. Hence, building a resilient trust and reputation management system that is robust against malicious activities becomes a challenging issue.

In summary, we believe that trust and reputation management systems will lead various applications from the social web to ad-hoc networks in near future and there are needs for scalable and attack resilient reputation systems. In this thesis, we first propose an iterative trust and reputation management algorithm and its application to Delay Tolerant Networks (DTNs). Then, we introduce the first application of the BP algorithm for trust and reputation management and study its application for Peer-to-Peer (P2P) networks. Finally, relying on the similarities between the reputation and recommender systems and successful application of iterative algorithms on reputation systems, we study the application of BP algorithm for recommender systems. We briefly summarize these works in following.

1.1 Iterative Trust and Reputation Management and Its Application to Delay Tolerant Networks (DTNs)

One of the primary objectives of this thesis is to develop a trust and reputation management scheme that not only provides immunity against malicious ratings but also discourages the service providers from any unfair and discriminating behaviors.

Our work on reputation systems stems from the prior success in the use of iterative algorithms, such as message passing techniques and Belief Propagation (BP) [83,110] in the decoding of Low-Density Parity-Check (LDPC) codes in erasure channels [75, 98]. These algorithms rely on graph-based representations of codes, where decoding can be viewed as message passing between the nodes in the graph. Moreover, they are shown to perform at error rates near what can be achieved by the optimal scheme, maximum likelihood decoding, while requiring far less computational complexity (i.e., linear in the length of the code). We believe that these significant benefits offered by iterative algorithms can be tapped in to benefit the field of reputation systems.

To achieve this, focusing mainly on centralized reputation management systems, we first introduce the "Iterative Trust and Reputation Mechanism" (ITRM) [11] in Chapter 3. ITRM is an algebraic trust and reputation management scheme inspired by the earlier work on iterative decoding of error-control codes in the presence of stopping sets [74,75,98]. In this work, we show the benefit of using iterative algorithms to detect and filter out unreliable ratings in a trust and reputation management system. Then, in Chapter 4, we explore the application of ITRM for Delay Tolerant Networks (DTNs).

DTNs are relatively new class of networks [38], wherein sparseness and delay are particularly high. In conventional MANETs, the existence of end-to-end paths via contemporaneous links is assumed in spite of node mobility. It is also assumed that if a path is disrupted due to mobility, the disruption is temporary and either the same path or an alternative one is restored very quickly. In contrast, DTNs are characterized by intermittent contacts between nodes, leading to space-time evolution of multihop paths (routes) for transmitting packets to the destination. In other words, DTNs' links on an end-to-end path do not exist contemporaneously, and hence intermediate nodes may need to store, carry, and wait for opportunities to transfer data packets towards their destinations. Hence, DTNs are much more general than

MANETs in the mobile network space (i.e., MANETs are special types of DTNs). Applications of DTNs include emergency response, wildlife surveying, vehicular-to-vehicular communications, healthcare, military, and tactical sensing.

Compared to traditional MANETs, common problems in packet communication such as routing, unicasting, broadcasting and multicasting become sufficiently harder in DTNs even with lossless links (i.e., no packet erasures due to communication link). This increase in difficulty can be directly attributed to the lack of knowledge on the network topology, and the lack of end-to-end paths. Hence, the schemes for routing packets have to be primitive such as forwarding to the next available node, injecting multiple copies into available nodes and employing erasure block codes [101]. On the other hand, depending upon the model for mobility, efficient communication schemes for stationary ad-hoc networks can be extended partially or wholly to DTNs.

As in MANETs, adversary may mount several threats against DTNs to reduce the performance of the network. The most serious attacks are due to the Byzantine (insider) adversary in which one or more legitimate nodes have been compromised and are fully controlled by the adversary. A Byzantine-malicious node may mount the following attacks in order to give serious damage to the network: 1. Packet drop, in which the malicious node drops legitimate packets to disrupt data availability, 2. Bogus packet injection, in which the Byzantine node injects bogus packets to consume the limited resources of the network, 3. Noise injection, in which the malicious node changes the integrity of legitimate packets, 4. Routing attacks, in which the adversary tempers with the routing by misleading the nodes, 5. Flooding attacks, in which the adversary keeps the communication channel busy to prevent legitimate traffic from reaching its destination, and 6. Impersonation attacks, in which the adversary impersonates the legitimate nodes to mislead the network. We note that because of the lack of end-to-end path from a source to its destination in DTNs, routing attacks are not significant threats for such networks. Attacks on packet integrity may be prevented using a robust authentication mechanism in both MANETs and DTNs. However, packet drop is harder to contain because nodes' cooperation is fundamental for the operation of these networks (i.e., a group of nodes cooperate in routing each others' packets using multihop wireless links without any centralized control). This cooperation can be undermined by Byzantine attackers, selfish nodes, or even innocent but faulty nodes. Therefore, we focus on packet drop attack which gives serious damages to the network in terms of data availability, latency, and throughput. Finally, Byzantine nodes may individually or in collaboration attack the security mechanism (e.g., the trust management and malicious node detection schemes).

In MANETs, reputation-based trust management systems are shown to be an effective way to cope with adversary. By establishing trust with the nodes it has or has not directly interacted, a node in the network diagnoses other nodes and predicts their future behavior in the network. Hence, trust plays a pivotal role for a node in choosing with which nodes it should cooperate, improving data availability in the network. Further, examining trust values has been shown to lead to the detection of malicious nodes in MANETs. Despite all the progress for securing MANETs, achieving the same for DTNs leads to additional challenges. The special constraints posed by DTNs make existing security protocols inefficient or impractical in such networks as will be discussed in Section 2.2.1. Thus, In Chapter 4, we propose a distributed malicious node detection mechanism for DTNs [12] using ITRM which enables every node to evaluate other nodes based on their past behavior, without requiring a central authority. Our results show that the resulting scheme effectively provides high data availability and low latency in the presence of Byzantine attackers. We also show that the proposed iterative mechanism is far more effective than some well-known reputation management techniques (e.g., *Bayesian framework* and *EigenTrust*) in detecting Byzantine nodes.

1.2 *Belief Propagation for Trust and Reputation Management and Its Application to Peer-to-Peer (P2P) Networks*

In Chapter 5, we expand our work in Chapter 3 and introduce the first application of the BP algorithm (a fully probabilistic and iterative algorithm), on centralized trust and reputation management systems. Different from our initial work in Chapter 3, in this work, we view the reputation management problem as an inference problem and describe it as computing marginal likelihood distributions from complicated global functions of many variables. Further, we utilize the BP algorithm to efficiently (in linear complexity) compute these marginal probability distributions. Thus, we introduce the "Belief Propagation-Based Iterative Trust and Reputation Management Scheme" (BP-ITRM). We show the efficiency and robustness of BP-ITRM both via analysis and extensive simulations in a centralized setting.

In a distributed infrastructure, trust and reputation management is more complicated than in centralized solutions. Hence, in Chapter 6, we focus on P2P networks and explore the application of BP-based trust and reputation management algorithms in a completely decentralized environment in the presence of malicious peers mounting attacks. Peer-to-peer (P2P) networks 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), in P2P networks, every peer plays the role of both a client and a server. In other words, each peer provides access to its resources (e.g., processing power or storage) as a server without the need for a central authority. P2P networks 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 clients get to rate the servers based on the quality of the transactions. A trust and reputation management mechanism is a promising method to protect the client by forming some foresight about the servers 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. As we discussed before, trust and reputation management systems are also subject to malicious behaviors. Malicious peers may attack the system to undermine (or boost) the reputation values of certain peers. Hence, building a resilient trust and reputation management system that is robust against malicious activities in a decentralized environment 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. Thus, in Chapter 6, for the first time, we utilize the BP algorithm in the design and evaluation of distributed trust and reputation management systems for P2P networks. We introduce the "Belief Propagation-Based Trust and Reputation Management for P2P Networks" (BP-P2P). We show via analysis and simulations that BP-P2P is resilient against attacks in a distributed environment. Further, we 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.

1.3 Application of Belief Propagation for Recommender Systems

Relying on our success in the reputation management problem, in Chapter 7, we extend the BP-based technique to arrive at scalable, accurate and robust recommender systems. Today, the quantity of available information grows rapidly, overwhelming consumers to discover useful information and filter out the irrelevant items. The explosive growth of the Internet has made this issue increasingly more serious. Thus, the user is confronted with a big challenge of finding the most relevant information or item in the short amount of time. Without some support, the process of filtering out irrelevant items and finally selecting the most appropriate one could be very difficult. Recommender systems are aimed at addressing this overload problem, suggesting to the users those items that meet their interests and preferences the best in a particular situation and context. These systems are used to direct users towards items they will like while interacting with large information spaces. More generally, recommender systems can learn about user preferences and profile over time, based on data mining algorithms, and automatically suggest products (from a large space of possible options) that fit the user needs.

Currently, recommender systems are used in a variety of application domains, e.g., books, movies, and music. Most well-known commercial web sites such as eBay, Amazon and Netflix use some types of recommender systems. Further, recommender systems have applications in advertisements; which is a successful source of income for Google and social networking web sites. By finding similarity among people's choices, recommender systems can be used for the customer directed advertising in which users are directed toward those items that meet their needs and preferences the best. Hence, it is foreseeable that the social web is going to be driven by these recommender systems.

The online movie rental service, Netflix, emphasizes the importance of recommender systems and the need to improve them by saying that "if there is a much better approach, it could make a big difference to our customers and our business" [7]. Indeed, in September 2009, the company awarded a $1 million prize to a team of engineers, statisticians and researchers that improved the accuracy of its movie recommendation system by 10%. "Personalized recommendations," says Brent Smith, Amazon's director of personalization, "are at the heart of why online shopping offers so much promise".

However, there are certain challenges to design accurate and scalable recommender systems. On one hand, unfortunately, recommender systems have to operate on incomplete profiles because users either do not like to disclose lots of personal information and preferences, and/or are not completely aware about their preferences. On the other hand, with the rapid growth of information flow, an increasing number of applications require recommender systems to make predictions without full knowledge of the problem they are trying to solve. The available data for the recommender systems is incomplete, uncertain, inconsistent and/or intentionally-contaminated. Challenges of this sort underlie the prediction problem in electronic commerce (where relevant information is hidden by parties who may have an incentive to misreport it), and online services (where the quality of predictions in the present depends on information revealed only in the future). Hence, new research needed to focus on algorithms which meet these challenges in the face of such uncertainty and yet maintain computational efficiency.

The two main collaborative filtering approaches that emerged as victorious from the Netflix Prize [7] are neighborhood methods and latent factor models. Neighborhood methods use similarity functions such as the Pearson Correlation or Cosine Distance to compute sets of neighbors to a user or an item. Recommendations are then computed by using data from those neighbors. On the other hand, latent factor models such as Matrix Factorization [93] solve the recommendation problem by decomposing the user-item matrix and learning latent factors for each user and item. The underlying assumption is that both users and items can be modeled by a reduced number of factors. This approach has proven to be the most accurate method in the Root Mean Square Error (RMSE) sense. However, most existing and highly popular recommender systems are shown to be prone to malicious behavior [29, 97] and they have scalability issues. In other words, they fall short of incorporating the attack profiles and the extra noise generated by the malicious users. Further, each new update (using the most recent data or ratings) for a particular active user requires to solve the entire problem for every user in the system, making it unattractive for large scale systems. On the other hand, Matrix Factorization methods are optimized to minimize RMSE. However, it is widely argued that RMSE cannot serve as a good proxy for usage precision accuracy [31]. Therefore, there is yet a need to develop scalable and dependable schemes. Thus, we formulate the recommender system problem as finding the marginal probability distributions of the unknown variables on a factor graph and we introduce the "Belief Propagation-Based Iterative Recommender System" (BPRS) in Chapter 7. We show that BPRS computes the recommendations for each user instantaneously (with linear complexity) using the most recent data and without requiring a training period. Further, we show that BPRS also provides comparable usage prediction and rating prediction accuracy to other popular methods such as Correlation-based neighborhood model (CorNgbr) [18] and Singular Value Decomposition (SVD) [102].

Finally, Chapter 8 summarizes the completed work and points out some of the possible future research directions.

CHAPTER II

BACKGROUND

In this chapter, we review the work related to the trust and reputation management schemes, security of Delay Tolerant Networks (DTNs), trust and reputation management in Peer-to-peer (P2P) networks, and recommender systems.

2.1 Trust and Reputation Management for Online Service Provision

We may classify reputation mechanisms for centralized systems as i) global reputation systems, where the reputation of a service provider is based on the ratings from general users [25,68], and ii) personalized reputation systems (i.e., recommender systems), where the reputation of a service provider is determined based on the ratings of a group of particular users, which may be different in the eyes of different users [35,108] (personalized reputation systems will be discussed in Section 2.4). The most famous and primitive global reputation system is the one that is used in eBay. Other well-known web sites such as Amazon, Epinions, and AllExperts use a more advanced reputation mechanism than eBay. Use of the *Bayesian Approach* is also proposed in [25,103]. Finally, [35] proposed to use the *Cluster Filtering* method [64] for reputation management. We briefly review these schemes in the following.

2.1.1 Commercial and Live Reputation Systems

The most famous and primitive global reputation system is the one that is used in eBay [5]. In eBay, after each transaction, sellers and buyers rate each other with the ratings 1 (positive), 0 (neutral) or -1 (negative), and the total rating of a peer is the sum of the individual ratings it received from the other peers. To provide information about the recent behavior of a peer, ratings about the past 6 months, ratings about the past month and the ratings about the past 7 days are kept separately. It is shown in [78] that, even this simple reputation mechanism provides the sellers with high reputation to sell their items more than the other sellers. However, there are a few major problems about the reputation scheme of eBay as well. Since all individual ratings are weighted equally, the unfair ratings (the ones coming from the malicious peers) are not filtered, and hence, they effect the reputation values of the sellers significantly. Moreover, since each peer initially starts with a reputation of 0, any peer with a negative reputation value may sign in to the system again with a new ID to increase its reputation value to 0. Another problem about the eBay is that since peers can see the ratings of each other in a transaction, peers give good ratings to each other most of the time which causes the reputation values to increase for each peer. EBay charges each seller a fee for selling an item to prevent fake transactions between peers (to avoid the collaboration of the peers to increase each others reputations). However, the loss due to the fake transactions can be compensated by the gain after having a high reputation value.

The well known online shopping site Amazon also uses a reputation system to rate its products [4]. Members give ratings to the products between 1 and 5, and each member is treated equally for their ratings as in eBay. The difference of Amazon from eBay is that users can also vote on the reviews of the other users on the products. This mechanism determines each reviewers rank as a function of the helpful votes it received.

Epinions [6], is a product review site in which users can rate and review items. Similar to Amazon, users can also give ratings to the reviews. Hence, the ratings of members on a review and on a product are considered separately. As a result of this, users are classified based on the quality of their ratings. In Epinions, users are motivated to write high quality reviews to the products by getting paid, and authors of more useful reviews earn more than the others.

An expert site, AllExperts [3], also uses a global centralized reputation system to rate the experts who provide service by answering the questions of the users. In [3], depending on the quality of the reply, the user who asked the question rates the expert on various aspects with a rating from 1 to 10. The average rating of an expert is basically the average of the individual ratings it receives from the users to whom it provided a service. The number of questions an expert answered is also displayed along with its average rating. However, there is no security mechanism against malicious peers and unfair ratings in [3], which makes this system vulnerable to malicious activities.

It is worth noting that the above reputation management mechanisms compute the average (or weighted average) of the ratings received for a product (or a peer) to evaluate the global reputation of a product (or a peer). Hence, these schemes are vulnerable to collaborative attacks by malicious peers.

2.1.2 Bayesian Approach

In Bayesian reputation systems [25,103], the *a posteriori* reputation value of a peer is computed combining its *a priory* reputation value with the new ratings received for that peer. The reputation of a peer is represented in the form of Beta PDF parameter tuple (α, β) (amount of positive and negative feedbacks) or the expectation value of the Beta PDF. When nothing is known, the *a priori* distribution is the uniform Beta PDF with $\alpha = 1$ and $\beta = 1$. Then, after observing r positive and s negative outcomes, the *a posteriori* distribution is the Beta PDF with $\alpha = r+1$ and $\beta = s+1$. A PDF of this type expresses the uncertain probability that future interactions will be positive. Further, the reputation score is commonly defined in the form of the probability expectation value of the Beta PDF. Moreover, Bayesian reputation systems use a threshold method to determine and update the report reliability (reliability of the ratings) of the peers. The server checks whether the received ratings are within a definite interval. Peers whose ratings lie within the interval are considered to be honest. Hence, reliability of a peer increases when the peer reports a reliable rating but decreases otherwise.

Since we present and evaluate our proposed trust and reputation management frameworks (ITRM and BP-ITRM) in a centralized setting, the most well-known Bayesian Approaches in Buchegger's work [25] and Whitby's work [103] can be considered as similar. In [25], if a rater's rating is deviated more than the deviation threshold d from the calculated reputation value, its trustworthiness value is updated accordingly. Further, if a rater's trustworthiness exceeds a definite threshold t, it is detected as malicious. Similarly, in [103], instead of using the deviation threshold, the authors check if the calculated reputation value for the service provider falls between a definite interval for each rater's rating distribution. Furthermore, we identify that both [25] and [103] have the same shortcoming against colluding malicious raters; both [25] and [103] first calculate the reputation value of a particular service provider, and then based on the calculated value, they adjust each rater's trustworthiness value. On the other hand, when the malicious raters collude, it is likely that the majority of the ratings to the victim service providers will be from malicious raters. In this scenario, the Bayesian Approach not only fails to filter the malicious ratings but it also punishes the honest raters which rates the victim service providers.

2.1.3 Cluster Filtering

Cluster Filtering [35] performs a dissimilarity test among the raters and then updates the reputation values of the peers using only the honest raters. Cluster Filtering [35] introduces a mechanism of controlled anonymity to avoid unfair ratings from malicious raters. To reduce the effect of unfair ratings, the authors first use collaborative filtering techniques [40, 80] to determine a neighborhood group of rater peers whose ratings over many subjects are similar. They then propose the Cluster Filtering approach [64] to filter out the unfair ratings. The idea of this approach is to apply a divisive clustering algorithm to separate the ratings into two clusters, the lower rating cluster and the higher rating cluster. Ratings in the lower rating cluster are considered as fair ratings. Ratings in the higher rating cluster are considered as unfair ratings, and therefore are excluded or discounted. To deal with the situation where ratings vary over time, the Cluster Filtering approach considers only the ratings within the most recent time window whose width is influenced by the frequency of fair ratings.

Different from these existing schemes, our proposed algorithms (ITRM [11] and BP-ITRM [16]) are graph based iterative algorithms motivated by the previous success on message passing techniques and belief propagation algorithms.

2.2 Security for Delay Tolerant Networks (DTNs)

Several works in the literature have focused on securing DTNs. In [88], the challenges of providing secure communication (i.e., confidentiality) in DTNs is discussed and the use of Identity-Based Cryptography (IBC) [32] is suggested. In [55], source authentication and anonymous communication as well as message confidentiality are provided using IBC. In [26], the use of packet replication is proposed to improve message delivery rate instead of using cryptographic techniques. We note that the existing techniques to secure DTNs are aimed to provide data confidentiality and authentication only. On the other hand, our proposed trust-based scheme provides malicious node detection and high data availability with low packet latency in the presence of Byzantine attacks. In MANETs, reputation-based trust management systems are shown to be an effective way to cope with adversary. In the following we discuss these systems and their impracticality for DTNs.

2.2.1 Trust Management in Mobile Ad-hoc Networks (MANETs)

The main goal for building a reputation system in MANETs is to protect the reactive routing protocol from attackers and increase the performance of the network. A recent review of these secure routing protocols for MANETs [73] indicates that these protocols either use the watchdog mechanism or Acknowledgement (ACK) messages to build trust values between the nodes. In MANETs, a node evaluates another by using either direct or indirect measurements. Building reputation values by direct measurement is either achieved by using the watchdog mechanism or by using the ACK from the destination. Building reputation values by just relying on the direct measurements and using the watchdog mechanism is proposed in [65, 69]. These schemes rely on monitoring the neighbor node to detect possible misbehavior. In other words, once a node forwards its packets to a specific node, it monitors the node by overhearing its transmission. Hence, a malicious node is detected by its neighbor when it drops a packet or changes the integrity of a packet. In [23, 24], the use of indirect measurements to build reputation values is also allowed while the watchdog mechanism is used to obtain the direct measurements. In these schemes, a node uses the reputation values that are established by some other node along with its own direct measurements. In [13, 15, 37, 61, 107], reputation values are constructed using the ACK messages sent by the destination node. In other words, a source node, which has established a path to its destination, would blame a path with a negative ACK and would attempt to use a path with higher credentials to increase the efficiency.

We note that these techniques are not applicable to DTNs due to the following reasons. In DTNs, a node cannot use the watchdog mechanism and monitor another intermediate node after forwarding its packets to it. This is because links on an end-to-end path do not exist contemporaneously, and hence an intermediate node needs to store, carry and wait for opportunities to transfer those packets. As a result, the node loses connection with the intermediate node which it desires to monitor. This implies that a Byzantine node in DTNs can get packets from a legitimate node, then move away and drop the packets. Similarly, relying on the ACK packets from the destination to establish reputation values would fail in DTNs because of the lack of a fixed common multihop path from the source to the destination. Even if we assume an ACK from destination to the source (which incurs large latency), this feedback packet travels to the source via intermediate nodes that are different from the set of nodes that delivered the data packet to the destination. More specifically, the source node, upon receiving a negative ACK, cannot decide which node on the forwarding path is to be blamed. Lastly, using indirect measurements is possible in DTNs. However, it is unclear as to how these measurements can be obtained in the first place.

2.3 Trust and Reputation Management in Peer-to-Peer (P2P) Networks

Trust and reputation management systems for P2P networks received a lot of attention [9, 30, 33, 43, 52, 53, 79]. In [79] and [52], authors cover most 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 [9] 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 [30, 33].

EigenTrust [53] is one of the most popular reputation management algorithms for P2P networks. In EigenTrust algorithm, each peer i rates another peer j by rating each downloaded file (from peer j) either as positive (if the downloaded file is authentic) or negative (if the downloaded file is fake). Each peer maintains a sum of all his transactions with other peers in a local trust vector. Then, the local trust values are aggregated around the network and normalized so that malicious peers will not be able to assign arbitrarily high trust values to other malicious peers. This normalization ensures that all trust values will lie between 0 and 1. Global reputation of each peer i is computed from the local trust values assigned to peer i by other peers. These local trust values are weighted by global reputations of assigning peers. This process iteratively continues until the global reputation values converge (the change in global reputation values drops below a threshold). 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. A malicious peer can attack the trust and reputation management system while providing a high quality service. Further, the EigenTrust algorithm relies on the presence of pre-trusted peers in the network which is not practical 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 [23] (Bayesian framework is discussed in Section 2.1.2). In schemes utilizing 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. The strength of our proposed approach (BP-P2P) stems from the fact that it tries to capture this correlation in analyzing the ratings and computing the trust and reputation values.

2.4 Recommender Systems

Techniques to build recommender systems [10, 81, 86, 87] can be classified into two main categories: i) content-based filtering [17, 46] in which the system uses behavioral data about a user to recommend items similar to those previously consumed by the user, and ii) collaborative filtering [40, 80] in which the system compares one user's behavior against the other users' behaviors and identifies items which were preferred by similar users. There are also hybrid methods combining these two techniques [27]. Collaborative filtering algorithms also fall into two general classes: i) memory-based algorithms [21, 45, 77] in which the value of an unknown rating is computed by aggregating the ratings of some other users for the same item, and ii) model-based algorithms [28, 48, 49, 51, 56–58, 67] in which the system uses the collection of the ratings to learn a model that is then used to make rating predictions. Methods that combine both memory-based and model-based algorithms are also suggested [71]. Memory-based algorithms are further classified into user-based [47, 62], item-based [36, 54, 84], and hybrid methods [100]. On the other hand, model-based algorithms include clustering methods [67], probabilistic methods [49], methods exploiting Singular Value Decomposition (SVD), Principal Component Analysis (PCA) and Maximum Margin Matrix Factorization (MMMF) techniques [41, 85, 93, 102].

The application of Bayesian networks and message passing algorithms for recommender systems is also studied in the past [34, 94]. In [94], the message passing technique is used to determine the latent factors of the users and items (as an alternative to SVD). In [34], because of the fuzziness associated with the ambiguity in the description of the ratings, a (non-iterative) inference is proposed among the users to remove this ambiguity. The key difference between our proposed approach and the other message passing-based methods is that, we describe the recommendation problem as computing marginal likelihood distributions from complicated global functions of many variables. To solve this problem whose complexity grows exponentially, we resort to the Belief Propagation (BP) algorithm whose computational efficiency (i.e., linear in the number of users) is driven by exploring the way in which the global functions factors into a product of simpler local functions. Inspired by successful applications of BP algorithms in various fields such as decoding of error correcting codes [59, 60, 66, 104], Artificial Intelligence [70], and reputation systems [16], we develop a new accurate and scalable recommender system.

CHAPTER III

ITERATIVE TRUST AND REPUTATION MANAGEMENT MECHANISM

3.1 Introduction

As we discussed in Chapter 1, trust and reputation are crucial requirements for most environments wherein entities participate in various transactions and protocols among each other. On the other hand, trust and reputation management systems are subject to various malicious behaviors. Hence, there is yet a need to develop reliable, scalable and dependable reputation management schemes that would also be resilient to various ways a reputation management system can be attacked. Focusing mainly on centralized reputation systems, the ultimate objective of this chapter is to develop a trust and reputation management scheme that not only provides immunity against malicious ratings but also discourages the service providers from any unfair and discriminating behaviors. To achieve this, we propose an algebraic iterative algorithm referred as "Iterative Trust and Reputation Mechanism" (ITRM). As in every trust and reputation management mechanism, we have two main goals: 1. Computing the service quality (reputation) of the peers who provide a service (henceforth referred to as Service Providers or SPs) by using the feedbacks from the peers who used the service (referred to as the raters), and 2. Determining the trustworthiness of the raters by analyzing their feedback about SPs. We consider the following major attacks that are common for any trust and reputation management mechanisms: i) Bad-mouthing, in which malicious raters collude and attack the SPs with the highest reputation by giving low ratings in order to undermine them, and ii) Ballot-stuffing, in which malicious raters collude to increase the reputation values of peers with low reputations. Further, we evaluate ITRM against some sophisticated attacks (which utilizes bad-mouthing or ballot-stuffing with a strategy) such as RepTrap [105] or the one in which malicious raters provide both reliable and malicious ratings to mislead the algorithm.

Our proposed iterative algorithm is inspired by the earlier work on the improved iterative decoding algorithm of LDPC codes in the presence of stopping sets [75, 98]. In iterative decoding of LDPC, every check-vertex (in the graph representation of the code) has some opinion of what the value of each bit-vertex should be. The iterative decoding algorithm would then analyze the collection of these opinions to decide, at each iteration, what value to assign for the bit-vertex under examination. Once the values of the bit-vertices are estimated, in the next iteration, those values are used to determine the satisfaction of the check-vertex values. The novelty of this work stems from the observation that a similar approach can be adapted to determine SPs' reputation values as well as the trustworthiness of the raters. Furthermore, the analysis of reputation systems resembles that of the code design problem. In LDPC, one of the goals is to find the decoding error for the a fixed set of check constraints. Similarly, in ITRM, our goal is to specify the regions of trust for the set of the system parameters. A region of trust is the range of parameters for which we can confidently determine the reputation values within a given error bound. We acknowledge, however, that we have a harder problem in the case of reputation systems as the adversary dynamics is far more complicated to analyze than the channel noise in the coding problem.

3.1.1 Contributions

The main strengths of the ITRM scheme are summarized in the following.

1. The proposed algorithm computes the reputations of the service providers accurately (with a small error) in a short amount of time in the presence of attackers.

2. ITRM is a robust and efficient methodology for detecting and filtering out

unreliable ratings (from malicious raters) in a short amount of time.

3. ITRM detects the malicious raters with a high accuracy, and updates their trustworthiness accordingly. Hence, ITRM enforces the malicious raters to execute low grade attacks in order to remain undercover.

4. The proposed ITRM algorithm has a computational complexity that is linear with the number of raters. Hence, ITRM is scalable and suitable for large scale implementations.

3.2 Iterative Trust and Reputation Management Mechanism (ITRM)

Let TR_j be the global reputation of the j^{th} SP. Further, TR_{ij} represents the rating that the peer i reports about the SP j, whenever a transaction is completed between the two peers. Moreover, R_i denotes the (report/rating) trustworthiness of the i^{th} peer as a rater[1]. The first step in developing ITRM is to interpret the collection of the raters and the SPs together with their associated relations as a bipartite graph, as in Fig. 1(a). In this representation, each rater corresponds to a *check vertex* in the graph, shown as a square and each SP is represented by a *bit vertex* shown as a hexagon in the graph. If a rater i has a rating about the j^{th} SP, we place an edge with value TR_{ij} from the i^{th} check-vertex to the j^{th} bit-vertex. As time passes, we use the age-factored values as the edge values instead. To each edge $\{ij\}$, a value $WR_{ij} = w_{ij}TR_{ij}$ is assigned, where WR_{ij} is the age-factored TR_{ij} value. The factor $w_{ij}(t)$ is used to incorporate the time-varying aspect of the reputation of the SPs (i.e., time-varying service quality). We use a known factor $w_{ij}(t) = \hat{\lambda}^{t-t_{ij}}$ where $\hat{\lambda}$ and t_{ij} are the fading parameter and the time when the last transaction between the rater i and the SP j occurred, respectively. If a new rating arrives from the i^{th} rater about

[1]All of these parameters (TR_j, TR_{ij} and R_i) may evolve with time. However, for simplicity, we omitted time dependencies from the notation.

the j^{th} SP, our scheme updates the new value of the edge $\{ij\}$ by averaging the new rating and the old value of the edge multiplied with the fading factor.

We consider slotted time throughout this discussion. At each time-slot, ITRM will be executed using the input parameters R_i and WR_{ij} to obtain the reputation parameters (e.g., TR_j) and the list of malicious raters (referred to as the blacklist). Initially, the blacklist is set empty. Details of ITRM may be described by the following procedure at the L^{th} time-slot. Let R_i and TR_{ij} be the parameter values prior to the present execution (the L^{th} execution) of ITRM algorithm. Let also TR_j^{ν} and TR_{ij}^{ν} be the values of the bit-vertex and the $\{ij\}^{th}$ edge at the iteration ν of the ITRM algorithm. Prior to the start of the iteration ($\nu = 0$), we set $TR_{ij}^{\nu=0} = TR_{ij}$ and compute the initial value of each bit-vertex (referred to as the initial guess $TR_j^{\nu=0}$) based on the weighted average of the age-factored edge values (WR_{ij}^{ν}) of all the edges incident to the bit-vertex j. Equivalently, we compute

$$TR_j^{\nu} = \frac{\sum\limits_{i \in A_j} R_i \times WR_{ij}^{\nu}}{\sum\limits_{i \in A_j} R_i \times w_{ij}(t)}, \tag{1}$$

where A_j is the set of all check-vertices connected to the bit-vertex j. It is interesting to note that the initial guess-values resemble the received information from the channel in the channel coding problem. Then, the first iteration starts (i.e., $\nu = 1$). We first compute the average inconsistency factor C_i^{ν} of each check-vertex i using the values of the bit-vertices (i.e., $TR_j^{\nu-1}$) for which it is connected to. That is, we compute

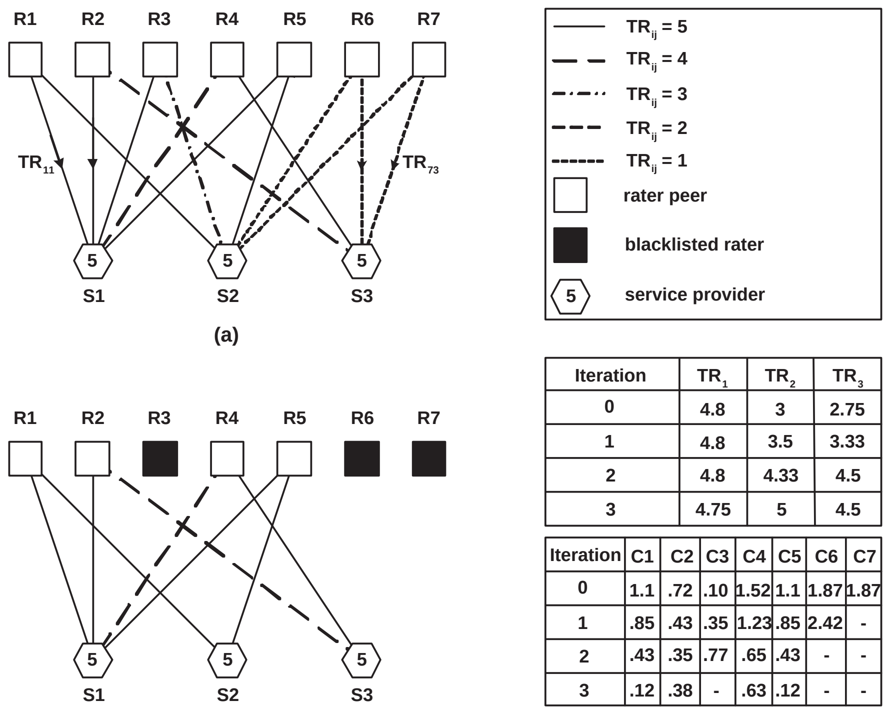
$$C_i^{\nu} = \frac{1}{\sum\limits_{j \in B_i} \hat{\lambda}^{t-t_{ij}}} \sum_{j \in B_i} d(TR_{ij}^{\nu-1}, TR_j^{\nu-1}), \tag{2}$$

where B_i is the set of bit vertices connected to the check-vertex i and $d(\cdot,\cdot)$ is a distance metric used to measure the inconsistency. We use the $\mathcal{L}^1$ norm (absolute value) as the distance metric, and hence,

$$d(TR_{ij}^{\nu-1}, TR_j^{\nu-1}) = |TR_{ij}^{\nu-1} - TR_j^{\nu-1}|\hat{\lambda}^{t-t_{ij}}. \tag{3}$$

After computing the inconsistency factor for every check-vertex, we list them is ascending order. Then, the check-vertex i with the highest inconsistency is selected and placed in the blacklist if its inconsistency is greater than or equal to a definite threshold τ (whose choice will be discussed later). If there is no check-vertex with inconsistency greater than or equal to τ, the algorithm stops its iterations. Once the check-vertex i is blacklisted, we delete its rating TR_{ij}^{ν} for all the bit-vertices j it is connected to. Then, we update the values of all the bit-vertices using (1). This completes the first iteration of ITRM. The iterative algorithm proceeds to other iterations exactly in the same way as the first iteration, updating the values of the bit-vertices and blacklisting some other check-vertices as a result. However, once a check-vertex is placed in the blacklist, for the remaining iterations it is neither used for the evaluation of TR_j values nor for the inconsistency measure of the check-vertices. We stop the iterations when the inconsistencies of all the check-vertices (excluding the ones already placed in the blacklist) fall below τ.

As an example, ITRM is illustrated in Fig. 1 for 7 raters, 3 SPs, and $\tau = 0.7$. It is assumed that the rates are integer values from $\{1, \ldots, 5\}$ and the actual reputations of the SPs, $T\hat{R}_j$, are equal to 5. For simplicity, we assumed w_i's to be equal to 1 and R_i's to be equal for all raters. Furthermore, we assumed that the peers 1, 2, 3, 4, and 5 are honest but 6 and 7 are malicious raters. The malicious raters (6 and 7) mount the bad-mouthing attack in this example by rating the SPs with $T\hat{R}_j = 5$ as 1 (to degrade their reputations). Fig. 1(a) shows the TR_{ij} values (illustrated by different line-styles) prior to the execution of ITRM. The TR_j values and the individual inconsistencies of the raters after each iteration are also illustrated in Fig. 1(c). We note that the algorithm stops at the third iteration when all the raters have inconsistencies less than τ. Fig. 1(c) indicates how ITRM gives better estimates of TR_j's compared to the weighted averaging method (which is correspond to the zero iteration). Fig. 1(b) illustrates the edges after the final iteration of ITRM. It is worth noting that the malicious raters 6 and 7 are blacklisted and their ratings are accordingly deleted. Moreover, rater 3, although honest, is also blacklisted at the third iteration. We note that this situation is possible when an honest but faulty rater's rating have a large deviation from the other honest raters.



(a)

Iteration	TR_1	TR_2	TR_3
0	4.8	3	2.75
1	4.8	3.5	3.33
2	4.8	4.33	4.5
3	4.75	5	4.5

Iteration	C1	C2	C3	C4	C5	C6	C7
0	1.1	.72	.10	1.52	1.1	1.87	1.87
1	.85	.43	.35	1.23	.85	2.42	-
2	.43	.35	.77	.65	.43	-	-
3	.12	.38	-	.63	.12	-	-

(b)

(c)

Figure 1: Illustrative example of ITRM.

