A Robust Distributed Reputation Mechanism for Peer-to-Peer Systems

Rahim Delaviz Aghbolagh

A Robust Distributed Reputation Mechanism for Peer-to-Peer Systems

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 1 Oktober 2013 om 15:00 uur

door **Rahim DELAVIZ AGHBOLAGH**

Master of Technology in Programvaruteknik för Distribuerade System,
Kungliga Tekniska Högskolan (KTH), Stockholm, Sweden
geboren te Ahar, IRAN

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. D.H.J. Epema

Samenstelling promotiecommissie:

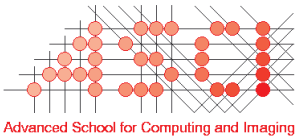
Rector Magnificus	voorzitter
Prof.dr.ir. D.H.J. Epema	Technische Universiteit Delft and TU Eindhoven, promotor
Dr.ir. J.A. Pouwelse	Technische Universiteit Delft, copromotor
Prof.dr.ir. P.F.A. Van Mieghem	Technische Universiteit Delft
Prof.dr. S. Etalle	Technische Universiteit Eindhoven
Prof.dr. W.J. Fokkink	Vrije Universiteit Amsterdam
Prof.dr. O. Babaoglu	University of Bologna, Italy
Dr. A. Datta	Nanyang Technological University, Singapore
Prof.dr.ir. H.J. Sips	Technische Universiteit Delft, reservelid

Published and distributed by: Rahim Delaviz Aghbolagh
E-mail: rahim@delaviz.nl

ISBN: 978-94-6186-208-2

Keywords: reputation systems, distributed systems, peer-to-peer systems, sybil, defense, scalability, BitTorrent, network analysis.

Printed in The Netherlands by: Wöhrmann Print Service.



The work described in this thesis has been carried out in the ASCI graduate school. ASCI dissertation series number 222.

This work was supported by the European Community's Sixth Framework Programme through the P2P-FUSION project (grant no. 035249)

Acknowledgments

Doing a PhD is like starting from the middle of the ocean and sailing to the shore. This journey is not doable without the help, the support, and the advice of many people on the path. Now that I am writing the final pages of this thesis, the acknowledgement, I am delightful to thank those who supported me to reach this point.

Dick, it was a great pleasure to work with you during the past four years. Your attention to detail, both in content and in writing, helped me a lot to improve the quality of this thesis. I will remember the times after our meetings, I was coming back to my office with papers marked in red. I appreciate your calm and encouraging personality, and the opportunity that you gave me to work on the subjects that I was interested in.

Johan, thanks a lot for giving me the opportunity to join the Tribler team at TU Delft. You are among the few academics with practical goals in mind. I wish you success in getting new grants and keeping the corridor busy. Also, thanks for approving my thesis from the middle of the wilderness in Tanzania.

Naza, special thanks goes to you. You helped me a lot during the critical years of my PhD. I found your inputs and criticisms very helpful, when I was frustrated by many trials and errors. Besides being a colleague, you were and will remain a great friend for all of us in the group.

Nitin, besides being a generous and helpful officemate you were a great friend as well. We explored a few places in the Netherlands and Belgium and it was quite fun. You liked to live and work in North America, I am sure with your level of intelligence and hard work, soon you will end up in a great place.

Dimitra, you are a true lady and a Computer Science researcher at the same time. I appreciate your inspiration on using the centrality measures in my first paper. I will remember how much we laughed at Fresco's botched restoration; one day we should visit that church :)

Boxun, we always had interesting discussion from "horseburger" to "bicology". I enjoyed a lot having dinner with you and Nitin at DailyWok and Burgerz and laughing at the names of the burgers, like "mother of all burgers" (with some tweaks). Also, I should thank you for your tips to improve the graphs that I was making for my papers.

Mihai, we learned tennis from each other by playing on Saturdays 10 a.m.. Your lunch

time stories about political corruptions in Romania were quite fun; in that regard, it seems that the politicians in my country are not alone.

Niels, the NewYork trip with you and your girlfriend, Janna, was very joyful. Specially the three 45 cm pizza at the Grimaldi pizzeria in Brooklyn, even Americans were looking at us strangely.

Adele, you are a good researcher (and a member of the Chinese People's party :)), one day you will become a successful professor at a top university.

Lucia, you were always raising interesting topics to talk about during the lunch time. Now, your new colleagues enjoying this talent of you; keep up the good work.

Riccardo, you are a friendly person and I always enjoyed talking with you about your trips to Thailand, Argentina and many more interesting places.

Boudewijn, you are a great developer, I am sure with a bit of teamwork you can develop high quality applications.

Arno, you were a generous and helpful colleague for each of us in the group. You are a nature lover; keep posting nice photos on Facebook.

Siqi, Yong, Jie, and Jianbin, I enjoyed playing badminton with you, even though I never won. One day, I will return as Forret Gump and will beat you all :)

Paulo, Munire, and Stephen, thanks for being so nice persons and keeping the ICT facilities working.

Reza, you are really a great friend. You were there when I needed you and I wish one day I be able to answer your generosity. I always enjoyed trash-talking with you when playing tennis.

Ali, Javad, Arash, Farshad, and Rouzbeh my dear Iranian friends at Delft. We had joyful times at Delft and enjoyed of visiting many places. With you I never felt homesick. I wish you all the best.

Ali, you were always a good friend for me during the last fourteen years that we know each other. I wish you many years of happy life with your little baby, Aida, and your wife, Faranak.

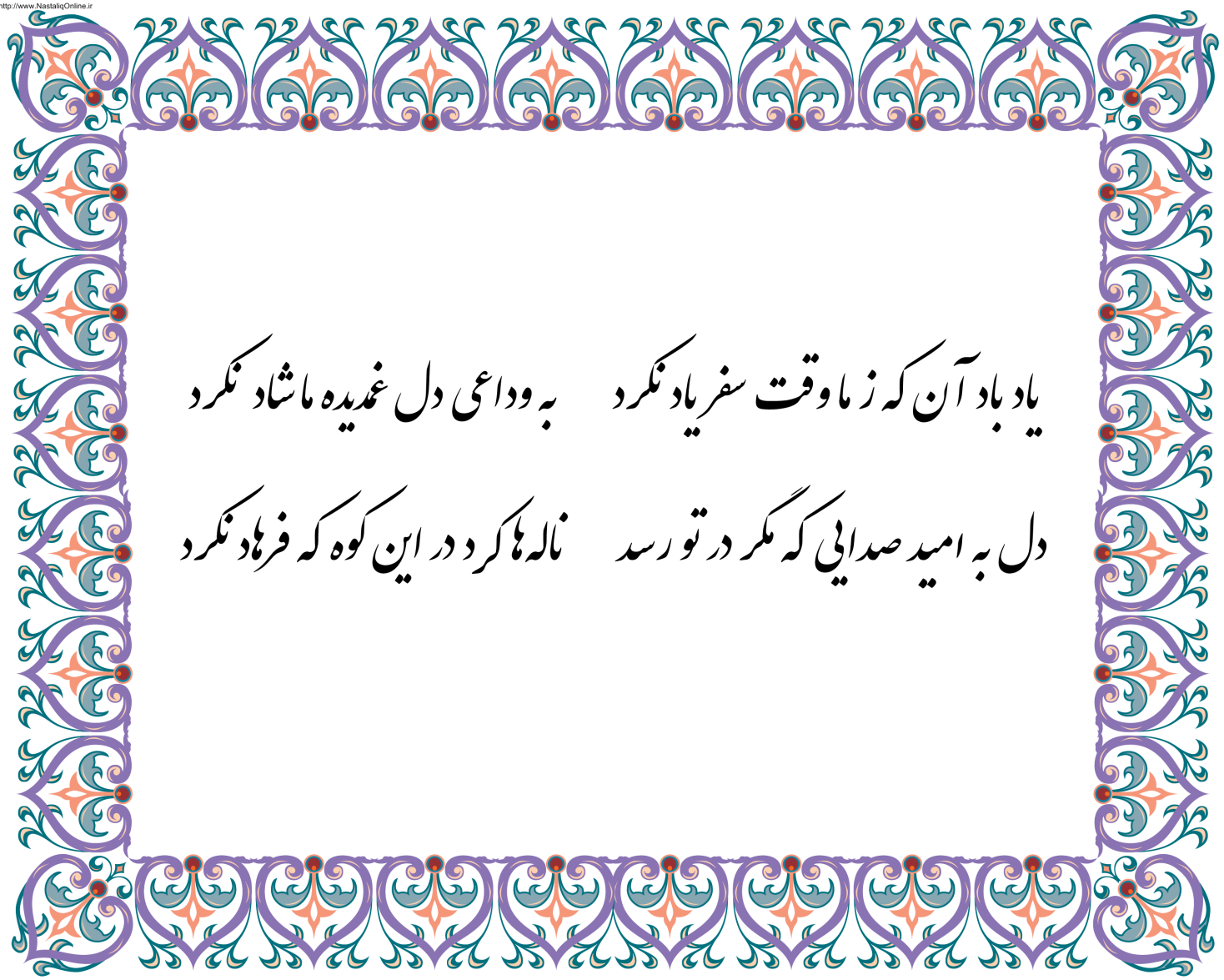
Ali, your were a nice colleague when we were working at PEC, Tehran. Thanks for being a reliable and helpful friend.

Sara, you have a high sense of humor and talking with you is very enjoyable. I wish one day you will run a successful tourist agency and will offer yourself good deals :)

Above all, I would like to thank my family for their love and constant support. You supported me when I needed you, you encouraged me when I was struggling and doubtful, and you filled my absence when I was needed. I love you all!

Mom and Dad, you supported me in all stages of my life and I wish I could have you in my defense. I missed you a lot and I am not able to express my feelings in words! I dedicate the following poem to you and to all who fought cancer [1].

[1] The poem is from the Iranian poet, Hafez, 1325-1389



یاد باد آن که ز ما وقت سفر یاد نکرد به وداعی دل غمدیده ما شاد نکرد

دل به امید صدایی که مگر در تو رسد ناله‌ها کرد در این کوه که فرهاد نکرد

Contents

Chapter 1

Introduction

In recent years, *Information Technology* has reshaped our private and social lives, it has created new business opportunities, and it has transformed many of our well-established processes into new forms. Using modern communications technology, it is common for two arbitrary persons in distant locations to encounter and transact with each other for different purposes, like buying/selling goods. In the *offline world*, due to its smaller scope and the existence of mature procedures that have come into existence in the course of time, interactions and their consequences are easy to manage. As an example, consider the act of *buying*, which we experience almost every day, and which is simply the process of exchanging a good for money. In such an action, the *trust* that the involving parties hold in each other facilitates the process. Such a simple interaction becomes complex in the *online world*, where it requires specific infrastructures and automated procedures. In the dispersed and large-scale environment of the online world, *online reputation mechanisms* can facilitate such a trust-based processes through scoring the system participants. The main idea behind reputation mechanisms is to use historical data to score the system participants and to make informative decisions in the future. In this regard, there are four standing questions: what type of data to collect, how to collect the required data, how to process the collected data, and how to score the entities in the system, such as the seller of a good in the online marketplace or the good itself? Besides, depending on the operational environment, extra concerns like security, may come into picture as well.

In this thesis, we are interested in *distributed reputation mechanisms*. In such a mechanism, collecting data and calculating reputation scores are done by the participants themselves, and there is no central and dedicated authority to perform these tasks. Such mechanisms are organic matches for *Peer-to-Peer (P2P) file-sharing* systems that are used for publishing and downloading content such as videos in a decentralized fashion. In a P2P file-sharing system, there is no central server to supply the content requesters with the content, and the contributions of the system participants are required for the good operation of the system. In P2P file-sharing systems, if everybody shares content eventually

everybody will obtain a high quality service, bur for rational peers (participants) providing resources, such as bandwidth, without getting anything in return is not interesting, and they need a stronger motivation to cooperate. One class of mechanisms that have been designed to motivate or even force peers to cooperate consists of reputation mechanisms.

Distributed reputation mechanisms for large-scale applications like P2P file-sharing come with many challenges. In this thesis, we address these challenges by designing, implementing, and analyzing techniques for making distributed reputation mechanisms more robust. We incorporate these techniques into the BarterCast reputation mechanism [68], which is already deployed in the Tribler BitTorrent client [82]. In BarterCast, peers use a gossiping protocol to inform other peers about their own upload and download activities. From the data received through this protocol, every peer locally builds a weighted, directed graph and computes reputation values for other peers in this graph. We make this mechanism more robust with respect to three important aspects. First, the reputation values should accurately reflect the true behaviors of the peers, and we study the opportunities for improving the accuracy of BarterCast. Secondly, as the open and distributed environment of P2P systems is lucrative for malicious actors, we study security issues in BarterCast and make it more secure. Thirdly, an acceptable tradeoff between scalability of the mechanism, in terms of communication, storage, and computation costs, and the accuracy of the computed reputation values is desired. We study this problem in BarterCast, and without compromising the reputation accuracy, we make it highly scalable. Moreover, through employing the data collected from the Tribler network, we perform a thorough and insightful study of the BarterCast mechanism from the perspective of network science, and we gain an understanding that may lead to further improvements.

1.1 Reputation Mechanisms

According to the Oxford dictionary[1] the term *reputation* is defined as "the beliefs or opinions that are generally held about someone or something". As can be understood from this definition, the reputation of an entity consists of the collective opinions about it that is built up in the course of time. The concept of reputation is closely related to the concept of *trust*, which is defined as "the firm belief in the reliability, truth, or ability of someone or something". In general, trust is built between a pair of people, the one who *trusts* and the one who is *trusted*, but reputation is widespread, and is the accumulation of the opinions that are held about somebody or something. Notice that these are just general definitions of these concepts, and depending on the application, the distinction between reputation and trust may become blurry. Below we briefly discuss where reputation systems come from and we elaborate on the building blocks of the online versions of these mechanisms.

[1] http://oxforddictionaries.com

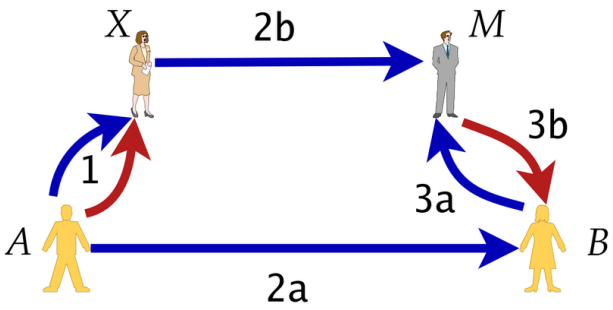


Figure 1.1: An example of a Hawala transaction (red lines indicate physical money transfers and blue lines indicate communications).

1.1.1 An Anecdotal History of the Concept of Reputation

Before we turn to an in-depth discussion of reputation mechanisms, we briefly present one of the origins of trust-based trading in old times and how it influenced modern reputation mechanisms. Before the existence of banks and modern financial transaction facilities, *trust-based* trading was the main way of doing long-distance business. A famous example of a historical and trust-based mechanism is *Hawala*[2], which is still legally used in Middle East countries like Iran, or illegally by those who are in search of special interests like money laundering [50]. Hawala is solely based on trust and it originated centuries ago in the Middle East, when traders started doing business with faraway places and there was a threat of robbery on the highway when carrying physical money. Figure 1.1[2] presents a Hawala transaction that takes place between A and B; here X and M are the Hawala brokers. The steps of this transaction are as follow. First, A approaches X and gives her an amount of money to be transfered to B and shares a password with her; here A trusts X to do a safe payment to B. Second, the broker X contacts M, generally they are located faraway, and informs him on the agreed fund and password and asks him to pay B if B gives the right password. In this step, M trusts X for the later settlement and payback by her. Third, B, who already has been informed by A about the amount and the shared password, goes to M and completes the transaction. Notice that during this transaction, no money is transferred between X and M, but later on they accumulate all their transactions and perform a clearance.

1.1.2 Online Reputation Mechanisms

Even though the spread of modern transactional systems has reduced the need for trust-based money transfers like Hawala, the emergence of online services, markets, and col-

[2]http://en.wikipedia.org/wiki/hawala

laborative applications has created new opportunities for reputation and trust mechanisms. Nowadays, almost everybody with Internet access performs online transactions, e.g., in the forms of buying a product, booking an accommodation, getting advice on a product or service, and hundreds of similar services. In comparison with a physical transaction, an online interaction differs from the offline version in many ways. Besides the technical differences, the involving parties do not have a prior interaction experience with each other, and they may not have any idea of the trustworthiness of the other party. Moreover, due to the ease of initiating such a transaction, and the limited information about the counter-party's reliability, misbehaving, fraud, and misuse of the system are highly plausible in such systems. So, providing the users with an acceptable level of assurance about the reliability of the product or service is required for the healthy and wealthy (high number of users) operation of the system. In such a situation, reputation mechanisms can help users to get through.

The fundamental idea behind (online and offline) reputation mechanisms is twofold. First, human behavior does not change radically over time, and what we have done in the past is a good indicator of how we may behave in the future. Secondly, through acquiring individual experiences about an entity, e.g., the trustworthiness of a seller or the reliability of a product, we can build a reusable and valuable knowledge-base for making informed decisions in the future. Figure 1.2 presents this concept in a simple way. The input to the reputation mechanism may come from many sources; the main source is feedback on past transactions that is provided by the system participants through rankings, comments, scores, or other forms as input [35]. The output of the mechanism is a score, a statement, or a rank that is assigned to the entity of interest, e.g., a person or a product. A user who wants to use the service but has no idea of whom to trust, consults the reputation scores to select a proper candidate to interact with. In Section 1.1.3, we will elaborate on the building blocks of typical online reputation mechanisms.