3.2.1 Managing Raters' Trustworthiness

We update the R_i values using the set of all past blacklists together in a Beta distribution [25]. Initially, prior to the first time-slot, for each rater-peer i, the R_i value is set to 0.5 ($\phi_i = 1$ and $\varphi_i = 1$). Then, if the rater-peer i is blacklisted, R_i is decreased by setting

$$\varphi_i(t+1) = \bar{\lambda}\varphi_i(t) + (C_i + 1 - \tau)^{\delta}, \tag{4}$$

otherwise, R_i is increased by setting

$$\phi_i(t+1) = \bar{\lambda}\phi_i(t) + 1, \tag{5}$$

where $\bar{\lambda}$ is the fading parameter and δ denotes the penalty factor for the blacklisted raters. We note that updating R_i values via the Beta distribution has one major disadvantage. An existing malicious rater with low R_i could cancel its account and sign in with a new ID (*whitewashing*). This problem may be prevented by updating R_i's using the method proposed in [108].

3.3 Security Evaluation of ITRM via User Modeling

In order to facilitate future references, frequently used notations are listed in Table 1.

Table 1: Notations and definitions.

D	Number of malicious raters
H	Number of honest raters
N	Number of service providers
W	$D/(D+H)$ (i.e., fraction of malicious raters)
m	Rating given by an honest rater
n	Rating given by a malicious rater
X	Total number of malicious ratings TR_{ij} received by a victim SP
d	Total number of newly generated ratings, per time-slot, by an honest rater
b	Total number of newly generated ratings, per time-slot, by a malicious rater
$\hat{b}$	Total number of newly generated attacking/malicious ratings, per time-slot, by a malicious rater
Δ	$\hat{b}/b$ (i.e., fraction of attacking ratings per time-slot)
μ	Total number of un-attacked SPs rated by an honest rater

3.3.1 Analytic Evaluation

We adopted the following models for various peers involved in the reputation system. We assumed that the quality of SPs remains unchanged during time-slots. We provided the evaluation for the bad-mouthing attack only, as similar results hold for ballot-stuffing and combinations of bad-mouthing and ballot-stuffing. We let $\hat{TR}_j$ be the actual reputation value of the j^{th} SP. Ratings (i.e., TR_{ij}) generated by the non-malicious raters are distributed uniformly among the SPs. We further assumed that m is a random variable with folded normal distribution (mean $T\hat{R}_j$ and variance 0.5), however, it takes only discrete values from 1 to 5. Furthermore, the R_i values for all the raters are set to the highest value (i.e., $R_i = 1$) for simplicity (which reflects the worst case). Finally, we assumed that d is a random variable with Yule-Simon distribution, which resembles the power-law distribution used in modeling online systems, with the probability mass function $f_d(d;\rho) = \rho B(d, \rho+1)$, where $B(\cdot,\cdot)$ is the Beta function. For modeling the adversary, we made the following assumptions. We assumed that the malicious raters initiate bad-mouthing and collude while attacking the SPs. Further, the malicious raters attack the same set Γ of SPs at each time-slot. In other words, Γ represents a set of size $\hat{b}$ in which each SP has an incoming edge from all malicious raters. The following discussions are developed for the time-slot t.

τ-eliminate-optimal Scheme: We declare a reputation scheme to be τ-eliminate-optimal if it can eliminate all the malicious raters whose inconsistency (measured from actual reputation values $T\hat{R}_j$ of SPs) exceeds the threshold τ. Hence, such a scheme would compute the reputations of the SPs by just using the honest raters. Naturally, we need to answer the following question: For a fixed τ, what are the conditions to have a τ-eliminate-optimal scheme? The conditions for ITRM to be a τ-eliminate-optimal scheme are given by the following lemma:

Lemma 3.3.1. *Let Θ_j and d_t be the number of unique raters for the j^{th} SP and the total number of outgoing edges from an honest rater in t elapsed time-slots, respectively. Let also Q be a random variable denoting the exponent of the fading parameter $\hat{\lambda}$ at the t^{th} time-slot. Then ITRM would be a τ-eliminate-optimal scheme if the conditions*

$$\sum_{r\in\Lambda} \Psi_r \geq (\hat{b}m + b\tau) \tag{6a}$$

and

$$\frac{\mu}{d_t} > 1 - \frac{\Theta\hat{\lambda}^Q\Delta}{D} \tag{6b}$$

are satisfied at the t^{th} *time-slot, where*

$$\Psi_r = \frac{mX + n\Theta_r\hat{\lambda}^Q}{X + \Theta_r\hat{\lambda}^Q} \textit{ for } r \in \Lambda, \tag{7}$$

and Λ *is the index set of the set* Γ.

Proof. At each iteration, ITRM blacklists the rater i with the highest inconsistency C_i if $C_i \geq \tau$. Each malicious rater has $\hat{b}$ attacking ratings at each time slot. Moreover, the inconsistency of a malicious rater due to each of its attacking edge j is $\left(\frac{mX+n\Theta_j\hat{\lambda}^Q}{X+\Theta_j\hat{\lambda}^Q} - m\right)$, where $j \in \Gamma$. Therefore, the total inconsistency of a malicious rater (which is calculated considering both its attacking and non-attacking ratings) should be greater than or equal to τ to be blacklisted. This results the condition in (6a). Further, given $C_i \geq \tau$ for a malicious rater i, to have a τ-eliminate-optimal scheme, we require that the inconsistency of the malicious rater with the highest inconsistency exceeds the inconsistencies of all the honest raters so that the blacklisted rater can be a malicious one in all iterations. To make sure ITRM blacklists all malicious raters, the inconsistency of a malicious rater must be greater than the inconsistency of an honest rater at the 0^{th} iteration with a high probability. The inconsistency of a malicious rater at the t^{th} time slot is given by

$$\left(|\frac{mX + nc\lambda^Q}{X + c\lambda^Q} - m|\right)\Delta. \tag{8}$$

Similarly, the inconsistency of an honest rater at the t^{th} time slot is

$$\left(|\frac{mX + nc\lambda^Q}{X + c\lambda^Q} - n|\right)\frac{d_t - \mu}{d_t}. \tag{9}$$

Hence, to blacklist a malicious rater, we require the term in (8) be greater than that of (9) which leads to (6b). □

The design parameter τ should be selected based on the highest fraction of malicious raters to be tolerated. To determine the optimal value of τ, we start with Lemma 3.3.1. We use a waiting time t such that (6a) and (6b) are satisfied with high probability (given the highest fraction of malicious raters to be tolerated). Then, among all τ values that satisfy (6a) and (6b) with high probability, we select the highest τ value. The intention for selecting the highest τ value is to minimize the probability of blacklisting an honest rater. In the following example, we designed the scheme to tolerate up to $W = 0.30$ (i.e., 30% malicious raters). For the given parameters $D + H = 200$, $N = 100$, $\Delta = 1$, $\rho = 1$ and $\hat{\lambda} = 0.9$, we obtained the optimal $\tau = 0.4$. In Fig. 2, we illustrate the waiting time for ITRM to be τ-eliminate-optimal for different fractions of malicious raters. As shown in Fig. 2, for W lower than 0.30, the waiting time becomes shorter to have a τ-eliminate-optimal scheme for $\tau = 0.4$. However, the scheme may also blacklist a few non-malicious raters in addition to the malicious ones when W is actually less than 0.30. This is because the optimal value of τ is higher for a τ-eliminate-optimal scheme when W is actually less than 0.30.

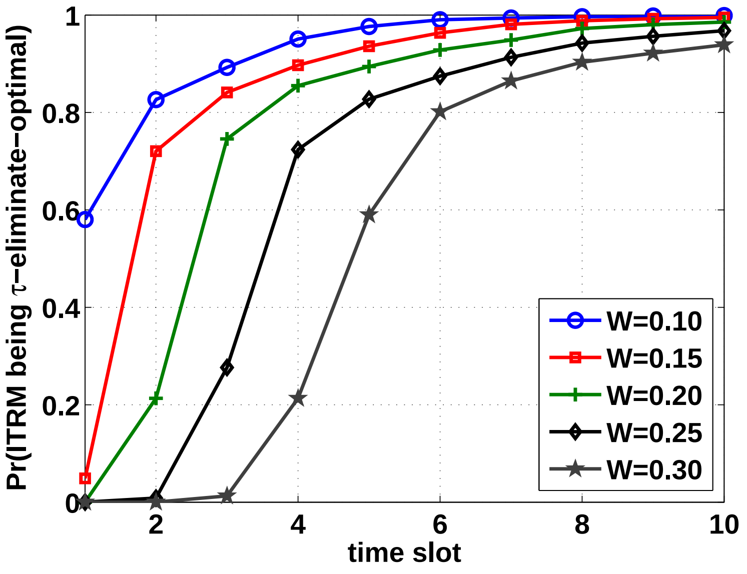


Figure 2: Waiting time for τ-eliminate-optimal.

3.3.2 Simulations

We compared the performance of ITRM with three well-known and commonly used reputation management schemes: 1) *The Averaging Scheme*, 2) *Bayesian Approach*, and 3) *Cluster Filtering.* The Averaging Scheme is widely used in well-known web sites such as Amazon and AllExperts (as discussed in Section 2.1.1). The Bayesian Approach [25] updates the TR_j values using a Beta distribution (as discussed in Section 2.1.2). For this scheme, we assumed a deviation threshold of 0.5 and a trustworthiness threshold of 0.75, which are the same parameters used in the original paper [25] (for details refer to [25]). Cluster Filtering [35] performs a dissimilarity test among the raters and then updates the TR_j values using only the honest raters (as discussed in Section 2.1.3).

We assumed that there were already 200 raters (all of which are honest and provide reliable ratings) and 50 SPs in the system. Moreover, a total of 50 time-slots have passed since the launch of the system. Further, ratings generated during previous time-slots were distributed among the SPs in proportion to their reputation values. After this initialization process, we introduced 50 more SPs as newcomers. Further, we assumed that a fraction of the existing raters changed behavior and became malicious after the initialization process. Hence, by providing reliable ratings during the initialization period (for 50 time-slots) the malicious raters increased their trustworthiness values before they attack. Eventually, we had $D + H = 200$ raters and $N = 100$ SPs in total. We further assumed that d is a random variable with Yule-Simon distribution as discussed in the analysis. At each time-slot, the newly generated ratings from honest raters are assigned to the SPs in proportion to the present estimate of their reputation values, TR_j. We obtained the performance of ITRM, for each time-slot, as the mean absolute error (MAE) $|TR_j - \hat{TR_j}|$, averaged over all the SPs that are under attack (where, $\hat{TR_j}$ is the actual value of the reputation). We used the following parameters throughout our simulations: $b = 5$, $\rho = 1$, $\hat{\lambda} = \bar{\lambda} = 0.9$, the penalty factor $\delta = 10$, and $\tau = 0.4$ (the choice of τ is based on the analytical results discussed in Section 3.3.1).

We have evaluated the performance of ITRM in the presence of bad-mouthing and ballot-stuffing. Here, we provide an evaluation of the bad-mouthing attack only, as similar results hold for ballot-stuffing. In all simulations, we considered the worst-case scenario in which the victims are chosen among the newcomer SPs with an actual reputation value of $\hat{TR_j} = 5$ in order to have the most adverse effect. The malicious raters do not deviate very much from the actual $\hat{TR_j} = 5$ values to remain under cover as many time-slots as possible (while still attacking). Hence, at each time-slot, the malicious raters apply a low intensity attack by choosing the same set of SPs from Γ and rating them as $n = 4$. We had also tried higher deviations from the $\hat{TR_j}$ value and observed that the malicious raters were easily detected by ITRM in fewer time-slots. Therefore, we identified the low intensity attack scenario as the most adverse one against the reputation management mechanism. We note that this attack scenario also resembles the RepTrap attack in [105] which is proved to be a strong and destructive attack that can undermine the reputation system. Further, by assuming that the ratings of the honest raters deviate from the actual reputation values, our attack scenario becomes even harder to detect when compared to the RepTrap. In Fig. 3, we show the performance of ITRM for this attack scenario after the newcomer SPs joined to the system and for different W values (with $\Delta = \hat{b}/b = 1$). We observed that ITRM guarantees significantly low errors regardless of the fraction of the malicious raters. As W becomes larger, it takes more time to get negligibly small error values (which is consistent with our analysis). The lags in the plots of ITRM in Fig. 3 correspond to waiting times to include the newcomer SPs into the execution of ITRM, computed based on our analytical results presented in Fig. 2.

Figures 4 and 5 illustrate the comparison of ITRM with the other schemes (i.e., Cluster Filtering, Bayesian Approach and Averaging Scheme) for the above attack scenario and when $W = 0.10$ and $W = 0.30$ of existing raters changed behavior and became malicious, respectively. Thus, the plots in Figs. 4 and 5 are shown from the time-slot the newcomers are introduced and existing raters changed behavior. We

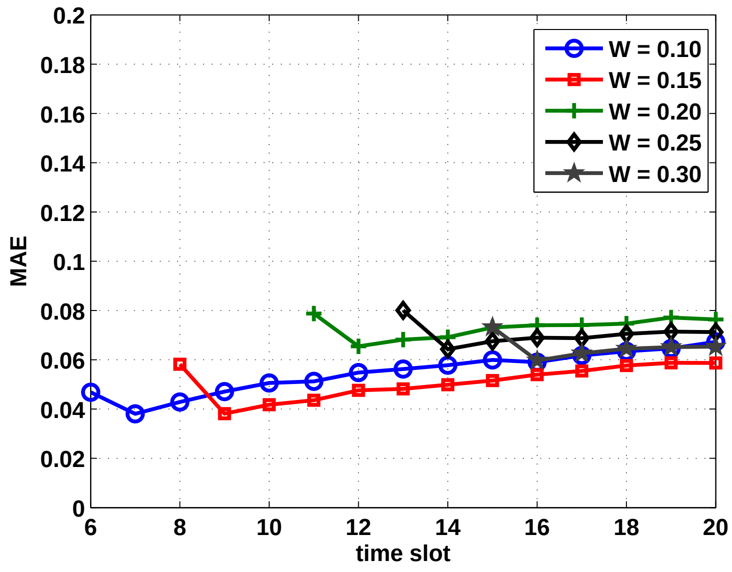


Figure 3: MAE performance of ITRM versus time for bad-mouthing and varying W.

note that for this simulation we set $\Delta = \hat{b}/b = 1$. Again, the lags in the plots of ITRM in Figs. 4 and 5 correspond to waiting times to include the newcomer SPs into the execution of ITRM, computed based on our analytical results. On the other hand, we executed the other 3 schemes starting from the first time-slot, since we observed that their performances were better that way. We also observed that the average number of iterations for ITRM is around 5 and it decreases with time and with decreasing fraction of malicious raters.

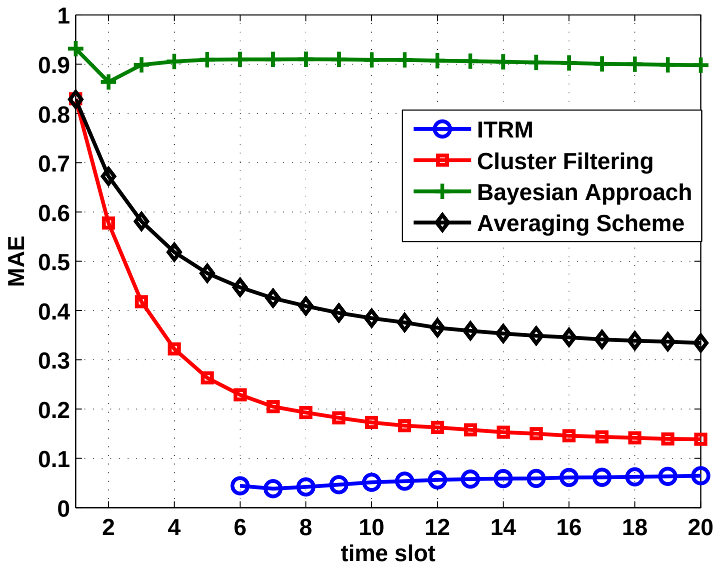


Figure 4: MAE performance of various schemes for bad-mouthing when $W = 0.10$.

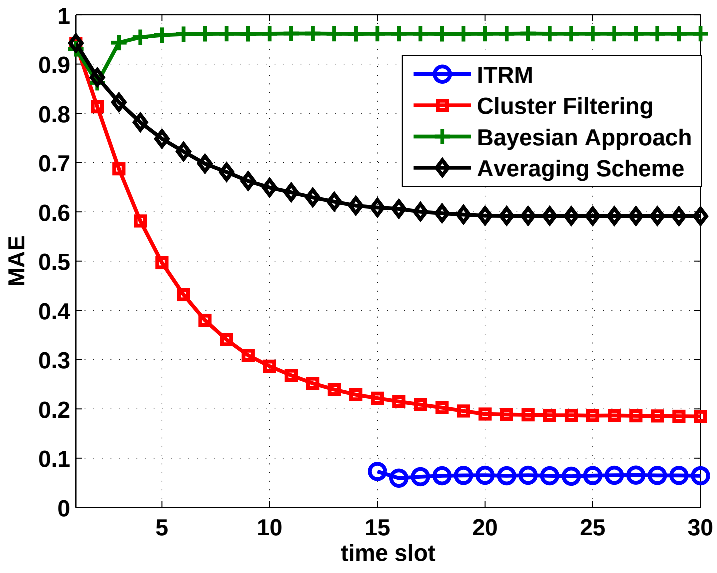


Figure 5: MAE performance of various schemes for bad-mouthing when $W = 0.30$.

We also evaluated the performance of ITRM when the malicious raters provide both reliable and malicious ratings to mislead the algorithm. In Fig. 6, we illustrate the performance of ITRM for this attack for $W = 0.10$ and different $\Delta = \hat{b}/b$ values.

We observed that as the malicious raters attack with less number of edges (for low values of $\hat{b}$), it requires more time slots to have negligibly low error values. Further, when the $\hat{b}$ values becomes very small ($\hat{b} = 1, 2$), it is hard to detect the malicious peers. On the other hand, although the malicious raters stay under cover when they attack with very less number of edges, this type of an attack limits the malicious raters' ability to make a serious impact (they can only attack to a small number of SPs). We note that for small values of $\hat{b}$, although not plotted, other reputation management mechanisms also fail to detect the malicious raters. Further, for different values of Δ and W, we observed that ITRM still keeps its superiority over the other schemes.

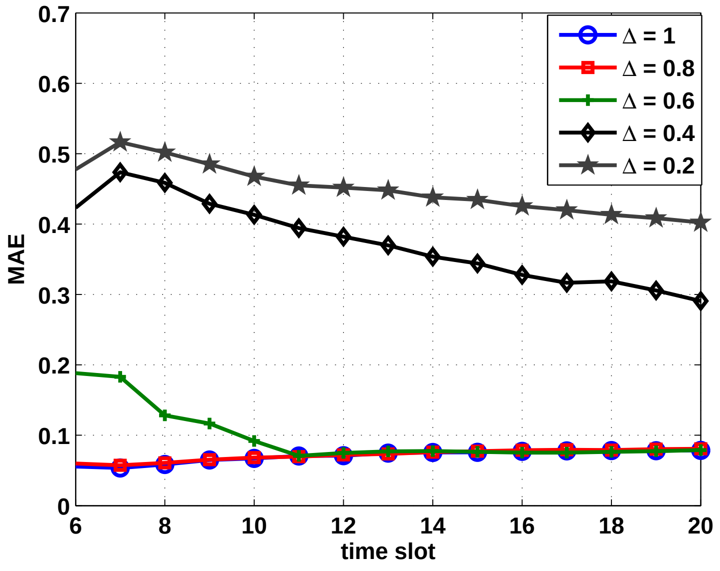


Figure 6: MAE performance of ITRM for bad-mouthing when $W = 0.10$ and for varying Δ.

From these simulation results, we conclude that ITRM significantly outperforms the Averaging Scheme and the Bayesian Approach in the presence of attacks. We identify that the reputation management scheme with the closest performance to ITRM is Cluster Filtering. However, the computational complexity of Cluster Filtering is much higher than ITRM. Specifically, the number of operations required in both methods is illustrated in Table 2. Therefore, while Cluster Filtering introduces quadratic complexity, the computational complexity of ITRM is linear with the number of raters. As a result, our proposed scheme is more scalable and suitable for large scale reputation systems.

Table 2: Computational complexity of Cluster Filtering and ITRM.

	ITRM	Cluster Filtering
Addition	$\mathrm{O}(D+H)$	$\mathrm{O}((D+H)^2)$
Multiplication	$\mathrm{O}(D+H)$	$\mathrm{O}((D+H)^2)$

3.4 Summary

In this chapter, we introduced the "Iterative Trust and Reputation Management Scheme" (ITRM). Our work is a graph based iterative algorithm motivated by prior success on message passing techniques for decoding LDPC codes. The proposed ITRM is a robust mechanism to evaluate the quality of the service of the service providers from the ratings received from the recipients of the service (raters). Moreover, it effectively evaluates the providers' reputations and the trustworthiness of raters while introducing a linear computational complexity with respect to the number of raters. We studied ITRM by a detailed analysis, and showed the robustness using computer simulations. Besides, we compared ITRM with some well-known reputation management schemes and showed the superiority of our scheme both in terms of robustness and efficiency.

CHAPTER IV

APPLICATION OF ITRM TO AD-HOC NETWORKS

4.1 Introduction

Delay Tolerant Networks (DTNs) are relatively new class of networks, wherein sparseness and delay are particularly high (as we discussed in Chapter 1). These special constraints posed by DTNs make existing security protocols inefficient or impractical in such networks. Our main objective in this chapter is to develop a security mechanism for DTNs which enables us to evaluate the nodes based on their behavior during their past interactions and to detect misbehavior due to Byzantine adversaries, selfish nodes and faulty nodes. The resulting scheme would effectively provide high data availability and packet delivery ratio with low latency in DTNs in the presence of Byzantine attackers. To achieve this goal, we aim at obtaining a reputation-based trust management system and an iterative malicious node detection mechanism for DTNs. Thus, we explore the application of Iterative Trust and Reputation Mechanism (ITRM), which is described in Chapter 3, on DTNs. We propose a distributed malicious node detection mechanism for DTNs using ITRM which enables every node to evaluate other nodes based on their past behavior, without requiring a central authority. We will show that the resulting scheme effectively provides high data availability and low latency in the presence of Byzantine attackers. We will also show that the proposed iterative mechanism is far more effective than some well-known reputation management techniques (e.g., *Bayesian framework* and *EigenTrust*) in detecting Byzantine nodes in a DTN environment.

We note that the security issues such as source authentication and data authentication have been previously studied for disconnected networks in [55, 88]. Hence, they are not considered in this research. The security objectives of our research are summarized as follows.

1. Data availability with low latency: Data availability should be ensured. Further, regeneration of the original messages at their destinations should not be delayed for a noticeable time by the Byzantine nodes.

2. A robust trust mechanism: Each node should be able to evaluate the average behaviors of the nodes that it has interacted with by the help of the feedbacks it receives from the other nodes. Moreover, this mechanism should be robust to individual and colluding Byzantine attackers.

3. Mitigating the Byzantine behavior: As we will explain in Section 4.2.1, malicious nodes may attack with different probabilities to hide from the trust management system. The network suffers the most due to malicious nodes with high attacking rates. Hence, containment of malicious nodes that do the most damage should be given the highest priority.

4. Detection of malicious nodes: As a result of the trust establishment, the system must be able to detect the Byzantine nodes without using a central authority. In particular, the performance of the detection algorithm should not be degraded by collaborative attacks against the trust management mechanism.

4.1.1 Contributions

The main contributions of our work are summarized in the following.

1. We introduce the application of ITRM into DTNs as an iterative trust management and malicious node detection scheme. The scheme provides high data availability and packet delivery ratio with low latency in the presence of Byzantine attackers.

2. The proposed algorithm computes the reputations of the network nodes accurately in a short amount of time in the presence of attackers without any central authority.

3. The proposed algorithm mitigates the impacts of Byzantine attackers proportional to their attack degrees. That is, the ones that are attacking with the highest strength are detected with higher probability.

4. Comparison of ITRM with some well-known reputation management techniques (e.g., *Bayesian framework* and *EigenTrust*) indicates the superiority of ITRM in terms of robustness against attacks in a realistic DTN environment. Further, the proposed algorithm is very efficient in terms of its computational complexity. Specifically, the complexity of ITRM is linear in the number of nodes. Hence, it is scalable and suitable for large scale implementations.

4.2 Trust Management and Adversary Detection in Delay Tolerant Networks (DTNs)

4.2.1 Adversary Models and Security Threats

As discussed in Section 4.1, we consider the challenging problem of countering Byzantine (insider) attacks (that give serious damage to the network in terms of data availability, latency and throughput). Broadly we consider two types of attack: 1. Attack on the network communication protocol, 2. Attack on the security mechanism.

Packet drop and packet injection (attack on the network communication protocol): An insider adversary drops legitimate packets it has received. This behavior of the malicious nodes has a serious impact on the data availability and the total latency of the network. Moreover, a malicious node may also generate its own flow to deliver to another (malicious) node via the legitimate nodes. As a result, bogus flows compete with legitimate traffic for the scarce network resources.

Bad-mouthing (ballot-stuffing) on the trust management (attack on the security mechanism): As it will be discussed, a legitimate node needs feedbacks from a subset of nodes to determine its trust on a specific node. When a malicious node is an element of this subset, it gives incorrect feedback in order to undermine the trust management system. Bad-mouthing and ballot-stuffing attacks attempt to reduce the trust on a victim node and boost the trust value of a malicious ally, respectively. A successful attack may result in an incorrect edge value (rating) from a non-malicious check-vertex in the graph representation in Fig. 1(a).

Random attack on trust management (attack on the security mechanism): A Byzantine node may adjust its packet drop rate (on the scale of zero-to-one) to stay under cover, making it harder to detect.

Bad-mouthing (ballot-stuffing) on the detection scheme (attack on the security mechanism): As it will be discussed, every legitimate node, in order to detect the nature of every network node, creates its own trust entries in a table (referred to as the node's rating table) for a subset of network nodes for which the node has collected sufficient feedbacks. Further, each node also collects rating tables from other nodes. When the Byzantine nodes transfer their tables to a legitimate node, they may victimize the legitimate nodes (in the case of bad-mouthing) or help their malicious allies (in the case of ballot-stuffing) in their rating table entries. This effectively reduces the detection performance of the system. Furthermore, malicious nodes can provide both reliable and malicious ratings to mislead the algorithm as discussed in Section 3.3.2. A successful attack adds a malicious check-vertex providing malicious edges (ratings) in the graph representation in Fig. 1(a).

During the evaluation of the proposed scheme, we assumed that malicious nodes may mount attacks on both the network communication protocol and the underlying security mechanism (trust and reputation management mechanism, ITRM) simultaneously. In the attack on the network communication protocol, we assumed that malicious nodes both drop the legitimate packets they have received from reliable nodes and generate their own flows to deliver to other (malicious) nodes via the legitimate nodes in order to degrade the network performance (i.e., data availability and packet delivery ratio) directly. In the attack on the security mechanism, we assumed that malicious nodes simultaneously execute “bad-mouthing (ballot-stuffing) on the trust management”, “random attack on trust management”, and “bad-mouthing (ballot-stuffing) on the detection scheme” (which are described above) to cheat the underlying trust and reputation management scheme (i.e., ITRM) and degrade the network performance indirectly. We study the impact of these attacks and evaluate our proposed scheme in the presence of these attacks (on the network communication protocol and the security mechanism) in Section 4.2.5. First, we study the impact of the attacks to cheat the underlying trust and reputation management mechanism alone and obtain the time required to detect all the malicious nodes in the network. Next, we study the impact of the “packet drop and packet injection attack” to the network performance (in terms of data availability and packet delivery ratio) while the malicious nodes also mount attacks on the underlying reputation mechanism.

As a result of our studies, we concluded that ITRM provides a very efficient trust management and malicious node detection mechanism for DTNs under the threat model discussed above. The most significant advantage of ITRM under the above threat model, in addition to resiliency to a high fraction of malicious nodes, is to let each network node accurately compute the reputation values of the other network nodes in a short time. Computing the reputation values in a short time is a very crucial issue in DTNs because of their unique characteristics (such as the intermittent contacts between the nodes). As a result of this advantage, each legitimate node detects and isolates the malicious nodes from the network to minimize their impact to the network performance (as will be illustrated in Section 4.2.5).

4.2.2 Network/Communication Model and Technical Background in Context

Before giving a high level description of our scheme, we will introduce the network/communication model and the main tools that we use for the system to operate.

Mobility model: We use both Random Waypoint (RWP) and Levy-walk (LW) mobility models for our study which are widely used for simulating DTNs. RWP model produces exponentially decaying inter-contact time distributions for the network nodes making the mobility analysis tractable. On the other hand, LW mobility is shown to produce power-law distributions that has been studied extensively for animal patterns and recently has been shown to be promising as a model for human mobility [82]. In the RWP mobility model [22], each node is assigned an initial location in the field and travels at a constant speed to a randomly chosen destination. The speed is randomly chosen from $[v_{min}, v_{max}]$ regardless of the initial location and destination. After reaching the destination, the node may pause for a random amount of time before the new destination and speed are chosen randomly. In LW mobility model [50, 72, 82], each flight length and pause time distributions closely match the truncated power-law distributions. Further, angles of movement are pulled from a uniform distribution. Our implementation of LW mobility model is based on the model in [82]. A step is represented by four variables, flight length (ℓ), direction (θ), flight time (Υt_f), and pause time (Υt_p). The model selects flight lengths and pause times randomly from their probability distributions $p(\ell)$ and $\psi(\Upsilon t_p)$ which are Levy distributions with coefficients α and β, respectively. Finally, regardless of the mobility model used, we assume a finite rate of packet transfer which forces the number of packets transmitted per contact to be directly proportional to the contact time.

Packet format: We require that each packet contains its two hop history in its header. In other words, when node B receives a packet from node A, it learns from which node A received that packet. This mechanism is useful for the feedback mechanism as discussed in Section 4.2.4.

Routing and packet exchange protocol: We assume that messages at the source are packetized. Further, the source node never transmits multiple copies of the same packet. Hence, at any given time, there is at most a single copy of each packet in the network. We assume only single-copy routing since reliable single-copy routing with packetization is achieved by encoding the data packets using rateless codes [63, 91] (as briefly discussed next) at the source node. The use of rateless coding improves reliability and latency in DTNs even when there is no adversary [99]. Furthermore, exchange of packets between two nodes follows a back-pressure policy. To illustrate this, assume node A and B have x and y packets belonging to the same flow f, respectively (where $x > y$). Then if the contact duration permits, node A transfers $(x - y)/2$ packets to node B belonging to flow f. As a result of the mobility model, each node has the same probability to meet with the destination of a specific flow. Hence, by using the back-pressure policy we equally share the resources (e.g., contact time) among the flows.

The packet exchange protocol also enforces fairness among multiple nodes that forwarded the same flow to a node. To clarify, let us assume that node A has some packets from a flow f (which were forwarded to it by χ different nodes) and based on the back-pressure policy, it needs to transfer some of them to node B. In this situation, node A must fairly select the packets based on their previous hops (which is available via the packet format discussed before). In other words, each packet that is received from a different node has the same probability to be selected for transfer. This mechanism is useful for the feedback mechanism as discussed later. Finally, when a node forwards a packet, it deletes it from its buffer.

Rateless Coding: Recently, a new class of efficient codes called rateless codes have been proposed that require no knowledge of channel parameters to perform near-optimally with a simple decoding algorithm [63]. The rateless encoder can potentially generate a limitless stream of encoded packets for a list of η input (information) packets. Fundamental to rateless coding is a probability distribution Ω on the set $\{1, 2, ...\eta\}$, i.e., the probability of the symbol i is given by $\Omega(i)$. To generate an encoded packet, the encoder generates an instance ξ of a random variable Ξ with the distribution Ω. The encoder then chooses ξ distinct input packets, say $P_{i_1}, ..., P_{i_\xi}$, from the available η input packets (each with, say, l bits) and declares the encoded packet to be $P_{i_1} \oplus P_{i_2} ... \oplus P_{i_\xi}$, where $\oplus$ denotes the packet-level XOR operation. In such a setup, it can be shown that when the decoder receives $\eta(1+\zeta_\eta)$ packets, where ζ_η is a positive number very close to zero, it can successfully decode all η input packets with high probability [63] (coding overhead ζ_η decreases as η increases).

Bloom filter: A Bloom filter is a simple space-efficient randomized data structure for representing a set in order to support membership queries [20]. A Bloom filter for representing a set U of G elements is described by an array of κ bits, initially all set to 0. It employs γ independent hash functions $\mathbb{H}_1, \ldots, \mathbb{H}_\gamma$ with range $\{1, \ldots, \kappa\}$. For every element $x \in U$, the bits $\mathbb{H}_1(x), \ldots, \mathbb{H}_\gamma(x)$ in the array are set to 1. A location can be set to 1 multiple times, but only the first change has an effect. To check if y belongs to U, we check whether all $\mathbb{H}_1(y), \ldots, \mathbb{H}_\gamma(y)$ are set to 1. If not, y definitely does not belong to U. Otherwise, we assume $y \in U$ although this may be wrong with some probability. Hence, a Bloom filter may yield a *false positive* where it suggests that y is in U even though it is not.

The probability of false positive is an important parameter in a Bloom filter. After all elements of U are hashed into the filter, the probability that a specific bit is 0 is

$$\left(1 - \frac{1}{\kappa}\right)^{\gamma G} \approx e^{-\gamma G/\kappa}. \tag{10}$$

Hence, the probability of false positive is

$$\tilde{p} = \left(1 - \left(1 - \frac{1}{\kappa}\right)^{\gamma G}\right)^{\gamma} \approx \left(1 - e^{-\gamma G/\kappa}\right)^{\gamma}. \qquad (11)$$

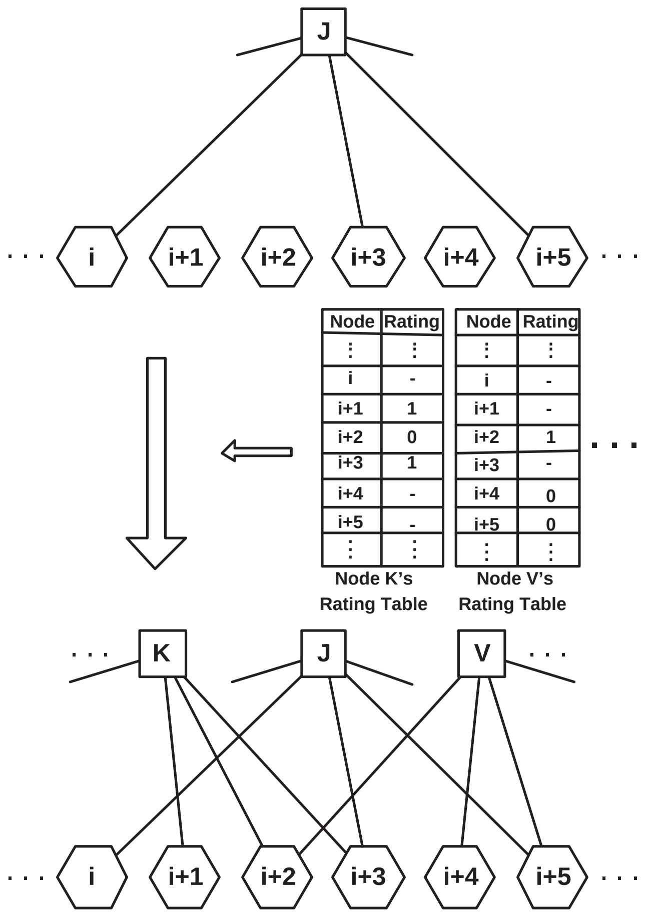
The network designer can arbitrarily decrease this probability to the expense of increasing communication overhead. We note that the false positive probability can be significantly reduced by using recently proposed techniques such as [44].

4.2.3 Iterative Malicious Node Detection for DTNs

In this section, we will describe how ITRM is adapted in DTNs as an iterative malicious node detection mechanism. We will pick an arbitrary node in the network and present the algorithm from its point of view throughout the rest of this discussion. We denote this node as a *judge* for clarification of our presentation. Further, the counterpart to the quality of a SP in the discussion of ITRM is the reliability of the node in DTN in faithfully following the network (routing) protocols to deliver the packets.