Reputation mechanisms are used widely in the context of eCommerce and they have facilitated fraud avoidance and brought trust into these systems. Amazon[3] and eBay[4] are two well known examples of successful eCommerce businesses that employ reputation mechanisms in their business models. Amazon uses different forms of feedbacks, e.g., scores, comments, videos, images, to rank products and also reviewers. In Amazon, based on the received scores, a reviewer can boost his rank and become one of the top-1000 reviewers [49]. In eBay, buyers and sellers can rate each other and when a transaction is complete, they can write a comment about their experience. Through accumulating the received scores, eBay assigns each buyer and seller a rank [49], which is very influential in future transactions. Several studies have shown that a seller's reputation has a significant and positive influence on the prices that he can obtain in eBay [45, 63].

[3] http://www.amazon.com
[4] http://www.ebay.com

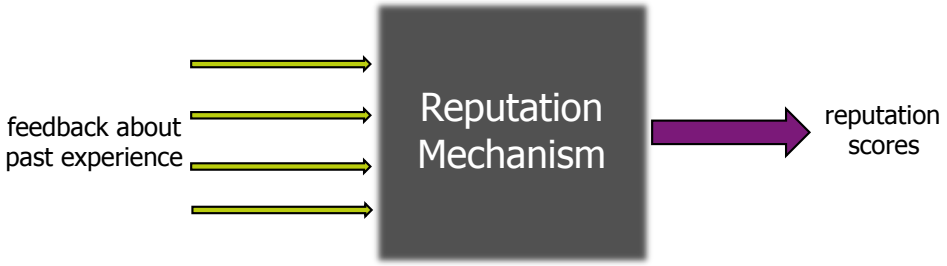


Figure 1.2: A reputation mechanism presented as a black box with its external interfaces.

Besides facilitating transactions in electronic markets, reputation mechanisms play an important role in distributed systems as well. For instance, they are used to filter out inauthentic contents in P2P file-sharing systems [25,51,104], to diminish the influence of malicious acts, and to push down inauthentic comments in a list of given comments[5] [12, 68,69,77]. Section 1.1.4 presents a summary of some distributed reputation mechanisms.

1.1.3 The Building Blocks of Reputation Mechanisms

The breakdown of a reputation mechanism into components can be realized in many ways. In this section we introduce two ways of structural decomposition of reputation mechanisms and elaborate on them.

The first breakdown is proposed by Marti et al. [66], who identify three main components for a reputation mechanism, and who present a breakdown of each component into smaller pieces, see Table 1.1. The three components are:

- *Information gathering*: This component is responsible for collecting information on the behavior of the system participants. Its sub-components deal with the issues of managing user identities, the trustworthiness of the information sources, aggregating information from different sources, and dealing with newcomers.

- *Scoring and ranking*: Once the information is collected, whether partially or completely, the next step is to compute a reputation score for an entity of interest. Such a computation may be done by an evaluator himself, by a centralized service, or in a quorum. A reputation score may be a binary value, a real number in a specific range; alternatively the mechanism designer may define qualitative values to represent the scores. Once the computation is done the evaluator should decide which entity, if any, to trust. This decision may be based on a minimum threshold or on the relative ranks of the entities.

[5] http://slashdot.org

- *Response*: This is the motivation engine of the reputation mechanism that encourages and discourages good and bad behaviors, respectively. The component may incentivize system participants for their good behavior, or it may punish the misbehaving participants.

Information gathering component	**Scoring and ranking component**	**Response component**
Identity scheme Information sources Information aggregation Stranger policy	Good vs. bad behavior Quantity vs. quality Time dependence Selection threshold Peer selection	Incentives Punishment

Table 1.1: Components and sub-components of a reputation system [66].

The second structural framework for reputation systems is proposed by Hoffman et al. [44], who put a lot of effort in the study of the attack and defense mechanisms in reputation systems. To compare different mechanisms, they have built an *analysis framework* that encompasses many types of reputation mechanisms. Their framework decomposes reputation mechanisms into the three components of *formulation*, *calculation*, and *dissemination*, where each component comprises smaller components, see Figure 1.3. The role of each component is:

- *Formulation*: This component contains the abstract mathematical specification of how the available information should be transformed into the reputation metric. The formulation can be realized through an explicit equation, e.g., summation of the inputs, or through an algorithm that transforms the input values. This component plays a critical role in the safe operation of the mechanism and any weakness may allow malicious manipulation of the outcome.

- *Calculation*: This component calculates the mathematical formulation of the reputation metric given a set of constraints. As an example, if the formulation is the summation of the inputs, this component should decide whether to use all values or just a subset of them.

- *Dissemination*: The dissemination component allows the system participants to obtain the information that they need to calculate reputation scores. This component can be categorized according to multiple aspects, e.g., centralized vs. distributed, deterministic vs. non-deterministic.

Figure 1.3 presents this framework in detail; it is more complete than the framework by Marti et al. [66].

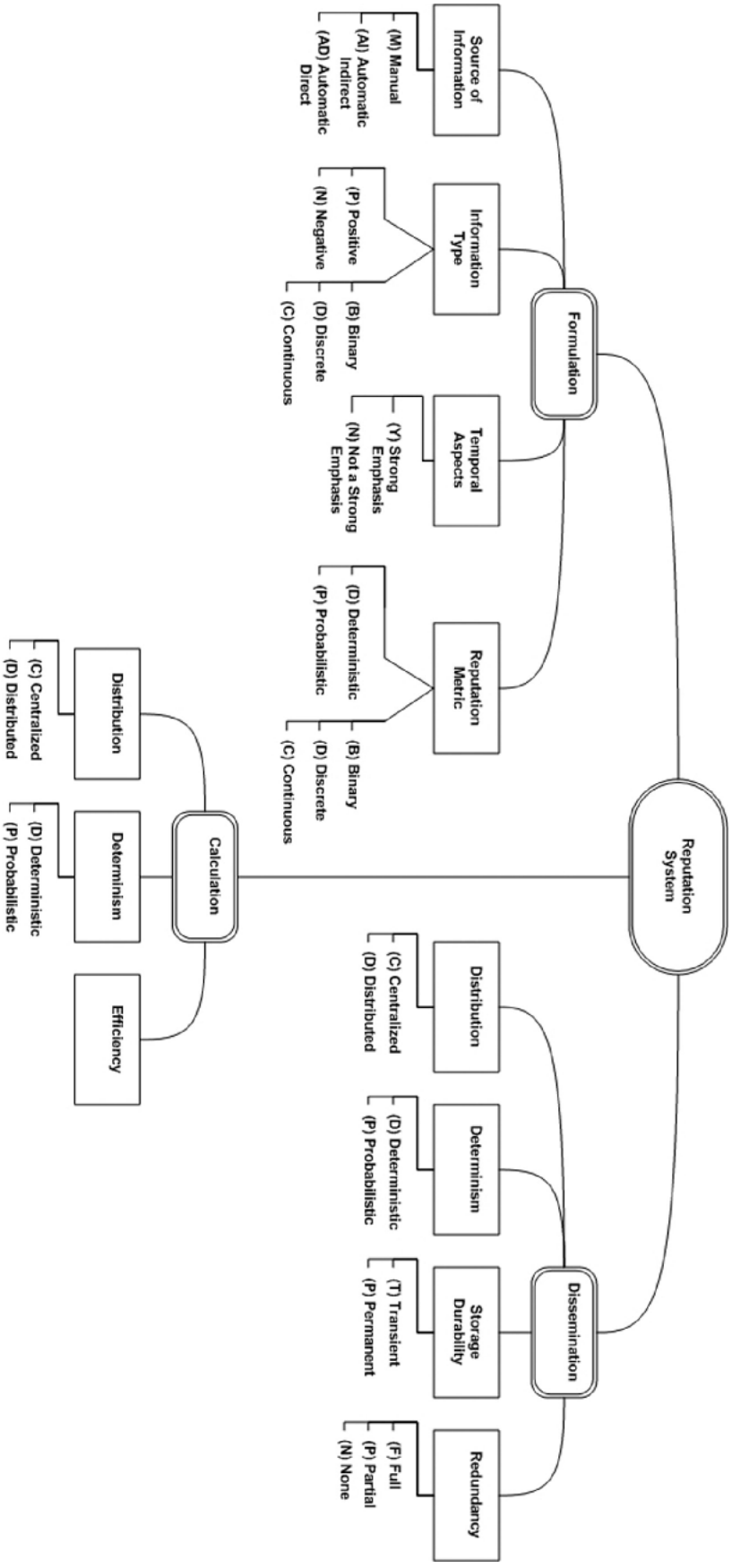


Figure 1.3: An analysis framework of reputation systems (taken from [44]).

1.1.4 Reputation Mechanisms in Distributed Systems

In this section, we provide a general overview of some of the main distributed reputations mechanisms. A more complete list is can be found in [44, 97].

- *PGrid*: This protocol was proposed by Aberer et al. [5] and it is one of the first distributed reputations mechanisms [5]. Even though the authors do not call their protocol PGrid, in this thesis to have a similar structure of writing we call it PGrid, like in [97]. In PGrid it is assumed that peers mainly are honest and only negative experiences are stated through *complaints*. The reputation mechanism collects, summarizes, and aggregates complaints about a peer; this process is done through a decentralized storage mechanism called P-Grid which was designed by the same author [4]. If the number of complaints about a peer is higher than the average in the system, the peer is not trusted any longer.

- *XRep*: This protocol was proposed by Damiani et al. [25], to be used in the Gnutella network. In this protocol each peer keeps track of the reputations of its neighbor peers and provides binary votes, positive or negative, about them. In this protocol both peers and files are assigned reputation values, and a combination of them are used to prevent the download of contaminated contents. Each peer has its own local repository and vote collecting is realized through broadcasting.

- *EigenTrust*: This protocol was proposed by Kamvar et al. [51], to prevent the spread of inauthentic contents in P2P file-sharing networks. In this protocol peers rate each other by the values of +1, 0 , or -1. The calculation of the global reputation values is based on these local opinions. To calculate the reputation values, the local opinions are put in a matrix and the left eigenvector of this matrix contains the global reputation values. In this protocol, a deterministic distributed framework (using multiple hosts for higher availability), which is based on Distributed Hash Tables, is responsible for calculating, storing, and distributing the reputation values.

- *TrustGuard*: Unlike the previous mechanisms, which are employed as a secondary mechanism to prevent misbehaving in the underlying applications, the TrustGuard mechanism [95] is designed to guard the reputation mechanism itself against malicious acts. It is designed to prevent three kinds of malicious acts, *oscillatory behaviors*, *fake transactions*, and *unfair ratings.* This mechanism uses a personalized oscillation guard, based on given opinions to similar partners in the past, to identify fake ratings. In this mechanism the rates are binary, and it uses a similar approach to PGrid for dissemination.

- *P2PRep*: This mechanism is designed by Aringhieri et al. [12], to deal with malicious actions in fully distributed and anonymous systems. The calculation is based

on *fuzzy* techniques, and two types of scores are considered for each peer, a *local* and a *global* score. Prior to using a network resource, a peer first queries for the locations of that resource, then polls for the reputations of the resource providers, and finally, using fuzzy techniques, calculates the reputations of the resource providers. In this mechanism only positive ratings are used (it is assumed that the participants are anonymous and negative values are pointless). The requests are broadcasted throughout the network but replies are uni-casted to the requester.

- *Credence*: Proposed by Walsh et al. [104], this protocol was designed to defend against file pollution in file-sharing networks. It was plugged into a customized version of the LimeWire BitTorrent client (in 2011 LimeWire was shutdown by court). This mechanism only ranks the contents, through employing individual votes by the peers; in order to overcome the problem of vote scarcity it uses a web-of-trust. Credence expects that honest peers vote for a content in a similar way and peers with similar voting patterns are trusted more than others. In this mechanism resources are rated as +1 (authentic) or -1 (inauthentic), it uses local databases to store the received votes from the neighbors, and it employs gossiping techniques for spreading information about unpopular contents.

1.2 BitTorent and Tribler

In this section we provide a brief overview of BitTorrent protocol and the Tribler project, and we present an overview of classes of incentive mechanisms in file-sharing networks. The Peer-to-peer technology has numerous applications in data storing and computing, but the most prominent of them is for *file-sharing*, which has been realized through the BitTorrent protocol [1]. Since the introduction of the first BitTorrent client in 2001, it has gained a lot of popularity for sharing video, music, and other bulky contents. In the BitTorent protocol, instead of publishing content on a server and downloading it directly, content is broken up into smaller pieces and each piece may be fetched from different peers. There is a centralized server called the *tracker*, which is responsible for tracking the available peers in a swarm, which is the set of online peers who are interested to upload or download a content. When a peer starts downloading a file, it contacts the tracker and gets a list of the addresses of peers who have some or all of the pieces of the file, then the process continues by directly asking and downloading pieces from the peers who already have them. The address(es) of the tracker(s) and metadata of the content are provided in a small file called the *.torrent* file of the content.

Since in BitTorrent there is no central content provider, sharing the available pieces of a content is crucial for the quality operation of the whole system. Looking from the local and selfish perspective of peers, if there is no control mechanism, then sharing content is

not in the interest of rational peers, and there should be a strong motivation for them to act cooperatively. Such selfish behavior is called *freeriding* [6], in which a freerider consumes much more resources than it provides in return. In BitTorrent, incentive mechanisms have been employed to withstand freeriding. The most widely used incentive mechanism is *tit-for-tat*, which was inspired by the outcome of an experiment run by Axelrod to determine the best strategy for iterative playing of the prisoner dilemma game [92]. Tit-for-tat is a pairwise mechanism, and a peer uploads to peers who reciprocate his uploads. Tit-for-tat is one of many forms of the incentive mechanisms that have been designed and proposed to be used in BitTorent. Table 1.2 gives an overview of different classes of incentive mechanisms along with their advantages and disadvantages. The content of this table is summarized from [67].

Tribler is a BitTorrent-based open source file-sharing client that is being developed at Delft University of Technology at Netherlands[6] with the aim of video-streaming and file-sharing, and it is used as a research vehicle for experimenting with and analyzing P2P algorithms by researchers at Delft University of Technology [75, 82, 86, 87]. This client has a number of distinctive features such as creating YouTube like channels, distributed search, and playback capabilities. The goal of the Tribler project is to build a fully distributed, anonymous, and user-friendly P2P content distribution framework. Tribler is an academic project and evolves continuously; Figure 1.4 shows a screenshot of version 6.0.0 of it. By entering relevant keywords in the top-left box one can search for content. Through the menus on the left side a user can browse the search results, the subscribed channels, the swarms he is currently participating in, and open the playback window while downloading the video. The central part of this screenshot shows the search results along with detailed information about each file.

The Tribler client is composed of a number of components, and two of the main components are BuddyCast [83] and BarterCast [68]. BuddyCast is the gossiping engine of Tribler and it collects and distributes all sorts of information that peers need to operate, e.g., content's metadata and the list of discovered peers. BarterCast is the incentive mechanism of Tribler. The focus of this thesis is to study and improve the robustness of the BarterCast mechanism in the aspects of accuracy, security, and scalability. In Section 1.3, we cover this mechanism in detail.

1.3 The BarterCast Reputation Mechanism

P2P file-sharing systems are characterized by large populations and high turnover. In such a setting, two interacting participants will often have no previous experience with each other, and will thus be unable to estimate each other's behavior in the system. De-

[6] http://www.tribler.org

Class	Description	Advantages	Disadvantages
Direct reciprocity	Solely based on the direct relationship between peers. Based on the short or long-term interaction history, a peer may collaborate or deny to give a service to the other peers (e.g., Tit-for-tat [22] and n-way exchange among rings of peers by Anagnostakis et al. [10]).	attack proof, easy to use, low overhead	highly subjective, inefficient in the long term, inefficient in large networks
Indirect reciprocity	Based on the concept of transitivity of contribution, which is the main pillar of trust networks [59] as well. If A contributes to B and B contributes to C, then C reciprocate A due to the contribution chain from A to C (e.g., one-hop reputations by Piatek et al. [81] and sybil-proof reputations by Landa et al. [55]).	high coverage in small-world networks [70]	difficulty in gathering reliable information, difficulty in instantaneous communication
Centralized reputation	A central server is responsible for collecting information about the peers' past behaviors. The server may calculate the final reputation score or just provide peers with the raw data to work on (e.g., all BitTorrent private communities like TVTorrents [3] and EZTVs [2]).	robustness, global, deterministic evaluations, easy to enforce policies	need a central server, users unreliability, unreliable feedbacks, low scalability
Decentralized reputation	The goal is to approximate the effect of the centralized reputation mechanism, but without having a central server. The designer should make a tradeoff between scalability and accuracy of the reputation scores. This class of mechanisms is the main focus of this thesis, and in Section 1.3 we elaborate more (e.g., EigenTrust [51] and BarterCast [68]).	highly scalable, decentralized	vulnerable to various attacks, subjective evaluations
Currency or credits	It is assumed that there is a kind of marketplace where through providing service, peers can earn spendable credit for future usage (e.g., [38, 43]).	credit validity, double spending, economic issues like inflation and deflation	sound theoretical background from economy

Table 1.2: Different classes of incentive mechanisms along with their advantages and disadvantages (the content of this table is summarized from [67]).

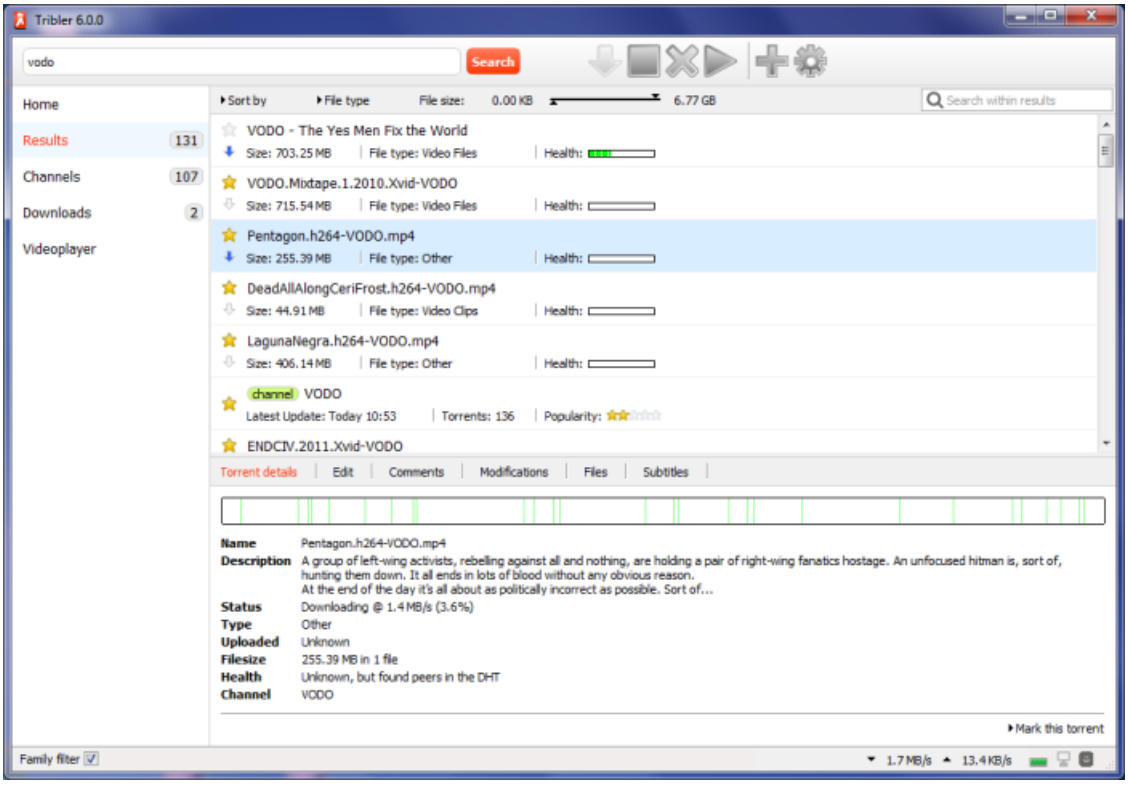


Figure 1.4: A screenshot of the main page of version 6.0.0 of Tribler.

spite such a complexity these systems can benefit from *reputation mechanisms* through which peers can evaluate the reputations of the system participants and are therefore able to identify good service providers. The BarterCast mechanism [68] is used by the Tribler Bittorent client to rank peers according to their upload and download behavior, and to prevent *free-riding*. In this mechanism, a peer whose upload is much higher than its download gets a high reputation, and other peers give a higher priority to it when selecting a bartering partner to exchange content with. In BarterCast, when two peers exchange content, they both log the cumulative amount of transferred data since the first data exchange and the identity of the corresponding peer in a *BarterCast record*. In BarterCast, to avoid misreporting about other peers data behaviors, each peer is only allowed to report about its own data exchange with other peers. This constraint decreases the dissemination rate of BarterCast records, and accordingly decreases the reputation accuracy. Moreover, the initial idea behind the design of BarterCast was to prevent *lazy free-riders* in the network [67], and the assumption was that all peers follow the protocol and there is no malicious peer in the network. Due to this assumption, the initial design of BarterCast does not contain security related components.

By exchanging BarterCast records, each peer creates its own current local view of the

upload and download activity in the system, and gradually expands its *partial graph*. The partial graph of peer i is $G_i = (V_i, E, \omega)$, where V_i is the set of peers that peer i has got informed about their activity through BarterCast records, and E is the set of weighted directed edges (u, v, w), with u, $v \in V_i$ and w the total amount of data transferred from u to v. Upon receipt of a BarterCast record (u, v, w), peer i either adds a new edge to its partial graph if it did not know u and/or v, or updates the weight of the edge $u \rightarrow v$ if it already exists in its partial graph.

In order to calculate the reputation of an arbitrary peer $j \in V_i$ at some time, peer i applies the maxflow algorithm [24] to its current partial graph to find the maximal flow from itself to j and vice versa. Maxflow is a classic algorithm in graph theory for finding the maximal flow from a source to a destination node in a weighted graph. When applying Maxflow to the partial graph, we interpret the weights of the edges, which represent amounts of data transferred, as flows. The original Maxflow algorithm by Ford-Fulkerson [24] tries all possible paths from the source to the destination, but in BarterCast only paths of length at most 2 are considered. Using the values $\Phi_2(x, y)$ as computed with the 2-hops Maxflow from x to y, the subjective reputation of peer j from peer i's point of view is calculated as:

$$R_i(j) = \frac{\arctan(\Phi_2(j, i) - \Phi_2(i, j))}{\pi/2}, \tag{1.1}$$

and so $R_i(j) \in (-1, +1)$. If the destination node j is more than two hops away from i, then its reputation is set to zero. The intuition behind this formula is twofold; first, with regard to the difference between incoming and outgoing flow it is a monotonic function that gives reputation values in the range of $(-1, +1)$. Second, the provided scaling by the $arctan$ function has the effect of giving higher value for the initial contribution of the starter peers (new comers). This ensures that a modest contribution of a new peer significantly affects its reputation. In the partial graph of Figure 1.5, peer i as the owner of the graph evaluates the reputation of peer j. In this graph, $\Phi_2(i, j) = 11$ and $\Phi_2(j, i) = 5$, and so $R_i(j) = -0.89$.

To compare BarterCast with other reputation mechanisms, we recall the general analysis framework from Figure 1.3 and cast the BarterCast reputation mechanism into this framework, see Figure 1.6.

1.4 Crawling Tribler

In this thesis, to collect the required dataset consisting of the BarterCast records of all (or at least, many) Tribler peers for analysis, we implemented a crawler and crawled the Tribler network. Except for some slight differences, the crawler works as an ordinary Tribler client. Discovery of the new peers is done through the BuddyCast protocol, which

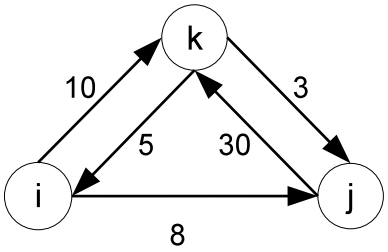


Figure 1.5: A partial graph in BarterCast.

is the gossiping engine of the Tribler client. When a new peer is discovered with this protocol, it is added to a list. The crawler hourly contacts all peers in this list and asks them for their latest BarterCast records by including the timestamp of the latest record it does have of each peer. Using the BarterCast records received by the crawler from each peer, we can reconstruct the partial graph of that peer in the same way the peer builds it.

The discovered peers have different ages, some of them having been installed and running for months and others just for a few days or even hours. So, when the crawler asks a peer for BarterCast records for the first time, it might receive very old records that are useless because they correspond to peers that were online in the past but no longer participate in the system. To mitigate this problem, when the crawler contacts a peer for the first time, it uses the start time of the crawl, that is, 00:00 hours on June 20, 2009, so that the discovered peers will only include BarterCast records fresher than the crawl start time in their replies.

Another problem in doing the crawling is the size of the reply messages. If a peer is asked for all its records at once, the reply message might be large and sending it may be problematic. To prevent this intrusive effect in the crawling, in each contact peers are only asked for 50 records that they have not sent already. Because of a potentially high churn rate, this limitation causes a side effect and for some of the peers that go offline the crawler is unable to fetch all their records. To have a reliable analysis, such incomplete views should be removed. Because in each contact a peer is limited to send at most 50 records, so with a high probability, having a multiple of 50 records from a peer means that it has not sent all its records. As a consequence, to filter out incomplete views, all views of the size of a multiple of 50 are removed.

To be able to sort the collected records and to account for the time difference with remote peers, the crawler asks peers to send for their local time as well. When the crawler receives such information, it logs the remote peer's time and its own local time. Using these two local times and the timestamp of the record (available in the record payload) the collected records can be sorted. If t_p and t_c denote the local time of the remote peer and the crawler, respectively, and t_r is the record timestamp, then the relative record occurrence

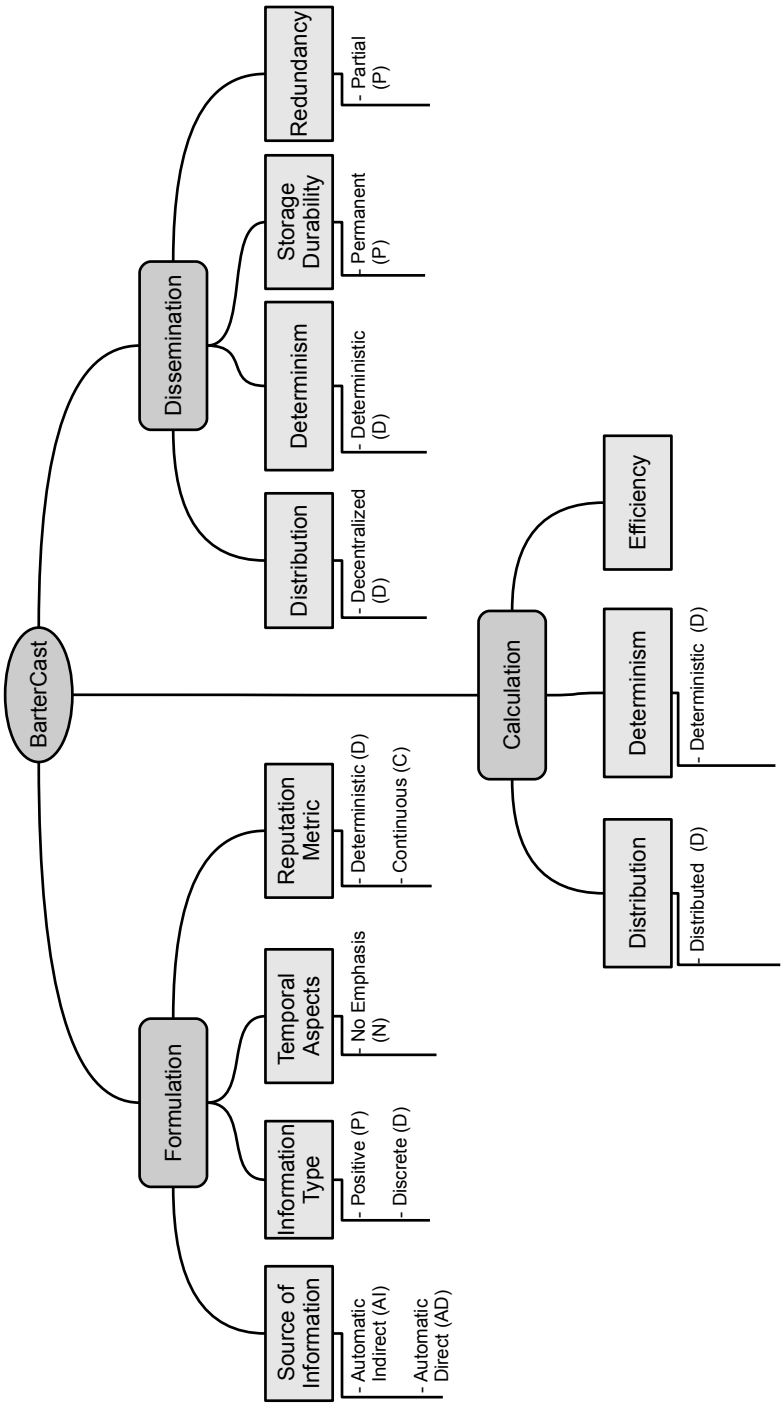


Figure 1.6: Casting the BarterCast mechanism into the reputation framework of Hoffman [44].

time is:

$$t_c - t_p + t_r \tag{1.2}$$

This relative time is used in the experiments to sort the BarterCast records.

In this thesis, the experiments and evaluations are based on the crawled data from different periods of time and with different volumes. In order to make a unified reference across the whole thesis, we assign a name tag to each dataset. Later on, we refer to these name tags when we explain the experiment processes. Here is the list of the datasets:

- **Dataset_1**: This dataset contains 547,761 BarterCast records from 2,675 different peers, collected from June 2009 until September 2009.
- **Dataset_2**: This dataset contains 2,837,422 BarterCast records from 11,176 different peers, collected from September 2010 until December 2010.
- **Dataset_3**: This dataset contains 37,072,073 BarterCast records from 77,289 different peers, collected from September 2010 until September 2012.

Notice that *Dataset_1* and *Dataset_2* are distinct but *Dataset_3* contains *Dataset_2*. Depending on the experiment, we provide additional information about each dataset in the relevant sections.

1.5 Problem Statement

Designing a distributed reputation mechanism comes with many challenges, and such a mechanism has requirements such as accuracy, security, and scalability. So far, many proposed mechanisms have never gone beyond a paper design into real operational mode [44]. Even though BarterCast is a deployed mechanism, its original design does not sufficiently fulfill these requirements mentioned above. In this thesis, we employ the deployed BarterCast mechanism as the mainline mechanism, and we design, implement, and analyze additions and modifications of it in order to make it robust with respect to accuracy, security, and scalability. In addition, through employing the data collected from the Tribler network, we do a thorough analysis of the BarterCast graph from the perspective of network science and gain insights for further improving this mechanism. In particular, we address in this thesis the following four research questions.

What is the accuracy and the coverage of BarterCast and how we can improve them? Two relevant and important questions associated with a reputation mechanism like BarterCast are what the accuracy of the reputation values is, and to what fraction of participants the mechanism can assign a meaningful reputation value (the coverage). In the end, the reputation values should reflect the true behavior of the peers, and without accurate reputation evaluations, the whole mechanism is useless. In this regard, we need

to define viable metrics for accuracy and coverage, and if the values for accuracy and coverage are not satisfactory we need to find ways to improve the rate of accuracy and coverage.

How can we make BarterCast resilient to different types of attacks? Due to the possibility of cheap identity creation, openness, and its decentralized nature, BarterCast is vulnerable to attacks like sybil attacks, whitewashing, and miss-reporting. In a sybil attack, through creating multiple fake identities an attacker benefits from the system without contributing. In whitewashing, a malicious user with a low reputation can get rid of its identity by creating a new one. Furthermore, through miss-reporting, an adversary can subvert the reputation values of others. These attacks hinder the natural operation of the system and they should be dealt with.

How we can make BarterCast scalable without compromising the accuracy? In online reputation mechanisms as BarterCast, providing the system participants (peers) with the appropriate information on previous interactions is crucial for accurate reputation evaluations. A naive solution is to provide all peers with all information, regardless of whether they need it or not, which may be very costly and not scalable. In order to have accurate reputation evaluations and at the same time to be able to scale to large numbers of peers, an elegant and scalable information dissemination solution is required.

What is the structure of the BarterCast network and what we can learn from it? In BarterCast, through collecting BarterCast records every peer builds a local graph which represents its view of the network. Besides, through combining all local graphs from all peers, we can build a single global graph which represents the whole network. Studying the global BarterCast graph from the perspective of network science can reveal many operational and performance aspects of it, and can help us to improve BarterCast in proper and effective ways.

1.6 Research Contributions and Thesis Outline

The contributions of this thesis are as follows.

Improving reputation accuracy and coverage (Chapter 2) In BarterCast, a peer calculates the reputations of other peers by applying the Maxflow algorithm to its partial graph; for efficiency reasons, only paths of at most two hops are considered. We identify and assess three potential modifications to BarterCast to improve its accuracy and coverage (fraction of peers for which a meaningful reputation can be computed). First, a peer executes Maxflow from the perspective of the node with the highest betweenness centrality instead of itself. Secondly, we assume a gossiping protocol that gives each peer complete information about upload and download activities in the system, and third, we lift the path length restriction in the Maxflow algorithm. To assess the impact of these modifications, we crawl the Tribler network and collect the upload and download actions

of the peers for three months. We apply BarterCast with and without these modifications on the collected data and measure the accuracy and coverage. This chapter is largely based on our work published in the *IEEE International Conference on Peer-to-Peer Computing 2010* [27].

Strengthening BarterCast against sybil attacks, whitewashing, and miss-reporting (Chapter 3) We study the opportunities for sybil attacks in BarterCast and we devise a method for making BarterCast sybil attack resilient, which we incorporate into BarterCast to obtain a protocol called SybilRes. In SybilRes, after an upload action, the uploading peer discounts the weights of the edges on the paths from the downloader to itself. As a consequence, due to the way reputations are computed, the reputation of a peer performing a sybil attack decreases fast. To mitigate the negative impact of edge weight discounting on the reputations of honest peers, after a download action, the downloading peer increases the weights of the edges on the paths from the uploader to itself. We demonstrate that SybilRes is effective in practice by means of trace-driven simulations using data collected from the Tribler network. Besides, due to modifications in the dissemination and formulation parts of the mechanism, SybilRes is robust against whitewashing and miss-reporting behaviors as well. This chapter is largely based on our work published in the *International Conference on Distributed Computing Systems 2012* [28].

Making BarterCast scalable (Chapter 4) In online reputation mechanisms, providing the system participants (peers) with the appropriate information on previous interactions is crucial for accurate reputation evaluations. A naive way of doing so is to provide all peers with all information, regardless of whether they need it or not, which may be very costly and not scalable. We propose a similarity-based approach, named SimilDis, for targeted dissemination of information in BarterCast. We propose two methods to derive peer similarity in the partial graph of a peer. The first method is based on incrementally maintaining a directed acyclic graph, and the second method is based on performing multiple nonuniform random walks in the partial graph. In both methods, each peer maintains a list of peers most similar to itself, and gives higher priority to them when disseminating information. We evaluate the accuracy and the cost of these methods using trace-driven simulations based on traces from the Tribler P2P file-sharing network. This chapter is largely based on our work published in the *International ACM Workshop on Scalable Trusted Computing 2012* [29].