Since direct monitoring is not an option in DTNs (as explained in Section 2.2.1), a judge node creates its own rating about another network node by collecting feedbacks about the node and aggregating them. Each judge node has a table (referred to as a *Rating Table*) whose entries (which are obtained using the feedback mechanism described in Section 4.2.4) are used for storing the ratings of the network nodes. In DTNs, due to intermittent contacts, a judge node has to wait for a very long time to issue its own ratings for all the nodes in the network. However, it is desirable for a judge node to have a fresh estimate of the reputation values of all the nodes in the network in a timely manner, mitigating the effects of malicious nodes immediately. To achieve this goal, we propose an iterative malicious node detection mechanism which operates by using the rating tables formed by other nodes (acting as judges themselves). The rating table of a judge node can be represented by a bipartite graph consisting one check-vertex (the judge node) and some bit-vertices (i.e., a subset of all the nodes in the network for which the judge node has received sufficient number of feedbacks to form a rating with high confidence). Besides, by collecting sufficient number of rating tables from other nodes, a judge node can generate a bipartite graph as in Section 3.2; which includes all the network nodes as bit-vertices. We illustrate this process at judge node J in Fig. 7 in which node J collects rating tables from other judge nodes (including K and V) and generates a bipartite graph including all network nodes as bit-vertices. Assuming N nodes in the network, a judge node may create a bipartite graph with N bit-vertices by collecting rating tables from $k - 1$ nodes each with at least s non-empty entries. Hence, the resulting graph would have k check-vertices (the k^{th} check vertex belongs the judge node). The parameters s and k are to be determined for high probability of detection while minimizing detection latency. Clearly, higher s and k reduces the detection error but increases the delay. We will discuss this issue in Section 4.2.5. Hence, when two nodes establish a contact in a DTN, they exchange their rating tables. Once a judge node collects sufficient number of tables each with sufficient number of non-empty entries, it can then proceed with the iterative algorithm to specify the reputation values for all the nodes.

To adapt the ITRM scheme for DTNs, we will present (feedback) ratings as "0" or "1", which results in binary reputation values. In this special case, the iterative reputation scheme becomes a detection scheme. That is, a node with a reputation value of zero would be interpreted as a malicious node. Therefore, the proposed scheme detects and isolates the malicious nodes from the network to minimize their impact. We note that we used binary rating values for simplicity of the setup. Alternatively, one may consider a setup where ratings are non-binary. In this scenario, when two nodes establish a contact, they may exchange packets with some probability associated with their reputation values (i.e., they may exchange packets proportional to their reputation values). Moreover, we did not incorporate R_i (trustworthiness) values for simplicity of simulations, and hence, we set all R_i values to one for the



Node	Rating
⋮	⋮
i	-
i+1	1
i+2	0
i+3	1
i+4	-
i+5	-
⋮	⋮

Node K's Rating Table

Node	Rating
⋮	⋮
i	-
i+1	-
i+2	1
i+3	-
i+4	0
i+5	0
⋮	⋮

Node V's Rating Table

Figure 7: Collecting and combining the rating tables at the judge node J.

application of ITRM in DTNs. In other words, we assume that the judge node does not have any previous knowledge about the nodes from which it receives the feedbacks and it trusts each node equally.

4.2.4 Trust Management Scheme for DTNs

In the proposed scheme, the authentication mechanism for the packets generated by a specific source is provided by a Bloom filter [20] and ID-based signature (IBS) [32]. Whenever a source node sends some packets belonging to the flow that is initiated by itself, it creates a Bloom filter output from those packets, signs it using IBS and sends it to its contacts. The Bloom filter output provides an authentication mechanism for the packets generated by a specific source. It is worth noting that whenever an intermediate node forwards packets belonging to a specific flow to its contact, it also forwards the signed Bloom filter output belonging to those packets for the packet level authentication at each intermediate node. We do not give further details of the authentication mechanism as source and data authentication for DTNs have been considered before [55, 88] and they are out of the scope of this work.

Our proposed feedback mechanism to determine the entries in the rating table is based on a 3-hop loop (referred to as *Indirect type I* feedback). We will describe this scheme by using a toy example between 3 nodes A, B, and C as follows. Let us denote the node that is evaluating as the *judge* (node A), the node that is being evaluated as the *suspect* (node B), and the node that was the direct contact of the suspect as the *witness* (node C). The basic working principle of the mechanism is that after the judge node has a transaction (in the form of passing some packets) with a suspect, the judge node waits to make contacts and receive feedback about the suspect from every node (i.e., witnesses) that has been in direct contact with the suspect. It is worth noting that this feedback mechanism is only used for constructing the entries in the judge node's rating table for a few network nodes. In overall, rating tables are collected from the contacts of the judge node and ITRM is applied to find the reputations of all network nodes (as described in Section 4.2.3).

Let assume that node A meets B, B meets C and C meets A at times t_0, t_1 and t_2, respectively, where $t_0 < t_1 < t_2$. Indirect type I feedback between nodes A, B and C is illustrated in Fig. 8. At time t_0, A and B execute mutual packet exchange as described in Section 4.2.2. When B and C meet at t_1, they first exchange signed time-stamps. Hence, when C establishes a contact with A, it can prove that it indeed met B. Then B sends the packets in its buffer executing the fairness protocol as discussed in Section 4.2.2. Moreover, suspect node B transfers the receipts it received thus far to the witness C. Those receipts include the proofs of node B's deliveries (including deliveries of the packets belonging to node A) thus far and are signed by the nodes to which its packets were delivered. We note that the receipts expire in time and deleted from the buffers of the witnesses. Hence, they are not accumulated in the buffers of the nodes. The lifetime of the receipts are determined based on the detection performance of the scheme (required time for the scheme to have a high malicious node detection accuracy) as will be described in Section 4.2.5. At the end of the contact, node C also gives a signed receipt to node B including the IDs of the packets it received from B during the contact. Finally, when the judge node A and the witness C meet, they initially exchange their contact histories. Hence, A learns that C has met B and requests the feedback. The feedback consists of 2 parts: i) Those receipts of B that are useful for A's evaluation (i.e., receipts which include the delivery proofs of the packets belonging to node A), and ii) If node C received node A's packets from node B, it sends the hashes of those packets to A for the latter's evaluation. We note that C can easily find out A's packets by just examining the headers explained in Section 4.2.2. From B's receipts, node A can determine if B followed the packet delivery procedure (which is described in Section 4.2.2) properly while delivering the packets forwarded by node A at time t_0 (B's receipts will reveal the packet deliveries of B after time t_0). Further, from the hashes of its own packets (if there is any received by node C), node A can determine if node B modified any of the packets before delivery.

If both parts of the feedback are verified by node A (if node B followed the packet delivery procedure for A's packets and delivered the packets properly), then the judge A makes a "positive evaluation" as 1. Otherwise, if either part of the feedback is not verified, the evaluation will be "negative" as 0. We note that if node C did not receive any packets belonging to node A, then node A's evaluation will be only based on the receipts of B which are provided by node C at time t_2 (i.e., node A will evaluate node B based on the receipts it received from node C, which is the first part of the feedback explained before). Each judge node uses the Beta distribution [25]

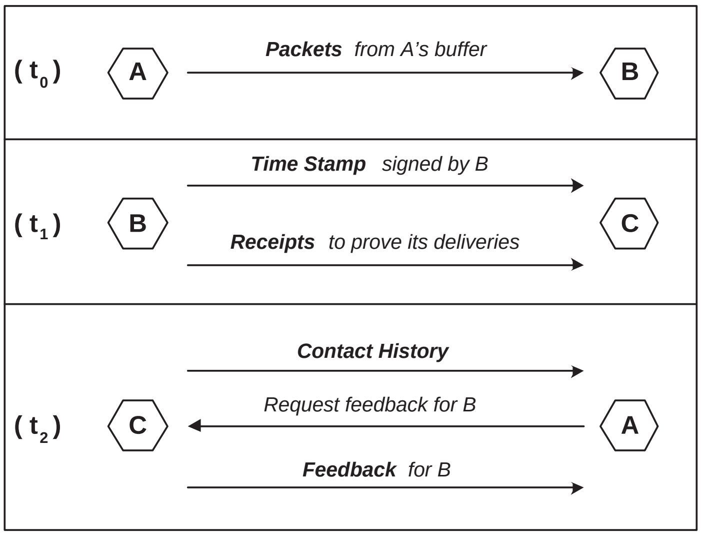


Figure 8: Indirect type I feedback between nodes A (judge), B (suspect) and C (witness).

to aggregate multiple evaluations it has made about a suspect using the associated feedbacks. The collection of multiple feedbacks generates the rating (verdict) of a judge node for a suspect node[1]. That is, if the aggregation of multiple feedbacks for a suspect node is bigger that 0.5, the suspect node is rated as 1 in the judge node's rating table. Otherwise, if the aggregation value is smaller than or equal to 0.5, the suspect node is rated as 0. We note that the feedbacks from the witnesses are not trustable. Because of the bad-mouthing (ballot-stuffing) and random attacks (discussed in Section 4.2.1), a judge node waits for a definite number of feedbacks to give its verdict about a suspect node with a high confidence. We will discuss this waiting time, the number of required feedbacks, and their interplay for different adversarial models in Section 4.2.5.

In the high level description of ITRM, it was implicitly assumed that the judge has a priori knowledge about the packet drop rate of the Byzantine node. This is unrealistic as the nodes may apply random attacks as in Section 4.2.1. To remove this

[1]ITRM utilizes the rating tables whose entries are associated verdicts to determine the final faith of a node. Hence, the verdicts will be further processed by ITRM.

assumption, we propose detection at different levels. We observed that the sufficient number of feedbacks that is required to give a verdict with high confidence depends on the packet drop rate of the Byzantine nodes. In other words, for a node with a higher drop rate, we would require fewer feedbacks than a node with a lower drop rate. Assume that we desire to perform detection at level $p_1 = 0.8$. This implies that after applying ITRM, each judge node would identify and isolate all the Byzantine nodes whose packet drop rates are p_1 or higher. Further, assume that the detection at level p_1 requires at least $\hat{M}_1$ feedbacks about a suspect node. The number of feedbacks depends on the confidence we seek at the accuracy of a verdict (before detection). The level of confidence is determined by the detection strategy. For instance, for ITRM, a confidence value in the order of 0.95 (out of 1) would be sufficient. Clearly, the number of feedbacks also depends on the detection level. The lower the detection level, the higher is the number of required feedbacks to maintain the same detection confidence. Hence, every judge stores together with its verdict the lowest level of detection at which the verdict can be used. Obviously, an entry verdict with lower detection level (e.g., $p = 0.6$) is also good for use in a high detection level (e.g., $p = 0.8$), but the inverse is not true. An entry is left empty if the judge does not have the sufficient number of feedbacks to give any verdict even at the highest detection level. We note that there is no pre-determined detection level for the proposed scheme. The judge node applies the ITRM for the lowest possible detection level (to minimize the impacts of malicious nodes) depending on the entries (number of feedbacks used to construct each entry verdict) in both its own rating table and the rating tables it collected from other nodes. The judge checks the detection level of each table entry (from both its own table and the collected tables) and performs the ITRM at the detection level of the entry verdict which is the largest. To clarify this, assume a judge node J collects rating tables from other judge nodes K and V as in Fig. 7. For this toy example, we assume that the judge node J will perform the ITRM by using only 3 rating tables (its own rating table and the ones collected from nodes K and V). We further assume that the rating table entry (in the rating tables of nodes J, K, and V) with the largest detection level has a detection level of j in the judge node J's rating table, and detection levels of k and v for nodes K and V's rating tables, respectively. Then, the judge node J performs the ITRM at the detection level of $max(j, k, v)$. The malicious nodes may try to survive from the detection mechanism by setting their packet drop rates to lower values. However, the proposed detection mechanism eventually detects all the malicious nodes (even the ones with lower packet drop rates) when the judge node waits longer times to apply the ITRM at a lower detection level. Further, as the drop rate of the malicious nodes gets lower, the negative impact of the malicious nodes gets less significant in terms of data availability and packet delivery ratio.

4.2.5 Security Evaluation

In this section, we give an analysis of the metrics of interest and illustrate our simulation results. Further, we compare the performance of ITRM with the well-known reputation management schemes (Bayesian framework [24] and EigenTrust [53]) in a realistic DTN environment. Finally, we show the performance of the proposed scheme for the malicious node detection, availability and packet delivery ratio via simulations (conducted using MATLAB). We assumed the mobility models (RWP and LW) of Section 4.2.2 with N nodes in the network. It is shown that the inter-contact time distributions of the LW can be modeled by a truncated Pareto distribution [50]. On the other hand, as we mentioned in Section 4.2.2, the fact that the inter-contact times of the RWP mobility model can be modeled as a Poisson process [42] makes the mobility analysis tractable. Therefore, for our analytical conclusions (in Lemmas 4.2.1 and 4.2.2), we assumed the RWP mobility model[2]. However, for the simulations, we

[2]Similar results can be obtained for the LW mobility model using a truncated Pareto distribution for the inter-contact times.

used both RWP and LW mobility models to evaluate the performance of the proposed scheme under different mobility models.

In all simulations, we fixed the simulation area to $4.5km$ by $4.5km$ (with reflecting boundaries) which includes $N = 100$ nodes each with a transmission range of $250m$ (which is the typical value for IEEE 802.11b). For the RWP model, we used $[v_{min}, v_{max}] = [10, 30]m/s$ and ignored the pause time for the nodes. For the LW mobility model, we set the speed of every node to $10m/s$. Further, we set the scale factors of flight lengths and pause times to 10 and 1, respectively. We used the Levy distribution coefficients of $\alpha = 1$ and $\beta = 1$. Finally, we set the maximum flight length and pause time to $4km$ and 2 hours, respectively.

Confidence on a Verdict: We let λ_i be the inter-contact time between two particular nodes. We analytically illustrated the waiting time of a judge node to collect sufficient number of feedbacks about a suspect (to give its verdict with high confidence) and evaluated the effect of random attack on the required number of feedbacks in the following. Let the random variables x, y and z represent the number of feedbacks received at a specific judge node A (about a suspect node B), total number of contacts that the suspect node B established after meeting A, and the number of distinct contacts of B after meeting A, respectively. The following lemma characterizes the time needed to receive M distinct feedbacks about a particular suspect node B at a particular judge node A for the RWP mobility model.

Lemma 4.2.1. *Let t_0 be the time that a transaction occurred between a particular judge-suspect pair. Further, let N_T be the number of feedbacks received by the judge for that particular suspect node since $t = t_0$. Then, the probability that the judge node has at least M feedbacks about the suspect node from M distinct witnesses at time $T + t_0$ is given by*

$$Pr(N_T \geq M) = \int_M^{\infty} \int_{-\infty}^{+\infty} f(x|z, T) f(z, T) dz dx. \tag{12}$$

Here, the distribution $f(x|z,t)$ is Poisson with rate $\lambda_i z/2$ and

$$f(z,t) = \int_{-\infty}^{+\infty} f(z|y,t) f(y,t) dy, \tag{13}$$

where $f(y,t)$ and $f(z|y,t)$ are both Poisson distributions with rates $(N-2)\lambda_i$ and $(N-2)\lambda_i - \lambda_i y/2$, respectively.

Proof. The probability that a particular judge node receives at least M feedbacks (from distinct witnesses) about a particular suspect node between time t_0 and $t_0 + T$ is given by

$$Pr(N_T \geq M) = \int_{M}^{\infty} f(x,T) dx, \tag{14}$$

where $f(x,t) = \int_{-\infty}^{+\infty} f(x|z,t) f(z,t) dz$. As a result of the RWP mobility model, it can be shown that $f(x|z,t)$ is Poisson with rate $\lambda_i z/2$ where z represents the number of distinct contacts of the suspect between time t_0 and $t_0 + T$ and x is the number of feedbacks received by the judge node (about the suspect) from a subset of those z contacts. Further, since there are N nodes in the network, it can be shown that the number of contacts established by any node has a Poisson distribution with rate $(N-1)\lambda_i$ (excluding itself). Therefore, the number of contacts the suspect established after the transaction with the judge, y, has a Poisson distribution with rate $(N-2)\lambda_i$ (excluding the judge node and the suspect node itself), and given y, the number of distinct contacts of the suspect, z, has a Poisson distribution with rate $(N-2)\lambda_i - \lambda_i y/2$. □

We studied the effect of random attack on the required number of feedbacks for a network with $N = 100$[3]. We denote the fraction of the Byzantine nodes in the network as W. As we discussed in Section 4.2.4, a judge node waits for a definite number of feedbacks to give its verdict about a suspect node with a high confidence. Figure 9 illustrates the variation of a (judge) node's confidence Σ on its verdict for

[3]The results illustrated (in Figs. 9 and 10) are independent of the mobility model used.

a suspect versus different levels of detection p. This is given for different number of feedbacks (M) when $W = 0.10$. As expected, a node has more confidence at higher detection levels and for high M values. As we discussed before, due to the bad-mouthing, ballot-stuffing and random attacks, a judge node waits for a definite number of feedbacks to give its verdict about a suspect node with a high confidence. Let $\hat{M}$ be the minimum number of feedbacks required about a specific suspect node for an acceptable confidence level on a verdict. In Fig. 10, the variance of $\hat{M}$ for different detection levels (p) and different W values is illustrated given a judge node has $\Sigma = 0.95$ confidence on its verdict ($\hat{M}$=M for $\Sigma \simeq 0.95$). Hence, we can say that a judge node needs more feedbacks about a suspect when there are more malicious nodes mounting bad-mouthing (or ballot-stuffing) on the trust management.

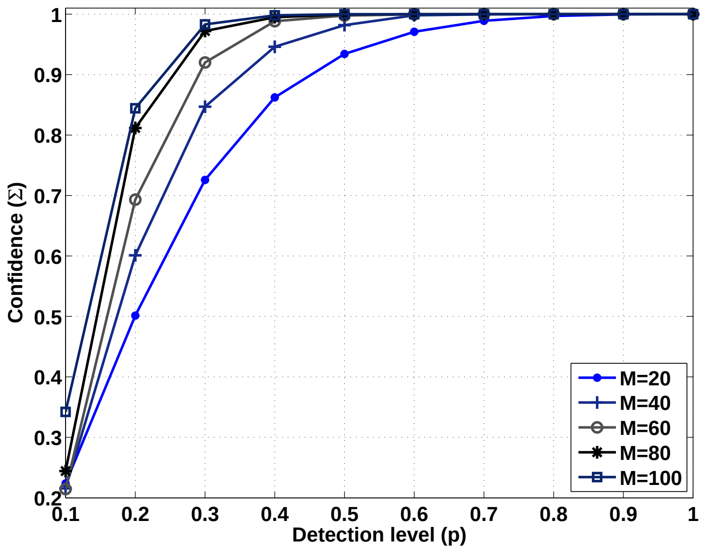


Figure 9: Confidence of a judge node on its verdict vs. the detection level for $W = 0.10$.

Detection Performance: We analytically illustrated the waiting time of a judge node before executing ITRM and evaluated the effects of attacks on the detection scheme for a network of size N in which the inter-contact time between two particular nodes is λ_i. Let $\hat{M}$ be the minimum number of feedbacks required about a specific

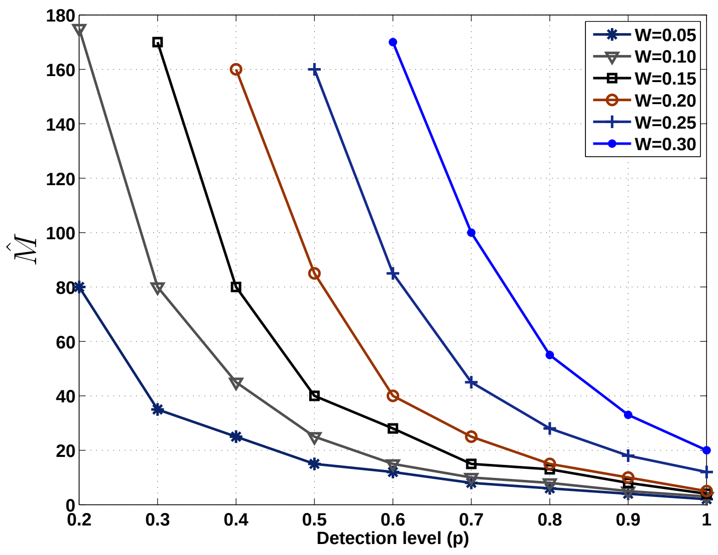


Figure 10: $\hat{M}$ versus the detection level when $\Sigma = 0.95$ for different values of W.

suspect node for an acceptable confidence level on a verdict. Further, let $\hat{T}$ be the time required to receive $\hat{M}$ feedbacks for a specific suspect. The following lemma along with the simulation results illustrated in Figs. 11, 12, 13 and 14 (which will be presented next) provide a good insight for a judge node about the instant it should apply ITRM (the proof is similar to that of Lemma 4.2.1).

Lemma 4.2.2. *Let a particular judge node start collecting feedbacks and generating its rating table at time $t = t_0$. Further, let $\hat{N}_T$ be the number of entries in the rating table of the judge node. Then, the probability that the judge node has at least s entries at time $t_0 + T$ is given by*

$$Pr(\hat{N}_T \geq s) = \int_{s}^{+\infty} \int_{-\infty}^{+\infty} f(z|x, T - \hat{T}) f(x, T - \hat{T}) dx dz, \tag{15}$$

where $f(x,t)$ and $f(z|x,t)$ are Poisson distributions with the rates $(N-1)\lambda_i$ and $(N-1)\lambda_i - \lambda_i x/2$ for the RWP mobility model, respectively.

We evaluated the performance of ITRM for different (k, s) pairs (where k is the number of rating tables collected at the judge node and s is the number of non-empty entries in each table). Moreover, we compared ITRM with the well-known Voting Technique in which a judge node decides on the type of a suspect based on the majority of the votes for that node. For the Voting Technique, we used the Indirect type I feedback as described in Section 4.2.4 (since direct monitoring is not possible in DTNs, we believe that this feedback mechanism is the only option for the nodes). However, in the Voting Technique, instead of utilizing the ITRM, a judge node decides on the type of a suspect node based on the majority of feedbacks it received (i.e., a suspect node is identified as a malicious node if it received more negative feedbacks than the positive ones).

We defined the *success* of a scheme as its capability of detecting all malicious nodes in the network (without identifying any reliable node as malicious by mistake). We illustrated the probability of success, S, of ITRM and the Voting Technique for different (k, s) pairs, and showed the time needed to obtain such a success probability. We used both RWP and LW mobility models (with the settings discussed before) in our simulations to evaluate the proposed scheme in a realistic DTN environment. In both mobility models, whenever two nodes establish a contact, a transaction occurs between them in the form of packet exchange. Further, the judge and malicious nodes start generating their rating tables and mounting their attacks at time $t = 0$, respectively.

We provide the evaluation only for the bad-mouthing on the detection scheme and bad-mouthing on the trust management only, as similar results hold for ballot-stuffing and combinations of bad-mouthing and ballot-stuffing. In particular, malicious nodes provide incorrect feedbacks to the judge nodes about their reliable contacts in order to cause the judge nodes to misjudge the types of reliable nodes (in their verdicts). As a result of the malicious feedback, a judge node may make a "negative evaluation" (as described in Section 4.2.4) on a reliable node. Second, the malicious nodes collaboratively victimize the reliable nodes (i.e., attack the same set of reliable nodes) in their own rating tables by rating them as "0" and forward these rating tables whenever they contact with reliable nodes to mislead the detection mechanism.

In Fig. 11, we illustrated S versus time for fixed values of k and varying s for the RWP mobility model. In Fig. 12, the s values are fixed and the parameter k is varied with increments of 5 for the RWP model. Similarly in Figs. 13 and 14, we illustrated S for ITRM and the Voting Technique with the LW mobility model. In all figures, time is measured starting from $t = 0$. Our results support the fact that RWP shows a more optimistic routing performance compared to LW since its high occurrences of long movements intensify the chance of meeting destinations [82]. Further, these results also give some indication of the false positive (tagging a reliable node as malicious) and false negative (labeling a malicious node as reliable) probabilities of the proposed scheme as well. As S increases, the probability that the scheme detects all malicious nodes gets higher along with the probability that the scheme identifies all reliable nodes as reliable. Similarly, as S decreases, the probability that the scheme labels a malicious node as reliable gets higher along with the probability that the scheme marks a reliable node as a malicious one. In other words, false positive and false negative probabilities are high when the probability of success is low as in Figs. 11, 12, 13 and 14. Furthermore, these results can also be used to determine the lifetimes of the receipts at the witness nodes. Knowing how long it takes to have a high success probability at a judge node for a given detection level, the witnesses can delete the receipts which have been stored for more than the sufficient time required for a high success probability from their buffers. Based on our simulation results, we concluded that ITRM significantly outperforms the Voting Technique by providing higher success rates in shorter time (regardless of the mobility model) which is a very crucial issue in DTNs. We obtained these results for the fraction of malicious nodes W is 0.10 and for a detection level of $p = 0.8$. However, we note that the required (k, s) pairs to obtain a high success probability do not change with the detection level, which only has an effect on $\hat{M}$. It is worth noting that even though the time required to get the high success probability increases with increasing W, the performance gap between ITRM and the Voting Technique remains similar for different values of W.

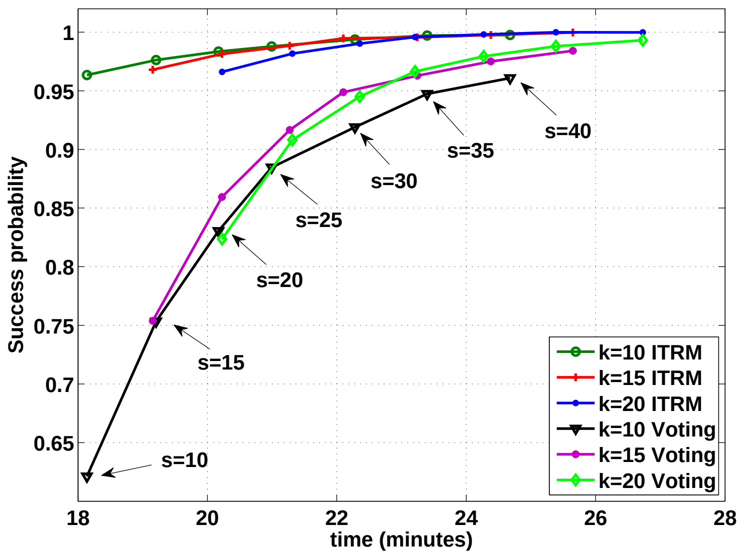


Figure 11: Probability of detection success for fixed k and varying s values with RWP mobility model for $W = 0.10$.

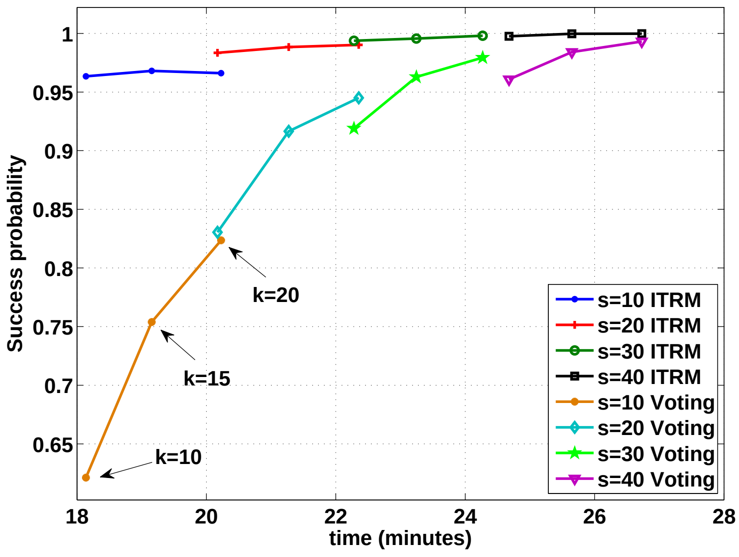


Figure 12: Probability of detection success for fixed s and varying k values with RWP mobility model for $W = 0.10$.

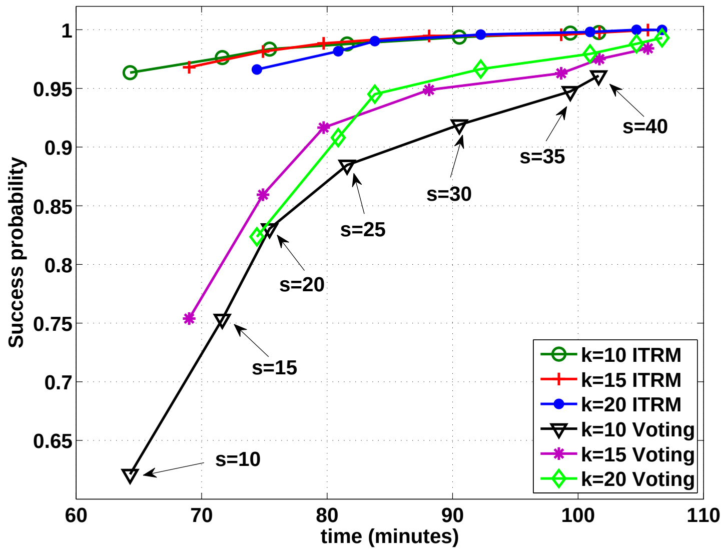


Figure 13: Probability of detection success for fixed k and varying s values with LW mobility model for $W = 0.10$.

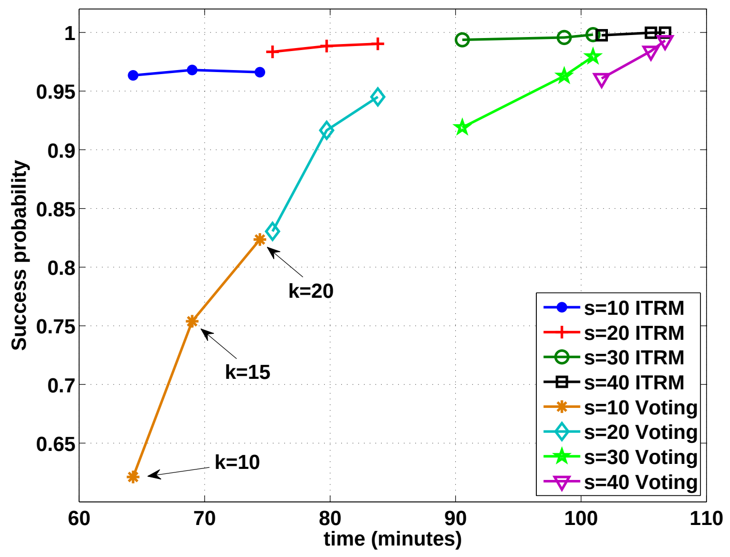


Figure 14: Probability of detection success for fixed s and varying k values with LW mobility model for $W = 0.10$.

In the rest of this section, we will present our simulation results for different network parameters and show the performance of the proposed scheme for Mean Absolute Error (MAE) in the computed reputation values, data availability, and packet delivery ratio. We note that we did not compare the proposed scheme with existing DTN security schemes such as [55] since none of the existing schemes is aimed to provide data availability and malicious node detection as in our work. Further, it is worth noting that there is no existing trust and reputation management mechanism for DTNs. In spite of this, we compared the proposed scheme with the Bayesian reputation management framework in [24] (which is also proposed as the reputation management system of the well-known CONFIDANT protocol [23]) and the EigenTrust algorithm [53] in a DTN environment. For the Bayesian framework [24], we used the parameters from the original work [24] (deviation threshold $d = 0.5$ and trustworthiness threshold $t = 0.75$). Further, we set the fading parameter to 0.9 (for details refer to [24]). It is worth noting that neither the original Bayesian reputation framework in [24] nor EigenTrust [53] is directly applicable to DTNs since both protocols rely on direct measurements (e.g., watchdog mechanism) which is not practical for DTNs as discussed in Section 2.2.1. Therefore, we implemented [24] and [53] by letting the judge nodes collect indirect measurements (feedbacks) from the witnesses using Indirect type I feedback as described in Section 4.2.4. Since direct monitoring is not possible in DTNs, we believe that this feedback mechanism is the only option for the nodes. Thus we assumed that, as in our scheme, each judge node collects feedbacks and forms its rating table. Further, each judge node exchanges its rating table with the other nodes upon a contact and then executes the reputation management protocol in [24] or EigenTrust [53].

We used the simulation settings described before with the LW mobility model. We assumed that a definite amount of time (4 hours) has elapsed since the launch of the system as the initialization period, during which new messages are generated by a Poisson distribution at rate $\lambda_m = 1/3000$ at the source nodes and transmitted to their respective destinations. Further, during this initialization period, rating tables were being created at the judge nodes. Then, at time $t = 0$ (after the initialization period)[4], we assumed legitimate nodes simultaneously start new flows to their destinations (while the previous flows may still exist) and attackers start mounting their attacks (both on the network communication protocol and the security system). Therefore, at time $t = 0$, we assumed each legitimate source node has 1000 information packets which are encoded via a rateless code for single-copy routing transmission. Hence, the number of encoded packets required by each destination to recover a message is roughly 1000[5]. We assumed packets with 128 bytes payloads and a data rate of 250 kbps for each link. We note that we used the same routing and

[4]Once the initialization period is elapsed, we set the time as $t = 0$.

[5]It can be shown that when the decoder receives $1000(1+\zeta_{1000})$ packets, where ζ_{1000} is a positive number very close to zero, it can successfully decode all 1000 input packets with high probability [63, 91].

packet exchange protocol for ITRM, Bayesian framework and EigenTrust algorithm (routing and packet exchange protocol is described in Section 4.2.2). We evaluated the data availability and packet delivery ratio for these new flows since time $t = 0$. Moreover, we let each judge node execute ITRM, Bayesian framework, or EigenTrust algorithm starting from time $t = 0$, and hence, we also evaluated the MAE since time $t = 0$. Thus, for all simulations, the plots are shown from time $t = 0$. The percentage of the Byzantine nodes in the network is denoted as W. For ITRM, the Bayesian framework in [24], and EigenTrust [53], we assumed that each judge node randomly picks 10 entries from each rating table it received in order to prevent the malicious users from flooding the mechanism with incorrect entries. We ran each simulation 100 times to get an average. We executed the experiment with different parameters in the LW mobility model (e.g., different Levy distribution coefficients, node speeds, etc.) and obtained similar trends. We further simulated the proposed scheme with the RWP mobility model with $[v_{min}, v_{max}] = [10, 30] m/s$ and ignoring the pause times. We obtained similar trends with the RWP model as the LW mobility model, and hence, we do not illustrate the results of the RWP mobility model.

As before, we present the evaluation only for the bad-mouthing on the detection scheme and bad-mouthing on the trust management (as described in Section 4.2.1), as similar results hold for ballot-stuffing and combinations of bad-mouthing and ballot-stuffing. Malicious nodes provide incorrect feedbacks to the judge nodes about their reliable contacts in order to cause the judge nodes to misjudge the types of reliable nodes (in their verdicts). Further, malicious nodes collaboratively victimize the reliable nodes in their rating tables by rating them as “0” and forward their rating tables whenever they contact with a reliable node to mislead the detection mechanism. In addition to the attacks on the security mechanism (i.e., the trust management and the detection algorithms), malicious nodes mount attacks on the network communication protocol by both dropping the legitimate packets they have received from reliable nodes (with different packet drop rates) and generating their own flows to deliver to other (malicious) nodes via the legitimate nodes. The ultimate goal of the adversary is to degrade the network performance (i.e., data availability and packet delivery ratio).

Mean Absolute Error (MAE): In Fig. 15, we compared the performance of ITRM with the Bayesian reputation management framework in [24] and the EigenTrust algorithm [53] (in the DTN environment presented before) in terms of MAE when the fraction of the malicious raters (W) is 0.30. In other words, for each legitimate judge, we computed the average MAE (between the actual reputation value and the computed reputation value) based on the reputation values computed at that judge node. Further, since each legitimate judge node computes the reputation values (of the other nodes) itself using ITRM, Bayesian framework or EigenTrust, we computed the average MAE over all legitimate nodes.

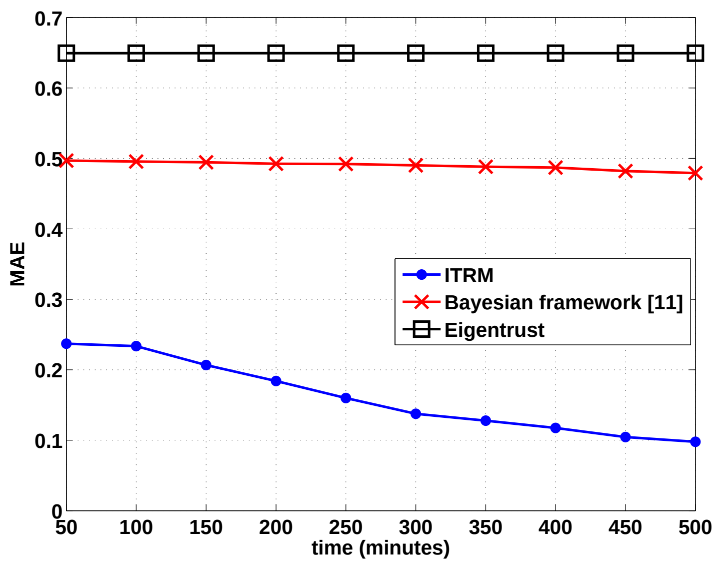


Figure 15: MAE performance of various schemes for bad-mouthing when $W = 0.30$.

From these simulation results, we conclude that ITRM significantly outperforms the Bayesian framework and the EigenTrust algorithm in the presence of attacks.

Further, for different values of W and for different parameters in the LW mobility model, we still observed the superiority of ITRM over the other schemes. We note that since the Bayesian framework shows a better performance than the EigenTrust in terms of MAE, we compare the performance of ITRM with the Bayesian framework for data availability and packet delivery ratio in the rest of this section.