Insights into the BarterCast network from the network science perspective (Chapter 5) In this chapter, we study the BarterCast mechanism from the perspective of network science and we provide a detailed analysis, which includes such network topology measures as the degree distribution, the node interconnectivity, the clustering coefficient, the community structure, and distance measures. Besides, we study the geographical spread and content sharing behavior of the system participants and correlate the results with their connectivity in the network. We interpret each evaluated measure in the

scope of reputation and file-sharing mechanisms and propose relevant implications and prospective applications for future designs. The global graph of BarterCast we study is based on data that we have collected during two years of crawling the Tribler file-sharing network. This chapter is largely based on our work published in the *International IFIP Networking Conference 2013* [30].

Conclusions (Chapter 6) In this chapter, we summarize our important findings in this thesis and provide suggestions for future study.

Chapter 2

Improving Reputation Accuracy and Coverage

The effectiveness of a reputation mechanism can be assessed with its *accuracy* and *coverage*. The accuracy measures to what extent the computed reputation values reflect the real behaviors of the system participants, and the coverage is the fraction of the system participants for which the mechanism is able to compute meaningful reputation values. Inaccurate or partial reputation evaluation leads to misjudgment, poor behavior, and finally, system degradation.

In this chapter we propose three modifications to the BarterCast reputation mechanism, and we evaluate the accuracy and the coverage of the original BarterCast reputation mechanism and of all combinations of these three modifications. First, rather than have each peer execute the Maxflow algorithm to compute reputations from its own perspective, we make each peer do so from the perspective of the node with the highest *betweenness centrality* [37] in its partial graph. The second modification consists in using a gossiping protocol that fully disseminates the BarterCast records in the whole system rather than limiting the exchange of these records to one hop. In the third modification we increase the maximal path length in the Maxflow algorithm to 4 or 6 instead of 2 as in the original BarterCast. In order to evaluate the original BarterCast reputation mechanism and our three modifications, we use *Dataset_1*, see Section 1.4. After filtering out the incomplete views from this dataset, we ended up with 416,061 records collected from 1,442 peers. This means that although 46% of the views are incomplete, they contain only 24% of the collected records. All the subsequent processing and analysis in this chapter is based only on complete views. From the records obtained from each peer, we emulate its reputation computations by reconstructing its *subjective view*, represented by the partial graph of the peer (in this thesis the terms partial graph and subjective view convey the same meaning). We then used this graph to execute the Maxflow algorithm with and without modifications.

The main contributions of this chapter are as follows:

1. We define appropriate metrics to quantitatively measure the accuracy and coverage of the BarterCast mechanism (Section 2.2).

2. We propose three modifications to the BarterCast mechanism, and using a set of data collected from the Tribler network for each combination of modifications we measure the change in the accuracy and coverage (Section 2.3).

3. We perform a statistical analysis to evaluate the significance of the improvements (Section 2.4).

2.1 Related Work

After the first release of BarterCast Seuken et al. [93] proposed an improvement to make it more resilient against misreporting attacks. Their solution is based on ignoring some of the feedback reports. Also, this solution could cut down the severity of the attack, but on the other hand it increases the feedback sparsity. Xiong et al. [107] show that the feedback sparsity is an issue in large distributed systems, and that a lack of enough feedback can lead to lower accuracy and coverage.

Besides BarterCast, several other distributed reputation mechanisms have been proposed for P2P systems, but they use different methods to calculate reputation values. EigenTrust [51] is based on summation of direct observations and indirect data and uses centralized *matrix operations* to compute the left Eigen vector. The CORE system [18] uses *arithmetic weighted averaging* on historical data to calculate reputation values.

2.2 Defining the Accuracy and Coverage Metrics

As the term *accuracy* indicates, it is a measure of how close an estimated reputation value is to an "objective" or real value. In a distributed mechanism like BarterCast, depending on how the feedback records are disseminated, peers may have different opinions about the reputation of a peer at the same time. Each peer also at each point in time has an *objective reputation* value, O_j, that is calculable only if the evaluator peer has a global view of the activity of all peers. In our case, only the crawler has such a view and using the collected data we can calculate the objective reputations. If U_j and D_j are the total upload and download by peer j, then its objective reputation is

$$O_j = \frac{\arctan(U_j - D_j)}{\pi/2}. \tag{2.1}$$

Using the objective and subjective reputations, the estimation error is defined as the absolute value of the difference between the subjective and objective values:

$$e(i,j) = abs(S_{ij} - O_j). \tag{2.2}$$

Higher estimation errors mean lower accuracy and vice versa.

Coverage is another important metric that expresses how well a node is located and can reach other nodes in the graph. Denoting by $F_h(.,.)$ the maximum flow computed with the Maxflow algorithm using all paths of length less than or equal to h, in the partial graph G the h-hop coverage of node i is defined as

$$c_G(i,h) = |\{u|F_h(i,u) > 0 \; or \; F_h(u,i) > 0\}|. \tag{2.3}$$

So the coverage of node i in a graph is the number of nodes at a distance at most h from node i with non-zero maximum flow to or from i. Dividing the coverage by the number of nodes normalizes it into the interval of $[0,1]$ and makes it possible to compare this metric in graphs of different size.

2.3 Problem Statement and Proposed Modifications

An analysis of the collected data set shows that the accuracy and the coverage with the current BarterCast mechanism are low and need to be improved. The mean of the estimation error is 0.664, which is the same as the average difference between two random values in the interval of possible reputation values, $(-1,+1)$. This means that a random guess for the subjective reputation value has the same precision as using the BarterCast mechanism. Similarly, the coverage of the BarterCast mechanism is very low at 0.032. In order to remedy this situation, we propose the following three modifications to the BarterCast mechanism.

2.3.1 Modification 1: Using Betweenness Centrality

Betweenness centrality has been introduced by Freeman [37] as a measure of the number of shortest paths passing through a node. In a graph $G = (V,E)$, if δ_{st} is the number of shortest paths between two arbitrary nodes s, t of G, and $\delta_{st}(v)$ is the number of these paths that pass through node v, then the betweenness centrality of node v is $\beta(v) = \sum_{s \neq v \neq t} \frac{\delta_{st}(v)}{\delta_{st}}$. A higher betweenness centrality means a higher participation of the node in connecting other nodes, and also a higher flow that passes through it. Another feature of this measure is that in contrast to connectivity (the sum of in and out degrees of a node), which is a local quantity, betweenness centrality is a quantity across the whole

graph; nodes with many connections may have a low betweenness centrality and vice versa [13]. Betweenness centrality has been used in the analysis of various topics, like transportation, social networks, and biological networks, but to the best of our knowledge it has not been used in reputation systems.

In the original BarterCast mechanism, a peer i as the owner of the partial graph G_i, in evaluating the reputation of peer j, runs the Maxflow algorithm to compute the maximum flow from itself to j and from j to itself. In the proposed modification, first node i finds the node with the highest betweenness centrality in G_i, and then replaces itself with that node in the Maxflow execution. By this change, the evaluator peer benefits from the centrality feature of the central node and uses the collected data in a better way.

2.3.2 Modification 2: Using Full Gossip

The second modification is obtained by changing the way BarterCast records are disseminated. In the original version, peers only use *1-hop* message passing and they are not allowed to forward the received records. Peers only report their own download and upload activities to the peers that are discovered by the BuddyCast protocol. This method limits the effect of misreporting but it is not efficient in spreading the BarterCast records. Specially if a peer goes offline, its upload and download activity are not disseminated, and when it comes online again, very few peers know about its activities. In this modification, instead of using 1-hop message passing, we assume that there is a *full gossiping* protocol that spreads records without the hop limitation, so that in principle all online peers eventually receive all propagated records.

2.3.3 Modification 3: Lifting the Maxflow Hop-Count Restriction

In the third modification we lift the restriction of 2 on the hop count in the Maxflow algorithm and increase it to 4 or 6 hops. With this change, more nodes are involved in the Maxflow algorithm and the chance of reaching a node, and so increasing the coverage, is increased.

2.4 Experimental Setup and Results

In this section we first explain our experimental set-up for assessing the accuracy and coverage of the original BarterCast mechanism and of the proposed modifications. In short, we emulate the creation of partial graphs using the BarterCast records received by the crawler, and we emulate their computation of the reputation values of those peers to which they appear to have uploaded data. Then we present the experimental results and compare the effect of the proposed modifications on accuracy and coverage. At the end

we do some statistical tests and determine whether the improvement level in accuracy is statistically significant or not.

2.4.1 Emulation of Full Gossiping

The subjective views collected by the crawler are only based on the standard 1-hop dissemination of BarterCast records. In order to evaluate the modification obtained with full-gossiping mode, we create artificial subjective views from the 1-hop subjective views. The full-gossip view at a certain point in time is the same for all peers, and is built from *all* BarterCast records received from all peers with a timestamp lower than that time. So here we assume perfect full gossip in that all BarterCast records with a certain timestamp have been received by all peers at the time indicated by the timestamp. It should be noted that when using full gossiping, the reputation computations may still yield different results when Maxflow is executed from the perspective of the local peer, but will give the same results when the local peer is replaced by the node with the highest betweenness centrality.

2.4.2 Experiment Design

In a large scale system like the one that the BarterCast mechanism is designed for, it is not required that every peer is able to evaluate the reputation of every other peer; peers just need to evaluate the reputations of the peers that they *encounter*. In the file-sharing system that we are studying, encountering means that a peer d contacts a peer s and asks s for some content, and peer s before responding to the request of d evaluates its reputation. When such an event happens, we say that s encounters d. In our experiment we try to emulate the encountering events and only do a reputation evaluation when processing a BarterCast record in order to build up a subjective view that indicates such an event.

Another point we consider in the experiment is that in a decentralized reputation mechanism like BarterCast, we cannot expect that immediately after joining the system, a peer is able to give a good evaluation of the reputations of the peers it encounters. The newly joining peers should be allowed to collect information during a *training* phase from already existing peers and grow their subjective views before starting the evaluation of reputations of others during the *testing* phase.

The starting point of our experiment consists of the time-ordered sequences of BarterCast records the crawler has received from all peers, which we can use to build their subjective views. We define the *availability interval* of a peer as the interval between the timestamps of the first and last record in the sequence of BarterCast records the crawler has received from it. In our experiment, every peer goes through two phases, a *training* phase and a *testing* phase. In the training phase of a peer, we reconstruct its subjective

view starting from the empty view by adding in sequence the BarterCast records of the first 80% of its availability interval. Only in the testing phase, peers evaluate the reputations of the peers they encounter. The testing phase is like the training phase, except that before adding an edge to its subjective view, a peer checks to see whether the conditions for encountering are satisfied. By checking these conditions we can detect the occurrence of an encountering event between two peers, and if required run the reputation evaluation process.

In the discussion below, we assume that the format of a BarterCast record is $[s, d, D, U, t]$, with t a relative timestamp and with D (U) the amount of data downloaded (uploaded) by peer s from (to) peer d until time t. When in the testing phase record $[s, d, D, U, t]$ of the subjective view G_i of peer i is processed, it is determined whether the reputation of peer d should be evaluated by peer i. This is only done if the following two conditions are satisfied:

1. $i = s$: The peer which uploads is also the owner of the partial graph, and it is the peer that should do the reputation evaluation.

2. $U > 0$: The record indicates an actual data upload.

In other words, if a record passes the above conditions, the reputation of the peer that does the downloading is evaluated by the peer that does the uploading, and the latter coincides with the peer for which the BarterCast record is processed (s evaluates d, and i and s coincide). The meaning of the two conditions on the BarterCast records is that apparently, peer i has done an upload to d, and when the BarterCast reputation mechanism would have been in use, this would have been the time that peer i should have invoked it.

When processing BarterCast records in the testing phase, the peers whose reputations should be evaluated by other peers, are categorized as *newcomers* or *existing peers*. The newcomers are those peers that have not done any download or upload activity in the past (before the relative time of the record that is processed), but the existing peers have done so and the crawler knows about their activity. To detect newcomers, let $[s, d, D, U, t]$ be the record that is being processed, and assume it has passed the above encountering checks, so peer s should evaluate d. To determine whether peer d is a newcomer or not, we consider all current subjective views, and if in any of these there exists a record $[s', d', D', U', t']$ with $s' = d$ or $d' = d$, $t' < t$, and $U' > 0$ or $D' > 0$, then d has been active in the past and is not a newcomer; otherwise it is.

Reputation evaluation for newcomers is meaningless, as without any previous information about a peer, there is no reputation to be calculated. So, in the results of the accuracy and the coverage below, only the existing nodes are considered and the newcomers are excluded. In our experiment, in which the training and testing phases take 80% and 20% of the availability intervals of the peers, respectively, the numbers of newcomers and existing peers are 140 and 123, respectively.

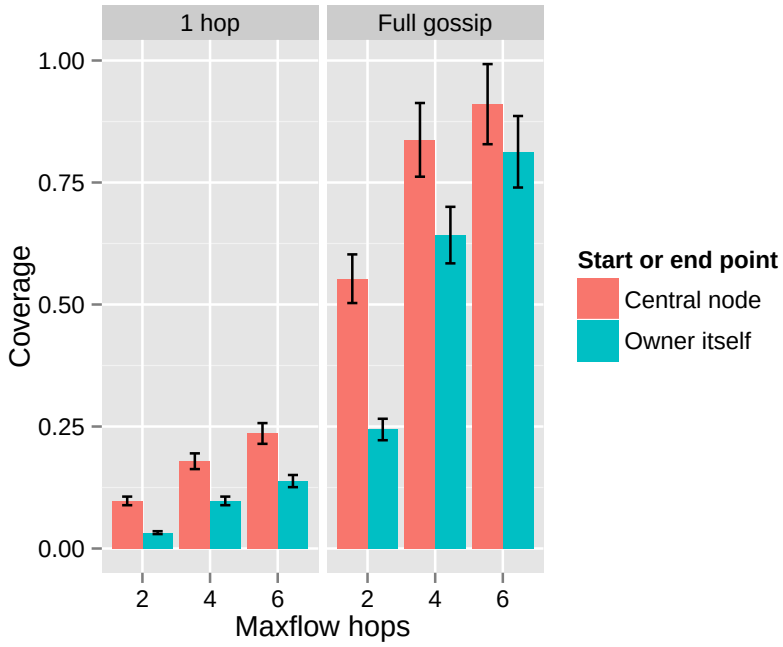


Figure 2.1: The coverage of the BarterCast mechanism for different parameter settings. (Error bars show the standard error of the mean.)

The explained experiment is run for each view one-by-one and in all combinations of the proposed modifications. For each combination, we assess the values of the accuracy and the coverage, and when all views are processed, the results are aggregated to compare the performance of the different combinations.

2.4.3 Coverage

The barchart in Figure 2.1 shows the number of covered peers for all combinations of the proposed modifications. It is expected that only existing peers can be covered by the evaluator peers, and so in all of our experiments the maximum possible value for the coverage is 123 (the number of existing peers). The left half of the graph shows the cases in which the central node is used in the Maxflow algorithm and the right half the view owner itself. As the graph shows, full gossiping boosts the coverage dramatically. Using the central node increases the coverage too, specially in 2-hops Maxflow, but for a larger number of hops, it is less effective. Increasing the number of hops has more or less the same influence as using the central node, and in both dissemination methods the biggest improvement is seen when we go from 2 to 4 hops.

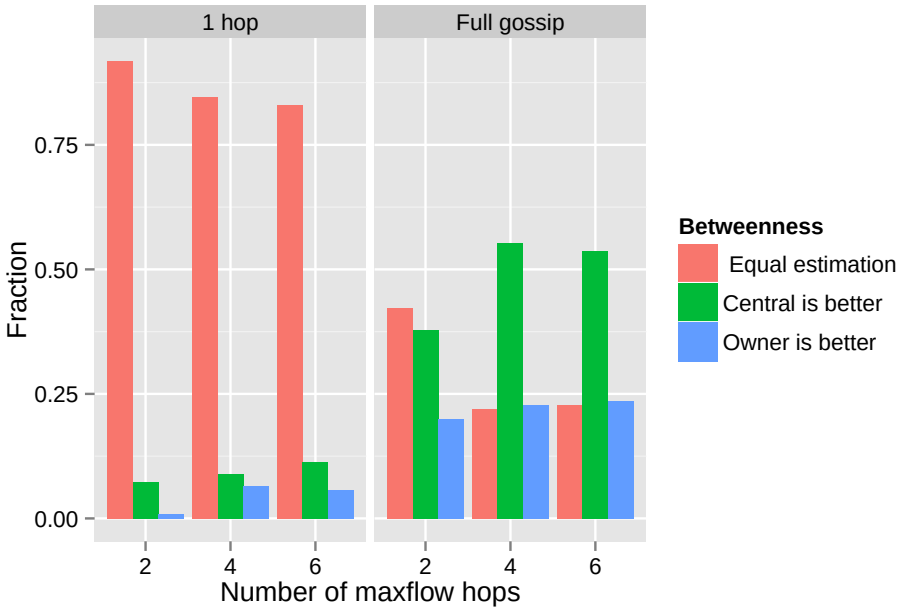


Figure 2.2: Comparing the accuracy of the central node against the view owner in the BarterCast mechanism.