Availability: We define the *availability* as the percentage of recovered messages (by their final destinations) in the network at a given time. In Figs. 16 and 17, we showed the percentage of recovered messages versus time for the following scenarios: i) when there is no defense against the malicious nodes and each malicious node has a packet drop rate of 1, ii) when a detection level of 0.8 is used by ITRM (in which each judge node is supposed to identify and isolate all the Byzantine nodes whose packet drop rates are 0.8 or higher), iii) when a complete detection is used by ITRM (in which all malicious nodes are supposed to be detected and isolated regardless of their packet drop rate), and iv) when the Bayesian reputation management framework in [24] is used to detect the malicious nodes. We note that in the second, third, and fourth scenarios, the packet drop rates by the malicious nodes are uniformly distributed between 0 and 1 in order to make the detection harder. Further, for the second, third, and fourth scenarios, we assume the attack on the security mechanism as described before.

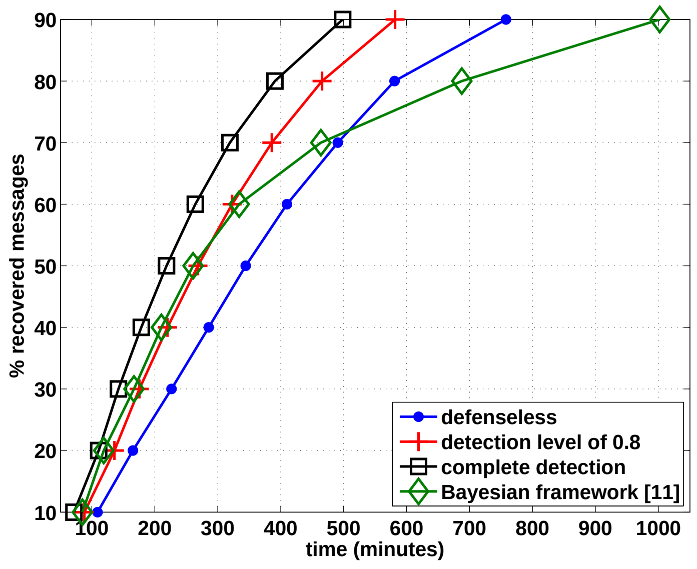


Figure 16: Fraction of the recovered messages versus time for $W = 0.10$ with LW mobility model.

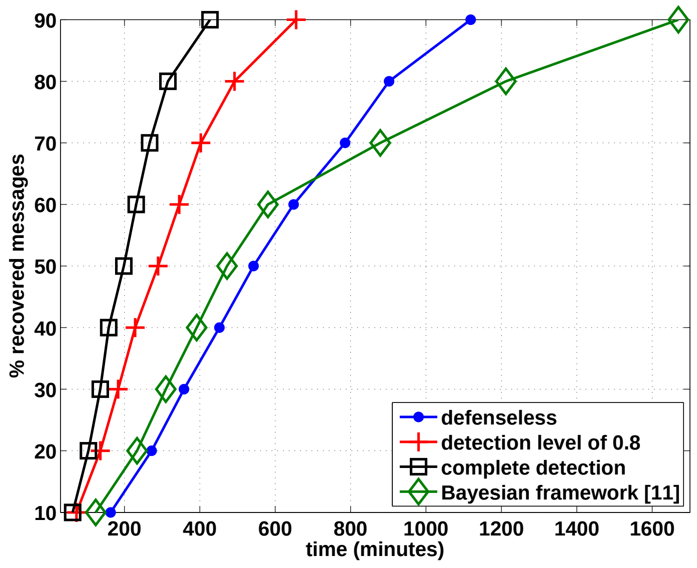


Figure 17: Fraction of the recovered messages versus time for $W = 0.40$ with LW mobility model.

The plots show that the percentage of recovered messages at a given time significantly decreases with increasing W for the defenseless scheme. On the other hand, we observed a considerable improvement in the percentage of recovered messages even after a high level detection ($p = 0.8$) using the proposed scheme. We further observed that the Bayesian reputation management framework in [24] fails to provide high data availability with low latency. This is due to the fact that when the malicious nodes collaboratively attack the reputation management scheme, reputation systems which rely on the Bayesian Approach (such as [24]) result in high MAE in the reputation values of the nodes (as illustrated in Fig. 15). Therefore, the reputation mechanism in [24] not only fails to detect all malicious nodes in the network, but it also labels some reliable nodes (which are victimized by the malicious nodes using the bad-mouthing attack) as malicious. Moreover, we considered the *reliable message delivery* as the probability of the delivery of a single specific message to its destination at any given time. Thus, the probability of recovery (of a specific message) at the destination node at any given time is plotted (while other flows still exist) in Figs. 18 and 19. These figures also illustrate the improvement in reliable message delivery as a result of the proposed scheme even after a high level detection. We again observed that the reputation mechanism in [24] fails to provide fast reliable message delivery due to the vulnerability of the Bayesian reputation management framework to detect malicious nodes.

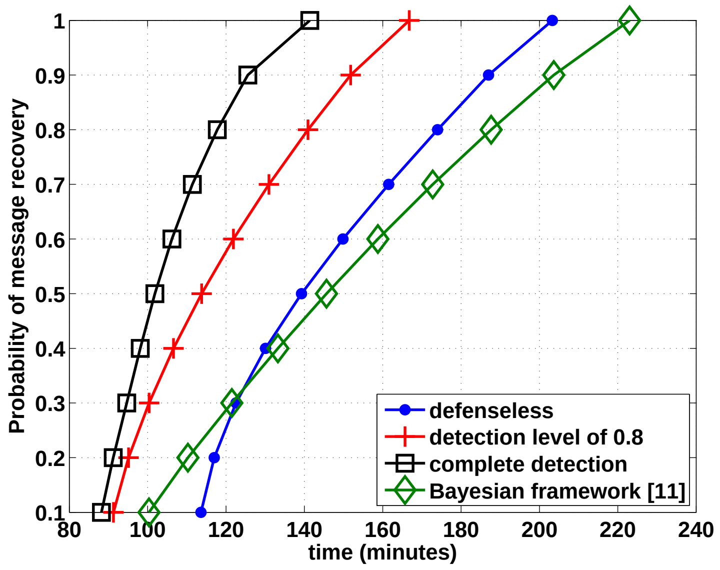


Figure 18: Probability of message recovery for a single flow versus time for $W = 0.10$ with LW mobility model.

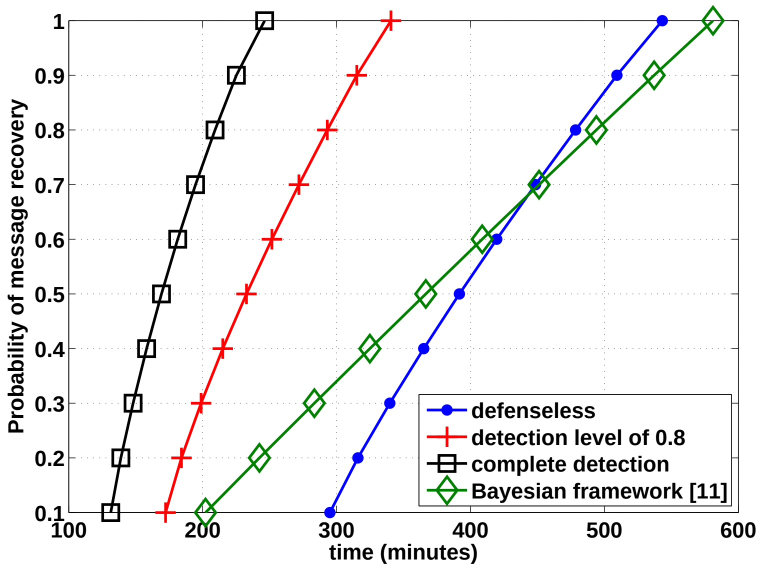


Figure 19: Probability of message recovery for a single flow versus time for $W = 0.40$ with LW mobility model.

Comparing the time required for a high success probability (for detection) in Figs. 13 and 14 and the time required to have high data availability at the receivers, we observed that the ITRM enables the judge nodes to calculate the reputations of all the network nodes in a relatively short amount of time. In other words, the time required to calculate the reputation values of all the network nodes at a judge node is significantly less than the time required for the transmission of a single message, which is a significant result for DTNs.

Packet Delivery Ratio: We define the packet delivery ratio as the ratio of the number of legitimate packets received by their destinations to the number of legitimate packets transmitted by their sources. Therefore, we observed the impact of malicious nodes on the packet delivery ratio and the progress achieved as a result of our scheme in Figs. 20 and 21. As before, we consider i) the defenseless scheme, ii) a detection level of 0.8, iii) a complete detection, and iv) the Bayesian reputation management framework in [24]. We observed a notable improvement in the packet delivery ratio as a result of the proposed scheme. As W increases, the packet delivery ratio of the defenseless scheme decreases significantly while our proposed scheme still provides a high packet delivery ratio even at the detection level of 0.8, which illustrates the robustness of the proposed scheme. Finally, we observed that the scheme in [24] fails to provide a high packet delivery ratio due to its vulnerability against colluding malicious nodes as discussed before.

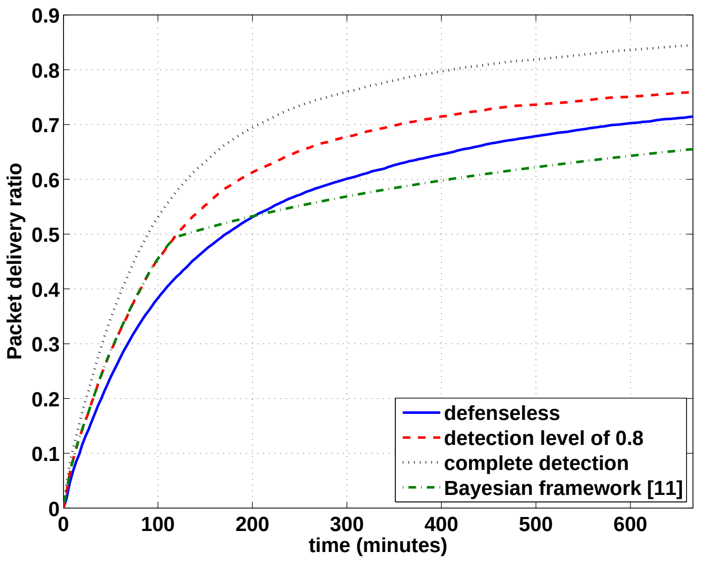


Figure 20: Packet delivery ratio versus time for $W = 0.10$ with LW mobility model.

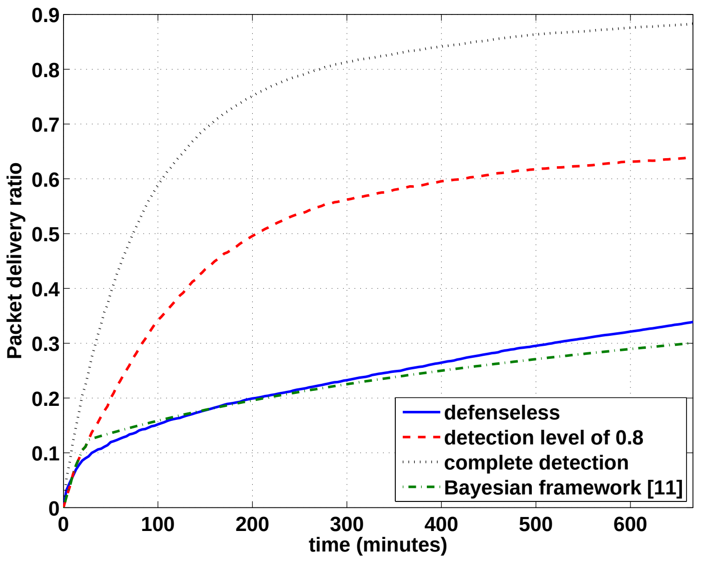


Figure 21: Packet delivery ratio versus time for $W = 0.40$ with LW mobility model.

Overhead due to the trust management scheme: Computation and communication overhead introduced due to the proposed trust management scheme is dominated by the generation, verification, and transmission of the IBS among the judge, suspect, and witness nodes. The crucial parameters of IBC to generate and verify the signatures are as follows [32]. i) G_1 and G_2 are an additive and a multiplicative group, respectively. Moreover, they are cyclic groups of order ϖ. ii) Φ and ϑ are two distinct generators for the group G_1. iii) The discrete logarithm problem is hard in both G_1 and G_2. iv) There is a bilinear pairing, $\hat{e} : G_1 \times G_1 \rightarrow G_2$, such as the modified Tate pairing on a supersingular elliptic curve [19]. The cryptographic primitives that are essential to sign and verify messages are scalar multiplication in G_1, M_{G_1}, exponentiation in G_2, E, and pairing, P. We consider IBS scheme in [32] to calculate the overhead of the proposed trust management scheme. In [32], $1M_{G_1} + 1E$ and $1M_{G_1} + 1E + 1P$ are required to sign and verify a signature, respectively. Moreover, the size of the signature is $G_1 \times Z^*_{\varpi}$. We define the average number of contacts a suspect node requires to deliver all packets it received from a particular judge node as Π. Hence, we calculated the extra overhead of the proposed trust management scheme until a judge node obtains s entries in its trust-table, each generated by at least $M = 20$ feedbacks ($M \geq 20$ is required for a verdict with high confidence as illustrated in Fig. 9). In Table 3 we illustrated the average number of signature generation (Sign), verification (Verify) and signature transfer (Trans) per a judge, suspect and witness node, respectively for $\Pi = 10$ and $\Pi = 20$ (we observed $10 \leq \Pi \leq 20$ on the average based on our simulations). It is worth noting that we did not consider the hash operations upon illustrating the overhead of the proposed trust management scheme because of the low complexity of the hashing operation. We can roughly say that verification of an IBS consumes 1000 times more power than hashing a 64-Byte message [76].

Further, the overhead caused by the extra messages between the nodes due to the security protocol is negligible when compared with the data packets. This is because the overhead due to the security mechanism is dominated by the signed receipts from the suspect nodes to prove the deliveries by the suspect nodes. As we mentioned before, knowing how long it takes to have a high success probability at a judge node

Table 3: Overhead of the proposed trust management scheme for $\Pi = 10$ and $\Pi = 20$.

	judge		suspect		witness	
Π	10	20	10	20	10	20
Sign	0	0	$0.1 \times s$	$0.1 \times s$	$0.1 \times s$	$0.1 \times s$
Verify	$7.9 \times s$	$19.9 \times s$	0	0	0	0
Trans	0	0	$17 \times s$	$32 \times s$	$0.2 \times s$	$0.4 \times s$

for a given detection level (from the results in Figs. 13 and 14), the witnesses can determine the lifetimes of the signed receipts. For example, in the LW mobility model used, the scheme provides a high probability of success (S) in approximately 70 minutes. Therefore, the lifetime of a signed receipt is estimated as 70 minutes, on the average. Moreover, for the chosen mobility model, each node establishes (on the average) 30 contacts in 70 minutes. This means that a suspect node transfers approximately 30 signed receipts to a witness node upon its contact. Since the length of the signature is about 20 bytes [109] and the size of a data packet is 128 bytes, 30 signed receipts can be delivered via 5 data packets. Considering the data rates of 250 kbps, the overhead of 5 data packets becomes negligible when compared to the entire message exchange between two nodes during the contact. This also shows that the proposed algorithm does not introduce a significant overhead burden on the network.

4.3 Summary

In this chapter, we introduced a robust and efficient security mechanism for delay tolerant networks (DTNs). The proposed security mechanism consists of a trust management mechanism and an iterative reputation management scheme (ITRM). The trust management mechanism enables each network node to determine the trustworthiness of the nodes with which it had direct transactions. On the other hand, ITRM takes advantage of an iterative mechanism to detect and isolate the malicious nodes from the network in a short time. We studied the performance of the proposed scheme and showed that it effectively detects the malicious nodes even in the presence of the attacks on the trust and detection mechanism. We also illustrated that the proposed scheme is far more effective than the Bayesian framework and EigenTrust in computing the reputation values in a DTN environment. Moreover, using computer simulations we showed that the proposed mechanism provides high data availability with low information latency by detecting and isolating the malicious nodes in a short time.

CHAPTER V

ITERATIVE TRUST AND REPUTATION MANAGEMENT USING BELIEF PROPAGATION

5.1 Introduction

As we discussed in Chapter 1, we believe that trust and reputation management systems will lead various applications from the social web to ad-hoc networks in near future and there are needs for scalable and attack resilient reputation systems. In this chapter, we introduce the first application of the *Belief Propagation* (BP) algorithm in the design and evaluation of trust and reputation management systems. In our previous work, inspired by the earlier work on iterative decoding of error-control codes in the presence of stopping sets [74, 75, 98], we proposed an algebraic iterative algorithm [11] referred as ITRM for reputation systems (described in Chapter 3) and showed the benefit of using iterative algorithms for trust and reputation management. Here, we expand this work and introduce a fully probabilistic approach based on the BP algorithm. Different from ITRM, in this chapter, we view the reputation management problem as an inference problem and describe it as computing marginal likelihood distributions from complicated global functions of many variables. Further, we utilize the BP algorithm to efficiently (in linear complexity) compute these marginal probability distributions. The work is inspired by earlier work on graph-based iterative probabilistic decoding of turbo codes and low-density parity-check (LDPC) codes, the most powerful practically decodable error-control codes known. These decoding algorithms are shown to perform at error rates near what can be achieved by the optimal scheme, maximum likelihood decoding, while requiring far less computational complexity (i.e., linear in the length of the code). We believe that the significant benefits offered by the iterative probabilistic algorithms can be also tapped in to benefit the field of reputation systems.

We introduce the "Belief Propagation-Based Iterative Trust and Reputation Management Scheme" (BP-ITRM). BP algorithm [59,70,110] (discussed in Section 5.1.2) is usually described in terms of operations on factor graphs. In BP-ITRM, the sellers (i.e., service providers) and buyers (i.e., consumers or raters) are represented via a factor graph on which they are arranged as two sets of variable and factor nodes that are connected via some edges. The reputation can be computed by message passing between nodes in the graph. In each iteration of the algorithm, all the variable nodes (sellers), and subsequently all the factor nodes (buyers), pass new messages to their neighbors until the reputation value converges. We show that the proposed iterative scheme is reliable (in filtering out malicious/unreliable reports). Further, we prove that BP-ITRM iteratively reduces the error in the reputation values of service providers due to the malicious raters with a high probability. We observe that this probability suddenly drops if the fraction of malicious raters exceeds a threshold. Hence, the scheme has a *threshold* property.

The proposed reputation management algorithm can be utilized in well-known online services such as eBay or Epinions. In eBay, each seller-buyer pair rate each other after a transaction. Thus, BP-ITRM can be used in eBay to compute the reputation values of the sellers and buyers along with the trustworthiness values of the peers in their ratings. Epinions, on the other hand, is a product review site in which users can rate and review items. Users can also give ratings to the reviews. Hence, the ratings of members on a review and on a product are considered separately. BP-ITRM can be utilized in such an environment to compute the reputations of the reviewers based on the ratings given by the users on the reviews. Although we present the proposed algorithm as a centralized approach, BP-based trust and reputation management can also be utilized in decentralized systems such as ad-hoc networks and P2P systems to compute the reputations of the peers in the network (as we will discuss in Chapter 6).

5.1.1 Contributions

The main contributions of our work are summarized in the following.

1. We introduce the first application of the Belief Propagation (BP) algorithm on trust and reputation management systems.

2. As the core of our trust and reputation management system, we use the BP algorithm which is proven to be a powerful tool on decoding of turbo codes and LDPC codes. Therefore, we introduce a graph-based trust and reputation management mechanism that relies on an appropriately chosen factor graph and computes the reputation values of service providers (sellers) by a message passing algorithm.

3. The proposed iterative algorithm computes the reputation values of the service providers (sellers) accurately (with a small error) in a short amount of time in the presence of attackers. The scheme is also a robust and efficient methodology for detecting and filtering out malicious ratings. Further, the scheme detects the malicious raters with a high accuracy, and updates their trustworthiness accordingly enforcing them to execute low grade attacks to remain undercover.

4. The proposed BP-ITRM significantly outperforms the existing and commonly used reputation management techniques such as the Averaging Scheme, Bayesian Approach as in [25, 103] and Cluster Filtering in the presence of attackers.

5.1.2 Belief Propagation

Belief propagation (BP) [59, 70, 110] is a message passing algorithm for performing inference on graphical models (factor graphs, Bayesian networks, Markov random fields). It is a method for computing marginal distributions of the unobserved nodes conditioned on the observed ones. Computing marginal distributions is hard in general as it might require summing an exponentially large number of terms. Hence, BP algorithm is usually described in terms of operations on a factor graph. A factor graph is a bipartite graph containing nodes corresponding to variables and factors with edges between them. A factor graph has a variable node for each variable x_i, a factor node for each function f_a, and an edge connecting variable node i to the factor node a if and only if x_i is an argument of f_a. The marginal distribution of an unobserved node can be computed accurately using the BP algorithm if the factor graph has no cycles. However, the algorithm is still well defined and often gives good approximate results even for the factor graphs with cycles (as it has been observed in decoding of LDPC codes).

BP algorithm simply works by passing messages between the factor and variable nodes on the factor graph. The message $\lambda_{a\to i}(x_i)$ from the factor node a to the variable node i can be interpreted as a statement about the relative probabilities that x_i is in its different states based on the function f_a. On the other hand, the message $\mu_{i\to a}(x_i)$ from the variable node i to the factor node a can be interpreted as a statement about the relative probabilities that x_i is in different states based on all the information node i has except for that based on the function f_a. The messages are updated according to the following rules [59]:

$$\lambda_{a\to i}(x_i) = \sum_{\mathbf{x}_a \backslash x_i} f_a(\mathbf{x}_a) \prod_{j\in N_a\backslash i} \mu_{j\to a}(x_j) \tag{16}$$

and

$$\mu_{i\to a}(x_i) = \prod_{c\in N_i\backslash a} \lambda_{c\to i}(x_i). \tag{17}$$

Here, $N_i\backslash a$ denotes all the nodes that are neighbors of node i except for node a. Further, $\sum_{\mathbf{x}_a\backslash x_i}$ denotes a sum over all the variables $\mathbf{x}_a$ that are arguments of f_a except x_i.

BP is commonly used in artificial intelligence and information theory. It has demonstrated empirical success in numerous applications including turbo codes, free energy approximation, satisfiability, and LDPC codes. While the optimal decoding technique of LDPC codes, maximum likelihood (ML) decoding, is an NP problem, BP algorithm provides a very efficient decoding that gets close to the bit error rate (BER) performance of the ML decoding when the code length becomes large. In other words, BP performs at error rates near what can be achieved by the optimal scheme while requiring far less computational complexity. Here, we exploit such benefits in trust and reputation management systems.

5.2 Belief Propagation for Iterative Trust and Reputation Management (BP-ITRM)

As in Chapter 3, we have two main goals: 1. computing the service quality (reputation) of the peers who provide a service (henceforth referred to as Service Providers or SPs) by using the feedbacks from the peers who used the service (referred to as the raters), and 2. determining the trustworthiness of the raters by analyzing their feedback about SPs. We assume two different sets in the system: i) the set of service providers, $\mathbb{S}$ and ii) the set of service consumers (hereafter referred as raters), $\mathbb{U}$. We note that these two sets are not necessarily disjoint. Transactions occur between SPs and raters, and raters provide feedbacks in the form of ratings about SPs after each transaction.

Let G_j be the reputation value of SP j ($j \in \mathbb{S}$) and T_{ij} be the rating that rater i ($i \in \mathbb{U}$) reports about SP 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 rater. In other words, R_i represents the amount of confidence that the reputation system has about the correctness of any feedback/rating provided by rater i. All of these parameters may evolve with time. However, for simplicity, we omitted time dependencies from the notation. We assume there are u raters and s SPs in the system (i.e., $|\mathbb{U}| = u$ and $|\mathbb{S}| = s$). 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 SPs and the trustworthiness values of the raters, respectively. Further, let $\mathbb{T}$ be the $s \times u$ SP-rater matrix that stores the rating values (T_{ij}), and $\mathbb{T}_i$ be the set of ratings provided by rater i. We consider slotted time throughout this discussion. At each time-slot (or epoch), the iterative reputation algorithm is executed using the input parameters $\mathbb{R}$ and $\mathbb{T}$ to obtain the reputation parameters (e.g., $\mathbb{G}$). After completing its iterations, the BP-ITRM scheme outputs new global reputations of the SPs as well as the trustworthiness ($\mathbb{R}$ values) of the raters. 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 (we implemented the proposed scheme for both cases and illustrate its performance in Section 5.3.3).

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 SP j. Loosely speaking, the present Bayesian approaches [25, 103] solve for these marginal distributions separately, leading to poor estimates as they neglect the interplay of the entire evidence. In contrast, 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 raters. Then, clearly, each marginal probability function $p(G_j|\mathbb{T}, \mathbb{R})$ may be obtained as follows:

$$p(G_j|\mathbb{T}, \mathbb{R}) = \sum_{\mathbb{G}\setminus\{G_j\}} p(\mathbb{G}|\mathbb{T}, \mathbb{R}), \tag{18}$$

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

Unfortunately, the number of terms in (18) grows exponentially with the number of variables, making the computation infeasible for large-scale systems even for binary reputation values. However, we propose to factorize (18) to local functions f_i using a factor graph and utilize the BP algorithm to calculate the marginal probability distributions in linear complexity. A factor graph is a bipartite graph containing two sets of nodes (corresponding to variables and factors) and edges incident between two sets. Following [59], 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 is still well-defined and empirically often gives good approximate results for the factor graphs with cycles [106].

To describe the reputation system, we arrange the collection of the raters and the SPs together with their associated relations (i.e., the ratings of the SPs by the raters) as a bipartite (or factor) graph, as in Fig. 22. In this representation, each rater peer corresponds to a factor node in the graph, shown as a square. Each SP is represented by a variable node shown as a hexagon in the graph. Each report/rating is represented by an edge from the factor node to the variable node. Hence, if a rater i ($i \in \mathbb{U}$) has a report about SP j ($j \in \mathbb{S}$), we place an edge with value T_{ij} from the factor node i to the variable node representing SP j. We note that the T_{ij} value between rater i and SP j is the aggregation of all past and present ratings between these two peers as described in the following. If any new rating arrives from rater i about SP j, our scheme updates the value T_{ij} by averaging the new rating and the old value of the edge multiplied with the fading factor. The factor $\gamma_{ij}(t)$ is used to incorporate the fading factor of the SPs' reputation (service quality). We use a known 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 rater i and SP j occurred, respectively. The parameter ϑ is chosen to be less than one to give greater importance to more recent ratings.

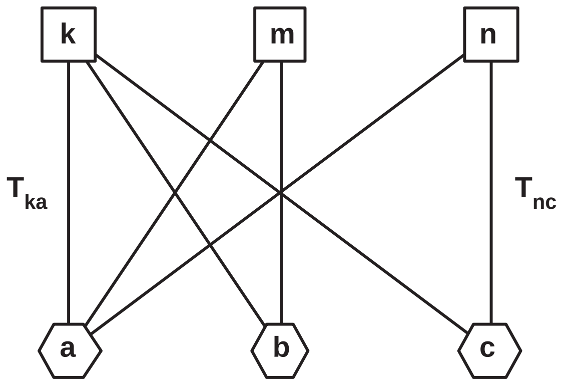


Figure 22: Factor graph between the SPs and the raters in (20).

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[1]:

$$p(\mathbb{G}|\mathbb{T},\mathbb{R}) = \frac{1}{Z}\prod_{i\in\mathbb{U}} f_i(\mathcal{G}_i, \mathbb{T}_i, R_i), \tag{19}$$

where Z is the normalization constant and $\mathcal{G}_i$ is a subset of $\mathbb{G}$. Hence, in the graph representation of Fig. 22, 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 rater. As an example, the factor graph in Fig. 22 corresponds to

$$p(G_a, G_b, G_c|\mathbb{T},\mathbb{R}) = \frac{1}{Z} f_k(G_a, G_b, G_c, T_{ka}, T_{kb}, T_{kc}, R_k)\times$$
$$f_m(G_a, G_b, T_{ma}, T_{mb}, R_m) \times f_n(G_a, G_c, T_{na}, T_{nc}, R_n). \tag{20}$$

We note that using (20) in (18), one can attempt to compute the marginal distributions. However, as discussed before, this can get computationally infeasible. Instead, we utilize the BP algorithm to calculate the marginal distributions of the variables in $\mathbb{G}$.

We now introduce the messages between the factor and the variable nodes to compute the marginal distributions using BP. We note that all the messages are

[1]It is shown that such a factorization eventually gives the marginal probability distributions via the BP algorithm [59].

formed by the algorithm that is ran in the central authority. To that end, we choose an arbitrary factor graph as in Fig. 23 and describe message exchanges between rater k and SP a. We represent the set of neighbors of the variable node (SP) a and the factor node (rater) k as $\mathbf{N_a}$ and $\mathbf{N_k}$, respectively (neighbors of a SP are the set of raters who rated the SP while neighbors of a rater are the SPs whom it rated). Further, let $\Xi = \mathbf{N_a} \backslash \{k\}$ and $\Delta = \mathbf{N_k} \backslash \{a\}$. The BP algorithm iteratively exchanges the probabilistic messages between the factor and the variable nodes in Fig. 23, updating the degree of beliefs on the reputation values of the SPs as well as the confidence of the raters on their ratings (i.e., trustworthiness values) at each step, until convergence. 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.

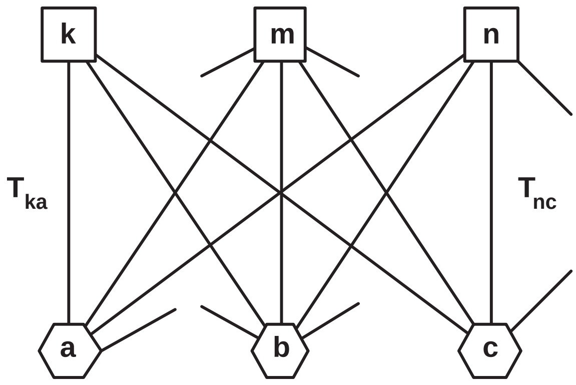


Figure 23: Setup of the scheme.

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

$$\lambda_{k \to a}^{(\nu)}(G_a^{(\nu)}) = \sum_{\mathcal{G}_k^{(\nu)} \backslash \{G_a^{(\nu)}\}} f_k(\mathcal{G}_k^{(\nu)}, \mathbb{T}_k, R_k^{(\nu-1)}) \prod_{x \in \Delta} \mu_{x \to k}^{(\nu-1)}(G_x^{(\nu)}), \tag{21}$$

where $\mathcal{G}_k^{(\nu)}$ 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. 24. Further, $R_k^{(\nu-1)}$ (the trustworthiness of rater k calculated at the end of $(\nu-1)^{th}$ iteration) is a value between zero and one and can be calculated as follows:

$$R_k^{(\nu-1)} = 1 - \frac{1}{|N_k|} \sum_{i \in N_k} \sum_{x \in \{0,1\}} |T_{ki} - x| \mu_{i \to k}^{(\nu-1)}(x). \tag{22}$$

The above equation can be interpreted as one minus the average inconsistency of rater k calculated by using the messages it received from all its neighbors. Using (21) and assuming that the arguments of a local function at a factor node are independent from each other (to reduce the computational complexity), it can be shown that

$$f_k(\mathcal{G}_k^{(\nu)}, \mathbb{T}_k, R_k^{(\nu-1)}) = \prod_{i \in \mathbf{N_k}} f_k(G_i^{(\nu)}, \mathbb{T}_k, R_k^{(\nu-1)}). \tag{23}$$

Thus, the message in (21) becomes

$$\begin{aligned}\lambda_{k\to a}^{(\nu)}(G_a^{(\nu)}) =& f_k(G_a^{(\nu)}, \mathbb{T}_k, R_k^{(\nu-1)}) \times \\ & \Big\{ \sum_{\mathcal{G}_k^{(\nu)} \backslash \{G_a^{(\nu)}\}} \Big[\prod_{i \in \mathbf{N_k} \backslash \{a\}} f_k(G_i^{(\nu)}, \mathbb{T}_k, R_k^{(\nu-1)}) \prod_{x \in \Delta} \mu_{x \to k}^{(\nu-1)}(G_x^{(\nu)}) \Big] \Big\}.\end{aligned} \tag{24}$$

Since the second part of (24) is a constant, $\lambda_{k\to a}^{(\nu)}(G_a^{(\nu)}) \propto f_k(G_a^{(\nu)}, \mathbb{T}_k, R_k^{(\nu-1)})$, and hence, $\lambda_{k\to a}^{(\nu)}(G_a^{(\nu)}) \propto p(G_a^{(\nu)}|T_{ka}, R_k^{(\nu-1)})$, where

$$\begin{aligned}& p(G_a^{(\nu)}|T_{ka}, R_k^{(\nu-1)}) = \\ & \Big[(R_k^{(\nu-1)} + \frac{1-R_k^{(\nu-1)}}{2})T_{ka} + \frac{1-R_k^{(\nu-1)}}{2}(1-T_{ka})\Big] G_a^{(\nu)} + \\ & \Big[\frac{1-R_k^{(\nu-1)}}{2}T_{ka} + (R_k^{(\nu-1)} + \frac{1-R_k^{(\nu-1)}}{2})(1-T_{ka})\Big](1-G_a^{(\nu)}).\end{aligned} \tag{25}$$

This resembles the belief/pleusability concept of the Dempster-Shafer Theory [89,90]. Given $T_{ka} = 1$, $R_k^{(\nu-1)}$ can be viewed as the belief of the k^{th} rater that $G_a^{(\nu)}$ is one (at the ν^{th} iteration). In other words, in the eyes of rater k, $G_a^{(\nu)}$ is equal to one with probability $R_k^{(\nu-1)}$. Thus, $(1 - R_k^{(\nu-1)})$ corresponds to the uncertainty in the belief of rater k. In order to remove this uncertainty and express $p(G_a^{(\nu)}|T_{ka}, R_k^{(\nu-1)})$ as the probabilities that $G_a^{(\nu)}$ is zero and one, we distribute the uncertainty uniformly between two outcomes (one and zero). Hence, in the eyes of the k^{th} rater, $G_a^{(\nu)}$ is equal to one with probability $(R_k^{(\nu-1)} + (1 - R_k^{(\nu-1)})/2)$, and zero with probability $((1 - R_k^{(\nu-1)})/2)$. 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 raters. Instead, we consider the belief of each rater individually and calculate probabilities that $G_a^{(\nu)}$ being one and zero in the eyes of each rater as in (25). The above computation must be performed for every neighbors of each factor nodes. This finishes the first half of the ν^{th} iteration.

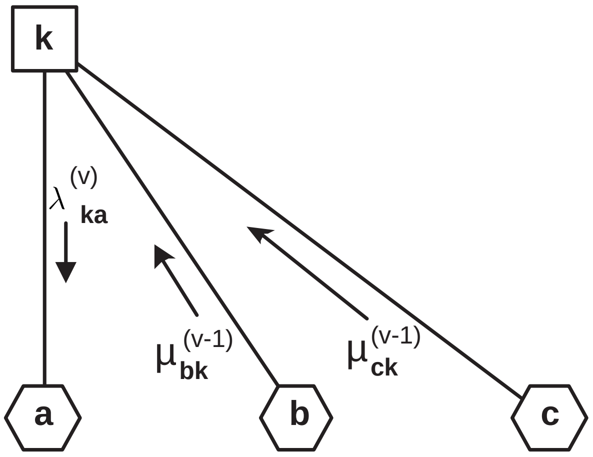


Figure 24: Message from the factor node k to the variable node a at the ν^{th} iteration.

During the second half, the variable nodes generate their messages (μ) and send to their neighbors. Variable node a forms $\mu_{a\to k}^{(\nu)}(G_a^{(\nu)})$ by multiplying all information it receives from its neighbors excluding the factor node k, as shown in Fig. 25. Hence, the message from variable node a to the factor node k at the ν^{th} iteration is given by

$$\mu_{a\to k}^{(\nu)}(G_a^{(\nu)}) = \frac{1}{\sum\limits_{h\in\{0,1\}} \prod\limits_{i\in\Xi} \lambda_{i\to a}^{(\nu)}(h)} \times \prod_{i\in\Xi} \lambda_{i\to a}^{(\nu)}(G_a^{(\nu)}) \tag{26}$$

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.

We note that the iterative algorithm starts its first iteration by computing $\lambda_{k\to a}^{(1)}(G_a^{(1)})$ in (21). However, instead of calculating in (22), the trustworthiness value R_k from the previous execution of BP-ITRM is used as initial values in (25).

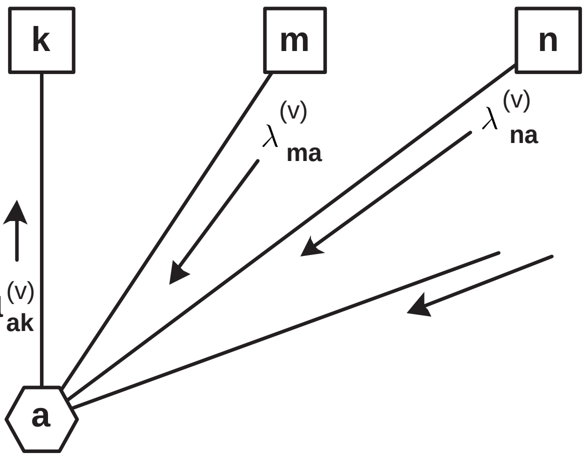


Figure 25: Message from the variable node a to the factor node k at the ν^{th} iteration.