2.4.4 Accuracy

In Figure 2.2 we show the fractions of nodes for which either the central node in the partial graph or the peer itself provides a better estimation of the reputation value for different numbers of hops in Maxflow and in both 1-hop and full-gossip dissemination. In practice, equal reputation estimation means that both reputation values are equal to 0. As the left hand of the figure (1-hop dissemination) shows, in more than 80% of the cases the central node and the view owner give the same estimation. When we move to full gossiping, the situation changes considerably, and using the central node gives better estimations. Especially with 4 and 6 hops, the number of cases that the central node is better is twice the number of cases that the view owner is better.

Figure 2.2 only shows which combination of the methods is better, but it does not tell how much they are better. To have a grasp of the improvement rate we compare the mean and the median of estimation errors. Figure 2.3 shows the mean and its standard error for all combinations of the modifications. As the graph shows, only changing the number of hops or using the central node does not improve much, and using the full gossiping is needed. Then, using both the central node and a higher number of hops decrease the estimation error, and when all modifications are applied, the mean of the errors becomes 0.404.

As the mean value may be biased by a small number of very high or very low values,

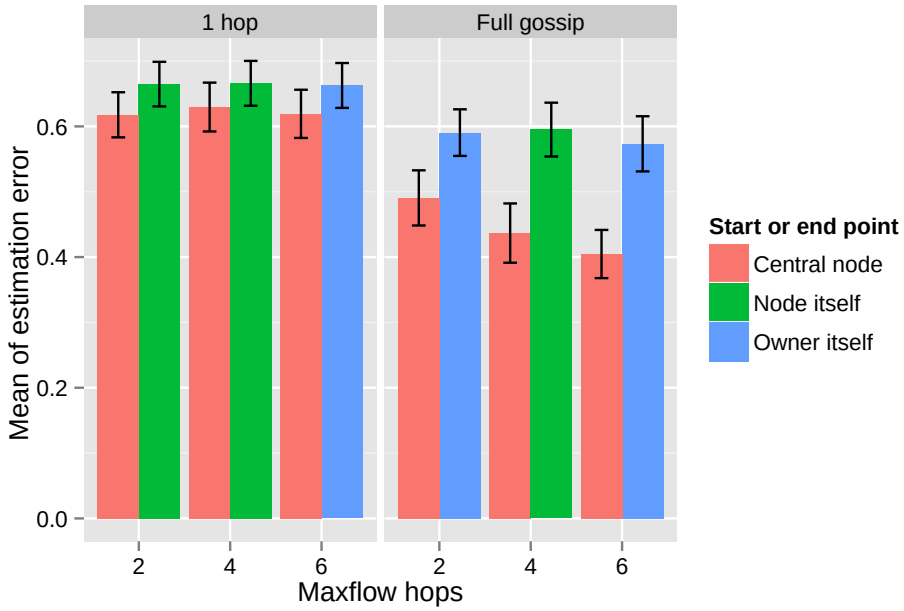


Figure 2.3: The mean of the estimation error in the BarterCast mechanism. (Error bars show the standard error of mean.)

we compare also the median of estimation errors in different scenarios. Figure 2.4 shows the median of the error in various situations. As it is seen, in 1-hop dissemination using the central node or higher Maxflow hops only have a little influence on the decrease of the median and using the full gossiping is required. In full gossiping mode using the central node is very effective and when it is combined with higher Maxflow hops the median is decreased by a factor of 10 and in the ultimate case it is pushed to below 0.09.

2.4.5 Statistical Analysis

The graphs with mean and median give an indication of the difference of these statistics in various scenarios, but they do not assess the significance of the differences. Figure 2.5 shows the density plots of the estimation errors in full gossiping and 6 hops Maxflow for both central node and view owner. As it is seen from these plots the estimation error values do not follow a Gaussian distribution. Because of the non-Gaussian property of the distributions, using a non-parametric test is preferred and in our analysis we use the Wilcoxon signed-rank test to compare the change of estimation errors. In this test, the null hypothesis is the equality of the medians and we test whether it is rejected or not. In each test we change one system parameter related to the three modifications of BarterCast discussed in this chapter, and test whether the change in error values is significant. Tables

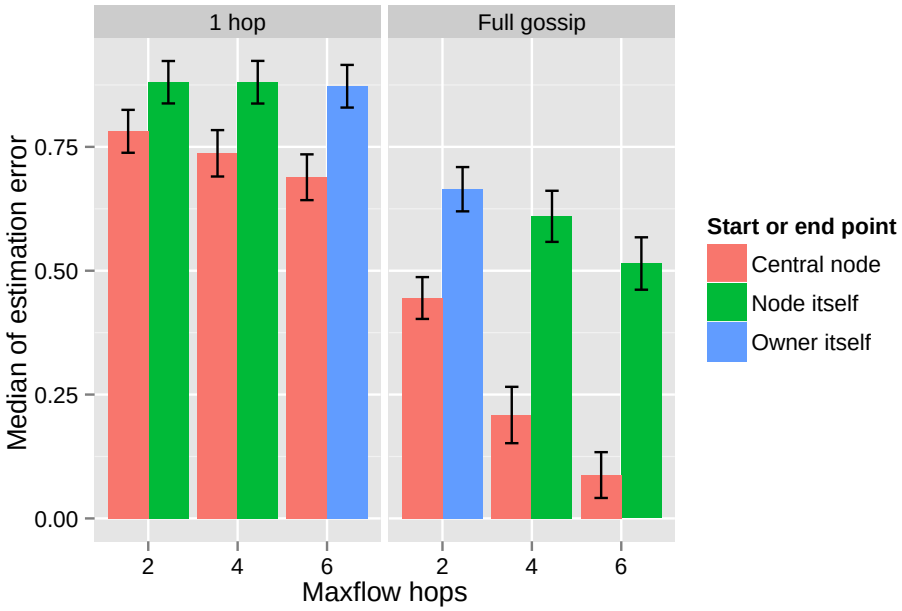


Figure 2.4: The median of the estimation error in the BarterCast mechanism. (Error bars show the standard error of median.)

2.1 to 2.4 contain the p-value and the test results. In the first table we change the dissemination method and compare error values in various combinations of the other parameters (first and second column). The third column shows the p-value, and the fourth column says whether with significant level of 95% the equality of the medians is rejected or not.

Table 2.2 is the same as Table 2.1 except that in this table the central node is compared with the view owner and as the last column shows, using the central node is only effective in full gossiping. Tables 2.3 and 2.4 contain the test results of changing the number of hops in 1-hop and full gossiping dissemination, respectively. In Table 2.3 we do not see any rejection, and increasing the number of hops in Maxflow does not help in 1-hop dissemination. In Table 2.4, which tests the effect of increasing the Maxflow hop number in full gossiping mode, we just see a single rejection in the last row, meaning that using more hops is useful only if the central node and full gossiping are applied.

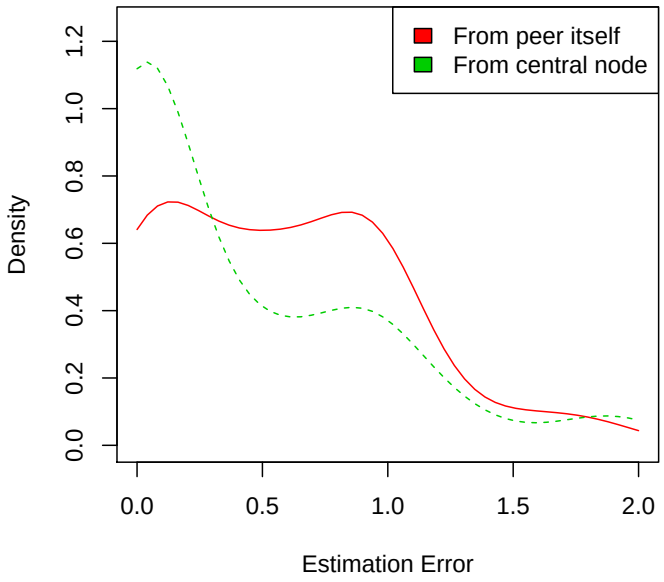


Figure 2.5: Density plot of the error values with full gossiping and 6-hop Maxflow

Table 2.1: Wilcoxon test for the difference in estimation errors for 1-hop dissemination and full gossiping.

number of hops in Maxflow	Maxflow start point	p-value	rejected? (95% sig. level)
2	view owner	0.0898	No
4	view owner	0.0585	No
6	view owner	0.0126	Yes
2	central node	0.0015	Yes
4	central node	1.617e-05	Yes
6	central node	2.031e-07	Yes

Table 2.2: Wilcoxon test for the difference in estimation errors for using central node and view owner.

number of hops in Maxflow	dissemination	p-value	rejected? (95% sig. level)
2	1-hop	0.3032	No
4	1-hop	0.3309	No
6	1-hop	0.2863	No
2	full gossip	0.0160	Yes
4	full gossip	0.0008	Yes
6	full gossip	0.0003	Yes

Table 2.3: Wilcoxon test for the difference in estimation errors using 1-hop dissemination and changing number of hops in Maxflow algorithm.

change in number of hops in Maxflow	Maxflow start point	p-value	rejected? (95% sig. level)
2 to 4	view owner	0.9686	No
4 to 6	view owner	0.8719	No
2 to 6	view owner	0.8318	No
2 to 4	central node	0.9721	No
4 to 6	central node	0.8297	No
2 to 6	central node	0.7936	No

Table 2.4: Wilcoxon test for the difference in estimation errors using full gossip dissemination and changing number of hops in Maxflow algorithm.

change in number of hops in Maxflow	Maxflow start point	p-value	rejected? (95% sig. level)
2 to 4	view owner	0.7633	No
4 to 6	view owner	0.4640	No
2 to 6	view owner	0.2862	No
2 to 4	central node	0.1700	No
4 to 6	central node	0.2440	No
2 to 6	central node	0.0117	Yes

2.4.6 Maxflow Runtime

In the BarterCast mechanism one of the reasons for limiting the Maxflow hops to 2 is to decrease the computation overhead. To analyze the overhead of increasing hop numbers, we run the Maxflow algorithm in 2 and 6 hops and compare the runtimes. Like the experiment for accuracy and coverage, also in this experiment the destination node in the

Maxflow algorithm is an existing node which is evaluated by the source node (a view owner). In the complete graph (a graph combined of partial graphs of all peers) for each pair of source and destination peers, we run the Maxflow algorithm 100 times and average the runtime. Figure 2.6 shows the experiment result, sorted by the runtime in 2 hops and compares it with the runtime in 6 hops. As it is seen, the runtimes are bounded between 200 and 450 ms and from the performance point of view their difference is negligible.

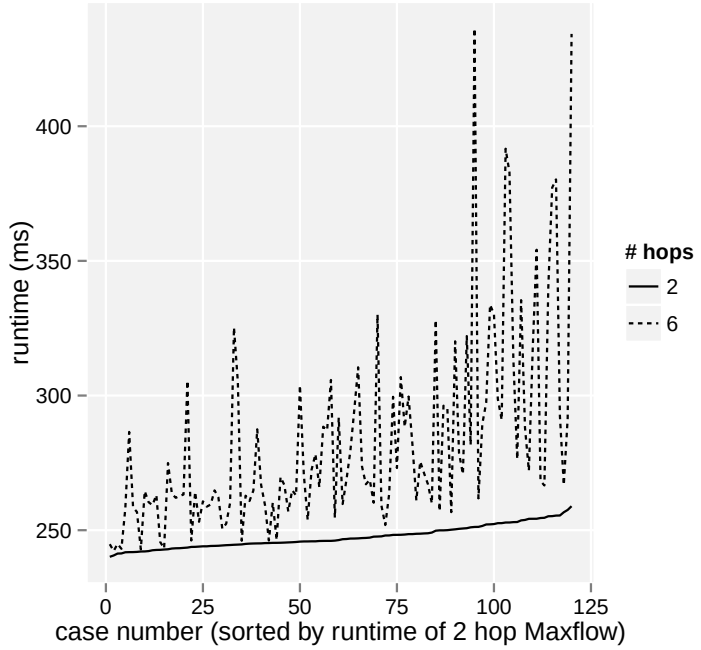


Figure 2.6: Runtime of the Maxflow algorithm with 2 and 6 hops.

2.5 Conclusion

In this chapter we have performed an empirical analysis of the accuracy and the coverage of the BarterCast reputation mechanism and proposed three applicable modifications to improve these values: using betweenness centrality, using full gossip instead of 1-hop dissemination of BarterCast records, and increasing the path length in the Maxflow algorithm. Our results show that using full gossip leads to the large improvement according to our metrics. Moreover, the other two modifications provide significant improvements, but only if combined with full gossip.

After understanding the improvements leveraged by changes in the design of BarterCast, some open questions related to the proposed improvements need now to be ad-

dressed. Full gossiping increases the dissemination performance, but it is more vulnerable to misreporting attacks, and indirect reports should be treated carefully. A possible solution for this problem could be to put the indirect reports received in a secondary view and to add them to the primary view, used for reputation evaluation, if they are received from more than a certain number of peers or by highly reputed peers. Another method to address the misreporting attack is the use of double signatures. In this solution, before disseminating a record, the content sender and receiver sign the associated BarterCast record using their private keys. Using this technique no other peer can eavesdrop and change the record. We address this problem in Chapter 3.

The second problem related to the proposed solutions is the performance of finding the node with the highest betweenness centrality in the partial graphs. The complexity for this calculation is considerable, of order $O(nm)$ for unweighted and $O(mn + n^2 \log n)$ for weighted graphs [15], where n and m are the number of nodes and edges, respectively. Our results are based on the unweighted version, and even with $O(nm)$ complexity the usage patterns observed so far hint at its practical feasibility. We deal with this problem in Chapters 4 and 5.

Chapter 3

Making BarterCast Resilient to Sybil Attacks

A crucial property of reputation mechanisms for P2P systems is their robustness against malicious actions, e.g., *sybil attacks*, which in open distributed environments are likely to happen. In a sybil attack the attacker creates many identities to subvert the systems in which he is attacking. In this chapter, we present a new version of BarterCast called SybilRes that we show to be resilient against sybil attacks.

In BarterCast, peers are identified by a permanent cryptographic id that is generated during Tribler installation, as explained in Section 1.3. As generating such an identity is cheap, it is possible for a modified client to generate many identities on behalf of a single user, and to perform a sybil attack in the following way. First, the attacker obtains a positive reputation from a potential victim peer's point of view, for instance, by uploading a large amount of data to that peer. Then she creates a number of *sybils*, and, by distributing false information about large uploads of those sybils to herself, she manages to have the sybils be connected to herself in the subjective graphs of other peers, and to have the sybils obtain high reputations at those peers as well. Finally, instead of her own identity, she uses her sybils to download, without losing her own reputation and without uploading to other peers, thus exploiting the system.

Like in BarterCast, in SybilRes an uploading peer first evaluates the reputation of a content requester before deciding on an upload. As a part of this evaluation, the uploader calculates the size of the flow on each edge in the maxflow from the requester to itself. After a successful upload, the uploader modifies its subjective graph by decreasing the weights of the edges with non-zero flows, which leads to a decrease in the subjective reputation of the downloader at the uploader. Before they have been used in attacks, the sybils of an attacker are only connected to her, and the flow from them to any other peer passes through a limited number of edges which connect the attacker to the rest of the network, the so-called *attack edges*. Because in the reputation evaluation of a sybil at

least one attack edge will carry a positive flow, by using a sybil the weight(s) of (an) attack edge(s), and accordingly, the reputations of the sybils, are discounted. Discounting edge weights may cause an honest peer to get a lower reputation than its real reputation. To mitigate this problem, another operation, which is run by the downloading peer, increases the weights of the edges that in the maxflow from the uploader to the downloader carry a positive flow.

For protocol evaluation, we use *Dataset_1* and *Dataset_2* (see Chapter 1.4), respectively. From the records obtained, we rebuild the subjective graphs of the peer. We then use these graphs to emulate sybil attacks with SybilRes and BarterCast, and assess the effectiveness of SybilRes in giving low reputations to sybils while keeping the reputations of the honest peers minimally affected.

The main contributions of this chapter are as follows:

1. We explain the problem of sybil attacks in the BarterCast mechanism and show how they can happen (Section 3.2).

2. We introduce an attack resistant version of BarterCast, which we call SybilRes, and we explain it in detail (Section 3.3).

3. We design appropriate experiments to measure the effectiveness of SybilRes in withstanding sybil attacks and we present the results (Section 3.6).

3.1 Related Work

Sybil attacks are a common issue in open distributed environments, and according to a survey by Hoffman et al. [44], most of the 24 reputation systems studied by them are susceptible to this type of attack. Cheng and Friedman [19] have formalized different conditions of sybil-proofness in a graph-based reputation mechanism, and they have proved that no symmetric reputation function (a function which is invariant under a renaming of the nodes) can be sybilproof.

Recently, the popularity growth of online social networks has spurred ideas to mitigate sybil attacks in decentralized environments. In effect, all of these solutions are based on the attacker's limitation on creating connections to the real entities in the network. Based on this idea, Yu et al. [109] have proposed SybilGuard for detecting sybils. SybilGuard uses random walks on a graph that describes past interactions in the system. Later, they proposed a modified version of it, called SybilLimit [108], where peers do multiple but shorter random walks than in SybilGuard. SybilInfer [26], by Danezies and Mittal, is another similar work which is based on the same assumptions as SybilGuard and SybilLimit. In SybilInfer, after doing multiple random walks, a peer uses a Bayesian inference method to determine whether other peers are legitimate or not. Another comparable work

is SumUp [100], that is designed to limit the capability of an attacker in bogus voting for a specific object. SumUp leverages social relations among the accounts and applies a specific method to assign edge weights. By studying SybilGuard, SybilLimit, SybilInfer, and SumUp, Viswanath et al. [102] have found that despite the differences in the algorithms, all of them are trying to find *local communities* around a trusted node. Besides, these methods have a serious usability limitation in multi-component real-world graphs, and they are highly vulnerable to targeted sybil attacks, if the attacker stays close to the trusted node(s) [102].

The concept of reputation is closely linked to that of *trust*, and similar to reputation systems, trust mechanisms are also prime targets for malicious acts. The first attack resistant mechanism metric was introduced by Reiter et al. [89]. In their proposal, in a network of trust relations, paths in which every node trusts its successors are used to evaluate the trustworthiness of a target node. The number of disjoint bounded-length paths from a trusted source node to a target node has to exceed some threshold for trusting that target node.

Leveraging social relations, Spear et al. [94] have introduced KarmaNet to build trusted social paths between peers and to create judicious forwarders. A distinctive property of KarmaNet is that it defines three types of evaluative actions, *initiate, forward*, and *route*, based on which a peer evaluates the trustworthiness of its direct neighbors. An action or a combination of actions is used to prevent different kinds of attacks. Regarding sybil attacks, by decreasing the initiation capability of sybils, KarmaNet introduces an upper bound for the number of spams that a sybil can generate and send to others. In order to recover from a bad karma, the attacker should do positive acts for other honest peers in the network. In comparison to SybilRes, the increase and decrease of karma in KarmaNet resembles edge weight increase and decrease in SybilRes. Ostra [73], is another social network-based unwanted-message preventing mechanism, and it is based on the number of trusted communications a node has built. In terms of reducing link credits, Ostra is similar to SybilRes, but unlike SybilRes it needs a trusted entity to observe users actions and associate them to their identities.

Another trust mechanism which in the sense of building a graph and using maxflow has some similarities with SybilRes is Advogato [59]. In Advogato, members refer to each other based on the perceived skills, and each node is assigned an integer capacity. By adding some edges and nodes to this referral graph, a corresponding weighted graph is created. A run of maxflow from the source node to a special node, called “supersink”, in the modified graph identifies the trusted nodes. Despite some similarities, opposed to Advogato, SybilRes is distributed rather than centralized and there is no need for the trusted source node. Besides, the reputation metric in SybilRes takes values in the interval $[0, 1]$, while the trust metric in Advogato is binary (accepted or rejected).

BarterCast was introduced by Meulpolder et al. [68] to prevent *lazy free-riding* in the

context of P2P file-sharing. The initial assumption in BarterCast was that every body follows the protocol and there are no malicious peers in the system. The validity of this assumption is debatable and it depends on the popularity of the application that uses the protocol; the more popular the application, the greater the chance of being attacked. Seuken et al. [93] have studied the vulnerability of BarterCast to another kind of attack called *misreporting*, and proposed a new method, called DropEdge, to remedy it. Our SybilRes protocol is based on the idea of limited conductance of the flow from sybils to non-sybil peers, similar to the mentioned social network based defense mechanisms. However, in our approach there is no need for trusted source node(s), or knowledge about the whole network, which are hard to attain in the real world. Also, despite SybilLimit-like mechanisms, in SybilRes, we are not trying to detect and ban sybils, but by discouraging sybil-like behavior we make the system robust, like SumUp.

3.2 The Vulnerability of BarterCast to Sybil Attacks

In this section, we first introduce the BarterCast mechanism, and then we explain how by using a number of sybils an attacker can download from other peers while circumventing the rightful reputation loss this implies.

BarterCast is vulnerable to sybil attacks, because by first doing an upload and then using sybils, a malicious user can keep up her reputation and still download from benevolent peers. In such an attack, the attacker first acts as a benevolent peer and gains a positive reputation at other peers by means of uploads. The selection of whom to upload depends on the strategy followed by the attacker, but for a successful attack the reputation of the attacker at a prospective victim should be high. After obtaining a positive reputation, the attacker creates a number of sybils and, by disseminating a number of (fake) BarterCast records, reports to the other peers that those sybils have performed substantial uploads to her. By adding the corresponding edges to its subjective graph by a prospective victim, a path is created from the sybils to the victim, and the sybils get a high reputation at the victim and become ready to be used by the attacker.

In BarterCast, after a data transfer action, the reputation of the downloader at the uploader decreases because either a new edge (with a positive weight) is created from the uploader to the downloader, or, if such an edge already existed, its weight is increased. In a sybil attack, instead of using her own identity, the attacker uses the identity of one of her sybils to download. By this strategy, even though the reputation of the sybil used is decreased, the attacker's reputation is not affected, and she can replace the sybil used with a new one. Since allocation of the available bandwidth is based on reputation values, subverting the reputation values by an attacker is a threat and it may degrade the service quality of the system.

Figure 3.1 shows an example of a sybil attack in BarterCast, where a and v_1, v_2, and

v_3 are the attacker and the victims, respectively, and $s_1, \ldots, s_4$ are the sybils of a. By uploading to v_1, peer a connects herself to the rest of the network, and through the fake edges $s_1 \rightarrow a$,..., $s_4 \rightarrow a$ she connects her sybils to the network. If peers use 2-hop maxflow, v_1 finds a path from every sybil of a to itself, and gives a high reputation of 0.936 to them, according to Eq. (1.1). The situation would be worse when maxflow is applied with a higher number of hops, because then using the same attack edge the attacker may attack multiple peers. For example, with 3-hop maxflow, the peers v_2 and v_3 will find incoming paths from the sybils of a, and a is able to attack them as well. Figure 3.1 shows a case of a single attacker, but in general, a group of attackers can try to collude by setting up any fake network among themselves and their sybils and by connecting themselves to other peers.

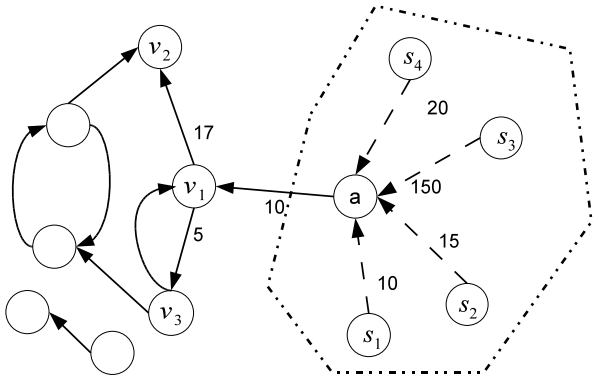


Figure 3.1: An example of a sybil attack in BarterCast (peer a is the attacker, peers s_1, ..., s_4 are her sybils, and the other peers are prospective victims).

3.3 The SybilRes Protocol

In this section, first we describe the main idea behind SybilRes, and then we explain its elements in detail.

3.3.1 The Main Idea of SybilRes

In the subjective graph of a peer, an attacker, her sybils, and the edges between them constitute a subgraph that we call the *sybil region*; the rest of the graph is called the *non-sybil region*. The edges that connect the sybil region to the non-sybil region are called the *attack edges*. Attack edges are created by an attacker by uploading data to honest peers and all the flow between the sybil and non-sybil regions has to pass through these edges. For example, in Figure 3.1 the area inside the dashed line is the sybil region of the attacker

a, and the edge $a \rightarrow v_1$ is an attack edge. Creating an attack edge is not free, because it requires the attacker to do real uploading to a non-sybil peer in the network. The non-free creation of such edges is the Achilles' heel of the attacker.

In SybilRes, after a peer p has uploaded to peer q, it decreases the weights of the edges on the paths from q to p in its subjective graph. These are the edges that are involved in carrying flow from q to p, and they cause q to get a positive reputation at p that was high enough for p to decide to upload to q. The rate of decrease of an edge's weight depends on the amount of flow carried by it and the amount of data transferred after the reputation evaluation. The consequence of the weight decreasing is that in the next reputation evaluation, the maxflow from q to p, and so the reputation of q at p, will be lower. On the downloading side, peer q applies a similar strategy as p, but instead of decreasing, it increases the weights of those edges that in a content request by p will be used by q to evaluate the reputation of p.

As the attack edges are on the paths from the sybils to a victim peer, using the weight-decreasing strategy, their weights, and accordingly, the reputations of the attacker and her sybils, will likely decrease fast. As a result, in order to keep up her and her sybils' reputations, the attacker needs to keep uploading to non-sybil peers in the network, which is the desired behavior.

When an attacker from a set of colluding peers performs a sybil attack, it can do so through one of its "own" attack edges, or though an attack edge of another colluder to which it has a path. In the former case, its own reputation will definitely suffer because of the discounting of the weight of the attack edge, but in the latter case, the reputation of the other colluder will suffer. No matter what fake network a group of colluding attackers create, the joint set of attack edges remains a bottleneck and restricts their reputations. As a consequence, collusion does not pay in SybilRes, and the optimum strategy for an attacker is to have exclusivity on the attack edges she uses.

3.3.2 Edge Accounting

We define the *contribution* of an edge in the maximum flow from one node to another in a weighted graph as the amount of flow that passes through that edge (which may not be uniquely defined, see Section 3.4). An edge *participates* in the maxflow if its contribution is non-zero. For example, in Figure 1.5, in the maxflow from i to j the contributions of the edges $i \rightarrow k$ and $k \rightarrow j$ are 3, and for the edge $i \rightarrow j$ it is 8. Dividing an edge's contribution by the total maxflow gives the *normalized contribution* of the edge, which is a value in the range $[0, 1]$.

Recall that every data upload action is preceded by a reputation evaluation. In SybilRes, after a peer p performs an upload to peer q, p *charges* the edges that contributed to calculating the maxflow from q to p in its subjective graph. The amount by which an edge

is charged depends on its contribution and on the amount uploaded after the reputation evaluation. Edges that contribute more are charged more, and a higher upload leads to a higher charge for all participants.

In order to account for data transfers and charge the participating edges, besides the real edge weight, which represents the amount of data transferred, each edge of a subjective graph is assigned an *effective weight* too. When an edge is added to a subjective graph, its effective weight is initialized to its real weight, and later, if it participates in a calculation of the maxflow from a downloading peer, its effective weight is decreased. In contrast to BarterCast, in which the real weight of an edge is used in reputation evaluations, in SybilRes the effective weight of an edge replaces its real weight in the maxflow algorithm.

Because of the dynamic nature of the network, the real weight of an edge may increase over time due to data transfers. To have precise reputation evaluations, the corresponding updates should be reflected in the effective weights of the edges too. More precisely, if w_{ij} and e_{ij} are the real and effective weights of edge $i \to j$ in the subjective graph of some peer, respectively, then if that peer receives a BarterCast record about a new transfer of size Δ from i to j, then both the real and effective weights of the edge are increased by Δ.

In SybilRes, assigning and updating the effective weights of edges are done locally by each peer within its own subjective graph, and the effective weights are not sent to other peers (but see also Section 3.3.5). It is therefore possible for an edge to have different effective weights in different subjective graphs even though it has the same real weights. In the following sections we explain how the effective weights of the edges are decreased or increased over time.

3.3.3 Edge Charging

Suppose that when evaluating the reputation of peer q, peer p finds in its subjective graph a set $K_h = \{k_1, k_2, ..., k_m\}$ of m *distinct paths* of length at most equal to h from q to itself. Two paths are distinct if they do not consist of exactly the same sets of edges. Each path $k \in K_h$ causes some flow from q to p, denoted by ϕ_k, and the total maxflow from q to p using paths of lengths at most h is equal to $\Phi_h(q,p) = \sum_{k \in K_h} \phi_k$. If an edge $i \to j$ belongs to at least one of these paths, its maxflow contribution is equal to:

$$c_{ij} = \sum_{k \in K_h, i \to j \in k} \phi_k. \tag{3.1}$$

The normalized contribution of the edge $i \to j$ is defined as:

$$n_{ij} = \frac{c_{ij}}{\Phi_h(q,p)}. \tag{3.2}$$

Calculating the contributions is done just before the start of an upload session, and it is independent of how much data is going to be transfered during the session. If e_{ij} is the effective weight of the edge $i \rightarrow j$ just before the session, then its effective weight after the session is:

$$e_{ij}^{'} = (1 - f(n_u, n_{ij})) \times e_{ij}, \tag{3.3}$$

where f is the *charging function* that is used to adjust the effective weights and $n_u \in [0, 1]$ is the normalized value of the data transfer during the session (in Section 3.3.7 we explain how the data transfer normalization is done). The function f should have the properties: 1) f has values in $[0, 1]$, 2) f is increasing in n_u and n_{ij}, and 3) if n_u or n_{ij} is equal to zero, then $f(n_u, n_{ij})$ is equal to zero. Besides, in the design of f some implicit restrictions should be considered as well:

- As both the attack edges and the legitimate edges are charged, a steep charging could lead to many high reputation evaluation errors for honest peers.
- A very mild charging may not be effective enough, and the attacker can still subvert the system and get benefit.

Considering these constraints we design the charging function as:

$$f(n_u, n_{ij}) = \frac{(n_u \times n_{ij})^{\gamma}}{\theta}, \tag{3.4}$$

where $\gamma > 0$ and $\theta \geq 1$ are the parameters that define how strict or forgiving SybilRes is.

3.3.4 Edge Recovery

A side effect of the charging strategy is that the honest peers whose outgoing edges become weaker, unfairly get lower reputations. Conversely, it is possible that due to the decrease in the effective weights of its incoming edges, a peer may get a higher reputation than it has without charging. To solve this problem, we rely on a behavioral difference between sybils and non-sybils, namely, that in contrast to sybils, which only download, benevolent peers perform uploads as well. If sybils would behave normally and also perform upload, the whole problem would be solved. Based on this fact, in addition to charging edges, SybilRes also *recovers* the effective weights of the edges.

To allow for recovery, at the end of a data transfer session from peer p to peer q, the downloading peer q increases the effective weights of those edges in its subjective graph that in the future could be used by q to evaluate the reputation of p. Indeed, these are the same edges that are charged by q if it uploads to p. To find these edges, before downloading, peer q runs the maxflow algorithm from p to itself and determines the participating

edges. After the download, the effective weight is recovered through the relation:

$$e'_{ij} = \min(w_{ij}, (1 + g(n_d, n_{ij})) \times e_{ij}), \tag{3.5}$$

where g is the *recovery function*, n_d is the normalized download, and n_{ij}, w_{ij}, and e_{ij} are the normalized contribution, the real weight, and the effective weight of the participating edge $i \to j$ just before the recovery.

Using the recovery strategy, not only the uploading peer but all the peers in the paths to it are rewarded, and the goodness of the uploading is echoed in the network. Like the charging process, the recovery of the effective weights is only done locally and peers do not send this information to others. The recovery function g has the same properties as the charging function f, and we let it have the same form as in Eq. (3.4), except that the values of the parameters γ and θ may differ. In order to incentivize peers to upload in time, the effective weights should be recovered less than what they are charged, that is, for any $x, y \in [0, 1]$, $g(x, y) < f(x, y)$.

3.3.5 The Global Reputations of Peers

Since the charge and recovery processes are done locally by each peer in its own subjective graph, an attack edge with a low effective weight at some peers may still have an effective weight equal to its real weight in some other subjective graphs. Hence, using the same attack edge, the attacker can still attack multiple peers. For example, in Figure 3.1, through the attack edge $a \to v_1$ and with 3-hop maxflow, the attacker a can attack peers v_1, v_2, and v_3. In SybilRes, the possibility of using the same attack edge to attack multiple peers depends on the number of hops used in the maxflow algorithm. With two hops, each attack edge can only be used to attack the peer who is directly connected to the attacker, but in general, each attack edge can be used to attack all peers that are reachable from the sybils within the number of hops used in maxflow.

To prevent the use of a single attack edge to attack multiple peers, we extend the charge and recovery processes to the whole network by having the receivers of a BarterCast record perform the same charge and recovery operations in their own subjective graphs that were performed by the uploader and the downloader peers recorded in the BarterCast record, respectively. With this strategy, peers can keep up with the updates of the effective weights and have an accurate view about the reputations of other peers without sending additional messages. To be more precise, suppose peer p uploads some amount to peer q. According to the SybilRes protocol, a new record is disseminated that reports this transfer. When a peer receives such a record, before updating its subjective graph with the edge $p \to q$, it performs the charging that p did and the recovery that q did in their own subjective graphs, charging the edges on the paths from p to q and recovering the edges in the reverse direction. Since peers may receive the disseminated records in different

orders, the effective weight of an edge in different subjective graphs may not converge exactly to the same value, but the general trend will be similar.

The reason for not disseminating the effective weights is to avoid the misreporting of these values by malicious peers. For the case of real edge weights, misreporting is prevented by double signing the BarterCast records by the uploader and downloader. The signing and authenticity checking of a record is done using the peers' public and private keys that are already available when Tribler is installed. For the effective weight, using double signatures does not help, because the effective weight of an edge can be updated later as the result of data transfers actions by other peers, over which the uploading or downloading peers do not have any control.

3.3.6 The Reputation Function in SybilRes

In an open environment, where obtaining a new identity is cheap, keeping negative reputations about others is meaningless, because they can easily be whitewashed, and peers can reset their negative background. Considering this generic problem and without losing the bartering property of BarterCast, we reformulate the reputation function of Eq. (1.1). In the new formulation, not only the effectiveness of SybilRes is increased, but the whole system becomes more robust against whitewashing.