The iterations stop when all variables in $\mathbb{G}$ converge. Therefore, at the end of each iteration, the reputations are calculated for each SP. To calculate the reputation value $G_a^{(\nu)}$, we first compute $\mu_a^{(\nu)}(G_a^{(\nu)})$ using (26) but replacing Ξ with $\mathbf{N_a}$, and then we set $G_a^{(\nu)} = \sum_{i\in\Upsilon} i\mu_a^{(\nu)}(i)$.

5.3 Security Evaluation of BP-ITRM via User Modeling

In this section, we mathematically model and analyze BP-ITRM. Moreover, we support the analysis via computer simulations and compare BP-ITRM with the existing and commonly used trust management schemes. In order to facilitate future references, frequently used notations are listed in Table 4.

5.3.1 Attack Models

We consider two major attacks that are common for any trust and reputation management mechanisms. Further, we assume that the attackers may collude and collaborate with each other:

Table 4: Notations and definitions.

$\mathbb{S}$	The set of service providers (SPs)
$\mathbb{U}_M$	The set of malicious raters
$\mathbb{U}_R$	The set of reliable raters
r_h	Report (rating) given by a reliable rater
r_m	Report (rating) given by a malicious rater
d	Total number of newly generated ratings, per time-slot, per a reliable rater
b	Total number of newly generated ratings, per time-slot, per a malicious rater

- **Bad-mouthing:** Malicious raters collude and attack the service providers with the highest reputation by giving low ratings in order to undermine them. It is also noted that in addition to the malicious peers, in some applications, bad-mouthing may be originated by a group of selfish peers who attempt to weaken high-reputation providers in the hope of improving their own chances as providers.

- **Ballot-stuffing:** Malicious raters collude to increase the reputation value of peers with low reputations. Just as in bad-mouthing, in some applications, this could be mounted by a group of selfish consumers attempting to favor their allies.

5.3.2 Analytic Evaluation

We adopted the following models for various peers involved in the reputation 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 quality of each service provider remains unchanged during time-slots. Moreover, the rating values are either 0 or 1 where 1 represents a good service quality. Ratings generated by the non-malicious raters are distributed uniformly among the SPs (i.e., their ratings/edges in the graph representation are distributed uniformly among SPs). We further assumed that the rating value r_h (provided by the non-malicious raters) is a random variable with Bernoulli distribution, where $Pr(r_h = \hat{G}_j) = p_c$ and $Pr(r_h \neq \hat{G}_j) = (1 - p_c)$, and $\hat{G}_j$ is the actual value of the global reputation of SP j. To the advantage of malicious raters, we assumed that a total of T time-slots had passed since the initialization of the system and a fraction of the existing raters change behavior and become malicious after T time-slots. In other words, malicious raters behaved like reliable raters before mounting their attacks at the $(T+1)^{th}$ time-slot. Finally, we assumed that d is a random variable with Yule-Simon distribution, which resembles the power-law distribution used in modeling online systems [92], with the probability mass function $f_d(d; \rho) = \rho B(d, \rho + 1)$, where B is the Beta function. For modeling the adversary, we made the following assumptions. We assumed that the malicious raters initiate bad-mouthing and collude while attacking the SPs (they attack the SPs who have the highest reputation values by rating them as $r_m = 0$). Further, the malicious raters attack the same set Γ of SPs at each time-slot. In other words, we denote by Γ the set of size b in which every victim SP has one edge from each of the malicious raters. We wish to evaluate the performance for the time-slot $(T + 1)$. It is worth noting that even though we discuss the details for bad-mouthing attack, similar counterpart results hold for ballot-stuffing and combinations of bad-mouthing and ballot-stuffing as well.

ϵ-optimal Scheme: The performance of a reputation scheme is determined by its accuracy of estimating the global reputations of the SPs. We declare a reputation scheme to be ϵ-optimal if the mean absolute error (MAE) ($|G_j - \hat{G}_j|$) is less than or equal to ϵ for every SP.

Naturally, we need to answer the following question: For a fixed ϵ, what are the conditions to have an ϵ-optimal scheme? In order to answer this question we require two conditions to be satisfied: 1) the scheme should iteratively reduce the impact of malicious raters and decrease the error in the reputation values of the SPs until it converges, and 2) the error on the G_j value of each SP j should be less than or equal to ϵ once the scheme converges. In the following, we obtained the condition to arrive at the ϵ-optimal scheme. Although the discussions of the analysis are based on bad-mouthing attack, 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 global reputation values of the victim SPs. Hence, G_j value of a victim SP j should be a non-decreasing function of iterations. This leads to the first condition on the ϵ-optimal scheme.

Lemma 5.3.1. (Condition 1): *The error in the reputation values of the SPs decreases with each successive iterations (until convergence) if $G_a^{(2)} > G_a^{(1)}$ is satisfied with high probability for every SP a ($a \in \mathbb{S}$) with $\hat{G}_a = 1$*[2].

Proof. Let $G_a^{(\omega)}$ and $G_a^{(\omega+1)}$ be the reputation value of an arbitrary SP a with $\hat{G}_a = 1$ calculated at the $(\omega)^{th}$ and $(\omega+1)^{th}$ iterations, respectively. $G_a^{(\omega+1)} > G_a^{(\omega)}$ if the following is satisfied at the $(\omega+1)^{th}$ iteration.

$$\prod_{j\in\mathbb{U}_R\cap\mathbf{N_a}} \frac{2p_cR_j^{(w+1)}+1-R_j^{(w+1)}}{-2p_cR_j^{(w+1)}+1+R_j^{(w+1)}} \prod_{j\in\mathbb{U}_M\cap\mathbf{N_a}} \frac{1-\hat{R}_j^{(w+1)}}{1+\hat{R}_j^{(w+1)}} >$$
$$\prod_{j\in\mathbb{U}_R\cap\mathbf{N_a}} \frac{2p_cR_j^{(w)}+1-R_j^{(w)}}{-2p_cR_j^{(w)}+1+R_j^{(w)}} \prod_{j\in\mathbb{U}_M\cap\mathbf{N_a}} \frac{1-\hat{R}_j^{(w)}}{1+\hat{R}_j^{(w)}}, \tag{27}$$

where $R_j^{(w)}$ and $\hat{R}_j^{(w)}$ are the trustworthiness values of a reliable and malicious rater calculated as in (22) 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 $j \in \mathbb{U}_M \cap \mathbf{N_a}$ and $R_j^{(w+1)} \geq R_j^{(w)}$ for $j \in \mathbb{U}_R \cap \mathbf{N_a}$. Thus, (27) 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 $j \in \mathbb{U}_M \cap \mathbf{N_a}$ and $R_j^{(w+1)} < R_j^{(w)}$ for $j \in \mathbb{U}_R \cap \mathbf{N_a}$. Hence, (27) is not satisfied at the $(w+1)^{th}$ iteration. Therefore, if $G_a^{(\omega)} > G_a^{(\omega-1)}$ holds for some iteration ω, then the

[2] *The opposite must hold for any SP with $\hat{G}_a = 0$.*

BP-ITRM algorithm reduces the error on the global reputation value (G_a) until the iterations stop[3], and hence, it is sufficient to satisfy $G_j^{(2)} > G_j^{(1)}$ with high probability for every SP j with $\hat{G}_j = 1$ (the set of SPs from which the victims are taken) to guarantee that BP-ITRM iteratively reduces the impact of malicious raters until it stops. $\square$

Although because of the *Condition* 1, the error in the reputation values of the SPs decrease with successive iterations, it is unclear what would be the eventual impact of malicious raters. Hence, in the following, we derive the probability P for ϵ-optimality.

Lemma 5.3.2. (Condition 2): *Suppose that the Condition 1 is met. Let ν be the iteration at which the algorithm has converged. Then, BP-ITRM would be an ϵ-optimal scheme with probability P, where P is given as below:*

$$P = \prod_{a \in \mathbb{S}} Pr\Big\{\epsilon \geq 1 - \frac{\varpi}{\varpi + \zeta}\Big\} \tag{28}$$

where,

$$\varpi = \prod_{j \in \mathbb{U}_R \cap \mathbf{N_a}} (2p_c R_j^{(\nu+1)} + 1 - R_j^{(\nu+1)}) \prod_{j \in \mathbb{U}_M \cap \mathbf{N_a}} (1 - \hat{R}_j^{(\nu+1)}), \tag{29a}$$

$$\zeta = \prod_{j \in \mathbb{U}_R \cap \mathbf{N_a}} (-2p_c R_j^{(\nu+1)} + 1 + R_j^{(\nu+1)}) \prod_{j \in \mathbb{U}_M \cap \mathbf{N_a}} (1 + \hat{R}_j^{(\nu+1)}). \tag{29b}$$

Proof. Given *Condition* 1 is satisfied, G_a value of an arbitrary SP a (with $\hat{G}_a = 1$) increases with iterations. Let BP-ITRM converges at the ν^{th} iteration. Then, to have an ϵ-optimal scheme, G_a value calculated at the last iteration of BP-ITRM ($G_a^{(\nu)}$) should result in an error less than or equal to ϵ for every SP. That is, the following should hold for every SP.

$$1 - G_a^{(\nu)} \leq \epsilon. \tag{30}$$

[3]Since the rating values are either 0 or 1, G_a values cannot be negative or above 1. Further, since the error decreases with each successive iterations, G_a values converge at some iteration.

Further, if the scheme continues one more iteration after convergence, it can be shown that

$$G_a^{(\nu+1)} = G_a^{(\nu)}. \tag{31}$$

Thus, combining (30) and (31) leads to (28). □

We note that *Conditions* 1 and 2 in Lemmas 5.3.1 and 5.3.2 are to give an insight about the performance of the algorithm prior to the implementation. Hence, these conditions do not need to be checked at each execution of BP-ITRM in the real-life implementation of the algorithm.

Finally, the variation of the probability of BP-ITRM being an ϵ-optimal scheme over time is an important factor affecting the performance of the scheme. We observed that given BP-ITRM satisfies *Condition* 1 (that the error in the reputation values of the SPs monotonically decreases with iterations), the probability of BP-ITRM being an ϵ-optimal scheme increases with time. This criteria is given by the following lemma:

Lemma 5.3.3. *Let P_{T+1} and P_{T+2} be the probabilities that BP-ITRM is ϵ-optimal at the $(T+1)^{th}$ and $(T+2)^{th}$ time-slots, respectively. Then, given Condition 1 holds at the $(T+1)^{th}$ time-slot, we have $P_{T+2} > P_{T+1}$.*

Proof. Due to the fading factor, the contributions of the past reliable ratings of the malicious raters to their R_i values become less dominant with increasing time. Let $R_i(T)$ and $\hat{R}_i(T)$ be the trustworthiness of a reliable and malicious rater at the T^{th} time-slot, respectively. Then, given that *Condition* 1 is satisfied at the $(T+1)^{th}$ time-slot, it can be shown that $R_i(T+1) \geq R_i(T)$ and $\hat{R}_i(T+1) < \hat{R}_i(T)$. Thus, the probability that BP-ITRM satisfies *Condition* 1 increases at the $(T+2)^{th}$ time-slot. □

In the following example, we illustrate the results of our analytical evaluation. The parameters we used are $|\mathbb{U}_M| + |\mathbb{U}_R| = 100$, $|\mathbb{S}| = 100$, $\rho = 1$, $\vartheta = 0.9$, $T = 50$,

$b = 5$ and $p_c = 0.8$. We note that there is no motive to select these parameters. We evaluated BP-ITRM with different parameters and obtained similar results. BP-ITRM works properly when the error in the reputation values of the SPs decreases monotonically with iterations until convergence. In other words, *Condition* 1 (in Lemma 5.3.1) is a fundamental requirement. In Fig. 26 we illustrated the probability of BP-ITRM to satisfy *Condition* 1 versus fraction of malicious raters. We observed that BP-ITRM satisfies *Condition* 1 with a high probability for up to 30% malicious raters. Further, we observed a threshold phenomenon. That is, the probability of BP-ITRM to satisfy *Condition* 1 suddenly drops after exceeding a particular fraction of malicious raters. Next, in Fig. 27, we illustrated the probability of BP-ITRM being an ϵ-optimal scheme versus fraction of malicious raters for three different ϵ values. Again, we observed a threshold phenomenon. As the fraction of adversary exceeds a certain value, the probability of BP-ITRM being an ϵ-optimal scheme drops sharply. Moreover, Fig. 28 illustrates the average ϵ values (ϵ_{av}) for which BP-ITRM is an ϵ-optimal scheme with high probability for different fractions of malicious raters. We observed that BP-ITRM provides significantly small error values for up to 30% malicious raters. We note that these analytical results are also consistent with our simulation results that are illustrated in the next section.

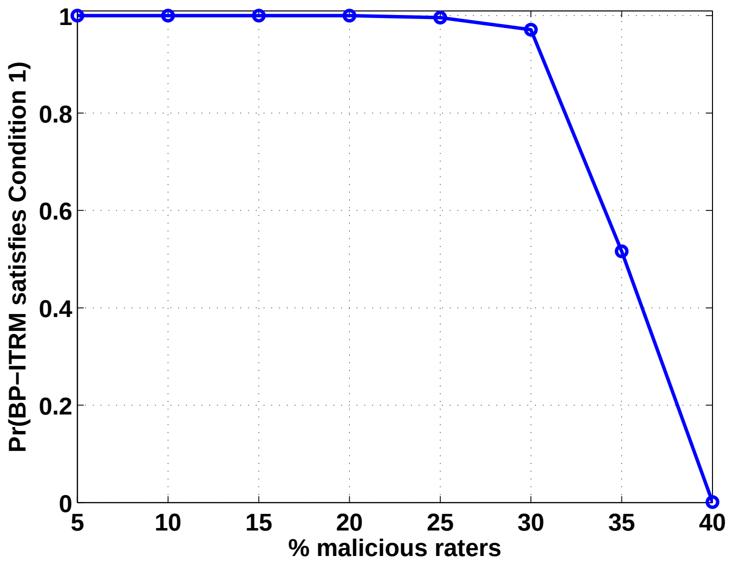


Figure 26: Probability of BP-ITRM to satisfy $Condition 1$ versus fraction of malicious raters.

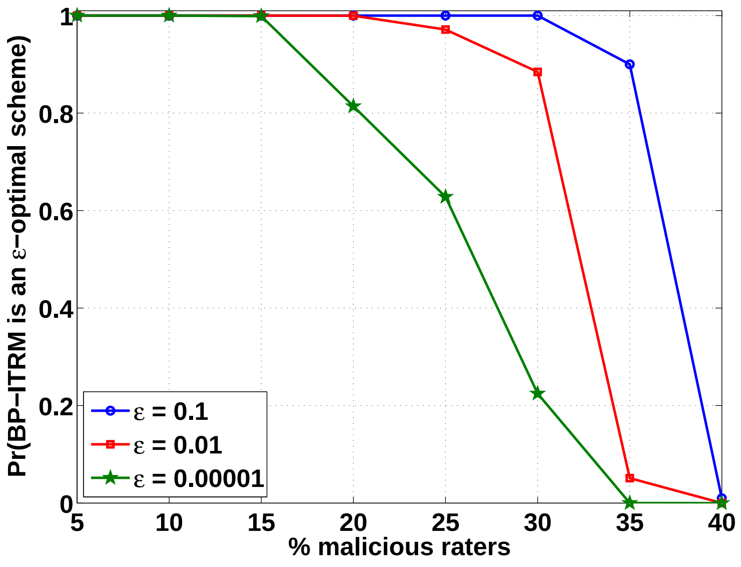


Figure 27: Probability that BP-ITRM is an ϵ-optimal scheme versus fraction of malicious raters for different ϵ values.

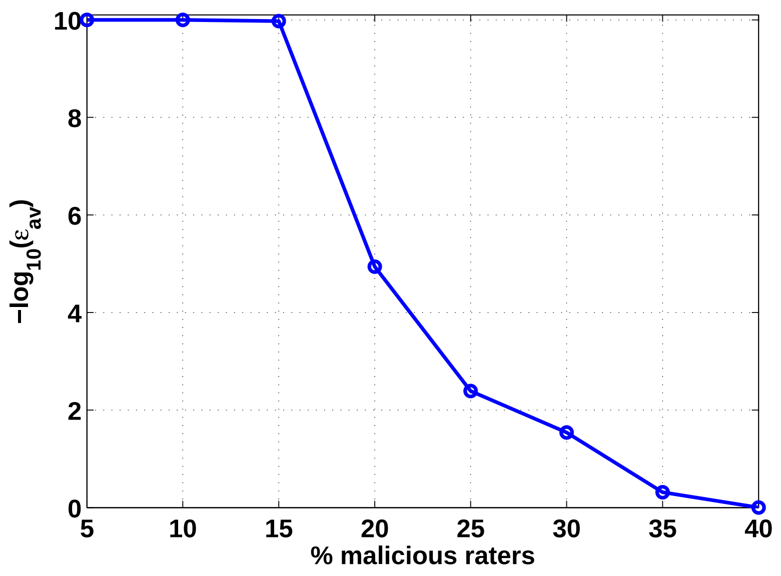


Figure 28: The average ϵ values for which BP-ITRM is an ϵ-optimal scheme with high probability versus fraction of malicious raters.

5.3.3 Simulations

We evaluated the performance of BP-ITRM in the presence of bad-mouthing, ballot-stuffing, and combinations of bad-mouthing and ballot-stuffing. Here, we provide an evaluation of the bad-mouthing attack only, as similar results hold for ballot-stuffing and combinations of bad-mouthing and ballot-stuffing. We compared the performance of BP-ITRM with three well-known and commonly used reputation management schemes: 1) *The Averaging Scheme*, 2) *Bayesian Approach*, and 3) *Cluster Filtering*. The Averaging Scheme is widely used as in eBay or Amazon (as discussed in Section 2.1.1). The Bayesian Approach [25, 103] updates G_j using a Beta distribution. We implemented the Buchegger's Bayesian approach in [25] (as discussed in Section 2.1.2) for the comparison with the deviation threshold $d = 0.5$ and trustworthiness threshold $t = 0.75$[4] (for details refer to [25]). As we discussed before, since we present and evaluate BP-ITRM in a centralized setting, Buchegger's work in [25]

[4]We note that these are the same parameters used in the original paper [25].

and Whitby's work in [103] can be considered as similar. In [25], if a rater's rating deviates beyond the deviation threshold d from the calculated reputation value, its trustworthiness value is modified accordingly. Further, if a rater's trustworthiness exceeds a definite threshold t, it is detected as malicious. Similarly, in [103], instead of using the deviation threshold, the authors check if the calculated reputation value for the SP falls between a definite interval for each rater's rating distribution. As we will discuss later, both [25] and [103] have the same problem against colluding malicious raters. Cluster Filtering [35, 64] (as discussed in Section 2.1.3), on the other hand, performs a dissimilarity test among the raters and then updates G_j using only the reliable raters. Finally, we compared BP-ITRM with our previous work on iterative trust and reputation management [11] (referred to as ITRM) to show the benefit of using BP.

We assumed that d is a random variable with Yule-Simon distribution (with $\rho = 1$ throughout the simulations) as discussed in Section 5.3.2. Further, the fading parameter is set as $\vartheta = 0.9$[5] and number of ratings, per time-slot, by a malicious rater as $b = 5$. Let $\hat{G}_j$ be the actual value of the global reputation of SP j. Then, we obtained the performance of BP-ITRM, for each time-slot, as the mean absolute error (MAE) $|G_j - \hat{G}_j|$, averaged over all the SPs that are under attack.

We assumed that the malicious raters collude and attack the SPs who have the highest reputation values (assuming that the attackers knows the reputation values) and received the lowest number of ratings from the reliable raters (assuming that the attackers have this information). We note that this assumption may not hold in practice since the actual values of the global reputations and number of ratings received by each SP may not be available to malicious raters. However, we assumed that this information is available to the malicious raters to consider the worst case

[5]We note that for the Averaging Scheme, Bayesian Approach, and Cluster Filtering we used the same fading mechanism as BP-ITRM (discussed in Section 5.2) and set the fading parameter as $\vartheta = 0.9$.

scenario. Further, the malicious raters collude and attack the same set Γ of SPs in each time-slot (which represents the strongest attack by the malicious raters). We further assumed that there are $|\mathbb{U}| = 100$ rater peers and $|\mathbb{S}| = 100$ SPs. Moreover, a total of $T = 50$ time-slots had passed since the lunch of the system, and reliable reports generated during those time-slots were distributed among the SPs uniformly. We note that we start our observations at time slot 1 after the initialization period.

Initially, we assumed that a fraction of the existing raters change behavior and become malicious after the start of the system (at time-slot one). Using all their edges, the malicious raters collude and attack the SPs who have the highest reputation values and received the lowest number of ratings from the reliable raters, by rating them as $r_m = 0$ (assuming the rating values are either 0 or 1). We note that this attack scenario also represents the RepTrap attack in [105] which is shown to be a strong attack. Further, we assumed that the rating r_h (provided by the non-malicious raters) is a random variable with Bernoulli distribution, where $Pr(r_h = \hat{G}_j) = 0.8$ and $Pr(r_h \neq \hat{G}_j) = 0.2$. First, we evaluated the MAE performance of BP-ITRM for different fractions of malicious raters ($W = \frac{|\mathbb{U}_M|}{|\mathbb{U}_M|+|\mathbb{U}_R|}$), at different time-slots (measured since the attack is applied) in Fig. 29[6]. We observed that the proposed BP-ITRM provides significantly low errors for up to $W = 30\%$ malicious raters. Moreover, MAE at the first time slot is consistent with our analytical evaluation which was illustrated in Fig. 28. Next, we observed the change in the average trustworthiness (R_i values) of malicious raters with time. Figure 30 illustrates the drop in the trustworthiness of the malicious raters with time. We conclude that the R_i values of the malicious raters decrease over time, and hence, the impact of their malicious ratings is totally neutralized over time. We further observed the average number of required iterations of BP-ITRM at each time-slot in Fig. 31. We conclude that the average number

[6]The plots in Figs. 29, 30, 31, 32, 33 and 34 are shown from the time-slot the adversary introduced its attack.

of iterations for BP-ITRM decreases with time and decreasing fraction of malicious raters. Finally, we compared the MAE performance of BP-ITRM with the other schemes. Figure 32 illustrates the comparison of BP-ITRM with the other schemes for bad-mouthing when the fraction of malicious raters (W) is 30%. It is clear that BP-ITRM outperforms all the other techniques significantly.

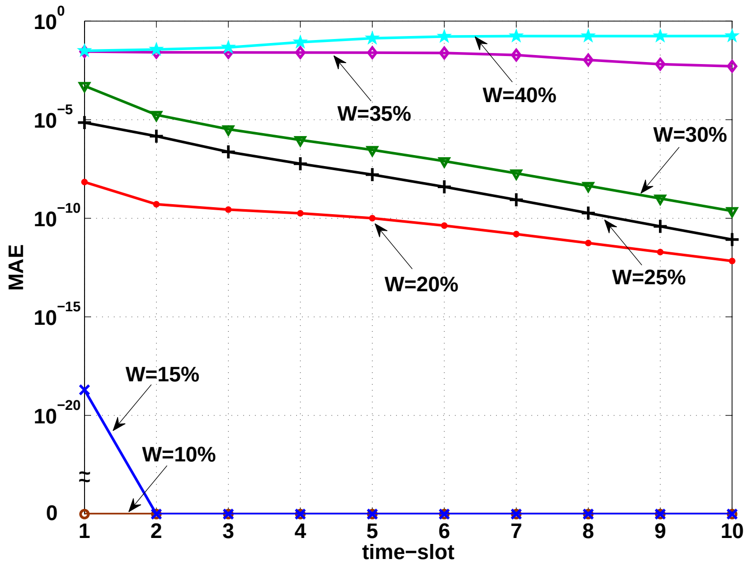


Figure 29: MAE performance of BP-ITRM versus time when W of the existing raters become malicious in RepTrap [105].

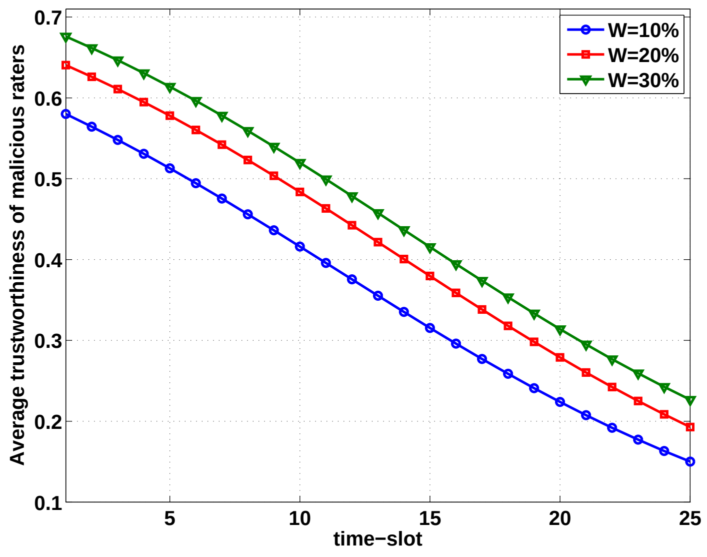


Figure 30: Change in average trustworthiness of malicious raters versus time for BP-ITRM when W of the existing raters become malicious in RepTrap [105].

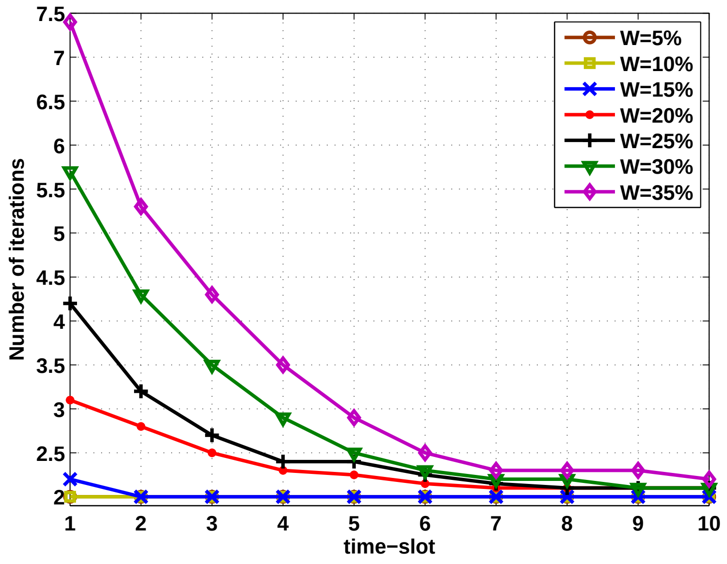


Figure 31: The average number of iterations versus time for BP-ITRM when W of the existing raters become malicious in RepTrap [105].

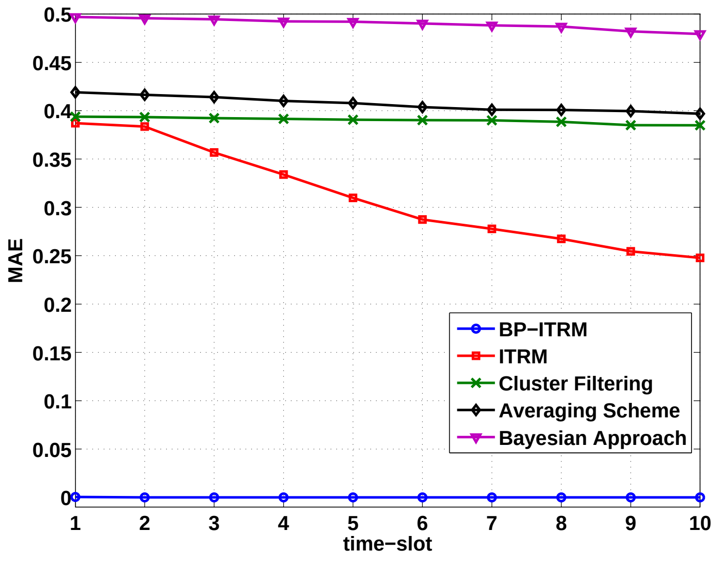


Figure 32: MAE performance of various schemes when 30% of the existing raters become malicious in RepTrap [105].

Next, we simulated the same attack scenario when ratings are integers from the set $\{1, \ldots, 5\}$ instead of binary values. We assumed that the rating r_h is a random variable with folded normal distribution (mean $\hat{G}_j$ and variance 0.5), however, it takes only discrete values from 1 to 5. Malicious raters choose SPs from Γ and rate them as $r_m = 4$. The malicious raters do not deviate very much from the actual $\hat{G}_j = 5$ values to remain undercover (while still attacking) as many time-slots as possible. We also tried higher deviations from the $\hat{G}_j$ value and observed that the malicious raters were easily detected by BP-ITRM. Figure 33 illustrates that BP-ITRM provides significantly low MAE for up to $W = 40\%$ malicious raters. We then compared the MAE performance of BP-ITRM with the other schemes in Fig. 34 and observed that BP-ITRM outperforms all the other techniques significantly.

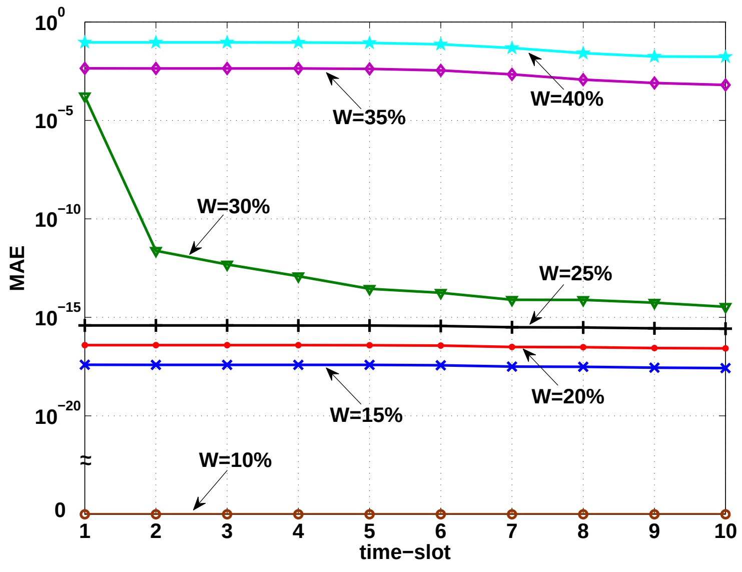


Figure 33: MAE performance of BP-ITRM versus time when W of the existing raters become malicious and rating values are integers from $\{1, \ldots, 5\}$ in RepTrap [105].

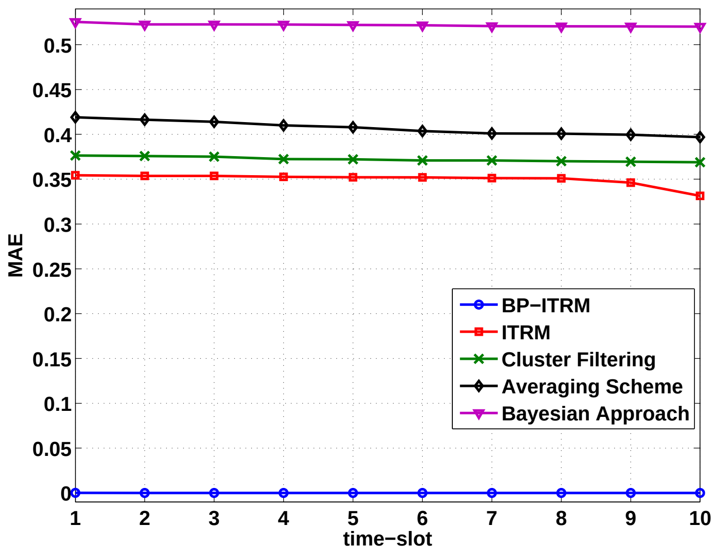


Figure 34: MAE performance of various schemes when 30% of the existing raters become malicious and rating values are from $\{1, \ldots, 5\}$ in RepTrap [105].

In most trust and reputation management systems, the adversary causes the most serious damage by introducing newcomer raters to the system. Since it is not possible for the system to know the trustworthiness of the newcomer raters, the adversary may introduce newcomer raters to the systems and attack the SPs using those raters. To study the effect of newcomer malicious raters to the reputation management scheme, we introduced 100 more raters as newcomers. Hence, we had $|\mathbb{U}_M| + |\mathbb{U}_R| = 200$ raters and $|\mathbb{S}| = 100$ SPs in total. We assumed that the rating values are either 0 or 1, r_h is a random variable with Bernoulli distribution as before, and malicious raters choose SPs from Γ and rate them as $r_m = 0$ (this particular attack scenario does not represent the RepTrap attack). We compared the MAE performance of BP-ITRM with the other schemes for this scenario in Fig. 35[7].

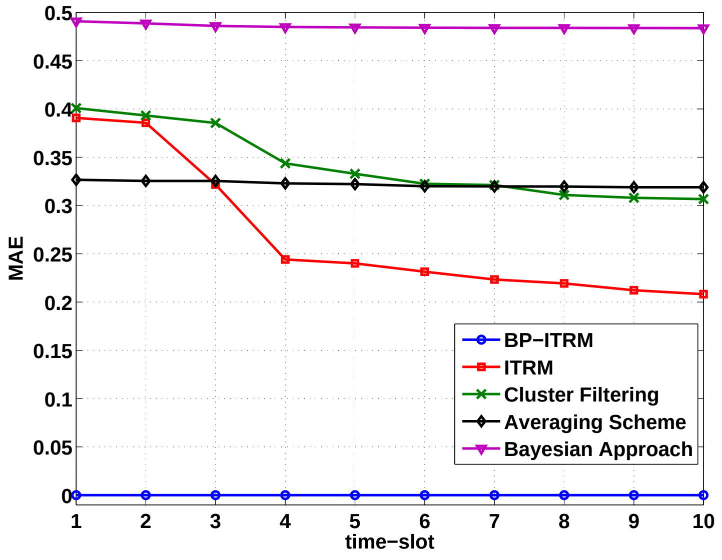


Figure 35: MAE performance of various schemes when 30% of the newcomer raters are malicious.

From these simulation results, we conclude that BP-ITRM significantly outperforms the Averaging Scheme, Bayesian Approach and Cluster Filtering in the presence of attackers. We identify that the Bayesian Approach performs the worst against the RepTrap attack and colluding attacks from malicious raters. Indeed, both [25] and [103] have the same shortcoming against colluding malicious raters. Both [25]

[7]The plot is shown from the time-slot the newcomers are introduced.

and [103] first calculate the reputation value of a particular SP, and then based on the calculated value, they adjust each rater's trustworthiness value. On the other hand, when the malicious raters collude (as in our attack scenario), it is likely that the majority of the ratings to the victim SPs will be from malicious raters. In this scenario, the Bayesian approach not only fails to filter the malicious ratings but it also punishes the reliable raters which rates the victim SPs. We also identify that ITRM (i.e., our algebraic iterative scheme described in Chapter 3) is the closest in accuracy to BP-ITRM. This emphasizes the robustness of using iterative message passing algorithms for reputation management.

Finally, assuming u raters and s SPs, we obtained the computational complexity of BP-ITRM as $\mathbf{max}(\mathrm{O}\big(cu\big), \mathrm{O}\big(cs\big))$ in the number of multiplications, where c is a small number representing the average number of rating edges per rater. In contrast, Cluster Filtering suffers quadratic complexity versus number of raters (or SPs).

5.4 Summary

In this chapter, we introduced the "Belief Propagation-Based Iterative Trust and Reputation Management Scheme" (BP-ITRM). Our work is an iterative probabilistic algorithm motivated by the prior success of message passing techniques and belief propagation algorithms on decoding of turbo codes and low-density parity-check codes. BP-ITRM relies on a graph-based representation of an appropriately chosen factor graph for reputation systems. In this representation, service providers and raters are arranged as two sets of variable and factor nodes that are connected via some edges. The reputation values of service providers are computed by probabilistic message passing between nodes in the graph until the convergence. The proposed BP-ITRM is a robust mechanism to evaluate the quality of the service of the service providers from the ratings received from the recipients of the service (raters). Moreover, it effectively evaluates the trustworthiness of the raters. We studied BP-ITRM by a detailed analysis and showed the robustness using computer simulations. We proved that BP-ITRM iteratively reduces the error in the reputation values of service providers due to the malicious raters with a high probability. Further, we observed that this probability demonstrates a threshold property. That is, exceeding a particular fraction of malicious raters reduces the probability sharply. We also compared BP-ITRM with some well-known reputation management schemes and showed the superiority of our scheme both in terms of robustness and efficiency.