In the new calculation, when peer i evaluates the reputation of peer j, like the previous formulation in Eq. (1.1), peer i calculates the h-hop maxflow from itself to j and from j to itself, denoted by $\Phi_h(i,j)$ and $\Phi_h(j,i)$, respectively. Letting $W_h(i,j) = arctan(\Phi_h(i,j))/(\pi/2)$, we define the non-negative subjective reputation of j at i as:

$$R_i^+(j) = W_h(j,i) \times (1 - W_h(i,j)). \tag{3.6}$$

The reputation metric $R_i^+(j) \in [0,1)$ has the properties that:

- If peer j whitewashes, then $W_h(j,i) = 0$ for all i, so its reputation is reset to zero.
- The best for an attacker j to do is to keep her received contributions $W_h(i,j)$ at zero, but SybilRes will decrease $W_h(j,i)$, and so the reputation of j will decrease.
- Like in BarterCast, if two peers contribute equally, the peer with the smaller received contribution has a higher reputation than the other one. A similar statement holds for equal received but larger given contributions.

3.3.7 Normalizing Data Transfers

The charging and recovery functions f and g from Eqs (3.3) and (3.5) are functions of the normalized data transfer and the edge contribution. The edge contribution is normalized

by Eq. (3.2), but the amount of data transferred is unbounded. To do a meaningful charge or recovery, peers should account for the rank of their transfer amount among all transfers that have already happened in the network. In this way, peers can calibrate their upload or download against the norm, and they can base their charge and recovery on this measure.

To find the normalized data transfer values we use the idea of *fractional rank*, which is often used to determine the ranking position of an individual in a population. We define the amounts of data transfers in the network as the population and use the fractional rank of a transfer amount as its normalized value. To calculate the fractional ranks, during the creation of its subjective graph, each peer maintains a sorted list of data transfers that it has heard about. Using this list, when peer p wants to determine the fractional rank of an upload or download of size x, it first adds x to this list and then computes the rank n_x of x by:

$$n_x = (B + 0.5E)/N, \tag{3.7}$$

where B is the number of items smaller than x, E is the number of items equal to x, and N is the total number of items in the list.

3.4 Finding Fair Contributions

The maximum flow from one node to another in a weighted directed graph may be realized in multiple ways, and an edge may have different flows in different realizations. As a consequence, the concept of contribution in Section 3.3 becomes vague and the contribution values are not uniquely defined. An example of this ambiguity is shown in Figure 3.2. In this graph, the maximum flow from s to d is 4.0, but there is an infinite number of possible realizations of this total flow. Among all solutions, except for the edge $s \rightarrow x$ whose contribution is fixed at 4.0, the contributions of the other edges vary. In SybilRes, in order to do unbiased edge charging and recovery, we need to find some notion of *fair* contributions of the participating edges. Fair contribution in the example of Figure 3.2 would be values of 2.0 for all edges except $s \rightarrow x$. We call a maxflow realization from node p to node q *fair* if the flow on an edge that is on at least one path from p to q is proportional to the sum of the capacities of the paths that it belongs to. The flow on the edge with a fair realization of the maxflow is taken as its contribution.

For SybilRes, we have devised a heuristic algorithm to approximate a fair maxflow realization. Let G be the directed graph in which we want to approximate the fair maxflow realization from a source to a destination node. In our algorithm, instead of calculating (a realization of) the maxflow with one execution of the Ford-Fulkerson (FF) algorithm in G, we repeatedly execute FF in a graph with the same nodes and edges as G but with randomized edge weights. The partial sum of an edge's weight in different executions ap-

proximates its original weight. We take as the approximate fair realization of the maxflow the sum of the maxflow realizations of the separate executions of FF. More precisely, each edge is assigned a *residual capacity* s, which is initialized to its original weight. Then the weight of the edge used in FF is a uniform random value in the range $[0, s]$. When the next execution of FF assigns a flow c to the edge, the residual capacity of the edge is set to $s - c$ and the process is repeated. As soon as the sum of the maxflows as computed by the repetitions of FF exceeds a certain fraction f_r of the real total maxflow in the original graph, the algorithm stops.

In maxflow executions, we run a modified version of the Ford-Fulkerson algorithm that uses depth-first search to find the paths from the source to the destination node. With this modification, edges that are only part of longer paths are not starved by the edges in the shorter paths. Although, during the path finding process the next outgoing edge from a node is chosen randomly with an equal chance for all edges, so all edges get a chance to participate.

For the example in Figure 3.2, with a value of f_r of 0.95, we get in 10 FF executions the approximate fair allocation with $c_{sx} = 3.97, c_{xy} = 1.85, c_{yd} = 1.85, c_{xz} = 2.12, c_{zd} = 2.12$.

A similar problem called *maximum balanced flow* was introduced by Minoux [71]. There, besides the edges' capacities, there is a rating function that specifies the upper bound of the flow on each edge. The difference between Minoux's and our problem is that in our case, the rating function and the upper bound of an edge is not known in advance.

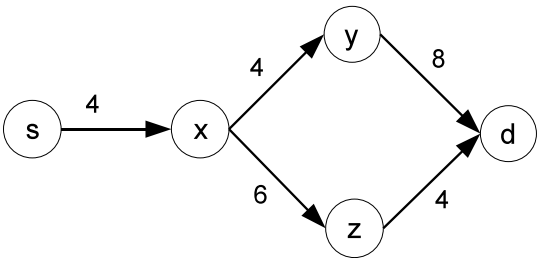


Figure 3.2: A sample subjective graph

3.5 Analysis of SybilRes

In this section, first present the result of applying SybilRes to the example of Figure 3.1 of Section 3.2, then we elaborate on the issues of false positive and false negative errors incurred by SybilRes. Finally, we discuss on the problems of churn, heterogeneity, and policy.

3.5.1 A Sybil Attack Example

In the example of Figure 3.1, first the attacker a uploads 10 units of data to peer v_1, and then she tries to download from v_1 using her sybils. Without loss of generality suppose that the sybils are created and used in chronological order. Also, for simplicity assume that irrespective of the upload amount from v_1 to a sybil, the charging function halves the effective weights of the participating edges. Table 3.1 contains the effective weight of the attack edge $a \rightarrow v_1$, maxflow from the sybils to v_1, and the reputations of the sybils (indeed the reputation of the attacker) at v_1. As calculated by SybilRes, at the time of using sybil s_4 the effective weight of the attack edge and the reputation of the attacker have already been decreased to 1.25 and 0.570, respectively.

Table 3.1: Applying BarterCast and SybilRes to the example of Figure 3.1

	BarterCast			SybilRes		
s_x	e_{av_1}	$\phi_2(s_x, v_1)$	$R_{v_1}(s_x)$	e_{av_1}	$\phi_2(s_x, v_1)$	$R^+_{v_1}(s_x)$
s_1	10.0	10.0	0.936	10.0	10.0	0.936
s_2	10.0	10.0	0.936	5.0	5.0	0.874
s_3	10.0	10.0	0.936	2.5	2.5	0.757
s_4	10.0	10.0	0.936	1.25	1.25	0.570

3.5.2 False Positives

In the sybil attack literature, most proposed defense mechanisms suffer from *false positive* (FP) and *false negative* (FN) errors [26, 108, 109]. In sybil-detection mechanisms, FP means that a peer is detected as a sybil but it is not, and FN means that a peer is a sybil but the mechanism is unable to detect it. In SybilRes, FN is interpreted as the forgiveness of the protocol when giving low reputations to sybils. So a smaller decrease in a sybil's reputation value means a higher FN and vice versa. Due to the charge and recovery strategies, a legitimate peer may get a higher or a lower reputation than its real reputation, and this difference is interpreted as an FP error.

We argue that because of the growing pattern of a subjective graph in the non-sybil region, the situations which lead to FP occur less, and if they happen, the change in the reputation is made up by the converse operation (charge or recovery):

- Due to peers' normal data transfer behavior in the non-sybil region, they are able to find diverse paths to other peers in the region. Having diverse paths means that the maxflow between two peers is split among multiple edges from different paths, and a group of specific edges (like attack edges) do not carry the whole flow. As a result, edges contribute less and are charged less.

- If because of other peers' download actions a peer's uploading edges are charged, then it is highly likely that its edges are recovered by other peers' upload actions as well. In total, for non-attack edges, the lost weight during charging is partially reimbursed during the recovery.

3.5.3 Churn, Heterogeneity, and Policy

In P2P systems, peers continuously move in and out of the system, and *churn* is a known phenomenon in such systems. In SybilRes, during a period of absence a peer may miss some disseminated records, but it keeps its local subjective graph, and so SybilRes will function properly when the peer becomes online again. By staying online, eventually the gossiping protocol will recover the missed information.

It is possible that some peers use the BarterCast protocol and some are updated to SybilRes. Since all computations are done locally and SybilRes does not change the way that peers interact with each other, both groups of peers can co-exist and interact transparently. The main difference is that peers running SybilRes will be sybil resilient.

Like in BarterCast, in SybilRes we do not deal with the policy of how to use the reputation values; this topic has been studied by Seuken et al. [93]. As a short comment, peers can either use the absolute reputation values or they can rank peers according to their reputations. Ranking peers is better than using the absolute values as it does not need extra settings of threshold values. With ranking, when there is a resource (bandwidth) shortage, highly ranked peers are preferred over lowly ranked peers, but if there are enough resources, newcomers can get bandwidth allocated and bootstrap easily.

3.6 Experimental Setup

We evaluate SybilRes from two aspects, which are its effectiveness against malicious behaviors and its accuracy in the evaluation of benevolent peers. The experimental setup is based on data collected from the Tribler network and on simulating the situations in which a peer evaluates another peer's reputation. For the simulation of sybil attacks, we artificially add an attacker and a number of sybils and compare the behavior of the system when using SybilRes and BarterCast.

3.6.1 Protocol Simulation

In the simulation of SybilRes, we gradually build up the subjective graphs of the peers and determine the situations in which peers evaluate each other's reputations. Building the subjective graphs is done using the time-ordered list of BarterCast records from the crawler starting from empty graphs. For reputation evaluations we use both BarterCast

and SybilRes. Moreover, in SybilRes in the subjective graphs of the uploading and downloading peers the effective weights of the participating edges are updated (according to the charge and recovery strategies). Suppose (i, j, U, D, t) is the next record from the trace that is going to be processed. If $U > 0$, before uploading, peer i has to evaluate peer j's reputation, and depending on the reputation value and the applied policy, peer i decides whether to upload to j or not. For simulation purposes we assume that uploading always happens. By applying BarterCast and SybilRes, peer i computes two subjective reputations, one for each protocol. If $D > 0$ then the above process is repeated by j and this time j evaluates i.

After the reputation evaluation, the edge $i \rightarrow j$ (with real and effective weight of U) is added to the subjective graphs of all peers that have already appeared in the trace, or, if it already exists, then its real and effective weights are updated. If i and/or j appears for the first time, then a new subjective graph for i and/or j is created and $i \rightarrow j$ becomes its (their) first edge. When adding a new edge to the subjective graphs, we assume there is an efficient mechanism that can provide peers with the latest disseminated records, e.g., using full gossiping [27]. After updating the subjective graphs, the process continues with the next record from the trace until all records have been processed.

3.6.2 Sybil Attack Emulation

To evaluate the effectiveness of SybilRes, we perform artificial sybil attacks. To select an attacker and a victim, after processing a certain fraction T_{pa} of the records so that the peers have meaningful subjective graphs, we randomly choose a pair (a, v) of attacker and victim, respectively, with the attacker having a positive reputation at the victim. Having only one pair of attacker and victim in each simulation run is representative for the operation of SybilRes, as we argued in 3.3.1 that collusion is not beneficial.

After choosing the attacker and victim, the next step is to create a number of sybils for the chosen attacker. To simulate an attack, after processing each record from the part remaining in the trace after the selection of the attacker and victim, with some probability, we create a *sybil record* and process it like a normal record. A sybil record is a BarterCast record of the form $(s_a^k, a, U_s, 0, t)$, where s_a^k is the k'th sybil of the attacker a and U_s is the pretended amount of data transferred from s_a^k to a. To have an effective attack, U_s should be larger than the sum of the weights of the outgoing edges of a so that the maxflow from the sybil to the victim is not limited by the attacker herself. Adding a sybil record to the subjective graph of the victim gives a positive reputation to the sybil at the victim and makes it ready to be exploited.

After creating a sybil with a positive reputation, we simulate an attack where the sybil downloads data on behalf of the attacker. In order to do so, immediately after processing the sybil record we create and process an *exploit record*. Such a record is of the form

Table 3.2: Simulation parameter setting

Parameter	Value	Section
θ for functions f and g	1.0	3.3.3
γ for function f	1.8	3.3.3
γ for function g	2.0	3.3.3
T_{pa} (fraction of pre-attack records)	0.5	3.6.2
U_s (transfered amount in a sybil record)	100 GB	3.6.2
D_s (transfered amount in an exploit record)	30 MB	3.6.2
h maxflow hops number (data of 2009)	4	3.3.6
h maxflow hops number (data of 2010)	3	3.3.6
f_r (pass rate in the fair contribution algorithm)	0.95	3.4

$(s_a^k, v, 0, D_s, t)$, where D_s is the amount of data the sybil s_a^k downloads from the victim v. Processing this record is equivalent to exploiting a sybil by the attacker, and this is the point at which SybilRes should give lower reputations to sybils.

To assess the effectiveness of SybilRes, we also run the simulation with the same pair of attacker and victim, but with the attacker using her own identity to download the same amount of content from the victim. The difference between the attacker's reputation in this case and in the case of using sybils is a measure of SybilRes's effectiveness.

Due to the charge and recovery strategies, with SybilRes benevolent peers may get higher or lower reputations, whereas BarterCast, which does not change edge weights, gives correct reputation values for these peers. To assess the resulting error in the reputation values, during simulation we record the reputations of the benevolent peers evaluated by other peers when using BarterCast and SybilRes. The difference between these values is the error incurred by SybilRes.

3.7 Results

In this section, we present an experimental evaluation of SybilRes. First, we run an experiment to find appropriate values for the parameters γ and θ in the charge and recovery functions f and g (see Eqs (3.3) and (3.5)). Then, we assess the protocol's robustness against sybil attacks, and the incurred errors in the reputations of benevolent peers by SybilRes. Finally, we evaluate the global reputation of attackers and the sensitivity of the protocol to the size of the exploit record. The values of the parameters that we use in the simulations are presented in Table 3.2.

3.7.1 Parameterizing the Charge and Recovery Functions

In SybilRes, we have to select appropriate values for the parameters γ and θ of the charge and recovery functions that simultaneously maximize the *effectiveness* and minimize the *incurred error*. We define the effectiveness as the minimum number of attacks after which the attacker loses M percent of her initial reputation, a higher number of attacks means a lower effectiveness and vice versa. The incurred error is defined as the fraction of benevolent peers whose reputations decrease by more than some threshold value V when SybilRes is used. To find appropriate values for γ and θ, we simulate SybilRes in scenarios with different values for γ and θ and measure the effectiveness and the incurred error in each scenario. For the charge function f, we experimented with values of γ in the range [0.2,1.8] and of θ in the range [1,4]; for the recovery function g, γ is 0.2 higher than in f (in order to give peers an incentive to upload, see Section 3.3.4) and θ has the same value as in f.

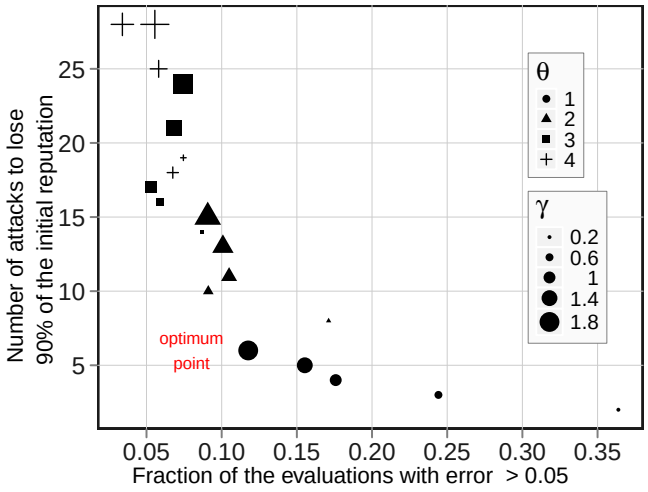
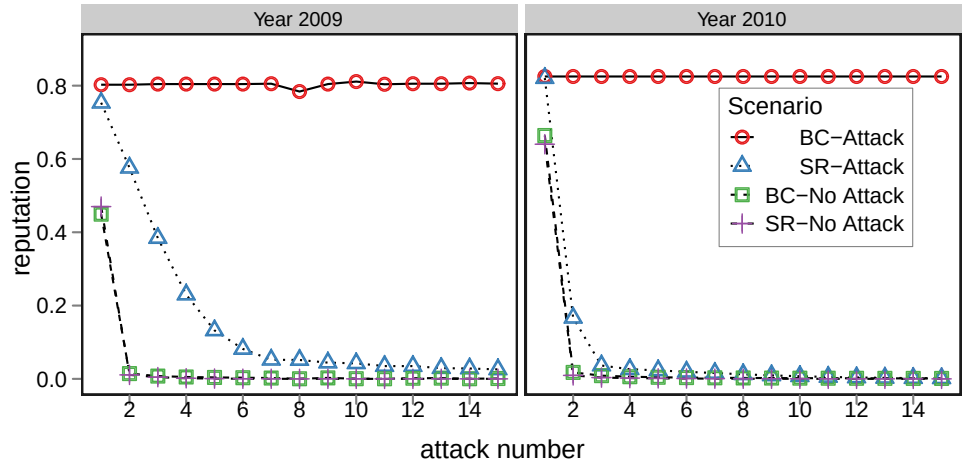


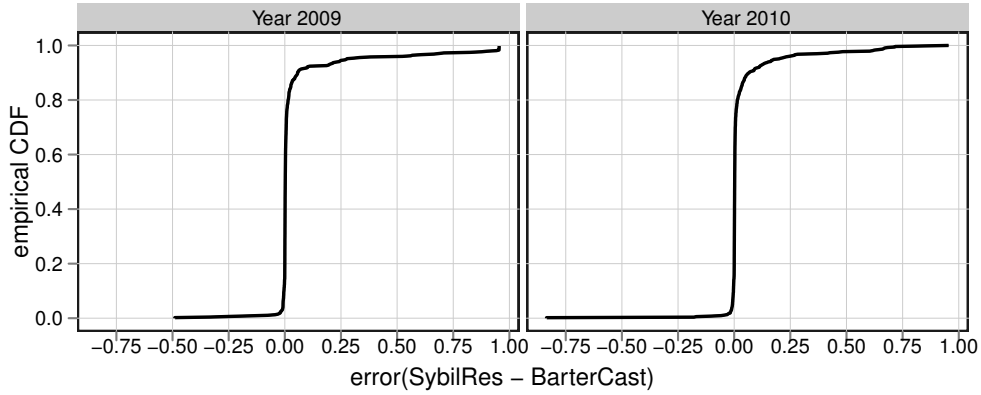
Figure 3.3: The effectiveness versus the error with SybilRes for different combinations of the parameters γ and θ of the charge and recovery functions.