CHAPTER VI

BELIEF PROPAGATION FOR TRUST AND REPUTATION MANAGEMENT IN DISTRIBUTED SYSTEMS

6.1 Introduction

Due to their size and the distributed architecture, Peer-to-peer (P2P) networks are highly vulnerable to attacks by the malicious peers (as we discussed in Chapter 1). The most common attack to P2P systems is in the form of injecting inauthentic files (or introducing viruses) to the network, which can be confronted by utilizing trust and reputation management systems. Thus, in this chapter, we introduce the first application of the Belief Propagation (BP) [59,70,110], an iterative probabilistic algorithm, in the design and evaluation of distributed trust and reputation management systems for P2P networks. Belief Propagation (BP) [59, 70, 110] 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 a BP-based reputation management algorithm for centralized environments [16] (as described in Chapter 5). However, in a distributed infrastructure, trust and reputation management is more complicated than in centralized solutions. Hence, different from our previous work in Chapter 5, in this chapter, we focus on P2P networks and explore trust and reputation management in a completely decentralized environment in the presence of malicious peers mounting attacks.

We introduce the "Belief Propagation-Based Trust and Reputation Management for P2P Networks" (BP-P2P). BP-P2P formulates the trust and reputation management problem as finding the marginal probability distributions of the reputation values. This problem, however, cannot be solved in a large-scale systems, 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 and lets the peers compute the reputation and trustworthiness values by distributed message passing between each other. We show that the proposed BP-based scheme is reliable (in filtering out malicious ratings and computing the reputation values) while being computationally efficient (i.e., linear in the number of peers). We further show that the communication overhead of BP-P2P is lower than the well-known EigenTrust algorithm which is particularly designed for P2P networks.

6.1.1 Contributions

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.

4. The proposed BP-P2P outperforms the existing and commonly used P2P reputation management techniques such as the EigenTrust algorithm [53] and the Bayesian Approach [24] (which is also proposed as the reputation management system of the well-known CONFIDANT protocol [23]) in the presence of attackers.

6.2 Belief Propagation-Based Trust and Reputation Management for P2P Networks (BP-P2P)

We assume two different sets in the system: i) the set of servers, $\mathbb{S}$ ($|\mathbb{S}| = s$) and ii) the set of clients, $\mathbb{U}$ ($|\mathbb{U}| = u$) . As opposed to our centralized approach (BP-ITRM), we assume that every peer in the network plays the role of both a client and a server (and hence, $u = s$ in a typical P2P network). 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 follows the protocol of message passing faithfully. In other words, 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 6.2.1, 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.

Similar to the discussion in Chapter 5, let G_j be the reputation value of server j ($j \in \mathbb{S}$), T_{ij} be the rating that client i ($i \in \mathbb{U}$) reports about server j ($j \in \mathbb{S}$), $\mathbb{T}$ be the $s \times u$ server-client matrix, $\mathbb{T}_i$ be the set of ratings provided by client i, and R_i be the trustworthiness of the peer i ($i \in \mathbb{U}$) as a client. Further, $\mathbb{G} = \{G_j : j \in \mathbb{S}\}$ and $\mathbb{R} = \{R_i : i \in \mathbb{U}\}$ are the collection of variables representing the reputations of the servers and the trustworthiness values of the clients, respectively. We consider slotted time throughout this discussion. At each time-slot, BP-P2P algorithm is executed using the input parameters $\mathbb{R}$ and $\mathbb{T}$ to obtain the reputation parameters at each peer. We note that different from the centralized algorithms, 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 [53]). 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.

Again, we approach the reputation management problem as finding s marginal probability functions, $p(G_j|\mathbb{T}, \mathbb{R})$, (each of which is associated with a variable G_j) given the observed data (i.e., evidence). As we discussed in Chapter 5, each marginal probability function $p(G_j|\mathbb{T}, \mathbb{R})$ may be obtained from the global function $p(\mathbb{G}|\mathbb{T}, \mathbb{R})$ by solving (18). However, the number of terms in (18) grows exponentially with the number of variables, making the computation infeasible. Further, (18) 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 (18) at each peer is not feasible in such distributed environments in which each peer has only a part of the evidence. Thus, we factorize (18) 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.

First, we arrange the collection of the clients and the servers together with the ratings as a factor graph, as in Fig. 23. 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 Fig. 23. Further, each rating is represented by an edge from the factor node to the variable node. We note that we use the same fading mechanism described in Section 5.2 (with the fading parameter ϑ) for the ratings to keep up with the temporal dynamics of the peers. Then, 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 in (19). Hence, in the graph representation of Fig. 23, 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. For the simplicity of the presentation we describe the message exchange between peer k (as a client) and peer a (as a server) in Fig. 23. 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). Factor and variable nodes in Fig. 23 iteratively exchange probabilistic messages following the BP algorithm, updating the degree of beliefs on the reputation values of the servers as well as the confidence of the clients on their ratings (i.e., trustworthiness values) 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. The message from the factor node (client) k to the variable node (server) a at the ν^{th} iteration is formed as

$$\lambda_{k\to a}^{(\nu)}(G_a^{(\nu)}) = \sum_{\mathcal{G}_k^{(\nu)}\setminus\{G_a^{(\nu)}\}} f_k(\mathcal{G}_k^{(\nu)}, \mathbb{T}_k, R_k^{(\nu-1)}) \prod_{x\in\Delta} \mu_{x\to k}^{(\nu-1)}(G_x^{(\nu)}), \tag{32}$$

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 and $\Delta = \mathbf{N_k^c}\setminus\{a\}$. This message transfer is illustrated in Fig. 36. Further, $R_k^{(\nu-1)}$ (the trustworthiness of client k calculated at the end of $(\nu-1)^{th}$ iteration) can be calculated as follows:

$$R_k^{(\nu-1)} = 1 - \frac{1}{|\mathbf{N_k^c}|} \sum_{i\in N_k^c} \sum_{x\in\{0,1\}} |T_{ki} - x| \mu_{i\to k}^{(\nu-1)}(x). \tag{33}$$

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 node. 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 $\mu_{a\to k}^{(\nu)}(G_a^{(\nu)})$ by multiplying all information it receives from its neighbors excluding the factor node k, as shown in Fig. 37. Hence, the message from variable node a to the factor node k at the ν^{th} iteration is given by

$$\mu_{a\to k}^{(\nu)}(G_a^{(\nu)}) = \frac{1}{\sum\limits_{h\in\{0,1\}} \prod\limits_{i\in\Xi} \lambda_{i\to a}^{(\nu)}(h)} \times \prod_{i\in\Xi} \lambda_{i\to a}^{(\nu)}(G_a^{(\nu)}), \tag{34}$$

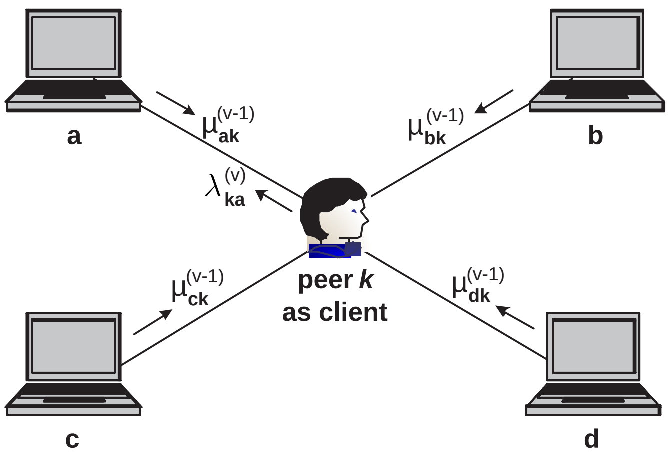


Figure 36: Message from the factor node (client) k to the variable node (server) a at the ν^{th} iteration.

where $\Xi = \mathbf{N_a^s} \backslash \{k\}$. 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. We note that the iterative algorithm starts its first iteration by computing $\lambda_{k\to a}^{(1)}(G_a^{(1)})$ in (32). However, instead of calculating in (33), the trustworthiness value R_k from the previous execution of BP-P2P is used as initial values in (32).

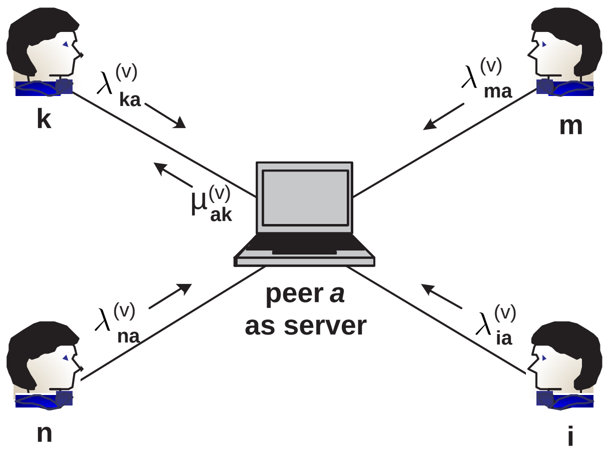


Figure 37: Message from the variable node (server) a to the factor node (client) k at the ν^{th} iteration.

BP-P2P stops after Ψ iterations (which is a pre-defined number and its selection will be discussed in Section 6.3.3). At the end of each iteration, the reputations are calculated at each server. To calculate the reputation value $G_a^{(\nu)}$, each server computes $\mu_a^{(\nu)}(G_a^{(\nu)})$ using (34) but replacing Ξ with $\mathbf{N_a^s}$, and then sets $G_a^{(\nu)} = \sum_{i=0}^{1} i\mu_a^{(\nu)}(i)$. 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.

6.2.1 Secure BP-P2P

As we discussed before, the fundamental scheme described in Section 6.2 is not completely secure since it assumes that every peer will honestly follow the protocol of message passing algorithm in BP. In other words, thus far we assumed that the messages associated with the reputation and trustworthiness values are computed faithfully according to the BP rules. We further assumed that each peer computes its own reputation and trustworthiness values and shares these values with the other peers upon an inquiry. However, it is clear that a malicious peer would not necessarily follow the BP rules and it would report its own reputation and trustworthiness values to the 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 [53]. Similar to [53], to assign score managers, we use a Distributed Hash Table (DHT) [95]. 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[1]. 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 [53]. Since it is not the main contribution of this work, 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 distributed 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 $\mathbb{H}_k = \{H_k^1, H_k^2, \dots, H_k^{\xi}\}$. We assume that each score manager of peer k knows: i) neighbors of peer k as a server (i.e., $\mathbf{N_k^s}$), and hence, the score managers of these neighbors, ii) neighbors of peer k as a client (i.e., $\mathbf{N_k^c}$), 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, each score manager generates messages as a server (variable node) on behalf of the peer they are responsible for. From now on, we refer a score manager as "client score manager" when it generates messages as a client, and

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

as "server score manager" when it generates messages as a server in order to avoid confusion. Thus, different from the fundamental scheme described in Section 6.2, BP messages are now between the client score managers and the server score managers. Due to this change, the factor graph in Fig. 23 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. 38 assuming $\xi = 3$. As illustrated in the figure, instead of connecting client k to the servers it rated (servers 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.

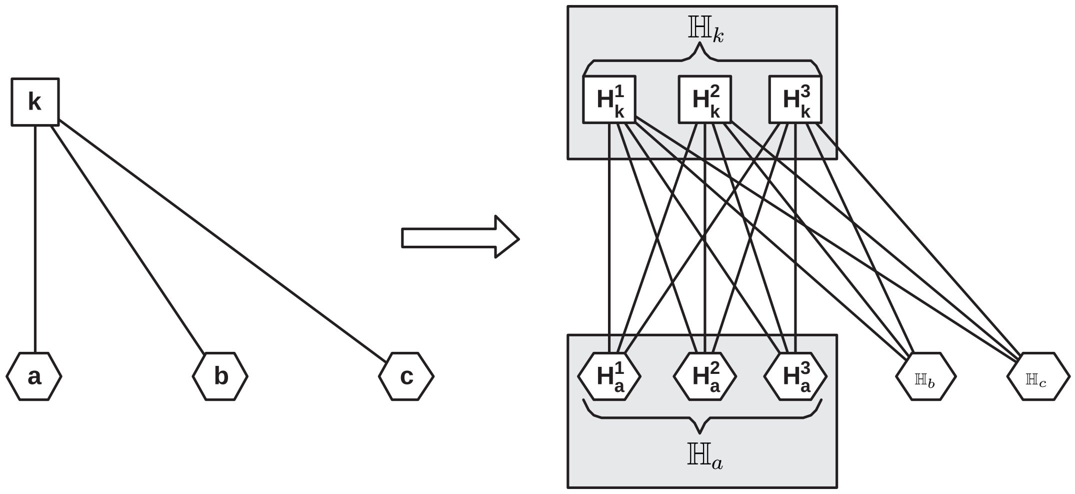


Figure 38: Utilizing score managers in BP-P2P.

Messages are exchanged between the score managers of the peers following the principles of the BP algorithm (as described in Section 6.2) and the algorithm stops after Ψ iterations (selection of Ψ will be discussed in Section 6.3.3). 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[2].

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 6.3.1. Further, we evaluate BP-P2P under this attack both analytically and via simulations in Sections 6.3.2 and 6.3.3, respectively.

6.2.2 Efficient BP-P2P

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

6.2.2.1 Computational Complexity

On 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.

6.2.2.2 Communication Overhead

In the fundamental scheme described in Section 6.2, 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

[2]The client i can also query the trustworthiness values of the score managers in $\mathbb{H}_j$ and compute the reputation value of server j using weighted average.

for each peer. Therefore, in this section, we modify the messages in (32) and (34) between the peers (between the score managers of the peers in the secure version described in Section 6.2.1) to reduce the communication overhead due to the 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 (32) 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, as we discussed in Section 5.2, it can be shown that $\lambda_{k\to a}^{(\nu)}(G_a^{(\nu)}) \propto p(G_a^{(\nu)}|T_{ka}, R_k^{(\nu-1)})$, where

$$\begin{aligned} &p(G_a^{(\nu)}|T_{ka}, R_k^{(\nu-1)}) = \\ &\Big[(R_k^{(\nu-1)} + \frac{1-R_k^{(\nu-1)}}{2})T_{ka} + \frac{1-R_k^{(\nu-1)}}{2}(1-T_{ka})\Big]G_a^{(\nu)} + \\ &\Big[\frac{1-R_k^{(\nu-1)}}{2}T_{ka} + (R_k^{(\nu-1)} + \frac{1-R_k^{(\nu-1)}}{2})(1-T_{ka})\Big](1-G_a^{(\nu)}). \end{aligned} \tag{35}$$

This resembles the belief/pleusability concept of the Dempster-Shafer Theory [89,90] (as we also discussed in Section 5.2). Given $T_{ka} = 1$, $R_k^{(\nu-1)}$ can be viewed as the belief of the client k that $G_a^{(\nu)}$ is one (at the ν^{th} iteration). In other words, in the eyes of client k, $G_a^{(\nu)}$ is equal to one with probability $R_k^{(\nu-1)}$. Thus, $(1 - R_k^{(\nu-1)})$ corresponds to the uncertainty in the belief of client k. In order to remove this uncertainty and express $p(G_a^{(\nu)}|T_{ka}, R_k^{(\nu-1)})$ as the probabilities that $G_a^{(\nu)}$ is zero and one, we distribute the uncertainty uniformly between two outcomes (one and zero). Hence, in the eyes of the client k, $G_a^{(\nu)}$ is equal to one with probability $(R_k^{(\nu-1)} + (1 - R_k^{(\nu-1)})/2)$, and zero with probability $((1 - R_k^{(\nu-1)})/2)$. We note that a similar statement holds for the case when $T_{ka} = 0$.

We now describe how we modify the BP messages in (35) and (34) 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 6.2.1). In principal, at each iteration, we let each score manager H_k^i 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_k^i computes the actual BP message in (35) or (34) using the broadcasted message (from H_k^i) and the information it already possesses. In the following, we discuss the details of this process.

Let $\mathbb{H}_{\mathbf{N_k^c}}$ denote the set of score managers of neighbors of client k. Instead of computing (35) for all its neighbors separately, each client score manager in $\mathbb{H}_k$ (in the secure version described in Section 6.2.1) only computes and broadcasts the updated trustworthiness value of the client k to its neighbors in $\mathbb{H}_{\mathbf{N_k^c}}$ instead of sending different messages to each of its neighbors. Since the score managers in $\mathbb{H}_{\mathbf{N_k^c}}$ know the rating value given by client k to the servers they are responsible for, each score manager in $\mathbb{H}_{\mathbf{N_k^c}}$ computes the actual message itself. For example, since the score managers of server a know T_{ka}, they compute the message in (35) by only using the broadcasted $R_k^{(\nu-1)}$ value.

Similarly, all messages from a server score manager (H_a^j) to its neighbors (in $\mathbb{H}_{\mathbf{N_a^s}}$) may be communicated simultaneously via a single broadcast step to decrease the communication overhead. The message from H_a^j to one of its neighbors i in $\mathbb{H}_{\mathbf{N_a^s}}$ is formed by multiplying all the messages received at H_a^j excluding the one received from the score manager i (similar to (34)). Thus, H_a^j 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 6.3). 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 6.3.3). On the other hand, in

EigenTrust [53] each score manager sends (on the average) $\mathbf{max}(\mathrm{O}\big(2\xi\Psi c\big), \mathrm{O}\big(2\xi\Psi v\big))$ 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 [53]. Therefore, we conclude that the proposed BP-based algorithm does not introduce a significant communication overhead to the network.

6.3 Security Evaluation

In this section, we mathematically model and analyze BP-P2P. Moreover, we support the analysis via computer simulations and compare BP-P2P with the existing and commonly used P2P reputation management mechanisms. In order to facilitate future references, frequently used notations are listed in Table 5.

Table 5: Notations and definitions.

$\mathbb{S}$	The set of 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

6.3.1 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.

6.3.1.1 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[3]. 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. 23). 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[4]). We note that this attack scenario also represents the RepTrap attack in [105] 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 $(T+1)$.

6.3.1.2 Malicious Score Managers

As we discussed in Section 6.2.1 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

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

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

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.

6.3.2 Analytical Evaluation

We adopted the following models for various peers involved in the P2P trust and reputation management system. 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. 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 6.3.1.

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. Similar to our centralized scheme in Chapter 5, 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 6.3.1), the system designed using these criteria will be robust against ballot-stuffing and combinations of bad-mouthing and ballot-stuffing as well.

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 (since the bad-mouthing attack is aimed to reduce the reputation values of the victim servers). This leads to the below lemma.

Lemma 6.3.1. *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 $\hat{G}_a = 1$*[5].

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.

$$\prod_{j \in \mathbb{H}^R_{\mathbf{N_a^s}} \cap \mathbb{U}_R} \frac{R_j^{(w+1)} + 1}{1 - R_j^{(w+1)}} \prod_{j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_R} \frac{1 - \hat{R}_j^{(w+1)}}{1 + \hat{R}_j^{(w+1)}} >$$
$$\prod_{j \in \mathbb{H}^R_{\mathbf{N_a^s}} \cap \mathbb{U}_R} \frac{R_j^{(w)} + 1}{1 - R_j^{(w)}} \prod_{j \in \mathbb{H}^M_{\mathbf{N_a^s}} \cap \mathbb{U}_R} \frac{1 - \hat{R}_j^{(w)}}{1 + \hat{R}_j^{(w)}}, \tag{36}$$

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 (33)) 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, (36) 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, (36) 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

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

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 it stops. $\square$

Although because of the Lemma 6.3.1, 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 6.3.1 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 below:

$$P = \prod_{a \in \mathbb{S}} Pr\Big\{\epsilon \geq 1 - \frac{\varpi}{\varpi + \zeta}\Big\} \tag{37}$$

where,

$$\varpi = \prod_{j \in \mathbb{H}^R_{\mathbf{N^s_a}} \cap \mathbb{U}_R} (R_j^{(\Psi+1)} + 1) \prod_{j \in \mathbb{H}^M_{\mathbf{N^s_a}} \cap \mathbb{U}_R} (1 - \hat{R}_j^{(\Psi+1)}) \prod_{j \in \mathbb{H}^M_{\mathbf{N^s_a}} \cap \mathbb{U}_M} (1 - \tilde{R}_j^{(\Psi+1)}) \tag{38a}$$

$$\zeta = \prod_{j \in \mathbb{H}^R_{\mathbf{N^s_a}} \cap \mathbb{U}_R} (1 - R_j^{(\Psi+1)}) \prod_{j \in \mathbb{H}^M_{\mathbf{N^s_a}} \cap \mathbb{U}_R} (1 + \hat{R}_j^{(\Psi+1)}) \prod_{j \in \mathbb{H}^M_{\mathbf{N^s_a}} \cap \mathbb{U}_M} (1 + \tilde{R}_j^{(\Psi+1)}). \tag{38b}$$

In (38), $R_j^{(\Psi+1)}$ 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 6.3.3). 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 [92], 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 6.3.1. We note that we also evaluated BP-P2P with different parameters and obtained similar results. In Fig. 39, we illustrated the probability of BP-P2P providing an 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. 40 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.

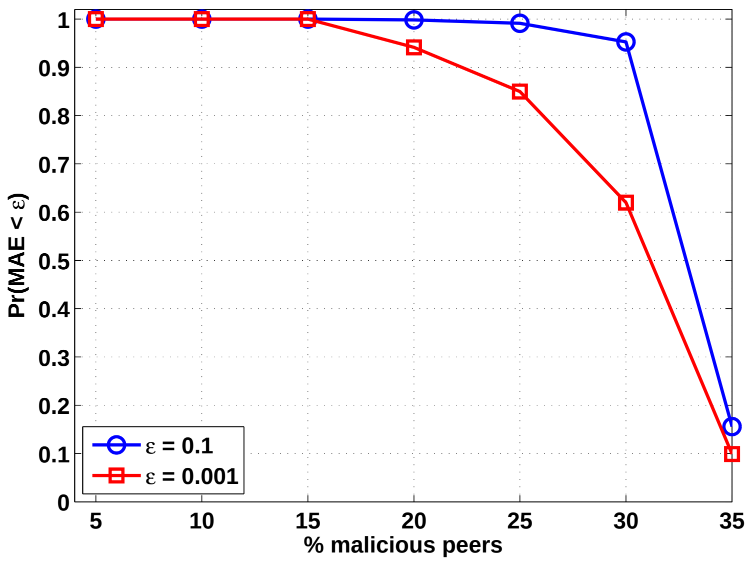


Figure 39: Probability of BP-P2P to provide an MAE that is less than ϵ versus fraction of malicious peers.

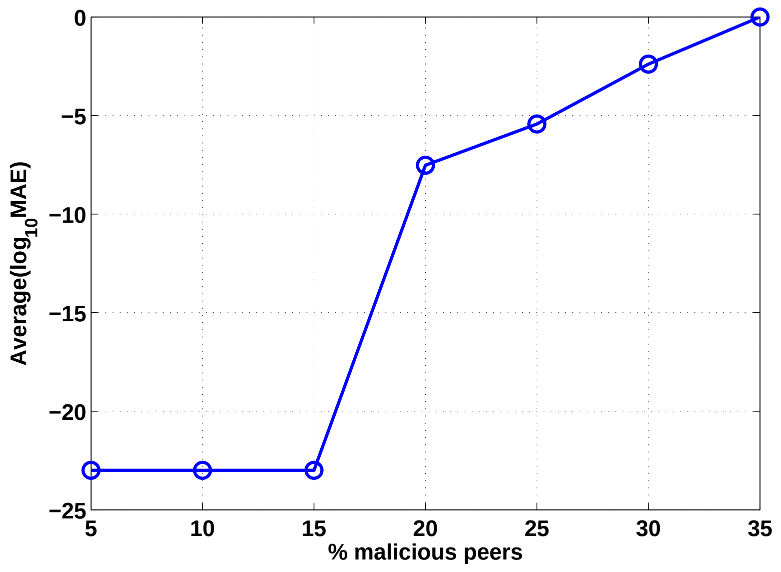


Figure 40: The average MAE values provided by BP-P2P with high probability versus fraction of malicious peers.

6.3.3 Simulations

We evaluated the performance of BP-P2P via computer simulations (via MATLAB) and compared BP-P2P with the Bayesian reputation management framework in [24] (which is also proposed as the reputation management system of the well-known CONFIDANT protocol [23]) and the EigenTrust algorithm [53] 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 6.3.2. 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 $\vartheta = 0.9$[6]. Further, we assumed that rating values are from the set $\Upsilon = \{0, 1\}$. Finally, we assumed the threat model described in Section 6.3.1 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

[6]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 $\vartheta = 0.9$.

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. For the Bayesian framework [24], we used the parameters from the original work [24] (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 [53] (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.

It is worth noting that in principle, BP-P2P performs better than the Bayesian framework since Bayesian approaches of [24] and [103] assume that the reputation values of the peers are independent. Hence, in these schemes, each reputation value is computed independent of the other peers' reputation values using the ratings given to each peer. However, this assumption is not valid because the ratings provided by the client peers induce a probability distribution on the reputation values of the server peers. These distributions are correlated because they are induced by the overlapping set of client peers. The strength of BP-P2P stems from the fact that captures this correlation in analyzing the ratings and computing the reputations. On the other hand, as we discussed in Section 2.3, 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. This is one of the reasons why we expect that the BP-P2P would perform better than the EigenTrust algorithm. Further, the EigenTrust algorithm computes the reputation values by a simple iterative weighted averaging mechanism which is vulnerable to collaborative attacks from the malicious peers. Therefore, the proposed BP-P2P algorithm is more powerful than the EigenTrust algorithm since it views the reputation management problem as an inference problem and computes the reputation values via probabilistic message passing between the peers, which is shown to be robust against colluding attackers in Chapter 5. Indeed, our simulation results (presented next) also support these arguments.

First, we determined the total number of iterations (Ψ) required for the BP-P2P algorithm. Thus, in Fig. 41[7], 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. 42. 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 43 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. We also compared the MAE performance of BP-P2P with the Bayesian Framework and the EigenTrust algorithm. Figure 44 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

[7]The plots in Figs. 41, 42 and 43 are shown from the time-slot the adversary introduced its attack.

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 6.3.1). In Fig. 45, 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. Next in Fig. 46, we evaluated the MAE performance of BP-P2P for different ξ values and for $W = 25\%$ malicious peers, at different time-slots. 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.

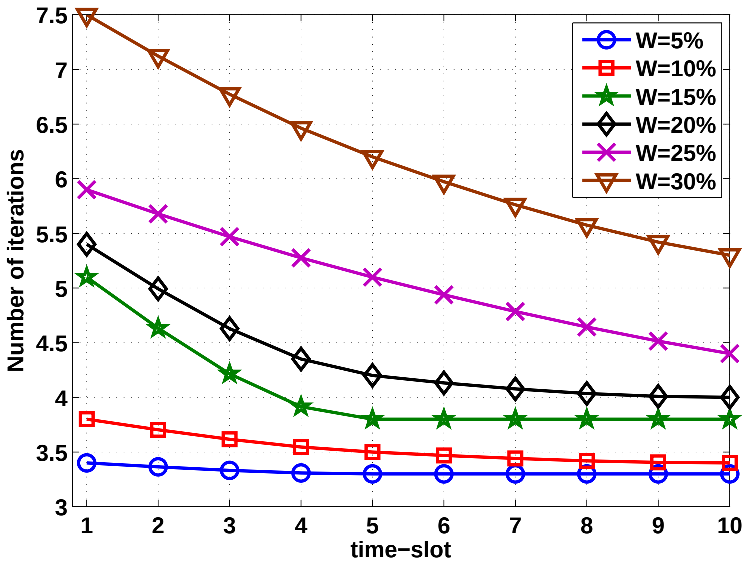


Figure 41: The average number of iterations versus time for BP-P2P to converge when W of the existing peers become malicious.

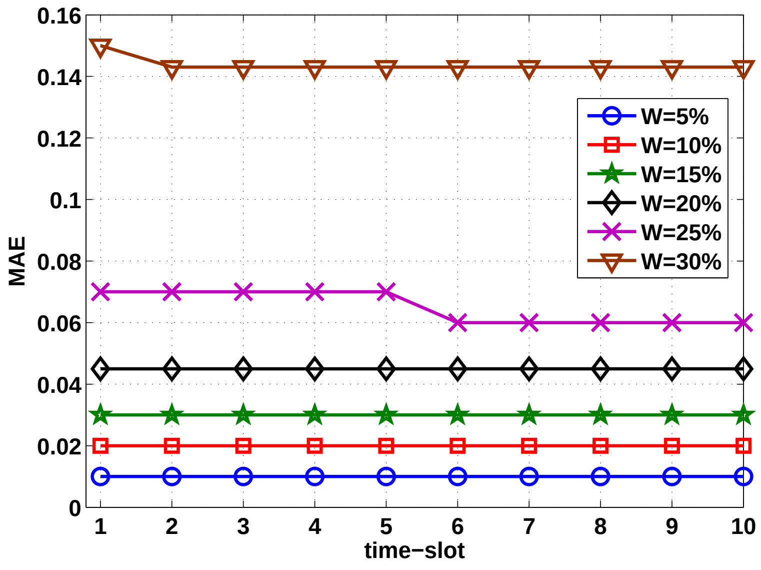


Figure 42: MAE performance of BP-P2P versus time when W of the existing peers become malicious.

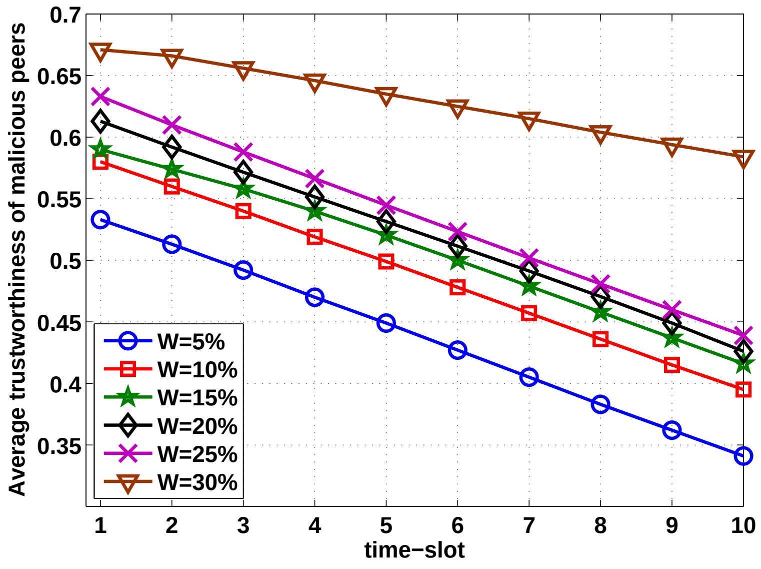


Figure 43: Change in average trustworthiness of malicious clients versus time for BP-P2P when W of the existing peers become malicious.

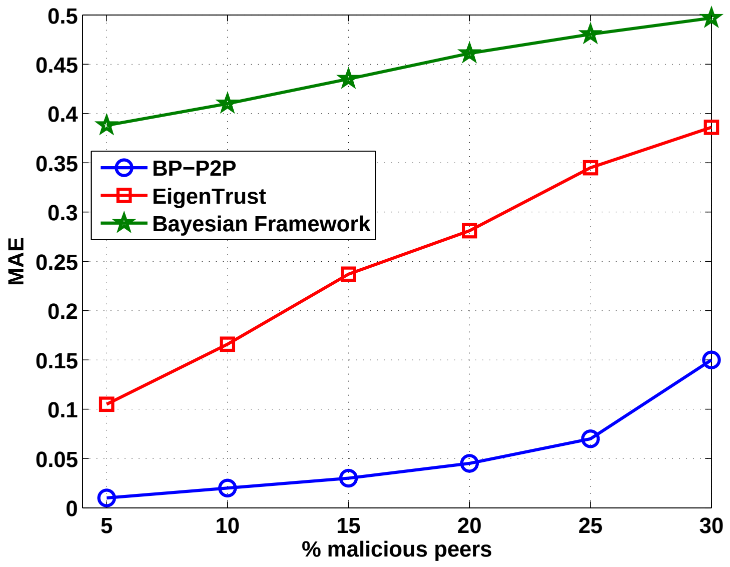


Figure 44: MAE performance of various schemes when different fractions of the existing peers become malicious at the first time-slot the attack is applied.

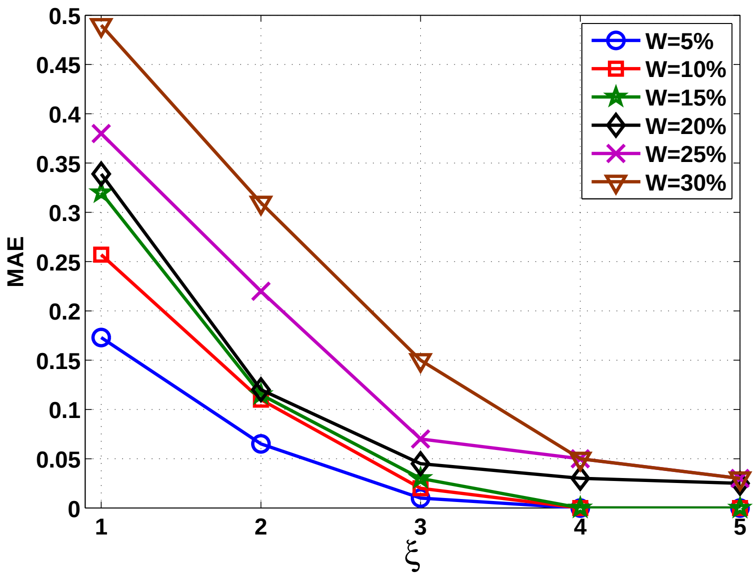


Figure 45: MAE performance of BP-P2P for different values of ξ and for different fractions of malicious peers at the first time-slot the attack is applied.

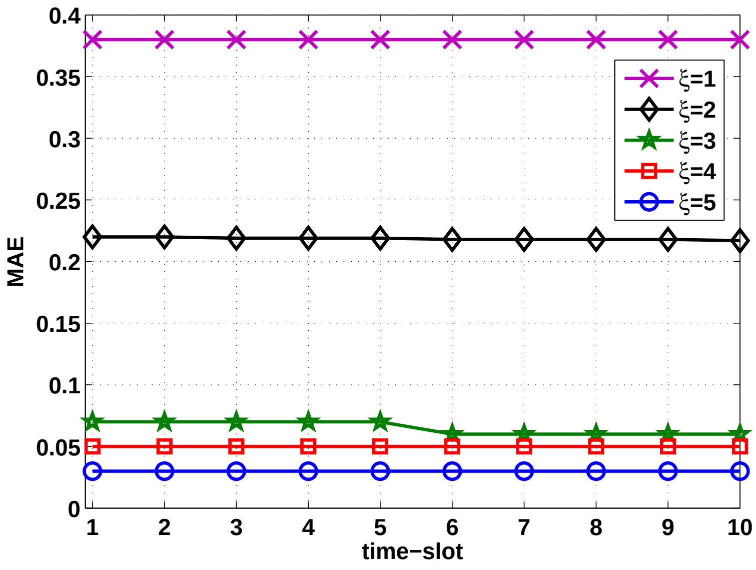


Figure 46: MAE performance of BP-P2P for different ξ values and for $W = 25\%$ malicious peers, at different time-slots.

Next, we simulated the same attack scenario (in which there are both malicious clients and malicious score managers) when ratings are integers from the set $\Upsilon = \{1, \ldots, 5\}$ instead of binary values[8] when $\xi = 3$. 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. 47 and observed that BP-P2P outperforms all the other techniques. We observed that BP-P2P provides significantly low MAE for up to $W = 30\%$ malicious clients. We further observed that the Bayesian Framework performs better than EigenTrust for this scenario.

[8]For the attack from the malicious score managers, we assumed a similar scenario to the binary case as discussed in Section 6.3.1.

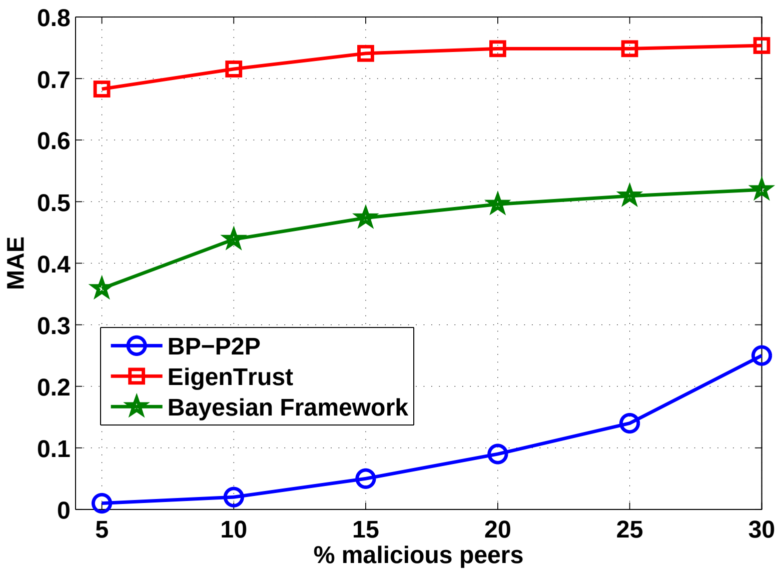


Figure 47: MAE performance of various schemes when the rating values are from $\Upsilon = \{1, \ldots, 5\}$.

Finally, we observed the impact of "malicious clients" and "malicious score managers" (described in Section 6.3.1) separately when $\xi = 3$ and ratings are binary values from the set $\Upsilon = \{0, 1\}$ (we kept all the other parameters the same as described before). In Fig. 48, we illustrated the MAE performance of BP-P2P when the attackers only attack as malicious clients (by collaboratively bad-mouthing the victim servers) and they behave reliably as score managers (by providing correct messages in the BP algorithm). Similarly, we studied the MAE performance when the attackers only attack as malicious score managers (by providing malicious messages in the BP algorithm) and behave reliably as clients (by providing reliable ratings). We observed that such an attack causes a significantly low MAE for up to $W = 30\%$ of attackers, and hence, we do not plot the results of this attack separately. We finally compared the impacts of these attacks when $W = 25\%$ of clients are malicious in Fig. 49. In Fig. 49, "hybrid attack" implies the attack in which the attackers attack both as malicious clients and malicious score managers. Further, "malicious clients" and "malicious score managers" imply the attacks in which the attackers only attack as malicious clients and the attackers only attack as malicious score managers, respectively. Based on these results, we conclude that the attacker has the most impact (in terms of MAE) when the malicious peers execute the "hybrid attack".

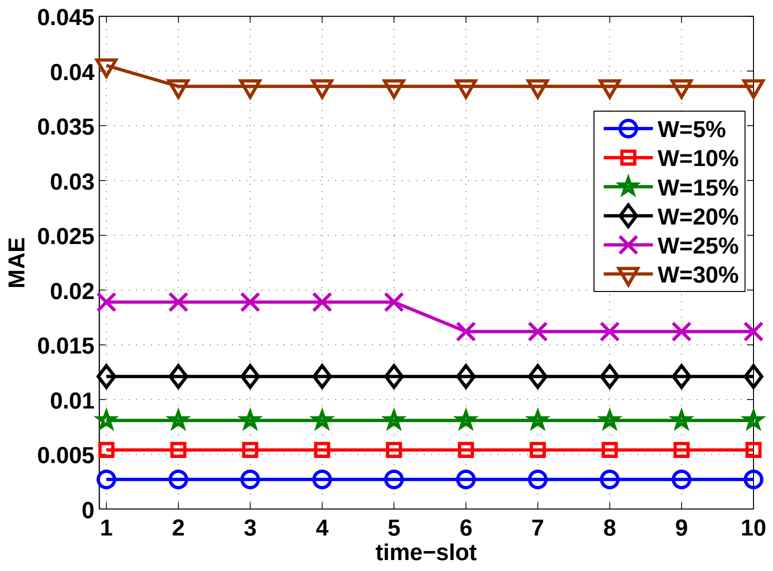


Figure 48: MAE performance of BP-P2P when the attackers only attack as malicious clients, at different time-slots.

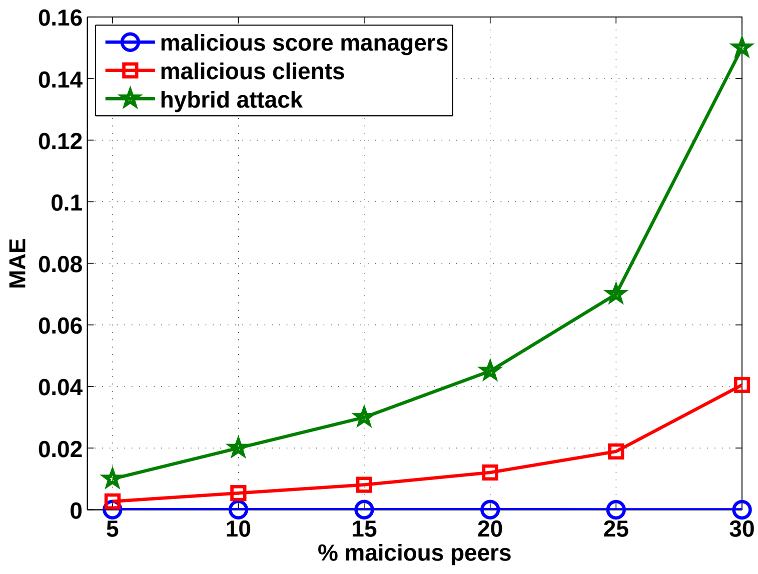


Figure 49: MAE comparison of different attack scenarios for different fractions of malicious peers at the first time-slot the attack is applied.

6.4 Summary

We introduced the first application of the belief propagation algorithm in the design and evaluation of distributed trust and reputation management systems for 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.

CHAPTER VII

BELIEF PROPAGATION-BASED ITERATIVE RECOMMENDER SYSTEM

7.1 Introduction

In the previous chapters, we viewed the reputation management problem as an inference problem whose solution involves computing marginal distributions. Then, we solved the problem efficiently (in linear complexity) by applying the Belief Propagation (BP) algorithm. We provided strong evidence that the proposed BP-ITRM (in Chapter 5) and BP-P2P (in Chapter 6) are superior to their counterparts. Relying on our success in the reputation management problem, we now extend the BP-based technique to arrive at scalable, accurate and robust recommender systems. As we discussed in Chapter 1, recommender systems are aimed at addressing the information overload problem, suggesting to the users those items that meet their interests and preferences the best in a particular situation and context. These systems are used to direct users towards items they will like while interacting with large information spaces. However, there are certain challenges to design accurate and scalable recommender systems. Hence, new research needed to focus on algorithms which meet these challenges. Thus, in this chapter we introduce the first application of Belief Propagation (BP), an iterative probabilistic algorithm, to solve the recommendation problem.

The key observation we make is that recommender systems deal with complicated global functions of many variables (e.g., users and items). By using a factor graph, we can obtain a qualitative representation of how the users and items are related on a graphical structure. Further, by using such a representation and the relations between the users and the items, global functions factor into a product of simpler local functions, each of which depends on a subset of the variables. Therefore, we propose to model the recommender system on a factor graph using which our goal is to compute the marginal probability distribution functions of the variables representing the ratings to be predicted for the users. However, we observe that computing the marginal probability functions is computationally prohibitive (i.e., exponential in the number of users and items) for large-scale recommender systems. Therefore, we propose to utilize the BP algorithm (discussed in Section 5.1.2) to efficiently compute these marginal probability distributions. BP algorithms are very powerful to solve inference problems, at least approximately. In fact, many algorithms (such as the forward-backward algorithm, Pearl's belief propagation for Bayesian networks, and iterative decoding of Gallager code) discovered in various scientific fields, although apparently different, are all special cases of the BP algorithm. The key role of the BP algorithm is that we can use it to compute the marginal distributions (of the variables representing the ratings to be predicted) in a complexity that grows only linearly with the number of nodes (i.e, users/items). For that reason, BP acts as a powerful tool to operate on statistical data encoded on some forms of large graphical models. We believe that the significant benefits offered by BP can be tapped in to benefit the field of recommender systems.

Hereafter, we refer to our scheme as the "Belief Propagation-Based Iterative Recommender System" (BPRS). In BPRS, the items and users are represented via a factor graph on which they are arranged as two sets of variable and factor nodes connected via some edges. The predictions of a user's ratings are computed by message passing between nodes in the graph. In each iteration of the algorithm, all the variable nodes (items), and subsequently all the factor nodes (users), pass new messages to their neighbors along with their edges on the graph until the recommendations converge. BPRS has several prominent features. First, it does not require to solve the problem for all users if it wishes to update the predictions for only a single active user and it does not require a training period to utilize the most recent data (ratings). Second, its complexity remains linear per single user, making it very attractive for large-scale systems. Therefore, it can update the recommendations for each active (online) user instantaneously using the most recent data (ratings). Further, we showed that BPRS provides comparable usage prediction and rating prediction accuracy to other popular methods such as the Correlation-based neighborhood model (CorNgbr) [18] and Singular Value Decomposition (SVD) [102], while it provides significant scalability advantages as discussed above. Furthermore, we are confident that this BP-based approach will enable us integrate data from different content domains to answer how the behaviors in one content domain represent the user preferences in another domain and provide resiliency against attacks. Therefore, we are very optimistic that this work promises a new direction for the recommender systems which will be scalable, accurate, and resilient to attacks.

7.1.1 Contributions

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 of recommender systems.

2. BPRS does not require solving the recommendation problem for all users if it wishes to update the recommendations for only a single active user using the most recent data (ratings).

3. BPRS computes the recommendations for each user with linear complexity and without requiring a training period.

4. BPRS iteratively reduces the error in the predicted ratings of the users until it

converges and it is comparable to the state of art methods such as Correlation-based neighborhood model (CorNgbr) and Singular Value Decomposition (SVD) in terms of rating and precision accuracy.

7.2 Belief Propagation for Recommender Systems (BPRS)

In the following, we discuss as to how we will give a new formulation of the recommender system problem as finding the marginal probability distributions of the unknown variables on a factor graph, using which we will arrive at a scalable and accurate recommender system (hereafter called BPRS).

We assume two different sets in the system: i) the set of users $\mathbb{U}$ and ii) the set of items (products) $\mathbb{I}$. Users provide feedbacks, in the form of ratings, about the items for which they have an opinion. The main goal is to provide accurate recommendations for every user by predicting the ratings of the user for the items that he/she did not rate before (unseen item). Here, we consider an arbitrary user z (referred as the active user) and compute the prediction of ratings for user z for unseen items to describe the algorithm. We assume u users and s items in the system (i.e., $|\mathbb{U}| = u$ and $|\mathbb{I}| = s$). Let $\mathbb{G}_z = \{G_{zj} : j \in \mathbb{I}\}$ be the collection of variables representing the ratings of the items to be predicted[1] for the active user z. Let also $\mathbb{R}_z = \{R_{zi} : i \in \mathbb{U}\}$ be the confidence of the system on the users for their ratings' reliability, given the active user is z. Further, we denote T_{ij} to represent the rating provided previously by user i about the item j. We denote $\mathbb{T}$ as the $s \times u$ item-user matrix that stores these ratings, and $\mathbb{T}_i$ be the set of ratings provided by the user i. We note that some rating entries could be missing (attributed to unseen items). To be consistent with the most of existing recommender systems, we assume that the rating values are integers from the set $\Upsilon = \{1, 2, 3, 4, 5\}$.

Our objective is to formulate the recommendation problem as making statistical

[1]A subset of these variables are already known as the corresponding items were rated by user z. Hence, they do not require any prediction.

inference about the ratings of users for unseen items based on observations. That is, given the past data evidence, what would be the likelihood (probability) that the rating takes a particular value? Here, the probability is the degree of belief to which the prediction of the rating is supported by the available evidence. This requires finding the marginal probability distributions of the variables in $\mathbb{G}_z$ conditioned on some observed preferences. However, computing these probability distributions is computationally prohibitive for large-scale recommender systems. Using the same principles discussed in Section 5.2, we represent the recommender system problem on a factor graph and utilize the BP algorithm to estimate these marginal probability distributions via message passing between the items and the users with a complexity that grows linearly with the number of variables in the system. As a result, we arrange the collection of the users and items together with the ratings provided by the users as a factor graph $g(\mathbb{U}, \mathbb{I})$. In this representation, each user corresponds to a factor node in the graph, shown as a square and each item is represented by a variable node shown as a hexagon. Further, each rating is represented by an edge from the factor node to the variable node. Then, since we consider the particular active user z, the factor graph is reduced to $g(\hat{\mathbb{U}}, \mathbb{I})$ (as in Fig. 50) by only keeping the users that are connected to z via a path in $g(\mathbb{U}, \mathbb{I})$ and removing all the other user nodes from the graph together with their edges. Eventually, the $g(\hat{\mathbb{U}}, \mathbb{I})$ graph has $|\hat{\mathbb{U}}| = \hat{u}$ users and $|\mathbb{I}| = s$ items.

As in Section 5.2, the joint probability function $p(\mathbb{G}_z|\mathbb{T}, \mathbb{R}_z)$ factors into a product of $\hat{u}$ local functions f_i, $(i \in \hat{\mathbb{U}})$, each having a subset of $\mathbb{G}_z$ as arguments similar to (19). Each local function is associated with a different factor node, and the function $f_i(\mathcal{G}_{zi}, \mathbb{T}_i, R_{zi})$ has the argument $\mathcal{G}_{zi}$ that is a subset of $\mathbb{G}_z$. Further, each local function f_i represents the probability distributions of its arguments given the amount of confidence of the system to the associated factor node (as a user) and the existing ratings of the associated user.

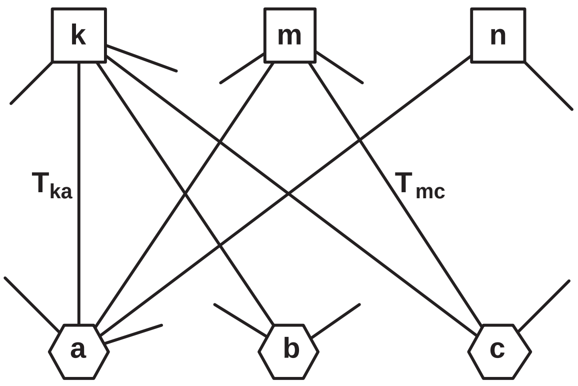


Figure 50: Graphical representation of the scheme from user z's point of view.

We now describe message exchange between user k and item a (in Fig. 50) provided that the active user is z. We represent the set of neighbors of the variable node a and the factor nodes k and z (in $g(\hat{\mathbb{U}}, \mathbb{I})$) as $\mathbf{N_a}$, $\mathbf{N_k}$, and $\mathbf{N_z}$, respectively. Further, let $\Xi = \mathbf{N_a} \backslash \{k\}$ and $\Delta = \mathbf{N_k} \backslash \{a\}$. Let $\mathbb{G}_{zj}^{(\nu)}$ be the value of variable $\mathbb{G}_{zj}$ at the iteration ν of the algorithm. The message $\mu_{a \to k}^{(\nu)}(G_{za}^{(\nu)})$ (from variable node a to the factor node k) denotes the probability that $G_{za}^{(\nu)} = \ell$ ($\ell \in \Upsilon$) at the ν^{th} iteration. On the other hand, $\lambda_{k \to a}^{(\nu)}(G_{za}^{(\nu)})$ (from factor node k to the variable node a) denotes the relative probabilities that $G_{za}^{(\nu)} = \ell$ ($\ell \in \Upsilon$) at the ν^{th} iteration, given T_{ka} and $R_{zk}^{(\nu-1)}$.

The message from the factor node k to the variable node a at the ν^{th} iteration is formed similar to the discussion on Section 5.2 as follows:

$$\lambda_{k \to a}^{(\nu)}(G_{za}^{(\nu)}) = \sum_{\mathcal{G}_{zk}^{(\nu)} \backslash \{G_{za}^{(\nu)}\}} f_k(\mathcal{G}_{zk}^{(\nu)}, \mathbb{T}_k, R_{zk}^{(\nu-1)}) \prod_{x \in \Delta} \mu_{x \to k}^{(\nu-1)}(G_{zx}^{(\nu)}), \tag{39}$$

where $\mathcal{G}_{zk}$ 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 the right half of Fig. 51. Further, $R_{zk}^{(\nu-1)}$ can be calculated as follows:

$$R_{zk}^{(\nu-1)} = 1 - \frac{1}{\rho |\mathbf{N_k}|} \sum_{i \in \mathbf{N_k}} \sum_{x \in \Upsilon} |T_{ki} - x| \mu_{i \to k}^{(\nu-1)}(x). \tag{40}$$

In the above equation, ρ, which is the highest possible deviation of a user, is set to 4 in this particular rating system, where the rating values are integers from the set

Υ. Thus, the reliability of users (in their ratings) is measured based on the messages formed by the algorithm. Using (39) and assuming that the predicted ratings in set $\mathcal{G}_{zk}$ are independent from each other at each intermediate step (as we discussed in Section 5.2), it can be shown that $\lambda_{k\to a}^{(\nu)}(G_{za}^{(\nu)}) \propto p(G_{za}^{(\nu)}|T_{ka}, R_{zk}^{(\nu-1)})$, where

$$p(G_{za}^{(\nu)} = \ell|T_{ka}, R_{zk}^{(\nu-1)}) =$$

$$\begin{cases} \Big[R_{zk}^{(\nu-1)} + (1-R_{zk}^{(\nu-1)}) \times \frac{1/|A_z-\ell|}{\sum\limits_{h\in\Upsilon} 1/|A_z-h|}\Big] & \text{if } T_{ka}=\ell \\ \\ \Big[(1-R_{zk}^{(\nu-1)}) \times \frac{1/|A_z-T_{ka}|}{\sum\limits_{h\in\Upsilon} 1/|A_z-h|}\Big] & \text{if } T_{ka}\neq\ell. \end{cases} \tag{41}$$

Here, $A_z = \frac{\sum_{i\in\mathbf{N_z}} T_{zi}}{|\mathbf{N_z}|}$ is the average rating of user z (for the items it previously rated). The way we compute the probabilities in (41) resembles the belief/pleusability concept of the Dempster-Shafer Theory [89] (similar to the discussion in Section 5.2). Given $T_{ka} = 1$, $R_{zk}^{(\nu-1)}$ can be viewed as the belief of user k that $G_{za}^{(\nu)}$ is one (at the ν^{th} iteration). In other words, in the eyes of user k, $G_{za}^{(\nu)}$ is equal to one with probability $R_{zk}^{(\nu-1)}$. Thus, $(1 - R_{zk}^{(\nu-1)})$ corresponds to the uncertainty in the belief of user k. In order to remove this uncertainty and express $p(G_{za}^{(\nu)}|T_{ka}, R_{zk}^{(\nu-1)})$ as the probabilities that $G_{za}^{(\nu)}$ is ℓ ($\ell \in \Upsilon$), we distribute the uncertainty among the possible outcomes (one to five) in proportion to A_z. Therefore, from user k's point of view, G_{za} is equal to one with probability $R_{zk}^{(\nu-1)} + (1-R_{zk}^{(\nu-1)}) \times \frac{1/|A_z-1|}{\sum\limits_{\gamma\in\Upsilon} 1/|A_z-\gamma|}$. On the other hand, it is equal to ℓ ($\ell \neq 1$) with probability $(1-R_k^{(\nu-1)}) \times \frac{1/|A_z-\ell|}{\sum\limits_{\gamma\in\Upsilon} 1/|A_z-\gamma|}$. We note that the above discussion assumed $T_{ka} = 1$ and similar statements hold for the cases when $T_{ka} = 2, 3, 4, 5$. The above computation in (41) must be performed for every neighbors of each factor node. This finishes the first half of the ν^{th} iteration.

In the second half of the ν^{th} iteration, we calculate the message $\mu_{a\to k}^{(\nu)}(G_{za}^{(\nu)})$ by multiplying all probabilities the variable node a received from its neighbors excluding

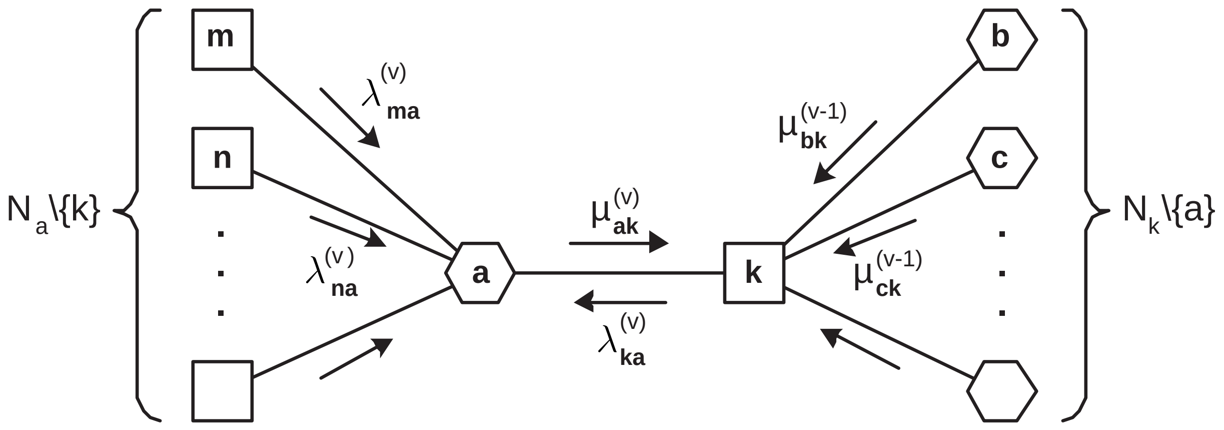


Figure 51: Message exchange between the factor node k and variable node a.

the one from the factor node k, as shown in the left half of Fig. 51. We note that the previous ratings of the active user play a key role in the algorithm. Hence, the values of those variables in $\mathbb{G}_z$ which are associated with the items already rated by the active user z are set to the corresponding ratings (i.e., $G_{zj} = T_{zj}$ if $j \in \mathbf{N_z}$). Thus, if $a \in \mathbf{N_z}$, the messages generated from the variable node a do not vary with iterations since the value of this variable node (G_{za}) is fixed based on the ratings of the active user. Therefore, the message from the variable node a to the factor node k at the ν^{th} iteration is given by

$$\mu_{a\to k}^{(\nu)}(G_{za}^{(\nu)} = \ell) =$$

$$\begin{cases} \frac{1}{\sum_{h\in\Upsilon}\prod_{i\in\Xi}\lambda_{i\to a}^{(\nu)}(h)} \times \prod_{i\in\Xi}\lambda_{i\to a}^{(\nu)}(G_{za}^{(\nu)}) & \text{if } a \notin \mathbf{N_z} \\ 1 & \text{if } a \in \mathbf{N_z} \text{ and } T_{za} = \ell \\ 0 & \text{if } a \in \mathbf{N_z} \text{ and } T_{za} \neq \ell. \end{cases} \tag{42}$$

The algorithm proceeds to the next iteration in the same way as the ν^{th} iteration. We clarify that the iterative algorithm starts by computing $\lambda_{k\to a}^{(1)}$ by using $R_{zk}^{(0)} = \varrho$, where ϱ $(0 < \varrho < 1)$ is the system's present confidence on the users for the reliability of their ratings computed at the previous execution of the algorithm. At the end of each iteration, the upper equation in (42), after following modification, is used to compute the prediction of ratings of the active user z. That is, we use the set $\mathbf{N_a}$ instead of Ξ in (42) to compute $\mu_a^{(\nu)}(G_{za}^{(\nu)})$ for every item a for which the active user z did not have any rating. Then, we set $G_{za}^{(\nu)} = \sum_{i=1}^{5} i\mu_a^{(\nu)}(i)$. The iterations stop when G_{zj} values converge for every item j.

7.3 Evaluation of BPRS

We evaluated the performance of BPRS using the 100K MovieLens dataset[2]. The dataset contains $100,000$ ratings from 943 users on 1682 items (movies) in which each user has rated at least 20 items. Further, the rating values are integers from 1 to 5. We implemented all the experiments via MATLAB on a 2 GHz PC with 4 GB RAM. We note that based on our simulations, we observed that BPRS converges, on the average, in 10 iterations. Therefore, for the remaining of this section, we either show our results during the first 10 iterations or after the 10^{th} iteration.

7.3.1 Prediction Accuracy

First, we evaluated the rating prediction accuracy of BPRS in terms of Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) metrics over the predicted ratings. For this evaluation, we used five different test datasets (i.e., we employed a five-fold cross-validation by evaluating BPRS using five different datasets) and we presented the average values resulting from these five datasets. We note that each test dataset is created by 80%/20% split of the full data into training and test data. We note that since this split is done randomly, each time we evaluated BPRS using a different training and test dataset. Then, we used the training data (80% of the whole dataset) to predict the ratings in the test dataset. It is worth noting that BPRS does

[2]Available at: http://www.grouplens.org/node/73.

not require a training period, and hence, we used the training set to compute the unknown ratings in the test data. Thus, we computed the RMSE and MAE as below:

$$\text{RMSE} = \sqrt{\frac{1}{|K|}\sum_{i\in\mathbb{U},j\in\mathbb{I}}(G_{ij}-\hat{G}_{ij})^2} \tag{43a}$$

$$\text{MAE} = \frac{1}{|K|}\sum_{i\in\mathbb{U},j\in\mathbb{I}}|G_{ij}-\hat{G}_{ij}|, \tag{43b}$$

where $|K|$ is the number of ratings (to be predicted) in the test dataset, $\hat{G}_{ij}$ is the actual value of the rating provided by user i for the item j in the test dataset, and G_{ij} is the predicted rating value by the algorithm. Next, we focus on some key components of BPRS, study their impacts to the prediction accuracy, and show the potential rooms for improvement in the proposed algorithm.

7.3.1.1 Handling the Uncertainty

In Section 7.2, to compute the message $\lambda^{(\nu)}_{k\to a}(G^{(\nu)}_{za})$ from the factor node k to the variable node a in (41), we distributed the uncertainty among the possible outcomes (one to five) in proportion to the average rating of the active user z (for the items it previously rated). Using this method, we observed that BPRS provides an RMSE of 0.9340 after running the algorithm for 10 iterations for each active user.

On the other hand, since the genres (i.e., types) of the movies are also available in the MovieLens dataset, we modified the above method and distributed the uncertainty in a more personalized way as follows. Given the active user is z, to compute $\lambda^{(\nu)}_{k\to a}(G^{(\nu)}_{za})$ from the factor node (user) k to the variable node (item) a, we first determine the genre (or genres if it has multiple genres) of item a[3]. We denote the genre (or the set of genres) of item a as κ_a. Then, we observe the histogram of the ratings provided by the active user z for the same genre (κ_a)[4] and distribute the uncertainty

[3]This information is available in the MovieLens dataset.

[4]We note that the rating histograms of users for the items in different genres can be computed offline.

in proportion to this histogram. That is, if the active user previously provided high ratings for the items in the same genre as κ_a, then we distribute most of the uncertainty to the higher ratings in proportion to the rating histogram of the active user for the items in the same genre as κ_a. Similarly, if the active user previously provided low ratings for the items in the same genre as κ_a, we distribute most of the uncertainty to the lower ratings. We note that if the active user z did not rate any items from this particular genre (κ_a), we distribute the uncertainty in proportion to the average rating of user z as we did previously. Using this more personalized technique, we observed that BPRS provides an RMSE of 0.9208 (after running the algorithm for 10 iterations for each active user). Thus, for the remaining of this section, we utilized this method to distribute the uncertainty and compute the message in (41).

7.3.1.2 Sampling the Graph

In Section 7.2, the factor graph $g(\mathbb{U}, \mathbb{I})$ is reduced to $g(\hat{\mathbb{U}}, \mathbb{I})$ by only keeping the users that are connected to z via a path (regardless of the length of the path) in $g(\mathbb{U}, \mathbb{I})$ and removing all the other user nodes from the graph together with their edges. Here, we want to observe the impact of sampling $g(\mathbb{U}, \mathbb{I})$ in a different way by only keeping the 2-hop neighbors of the active user z and removing all the other user nodes from the graph together with their edges. We note that the 2-hop neighbors of the active user z are the users who previously rated at least one common item with the active user z in the training dataset.

In Figs. 52 and 53, we show the RMSE and MAE provided by BPRS for these two different scenarios (i.e., when all users connected to each active user via a path are considered and when only the 2-hop neighbors of each active user are considered in the algorithm) for the first 10 iterations of the algorithm, respectively. Further, the RMSE and MAE values in Figs. 52 and 53 are listed in Tables 6 and 7. We observed that BPRS provides an RMSE of 0.9208 when all users connected to each active user via a path are considered, and an RMSE of 0.9198 when only the 2-hop neighbors of each active user are considered. Thus, we conclude that keeping only the 2-hop neighbors of each active user provides better performance in terms of both RMSE and MAE. Further, keeping fewer number of users also reduces the computational complexity as will be discussed later. We note that we also tried using at most 4-hop (i.e., only including 2-hop and 4-hop neighbors) and at most 6-hop neighbors (i.e., only including 2-, 4-, and 6-hop neighbors) of each active user as well. However, the results were almost the same as including all users connected to each active user via a path, and hence, we do not show those results here.

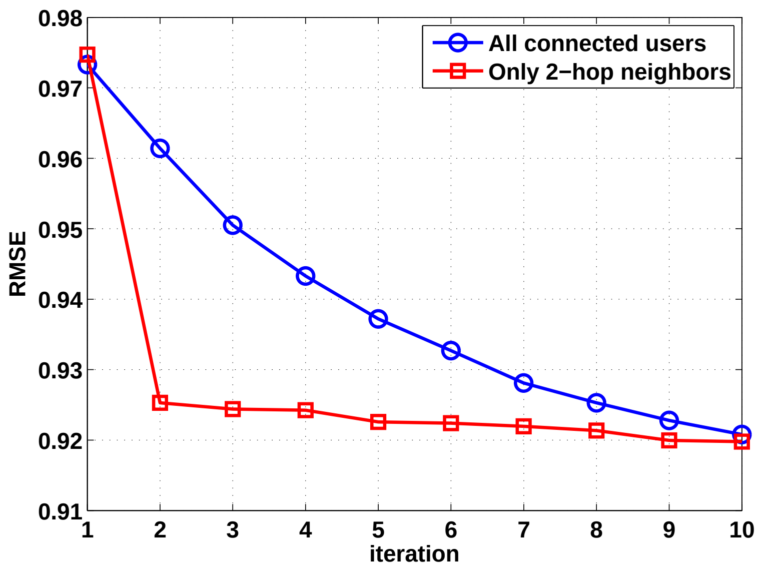


Figure 52: Performance of BPRS in RMSE vs. number of iterations when: (i) all users connected to each active user via a path, and (ii) only the 2-hop neighbors of each active user are considered.

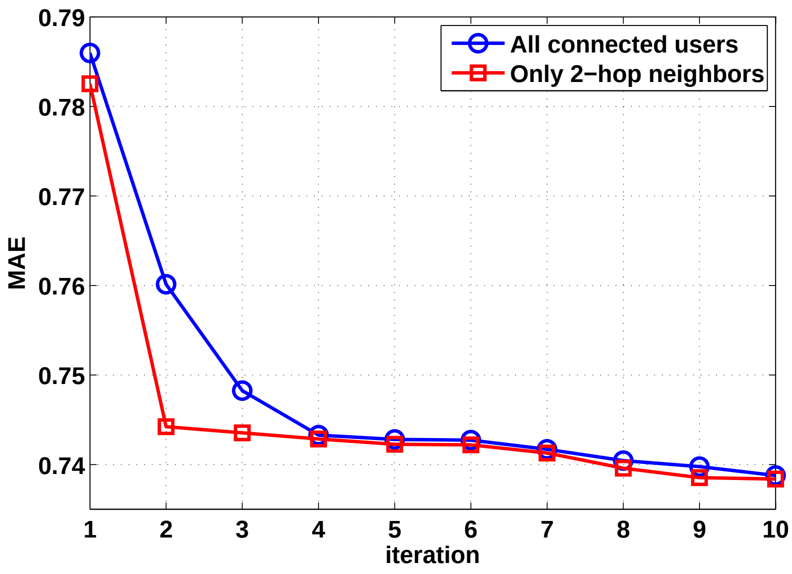


Figure 53: Performance of BPRS in MAE vs. number of iterations when: (i) all users connected to each active user via a path, and (ii) only the 2-hop neighbors of each active user are considered.

Table 6: Performance of BPRS in RMSE and MAE vs. number of iterations when all users connected to each active user via a path are considered.

iteration	1	2	3	4	5	6	7	8	9	10
RMSE	0.9733	0.9614	0.9505	0.9433	0.9372	0.9327	0.9281	0.9253	0.9228	0.9208
MAE	0.7860	0.7601	0.7483	0.7433	0.7428	0.7427	0.7417	0.7404	0.7398	0.7388

Table 7: Performance of BPRS in RMSE and MAE vs. number of iterations when only the 2-hop neighbors of each active user are considered.

iteration	1	2	3	4	5	6	7	8	9	10
RMSE	0.9747	0.9253	0.9244	0.9242	0.9226	0.9224	0.9220	0.9214	0.9200	0.9198
MAE	0.7825	0.7442	0.7436	0.7429	0.7423	0.7422	0.7413	0.7396	0.7385	0.7384

7.3.1.3 Biasing the Algorithm Toward the Active User

To give more weight to the ratings provided by the active user (for whom the algorithm computes the missing ratings), we studied further biasing the algorithm toward the active user by computing the user's reliability in (40) differently. In other words, instead of giving equal weight to each neighbor of user k while computing (40), we give more weight to user k's neighbors (in $\mathbf{N_k}$) which are also neighbors of the active user z. Thus, instead of computing the reliability as in (40), we compute the reliability of user k as below:

$$R_{zk}^{(\nu-1)} = 1 - \frac{1}{\rho[|\mathbf{N_k} \cap \mathbf{N_z}|(w-1) + |\mathbf{N_k}|]} \Big[\sum_{i \in (\mathbf{N_k} \cap \mathbf{N_z})} \sum_{x \in \Upsilon} w|T_{ki} - x|\mu_{i \to k}^{(\nu-1)}(x) + \sum_{j \in \mathbf{N_k} \setminus (\mathbf{N_k} \cap \mathbf{N_z})} \sum_{x \in \Upsilon} |T_{kj} - x|\mu_{j \to k}^{(\nu-1)}(x) \Big], \tag{44}$$

where w is the weight we give to the inconsistency of user k due to the items in $\mathbf{N_k} \cap \mathbf{N_z}$. We note that when $w = 1$, the weight given to each neighbor of user k becomes equal and (44) simplifies to (40).

In Figs. 54 and 55, we showed the RMSE and MAE performances of BPRS for different values of w, respectively. Again, we observed the performance when all users connected to each active user via a path are considered and when only the 2-hop neighbors of each active user are considered in the algorithm. Further, we listed the RMSE and MAE values of Figs. 54 and 55 in Tables 8 and 9. We again observed that when only the 2-hop neighbors of each active user are considered, BPRS provides better prediction accuracy. We further observed that there is an optimal point for w beyond which both RMSE and MAE increase. Further, when only the 2-hop neighbors of each active user are considered, RMSE gets its minimum value when $w = 5$ while MAE gets its minimum when $w = 7$.

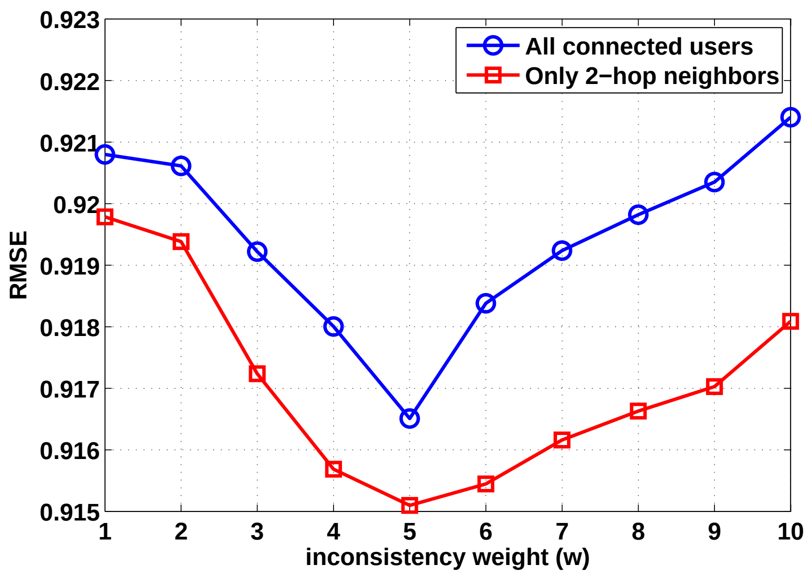


Figure 54: Performance of BPRS in RMSE vs. the inconsistency weight (w) when: (i) all users connected to each active user via a path, and (ii) only the 2-hop neighbors of each active user are considered.

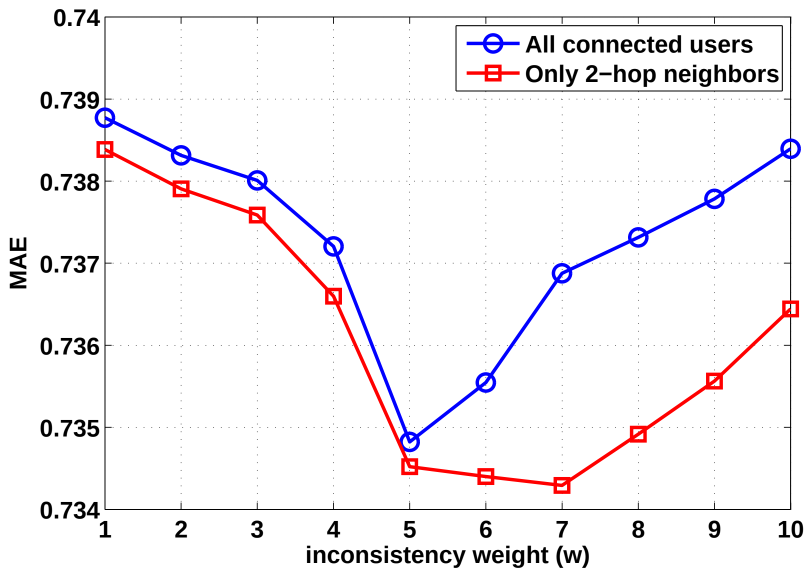


Figure 55: Performance of BPRS in MAE vs. the inconsistency weight (w) when: (i) all users connected to each active user via a path, and (ii) only the 2-hop neighbors of each active user are considered.

Table 8: Performance of BPRS in RMSE and MAE vs. the inconsistency weight (w) when all users connected to each active user via a path are considered.

w	1	2	3	4	5	6	7	8	9	10
RMSE	0.9208	0.9206	0.9192	0.9180	0.9165	0.9184	0.9192	0.9198	0.9204	0.9214
MAE	0.7388	0.7383	0.7380	0.7372	0.7348	0.7355	0.7369	0.7373	0.7378	0.7384

Table 9: Performance of BPRS in RMSE and MAE vs. the inconsistency weight (w) when only the 2-hop neighbors of each active user are considered.

w	1	2	3	4	5	6	7	8	9	10
RMSE	0.9198	0.9194	0.9172	0.9157	0.9151	0.9154	0.9162	0.9166	0.9170	0.9181
MAE	0.7384	0.7379	0.7376	0.7366	0.7345	0.7344	0.7343	0.7349	0.7356	0.7364

On the other hand, we observed that some users previously rated more common items with the active user, and hence, their opinions should be more valuable compared to the users who rated less (or no) common items with the active user. To study this, we modified the BP message in (42) in order to give more importance to the messages of the users who previously rated more common items with the active user as in the following.

Given the active user is z, we first determine the number of common items rated both by the active user z and the other users in the training dataset. Thus, we construct the vector $\Theta_z = \{\theta_{zi} : i \in \mathbb{U} \setminus \{z\}\}$, in which θ_{zi} denotes the number of common items rated both by user i and the active user z. Then, instead of computing as in (42), we compute the message from the variable node (item) a to the factor node (user) k as below:

$$\mu^{(\nu)}_{a\rightarrow k}(G^{(\nu)}_{za}=\ell) =$$

$$\begin{cases} \frac{1}{\sum\limits_{h\in\Upsilon}\prod\limits_{i\in\Xi}(\lambda^{(\nu)}_{i\rightarrow a}(h))^{\zeta_{zi}}} \times \prod\limits_{i\in\Xi}(\lambda^{(\nu)}_{i\rightarrow a}(G^{(\nu)}_{za}))^{\zeta_{zi}} & \text{if } a \notin \mathbf{N_z} \\ \\ 1 & \text{if } a \in \mathbf{N_z} \text{ and } T_{za}=\ell \\ 0 & \text{if } a \in \mathbf{N_z} \text{ and } T_{za}\neq\ell, \end{cases} \tag{45}$$

where

$$\zeta_{zi} = 1 + (\Psi - 1) \times \frac{\theta_{zi}}{max(\Theta_z)}, \tag{46}$$

and Ψ is the BP message weight used to give more weight to the messages of the users who previously rated more common items with the active user. We note that when $\Psi = 1$, the algorithm gives equal weight to the message of each user as we had previously.

In Figs. 56 and 57, we illustrate the RMSE and MAE provided by BPRS for different values of Ψ. Further, we listed the actual RMSE and MAE values of Figs. 56 and 57 in Tables 10 and 11. Once again, we observed that when only the 2-hop neighbors of each active user are considered, BPRS provides better prediction accuracy. Similar to the inconsistency weight (w), we observed that Ψ also has an optimal value ($\Psi = 2$) and after this value both RMSE and MAE start increasing.

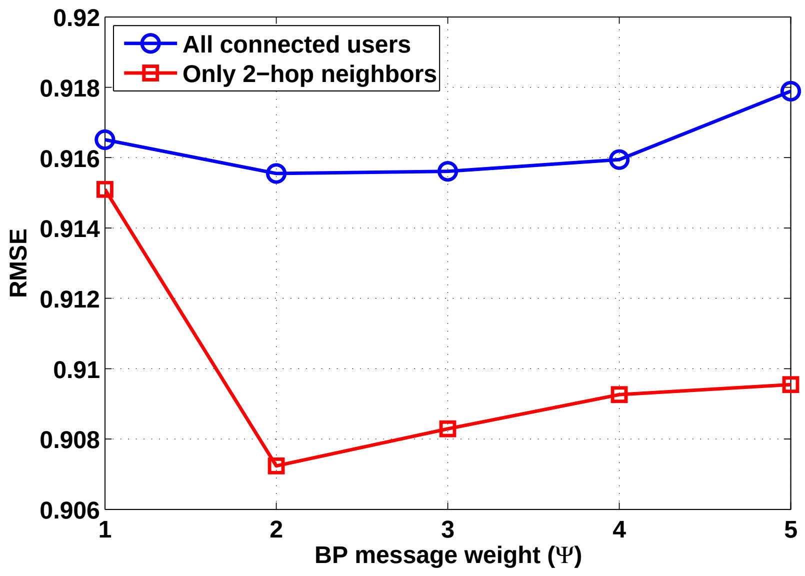


Figure 56: Performance of BPRS in RMSE vs. the BP message weight when: (i) all users connected to each active user via a path, and (ii) only the 2-hop neighbors of each active user are considered.

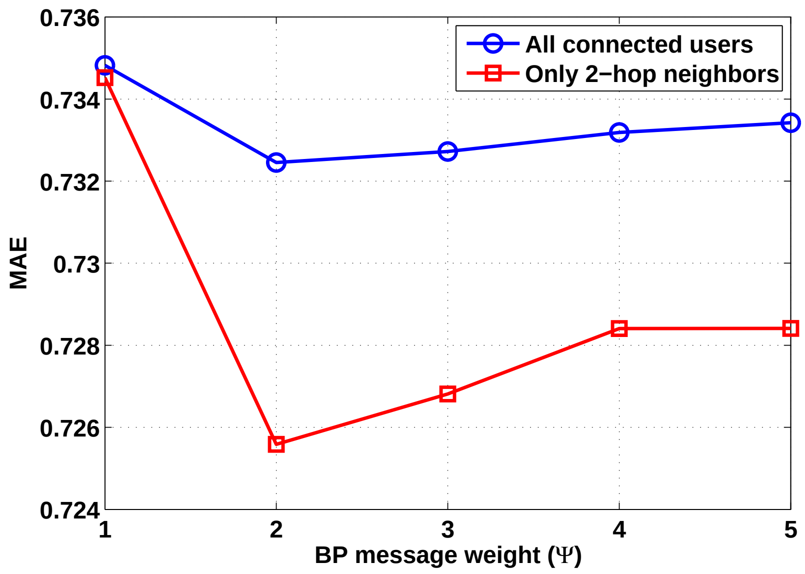


Figure 57: Performance of BPRS in MAE vs. the BP message weight when: (i) all users connected to each active user via a path, and (ii) only the 2-hop neighbors of each active user are considered.

Table 10: Performance of BPRS in RMSE and MAE vs. the BP message weight when all users connected to each active user via a path are considered.

BP message weight	1	2	3	4	5
RMSE	0.9165	0.9156	0.9156	0.9159	0.9179
MAE	0.7348	0.7325	0.7327	0.7332	0.7334

Table 11: Performance of BPRS in RMSE and MAE vs. the BP message weight when only the 2-hop neighbors of each active users are considered.

BP message weight	1	2	3	4	5
RMSE	0.9151	0.9072	0.9083	0.9093	0.9095
MAE	0.7345	0.7256	0.7268	0.7284	0.7284

7.3.1.4 Comparison with Other Schemes

We evaluated BPRS against some popular recommendation algorithms over the same dateset (100K MovieLens dataset). We report the results as 1. MovieAvg (which computes the predicting ratings for the movies by averaging all the received ratings for each movie) with an RMSE of 1.053, 2. Correlation-based neighborhood model (CorNgbr), which is one of the most popular cluster filtering methods, with an RMSE of 0.9406 [57], and 3. SVD latent factor model with 50 factors and RMSE of 0.9046 [57]. We conclude that BPRS, with an RMSE of 0.9072 (as illustrated in Table 11), is comparable to existing methods such as CorNgbr and SVD in terms of rating prediction accuracy. On the other hand, BPRS generates recommendations in linear complexity for each active user (as we will discuss in Section 7.3.3). Further, it updates the recommendations for each active user instantaneously using the most recent data (ratings) without solving the recommendation problem for all users (as we discussed in Section 7.1). Therefore, we claim that it has a significant advantage over the existing methods.

7.3.2 Precision and Recall

We also evaluated the usage precision accuracy of BPRS on the 100K MovieLens dataset. As such, we measured the percentage of the favored items that were suggested by the algorithm (i.e., recall) and the percentage of desirable recommendations (i.e., precision). For measuring precision and recall on the MovieLens dataset, we followed the method in [31]. We randomly sub-sampled 14% of the ratings from the MovieLens dataset to create a probe set[5]. Then, we let the training set M contain the remaining ratings and the test set T contain all the 5-star ratings from the probe set (i.e., T contains items relevant to the respective users). Next, for each test item i (rated by the user j) in the test set T, we randomly selected 500 additional items unrated by user j and we predicted the ratings for the test item i and for the additional 500 items via BPRS using the ratings in the training set M. To measure the precision and recall, we formed a top-N recommendation list by picking the N top ranked items (in terms of the value of their predicted ratings) as a result of BPRS. Then, if the test item is within the top-N recommendation list, we said that "we have a hit" (i.e., the test item i is recommended to the user), otherwise, "we have a miss". Finally, we computed the precision and recall introduced by BPRS by using the following two metrics:

$$\text{recall}(N) = \frac{\text{number of hits}}{|T|} \tag{47a}$$

$$\text{precision}(N) = \frac{\text{number of hits}}{N.|T|} = \frac{\text{recall}(N)}{N}, \tag{47b}$$

where $|T|$ is cardinality of set T. We illustrate recall versus N and precision versus recall provided by BPRS in Figs. 58 and 59, respectively when the inconsistency weight $w = 5$, BP message weight $\Psi = 2$, and only 2-hop neighbors of each active

[5]In [31], 1.4% of the rating are sub-sampled from the 1M MovieLens dataset. Therefore, we sub-sampled 14% of the ratings from the 100K MovieLens dataset to have approximately the same number of test items.

user are considered. As expected, as N (i.e., the number of items recommended by BPRS) increases, recall increases, however, precision decreases with increasing N. Further, we observed that the resulting precision and recall values by BPRS are comparable to other popular and accurate methods such as CorNgbr and SVD (refer to [31] for the precision and recall values of the other schemes).

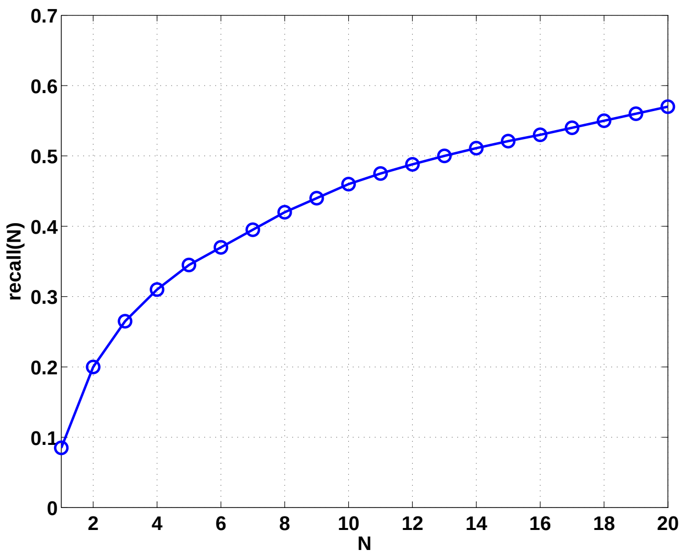


Figure 58: Recall.

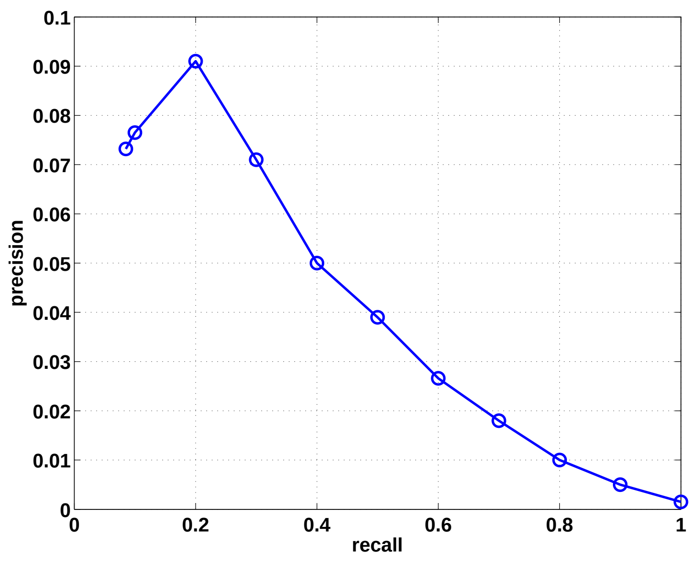


Figure 59: Precision vs Recall.

7.3.3 Computational Complexity

Assuming u users and s items in the system, we obtained the computational complexity of BPRS (in the number of multiplications) as $\mathbf{max}(\mathbb{O}(cs), \mathbb{O}(cu))$ per each active user, where c is the average number of nonzero elements in each row of the user-item matrix[6]. We note that due to the sparseness of the user-item matrix, the coefficient c is a small number. Further, as we discussed before, BPRS converges, on the average, in 10 iterations. Hence, we did not include the number of iterations in the complexity measure as it only introduces a small constant in front of the total complexity. This result indicates that BPRS can compute the recommendations for each active user very efficiently using the most recent data (ratings). On the other hand, SVD based recommender systems (whose prediction accuracy is slightly better than BPRS as discussed in Section 7.3.1.4) require solving the recommendation problem for all users even if they wish to update the recommendations for only a single

[6]We measure the complexity of BPRS per user as it can predict the ratings for each user separately using the most recent data (ratings); in contrast to other schemes.

active user using the most recent data. Further, the computational complexity of SVD based recommender systems is $\mathbb{O}(usk)$, where k is the number of latent factors used in the algorithm. Therefore, we claim that the BP-based approach toward the recommendation problem is very promising and can result in a new class of accurate and scalable recommender systems.

7.4 Summary

We introduced the "Belief Propagation-Based Iterative Recommender System" (BPRS). BPRS formulates the recommendation problem as making statistical inference about the ratings of users for unseen items based on observations. BPRS represents the recommendation problem on a factor graph and utilizes the belief propagation algorithm to efficiently estimate the marginal probability distributions of the variables representing the ratings of the items to be predicted via message passing between the items and the users. BPRS provides a complexity that remains linear per single active user, making it very attractive for large-scale systems. Further, it can update the recommendations for each active user instantaneously using the most recent data (ratings) and without solving the recommendation problem for all users. While providing these significant scalability advantages over the existing methods, we showed (via computer simulations using the 100K MovieLens dataset) that BPRS also provides comparable usage prediction and rating prediction accuracy to other popular methods such as Correlation-based neighborhood model (CorNgbr) and Singular Value Decomposition (SVD).

CHAPTER VIII

CONCLUSION

8.1 Contributions

In this dissertation, we explored and investigated new theoretical and practical challenges in the application of iterative algorithms and belief propagation in trust and reputation management and recommender systems.

First we proposed an algebraic iterative algorithm (referred as ITRM) for trust and reputation management in centralized environments. In a typical online transaction, the recipient of the service often has insufficient information about the service quality of the service provider before the transaction. Hence, the service recipient should take a prior risk before receiving the actual service. This risk puts the recipient into an unprotected position since he has no opportunity to try the service before he receives it. This problem gives rise to the use of reputation management systems in which reputations are determined by rules that evaluate the evidence generated by the past behavior of an entity within a protocol. Hence, after each transaction, a party who receives the service (referred to as the rater) provides (to the central authority) its report about the quality of the service provided for that transaction. The central authority collects the reports and updates the reputations of the service providers. Therefore, trust and reputation management systems are expected to lead various applications from social networks to ad-hoc networks in the near future. On the other hand, reputation management systems are subject to various manipulations, launched by a malicious participant or a group of colluding malicious participants. The proposed ITRM is a robust mechanism to evaluate the reputations of the service providers as well as the trustworthiness values of raters in their ratings. Comparison of ITRM with some well-known reputation management techniques (e.g., *Averaging Scheme*, *Bayesian Approach* and *Cluster Filtering*) indicates the superiority of the proposed scheme both in terms of robustness against attacks and efficiency. Furthermore, we showed that the computational complexity of the proposed ITRM is far less than the Cluster Filtering; which has the closest performance (to ITRM) in terms of resiliency to attacks. Specifically, the complexity of ITRM is linear in the number of users, while that of the Cluster Filtering is quadratic.

Then, we explored the application of ITRM for Delay Tolerant Networks (DTNs). DTNs are characterized by intermittent contacts between nodes, leading to space-time evolution of multihop paths for transmitting packets to the destination. Due to these special characteristics posed by DTNs, existing reputation systems are inefficient or impractical in such networks. On the other hand, adversary may mount several threats against DTNs to reduce the performance of the network. The most serious attacks are due to the Byzantine (insider) adversary in which one or more legitimate nodes have been compromised and fully controlled by the adversary. Among these attacks, packet drop is harder to contain because nodes' cooperation is fundamental for the operation of DTNs. Therefore, in this work, we considered the packet drop attack which gives serious damages to the network in terms of data availability, latency, and throughput. Thus, we introduced a distributed malicious node detection mechanism for DTNs using ITRM. The proposed work enables every node to evaluate other nodes based on their past behavior, without requiring a central authority. We showed that the resulting scheme effectively provides high data availability and low latency in the presence of Byzantine attackers. We also showed that the proposed iterative mechanism is far more effective than Bayesian framework and EigenTrust in computing the reputation values in a DTN environment.

Next, we extended ITRM and introduced the first application of the Belief Propagation (BP) algorithm, a fully probabilistic and iterative algorithm, for trust and reputation management (referred as BP-ITRM) in a centralized environment. In this work, we introduced, for the first time, a formulation in which the reputation management problem is described as an inference problem on a factor graph. This representation requires computing marginal likelihood distributions from complicated global functions of many variables. To solve this problem whose complexity grows exponentially, we resorted to BP whose computational efficiency (i.e., linear in the number of users) is driven by exploring the way in which the global functions factors into a product of simpler local functions. In BP-ITRM, the sellers (i.e., service providers) and buyers (i.e., raters) are represented via a factor graph on which they are arranged as two sets of variable and factor nodes that are connected via some edges. The reputation values of service providers are computed by message passing between nodes in the graph. We showed that the proposed iterative scheme is reliable (in filtering out malicious/unreliable ratings) while being computationally efficient (i.e., linear in the number of variables). Thus, it can be used as an effective and scalable reputation management system in many applications such as online services.

As expected, trust and reputation management is more complicated in distributed environments than in centralized solutions. Hence, next, we extended BP-ITRM and introduced the first application of BP algorithm in the design and evaluation of distributed trust and reputation management systems for P2P networks (referred as BP-P2P). In BP-P2P, the reputation and trustworthiness value of each peer is computed by distributed message passing among the peers on a factor graph. We compared BP-P2P with some well-known P2P reputation management schemes (such as the EigenTrust algorithm and the Bayesian Approach) and showed the superiority of BP-P2P in terms of its robustness against malicious behavior. Further, we showed that the computational complexity of BP-P2P grows only linearly with the number of peers in the network while its communication overhead is lower that the well-known EigenTrust algorithm.

Relying on the similarities between the reputation and recommender systems and successful application of iterative algorithms on reputation systems, finally, we introduced the application of BP algorithm to arrive at scalable, accurate and robust recommender systems. Recommender systems form a specific type of information filtering technique that aim to support users in their decision-making while interacting with large information spaces. There are two different sets in a recommender system: i) the set of users, and ii) the set of items (products). Users provide feedbacks, in the form of ratings, about the items for which they have an opinion. The main goal of a recommender system is to provide accurate recommendations for every user by predicting the ratings of the user for the items that he/she did not rate before. Our proposed mechanism (referred as BPRS) represents the recommendation problem on a factor graph and utilizes the BP algorithm to efficiently estimate the marginal probability distributions of the variables representing the ratings of the items to be predicted. We showed that BPRS computes the recommendations for each user with linear complexity and without requiring a training period. We further showed that BPRS is comparable to the state of art methods such as Correlation-based neighborhood model (CorNgbr) and Singular Value Decomposition (SVD) in terms of rating and precision accuracy.

8.2 Suggestions for Future Research

This dissertation opened up many interesting theoretical and practical research possibilities in trust and reputation management and recommender systems, and other related areas in ad-hoc networks and P2P networks. In the following, some of the interesting and potentially rich open directions for future research are listed.

- Scalability, complexity and convergence analysis of BP-ITRM and BPRS.
- Study of optimal attack strategies for BP-ITRM by the malicious users using a game theoretical approach.

- Evaluation of BP-ITRM and BPRS via real-life datasets and user studies.
- New applications of belief propagation-based trust and reputation management algorithm.
- New formulations of the recommender system problem on a factor graph, a pairwise Markov random field or a Bayesian network.
- Study of robust and manipulation resilient recommender systems.
- Utilizing BPRS to integrate data from different content domains.
- Analysis and impact of noisy belief propagation messages in a distributed environment.
- Study of various attacks on BP-P2P and prevention against malicious score managers.

REFERENCES

[1] "http://blog.auctionbytes.com/cgi-bin/blog/blog.pl?/comments/2009/11/1258208701.html." Oct. 2011.

[2] "http://pages.ebay.com/services/forum/feedback.html." Oct. 2011.

[3] "http://www.allexperts.com." Oct. 2011.

[4] "http://www.amazon.com." Oct. 2011.

[5] "http://www.ebay.com." Oct. 2011.

[6] "http://www.epinions.com." Oct. 2011.

[7] "http://www.netflixprize.com." Oct. 2011.

[8] "http://www.pewinternet.org/media-mentions/2004/americans-using-online-reputation-systems.aspx." Oct. 2011.

[9] Aberer, K. and Despotovic, Z., "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.

[10] Adomavicius, G. and Tuzhilin, A., "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 734–749, June 2005.

[11] Ayday, E., Lee, H., and Fekri, F., "An iterative algorithm for trust and reputation management," *ISIT '09: Proceedings of IEEE International Symposium on Information Theory*, 2009.

[12] Ayday, E., Lee, H., and Fekri, F., "Trust management and adversary detection in delay tolerant networks," *In Proceedings of IEEE Military Communications Conference (MILCOM)*, 2010.

[13] Ayday, E. and Fekri, F., "Using node accountability in credential based routing for mobile ad-hoc networks," *Proceedings of the Fifth IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2008.

[14] Ayday, E. and Fekri, F., "A belief propagation based recommender system for online services," *Proceedings of the fourth ACM conference on Recommender systems*, pp. 217–220, 2010.

[15] Ayday, E. and Fekri, F., "A protocol for data availability in mobile ad-hoc networks in the presence of insider attacks," *Elsevier Ad Hoc Networks*, vol. 8, pp. 181–192, Mar. 2010.

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

[17] Balabanovic, M. and Shoham, Y., "Fab: Content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, pp. 66–72, 1997.

[18] Bell, R. M. and Koren, Y., "Lessons from the netflix prize challenge," *ACM SIGKDD Explorations Newsletter*, vol. 9, pp. 75–79, December 2007.

[19] Blake, I., Seroussi, G., and Smart, N., "Advances in elliptic curve cryptography," *London Mathematical Society Lecture Note Series*, 2005.

[20] Bloom, B. H., "Space/time trade-offs in hash coding with allowable errors," *ACM Communications*, vol. 13, pp. 422–426, July 1970.

[21] Breese, J. S., Heckerman, D., and Kadie, C., "Empirical analysis of predictive algorithms for collaborative filtering," *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43–52, 1998.

[22] Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y.-C., and Jetcheva, J., "A performance comparison of multi-hop wireless ad hoc network routing protocols," *MobiCom '98: Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 85–97, 1998.

[23] Buchegger, S. and Boudec, J., "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)*, June 2002.

[24] Buchegger, S. and Boudec, J., "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.

[25] Buchegger, S. and J.Boudec, "Coping with false accusations in misbehavior reputation systems for mobile ad-hoc networks," *EPFL-DI-ICA Technical Report IC/2003/31*, 2003.

[26] Burgess, J., Bissias, G., Corner, M., and Levine, B., "Surviving attacks on disruption-tolerant networks without authentication," *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 61–70, 2007.

[27] Burke, R., "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, pp. 331–370, November 2002.

[28] Canny, J., "Collaborative filtering with privacy via factor analysis," *In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 238–245, 2002.

[29] Cheng, Z. and Hurley, N., "Effective diverse and obfuscated attacks on model-based recommender systems," *In Proceedings of the third ACM conference on Recommender systems*, pp. 141–148, 2009.

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

[31] Cremonesi, P., Koren, Y., and Turrin, R., "Performance of recommender algorithms on top-n recommendation tasks," *In Proceedings of the fourth ACM conference on Recommender systems*, pp. 39–46, 2010.

[32] Cui, S., Duan, P., and Chan, C., "An efficient identity-based signature scheme with batch verifications," *Proceedings of the 1st International Conference on Scalable Information Systems (InfoScale '06)*, p. 22, 2006.

[33] Damiani, E., di Vimercati, D. C., Paraboschi, S., Samarati, P., and Violante, F., "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.

[34] de Campos, L. M., Fernández-Luna, J. M., and Huete, J. F., "A collaborative recommender system based on probabilistic inference from fuzzy observations," *Fuzzy Sets Syst.*, vol. 159, pp. 1554–1576, June 2008.

[35] Dellarocas, C., "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.

[36] Deshpande, M. and Karypis, G., "Item-based top-N recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, pp. 143–177, January 2004.

[37] Dewan, P., Dasgupta, P., and Bhattacharya, A., "On using reputations in ad-hoc networks to counter malicious nodes," *Proceedings of the Tenth International Conference on Parallel and Distributed Systems (ICPADS04)*, 2004.

[38] Fall, K., "A delay-tolerant network architecture for challenged internets," *ACM SIGCOMM*, pp. 27–34, 2003.

[39] Ganeriwal, S. and Srivastava, M., "Reputation-based framework for high integrity sensor networks," *Proc. 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 66–77, 2004.

[40] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D., "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, pp. 61–70, December 1992.

[41] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C., "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[42] Groenevelt, R., Nain, P., and Koole, G., "The message delay in mobile ad hoc networks," *Performance Evaluation*, vol. 62, no. 1-4, pp. 210–228, 2005.

[43] Gupta, M., Judge, P., and Ammar, M., "A reputation system for peer-to-peer networks," *NOSSDAV '03: Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 144–152, 2003.

[44] Hao, F., Kodialam, M., and Lakshman, T. V., "Building high accuracy Bloom filters using partitioned hashing," *Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems*, pp. 277–288, 2007.

[45] Herlocker, J., Konstan, J. A., and Riedl, J., "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Information Retrieval*, vol. 5, no. 4, pp. 287–310, 2002.

[46] Herlocker, J. L. and Konstan, J. A., "Content-independent task-focused recommendation," *IEEE Internet Computing*, vol. 5, pp. 40–47, November 2001.

[47] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J., "An algorithmic framework for performing collaborative filtering," *In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 230–237, 1999.

[48] Herlocker, J. L., Konstan, J. A., and Riedl, J., "Explaining collaborative filtering recommendations," *In Proceedings of the ACM conference on Computer supported cooperative work*, pp. 241–250, 2000.

[49] Hofmann, T., "Latent class models for collaborative filtering," *In Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 688–693, 1999.

[50] Hong, S., Rhee, I., Kim, S. J., Lee, K., and Chong, S., "Routing performance analysis of human-driven delay tolerant networks using the truncated levy walk model," *Proceeding of the 1st ACM SIGMOBILE workshop on Mobility models*, pp. 25–32, 2008.

[51] Jin, R., Si, L., and Zhai, C., "A study of mixture models for collaborative filtering," *Inf. Retr.*, vol. 9, pp. 357–382, June 2006.

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

[53] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H., "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.

[54] Karypis, G., "Evaluation of item-based top-N recommendation algorithms," *In Proceedings of the tenth international conference on Information and knowledge management*, pp. 247–254, 2001.

[55] Kate, A., Zaverucha, G., and Hengartner, U., "Anonymity and security in delay tolerant networks," *Proceedings of the 3rd International Conference on Security and Privacy in Communication Networks (SecureComm07)*, 2007.

[56] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., Riedl, J., and Volume, H., "Grouplens: Applying collaborative filtering to usenet news," *Communications of the ACM*, vol. 40, pp. 77–87, 1997.

[57] Koren, Y., "Factorization meets the neighborhood: a multifaceted collaborative filtering model," *In Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 426–434, 2008.

[58] Koren, Y., Bell, R., and Volinsky, C., "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, pp. 30–37, August 2009.

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

[60] Kschischang, F. R., Frey, B. J., and andrea Loeliger, H., "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, 1998.

[61] Liu, K., Deng, J., Varshney, P. K., and Balakrishnan, K., "An acknowledgment-based approach for the detection of routing misbehavior in MANETs," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 536–550, May 2007.

[62] Liu, N. N. and Yang, Q., "Eigenrank: a ranking-oriented approach to collaborative filtering," *In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 83–90, 2008.

[63] Luby, M., "LT codes," *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, pp. 271–280, 2002.

[64] Macnaughton-Smith, P., Williams, W. T., Dale, M. B., and Mockett, L. G., "Dissimilarity analysis: A new technique of hierarchical subdivision," *Natue(202)*, pp. 1034–1035, 1964.

[65] Marti, S., Giuli, T., Lai, K., and Baker, M., "Mitigating routing misbehavior in mobile ad-hoc networks," *Proceedings of ACM International Conference on International Conference on Mobile Computing and Networking (MobiCom00)*, pp. 255–265, 2000.

[66] Mceliece, R. J., Mackay, D. J. C., and Cheng, J.-F., "Turbo decoding as an instance of Pearl's "belief propagation" algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 140–152, 1998.

[67] Mobasher, B., Dai, H., Luo, T., and Nakagawa, M., "Discovery and evaluation of aggregate usage profiles for web personalization," *Data Mining and Knowledge Discovery*, vol. 6, no. 1, pp. 61–82, 2002.

[68] Page, L., Brin, S., Motwani, R., and Winograd, T., "The pagerank citation ranking: Bringing order to the web," tech. rep., Stanford Digital Library Technologies Project, 1998.

[69] Paul, K. and Westhoff, D., "Context aware detection of selfish nodes in dsr based ad-hoc networks," *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM02)*, pp. 178–182, 2002.

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

[71] Pennock, D., Horvitz, E., Lawrence, S., and Giles, C. L., "Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach," *In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 473–480, 2000.

[72] Petz, A., Enderle, J., and Julien, C., "A framework for evaluating DTN mobility models," *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, pp. 94:1–94:8, 2009.

[73] Pirzada, A. A., McDonald, C., and Datta, A., "Performance comparison of trust-based reactive routing protocols," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 695–710, June 2006.

[74] Pishro-Nik, H. and Fekri, F., "On decoding of low-density parity-check codes on the binary erasure channel," *IEEE Transactions on Information Theory*, vol. 50, pp. 439–454, March 2004.

[75] Pishro-Nik, H. and Fekri, F., "Results on punctured low-density parity-check codes and improved iterative decoding techniques," *IEEE Transactions on Information Theory*, vol. 53, pp. 599–614, Feb. 2007.

[76] Potlapally, N., Ravi, S., Raghunathan, A., and Jha, N., "Analyzing the energy consumption of security protocols," *Proceedings of International Symposium of Low Power Electronics and Design*, Aug. 2003.

[77] Rashid, A. M., Lam, S. K., Karypis, G., and Riedl, J., "ClustKNN: a highly scalable hybrid model-& memory-based CF algorithm," *In Proceedings of WebKDD-06, KDD Workshop on Web Mining and Web Usage Analysis, at 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2006.

[78] Resnick, P. and Zeckhauser, R., "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system," *Workshop on Emprical Studies of Electronic Commerce*, 2002.

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

[80] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J., "GroupLens: an open architecture for collaborative filtering of netnews," *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pp. 175–186, 1994.

[81] Resnick, P. and Varian, H. R., "Recommender systems," *Communications of the ACM*, vol. 40, pp. 56–58, March 1997.

[82] Rhee, I., Shin, M., Hong, S., Lee, K., and Chong, S., "On the levy walk nature of human mobility," *INFOCOM '08: Processings of the IEEE International Conference on Computer Communications*, 2008.

[83] Richardson, T. J. and Urbanke, R. L., "The capacity of low-density parity check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, pp. 599–618, Feb. 2001.

[84] Sarwar, B., Karypis, G., Konstan, J., and Reidl, J., "Item-based collaborative filtering recommendation algorithms," *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, 2001.

[85] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. T., "Application of dimensionality reduction in recommender system - a case study," *ACM WebKDD Workshop*, 2000.

[86] Schafer, J. B., Konstan, J., and Riedi, J., "Recommender systems in e-commerce," *In Proceedings of the 1st ACM conference on Electronic commerce*, pp. 158–166, 1999.

[87] Schafer, J. B., Konstan, J. A., and Riedl, J., "E-commerce recommendation applications," *Data Min. Knowl. Discov.*, vol. 5, pp. 115–153, January 2001.

[88] Seth, A. and Keshav, S., "Practical security for disconnected nodes," *Proceedings of the 1st IEEE ICNP Workshop on Secure Network Protocols (NPSec)*, pp. 31–36, 2005.

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

[90] Shafer, G., "The Dempster-Shafer theory," *Encyclopedia of Artificial Intelligence*, 1992.

[91] Shokrollahi, A., "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, pp. 2551–2567, June 2006.

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

[93] Srebro, N., Rennie, J., and Jaakkola, T., "Maximum margin matrix factorizations," *In Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2005.

[94] Stern, D. H., Herbrich, R., and Graepel, T., "Matchbox: large scale online bayesian recommendations," *In Proceedings of the 18th international conference on World wide web*, pp. 111–120, 2009.

[95] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H., "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.

[96] Sun, Y., Yu, W., Han, Z., and Liu, K., "Information theoretic framework of trust modeling and evaluation for ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 305–317, Feb. 2006.

[97] Van Roy, B. and Yan, X., "Manipulation-resistant collaborative filtering systems," *In Proceedings of the third ACM conference on Recommender systems*, pp. 165–172, 2009.

[98] Vellambi, B. N. and Fekri, F., "Results on the improved decoding algorithm for low-density parity-check codes over the binary erasure channel," *IEEE Transactions on Information Theory*, vol. 53, pp. 1510–1520, April 2007.

[99] Vellambi, B. N., Subramanian, R., Fekri, F., and Ammar, M., "Reliable and efficient message delivery in delay tolerant networks using rateless codes," *MobiOpp '07: Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, pp. 91–98, 2007.

[100] Wang, J., de Vries, A. P., and Reinders, M. J. T., "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," *In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 501–508, 2006.

[101] Wang, Y., Jain, S., Martonosi, M., and Fall, K., "Erasure-coding based routing for opportunistic networks," *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 229–236, 2005.

[102] Weimer, M., Karatzoglou, A., and Smola, A., "Adaptive collaborative filtering," *In Proceedings of the 2008 ACM conference on Recommender systems*, pp. 275–282, 2008.

[103] Whitby, A., Josang, A., and Indulska, J., "Filtering out unfair ratings in bayesian reputation systems," *AAMAS '04: Proceedings of the 7th International Workshop on Trust in Agent Societies*, 2004.

[104] Wiberg, N., "Codes and decoding on general graphs," *PhD thesis, Linkping University, Sweden*, 1996.

[105] Yang, Y., Feng, Q., Sun, Y. L., and Dai, Y., "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.

[106] Yedidia, J. S., Freeman, W. T., and Weiss, Y., "Constructing free energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, pp. 2282–2312, 2005.

[107] Yu, W. and Liu, K. R., "Game theoretic analysis of cooperation stimulation and security in autonomous mobile ad-hoc networks," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 507–521, May 2007.

[108] Zacharia, G., Moukas, A., and Maes, P., "Collaborative reputation mechanisms in electronic marketplaces," *HICSS '99: Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume 8*, 1999.

[109] Zhang, C., Lu, R., Lin, X., Ho, P.-H., and Shen, X., "An efficient identitybased batch verification scheme for vehicular sensor networks," *INFOCOM '08: Processings of the IEEE International Conference on Computer Communications*, 2008.

[110] Zhang, J. and M.Fossorier, "Shuffled belief propagation decoding," *Proceedings of the 36th Asilomar Conference on Signals, Systems and Computers*, Nov. 2002.

VITA

Erman Ayday received his B.S. degree in Electrical and Electronics Engineering from the Middle East Technical University, Ankara, Turkey, in 2005. He received his M.S. degree from the School of Electrical and Computer Engineering (ECE), Georgia Institute of Technology, Atlanta, GA, in 2007. He is currently a Research Assistant in the Information Processing, Communications and Security Research Laboratory and pursuing his Ph.D. degree at the School of ECE, Georgia Institute of Technology, Atlanta, GA. His current research interests include wireless network security, game theory for wireless networks, trust and reputation management, and recommender systems. Erman Ayday is the recipient of 2010 Outstanding Research Award from the Center of Signal and Image Processing (CSIP) at Georgia Tech and 2011 ECE Graduate Research Assistant (GRA) Excellence Award from Georgia Tech.