For a reputation loss of $M = 90\%$ and an error threshold value of $V = 0.05$, Figure 3.3 plots the effectiveness against the incurred error for various combinations of values of γ (in f) and θ. As can be seen, with increasing values of θ the effectiveness drops, and for $\theta \geq 3$, the attacker is able to do at least 15 attacks before loosing 90% of her reputation. Similarly, there is a decreasing trend in the effectiveness against γ. On the other hand, a lower γ leads to a higher error, and there is a trade-off between effectiveness and error. To have a high effectiveness without an error that is too high, we select the point $\theta = 1$ and $\gamma = 1.8$ at the knee of the curve. This combination of parameter values gives an error

of around 0.12 and an effectiveness of at most 6 sybil attacks.



(a) The average reputation of attackers in SybilRes (SR) and BarterCast (BC).



(b) The CDF of the incurred errors in the reputation values of benevolent peers in SybilRes.

Figure 3.4: The effectiveness (a) and the error in the reputation values (b) with SybilRes (The results for SR-No Attack and BC-No Attack coincide almost completely.)

3.7.2 The Reputations of Peers in SybilRes

In this section we present the simulation results that show the effectiveness of SybilRes in combating sybil attacks in comparison with BarterCast when the attacker uses multiple sybils to attack the same victim. We will perform simulations for two scenarios, one in which the attacker actually employs sybils to download content, and one in which the attacker uses her own identity to do so (and, in fact, does not really "attack"). For each combination of protocol and scenario and for both of our traces, we run 100 simulations

with different attacker-victim pairs in each run, using the values of the parameters listed in Table 3.2. During the simulations, we record the reputations of the attackers at the victims after each attack, and we record the reputations of benevolent peers when they download as computed by the uploaders.

The simulation results are shown in Figure 3.4. Figure 3.4(a) shows how the average reputation of the attackers changes as the number of attacks increases, for different combinations of protocol and attack scenario. Each point in this figure represents the average for 100 simulations, and the quantity on the horizontal axis is the sybil (attack) number. As can be observed, even with the new reputation metric, with only non-negative values in Eq. (3.6), BarterCast is defeated by the attacker. In contrast, with SybilRes, after a few downloads (using the fifth sybil), the reputation of the attacker approaches her real reputation if she uses her own identity. Figure 3.4(b) presents the CDF of the errors incurred by SybilRes for the traces from 2009 and 2010. As can be seen, in 90% of the cases the error is almost equal to 0, and for 95% it is less than 0.3. To assess the protocol with different hops, for the trace of 2010, we use maxflow with 3 instead of 4 hops. With this trace, the SybilRes protocol is almost as effective as for the crawl of 2009, and the errors are even smaller.

3.7.3 Attacking Multiple Peers

As mentioned in Section 3.3.5, an attacker may attack multiple peers using the same attack edge when the number of hops used in maxflow is larger than 2. In this section we report an experiment to assess how SybilRes performs in the face of attacking multiple peers. For this assessment, after choosing the pair of attacker and victim, we identify those peers that can be attacked using the same attack edge that is used to attack the victim. During the simulation, after using each sybil by the attacker and the dissemination of the corresponding records, we have the potential victims (i.e., the peers reachable from the attacker along the attack edge within the number of hops as used by maxflow) evaluate the reputation of the attacker. Figure 3.5 presents the results for the data crawled in 2009. As can be seen, similarly as the reputation of the attacker at the actual victim, the average reputation of the attacker at the potential victims decreases to zero.

3.7.4 Varying the Attack Size

The results in Figure 3.4 are based on a data transfer of 30 MB for each exploit record, which is equal to 70th percentile of the edge weights in the collected data. Using this value makes sense, as with small values the attacker may not gain much benefit from her attack, and very large values may be suspicious. In order to assess the stability of SybilRes against this parameter, we do an experiment with different values for exploit record size,

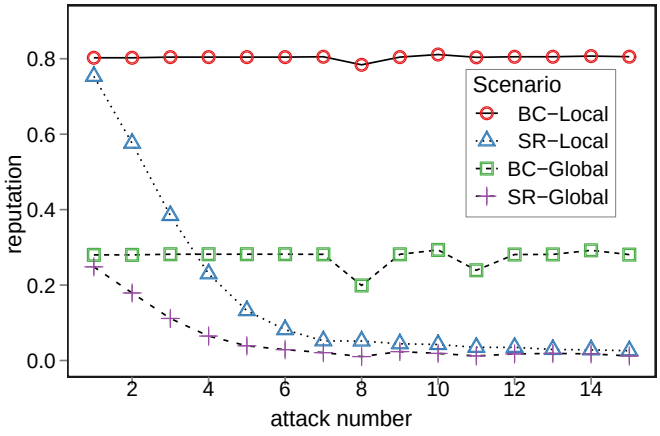
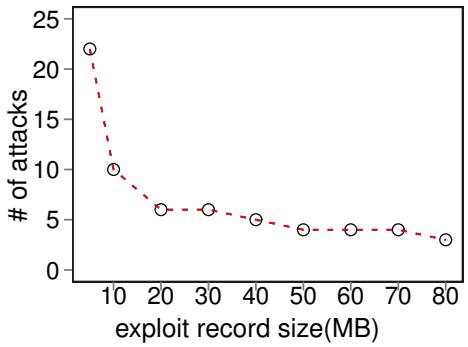
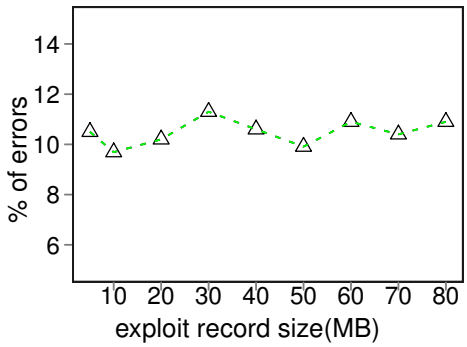


Figure 3.5: The average local reputation of the attacker at the actual victim and its average global reputation at the potential victims in BarterCast (BC) and SybilRes (SR) using the trace of 2009.

using trace from 2009 (the other parameters are same as in Table 3.2). For each value we do 100 runs. Figure 3.6(a) shows the average number of sybils before loosing 90% of the initial reputation, which varies from 22 for 5 MB to 2 sybils for 80 MB. This means that if the attacker does large exploits, she looses her reputation very fast. Figure 3.6(b) depicts the percentage of the cases that the reputation evaluation error for honest peers is larger than 0.1. As can be seen, there is no correlation between the record size and reputation error. In conclusion, playing with the data size the attacker can not manipulate the reputations of honest peers.



(a) Number of attacks to loose 90% of the initial reputation.



(b) Percentage of reputation evaluations with error > 0.1

Figure 3.6: Sensitivity of SybilRes against the data size of the exploit record.

3.8 Conclusion

In this chapter, we have introduced SybilRes, a redesign of the BarterCast reputation mechanism, which by updating edge weights in the subjective graphs of peers combats sybil attacks. The experimental results using the Tribler data set show that SybilRes is highly resilient against sybil behavior, and accurate in the evaluation of the reputations of honest peers. We believe that the edge accounting method of SybilRes (charge and recovery) can be adapted and used in any similar system which is based on, or can be transformed to a weighted graph, e.g., Ripple[1].

As future work, we are designing a scalable record dissemination method tailored for SybilRes. In this method, using its subjective graph a peer derives a similarity measure between itself and other peers that it knows, and when selecting a record receiver, it gives priority to peers with a higher similarity. By targeted dissemination, the load due to the dissemination of records will decrease without affecting the accuracy of reputations very much. Chapter 4 addresses this problem.

[1] http://ripple-project.org

Chapter 4

Targeted Information Dissemination

Providing efficient reputation management mechanisms at scale is an important step to provide trust in many distributed systems, like file sharing systems. A typical online reputation mechanism is composed of three main components: Formulation, Calculation, and Dissemination [44]. The dissemination component provides the other components with the required information to operate. More specifically, in reputation mechanisms in which the calculation component uses information from other participants (peers) on *interactions* in the system as input, peers will not be able to evaluate accurate reputations without an effective spread of this information. From the point of view of reputation *accuracy*, providing peers with more information is preferred, but from the point of view of *scalability*, uncontrolled and blind dissemination can be problematic in terms of communication, computation, and storage costs. This chapter deals with this trade-off in large-scale distributed reputation systems by providing a scalable dissemination method in the BarterCast reputation mechanism.

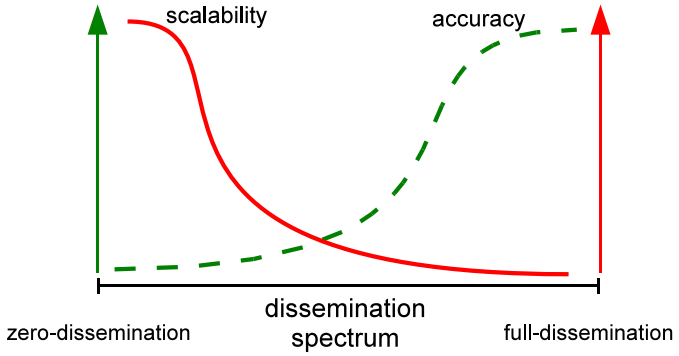


Figure 4.1: The information dissemination spectrum and the accuracy and scalability curves.

In online reputation mechanisms, information dissemination spans the spectrum from

zero to *full dissemination*, see Figure 4.1. With zero dissemination, participants only use their own direct experiences for evaluating reputations, and no information on interactions is spread. Such a mechanism works if the participants interact frequently with each other, and if having only the direct interactions is enough to have an accurate prediction about a counter party's future behavior. At the other extreme of the spectrum is full dissemination, where all participants receive information of all previous interactions. Although from the accuracy point of view this is desirable, full dissemination does not scale, and may even be unnecessary.

In large-scale online systems such as P2P file-sharing systems, peers will only interact with a subset of all peers, for instance, those peers who have *similar* tastes with respect to the content available in the system. Providing peers with information on all peers and interactions is then very inefficient. Rather, information dissemination targeted at similar peers may then be sufficient and much more efficient, which especially is important for power, memory, and computation constrained mobile devices. In BarterCast, peers build a *partial graph* with peers as nodes and interactions they have learnt about as edges. In Chapter 2 we have shown that performing full dissemination about all interactions between peers improves BarterCast's accuracy. However, this *full dissemination* approach incurs high operational costs. In this chapter, we propose a new low-cost dissemination mechanism for BarterCast called *SimilDis*, which without providing a full view to all peers leads to highly accurate reputation evaluations.

In SimilDis, we use either of two methods to compute peer similarity values, one of which is deterministic and of which is non-deterministic. In the first, each peer builds a labeled *similarity graph*, which is a directed acyclic graph (DAG), from its partial graph in which the labels indicate the similarities with the local peer. The second method is based on doing multiple non-uniform random walks (RW) in a peer's partial graph; then, the number of times a node is visited is a measure of its similarity to the local peer. This method was already used in [98, 99]. Both methods are solely based on a peer's local information. In order to evaluate SimilDis, we simulate it using traces from the Tribler network, and we assess its accuracy in evaluating reputations and the incurred communication, computation, and storage costs. The results show that SimilDis, compared with full dissemination, yields very low reputation evaluation errors, and causes the communication costs and the average size of the partial graphs to be reduced by two orders of magnitude.

The main contributions of this chapter are as follows:

1. We design and present a new similarity-based protocol for targeted disseminating of BarterCast records (Section 4.3).

2. We present in detail the DAG-based method for incrementally updating similarity values (Section 4.4).

3. Employing data collected from the Tribler network, we measure the accuracy and the overhead of two similarity-based dissemination methods and make a comparison it with the case of complete dissemination (Section 4.6).

4.1 Related Work

Three areas of research are very relevant to the topic of this chapter, which are gossip protocols, information dissemination in reputation and trust systems, and node similarity in graphs. In this section we give a review for each area.

Since their introduction for database synchronization [31], gossiping protocols have found various applications such as membership management [9], aggregation [48], multicasting [41], and information dissemination [33]. Gossiping protocols consist of three elements: *select-partner* (whom to send a message to), *select-to-send* (what to send), and *select-to-keep* (what to keep from the information received) [91]. The select-partner element plays a key role in the formation of the topology, induced by the protocol, and its effectiveness. Regarding information dissemination, gossiping protocols have a number of desirable properties like resilience to failures, fast convergence, load balancing, and high scalability, which make them suitable for distributed systems.

4.2 A General Overview of SimilDis

To have a scalable and accurate reputation mechanism and to provide the right information to the right peers, the selection of peers for sending records should be done carefully. One way to realize this goal is to use a similar technique to *semantic clustering* in distributed search mechanisms [23, 62]. In such a technique, based on a kind of semantic similarity, peers are clustered in a number of groups, and when a peer initiates a query, first it sends it to its group members, and only if the reply is not satisfying it asks outsiders. A similar technique to semantic searching can be used in the spread of interactions in reputation mechanisms as well.

The BarterCast mechanism can be decomposed into the three components of *dissemination*, *formulation*, and *calculation*. The role of the dissemination component is to gather and provide the other components with the new BarterCast records that have been spread in the network. In the new mechanism, the dissemination component is replaced by SimilDis and it differs from the previous dissemination component in two significant ways.

First, instead of 1-hop dissemination, peers are allowed to send the received records to other peers in the network. In current BarterCast, to avoid misreporting, peers only are allowed to spread records about their own direct interactions with other peers. In other

words, if p uploads to q then only p and q can inform other peers about this action, but the other peers are not allowed to disseminate it. This restriction limits the record reachability and decreases the reputation accuracy calculated by peers [27]. In SimilDis, to solve this problem, we allow peers to send the received records to other peers in their partial graphs. To prevent misreporting, instead of initiating plain records, the peers who are involved in a data transfer action sign the record with their private keys. With signed records, no one can tamper with and change the record content. Allowing peers to send the received records can increase the dissemination level, but it will also increase the communication, storage, and computation costs. These issues are addressed by *targeted dissemination*.

Secondly, in gossiping protocols choosing a right set of rumor receivers is crucial in building a desired overlay [47], and on the overall efficiency of the protocol [33]. Based on this idea, in SimilDis, using the partial graph G_p of peer p, we derive a similarity measure between p and the other peers in G_p. Using this similarity metric, peers who are similar to p get a higher priority to be chosen as the record receivers during the dissemination of BarterCast records. With this modification, the partial views of peers are concentrated around similar peers and only records that are of value are disseminated and kept by each peer.

In summary, SimilDis operates as follows. Besides its partial graph, each peer builds and maintains a limited-length, ordered *similarity list*. When sending a record, it selects a set of random peers from its similarity list as the record receivers. The details of the similarity computation and update processes are explained in Section 4.3.

4.2.1 Information Dissemination in Reputation and Trust Systems

As mentioned in the introduction, providing reputation evaluators with the right set of information is crucial for accurate evaluation. Hoffman et al. [44] have studied reputation systems from various dimensions and have defined four aspects for their dissemination component: *dissemination structure*, *dissemination approach*, *dissemination durability*, and *level of redundancy*. The dissemination structure specifies whether the information is collected and disseminated in a centralized fashion, like in eBay, or in a decentralized way, like in Credence [104] and EigenTrust [51]. The dissemination approach categorizes systems as deterministic or probabilistic. Deterministic approaches usually are based on a hierarchical structure [32] or they use DHT, as EigenTrust. Dissemination durability is mostly a matter of implementation, but in general there are two types of them, permanent storage systems which keep information for a long period, like EigenTrust and Credence, and volatile or short-term storage mechanisms, like ARA [42]. Finally, the redundancy aspect relates to the degree of information redundancy, and involves a tradeoff between scalability and reliability. Considering these aspects, we categorize BarterCast as distributed, probabilistic, long-term storage, and redundant, Figure1.6 in Chapter 1.

In computer science literature, the terms reputation and *trust* are closely related to each other, and sometimes they are used equivalently. Briefly, trust refers to a subjective opinion about an entity which is less general than reputation [39]. Trust mechanisms have been widely studied in various domains as multi-agent systems, P2P networks, ad-hoc networks, wireless sensors networks, and dozens of trust protocols have been proposed [39]. Despite their diversity, like in reputation systems, effective dissemination of behavioral information is a vital requirement for doing a meaningful trust inference [66]. Specially, in mobile and sensor networks, due to power and computation limitations, proper built of *web of trust* (the trust network) is critical for the scalable operation of the system [84]. The proposed method in this chapter, for targeted dissemination, is easily applicable in this area as well.

4.2.2 Node Similarity

Due to the high volume of generated information and the need to filter and categorize them, similarity measures have gained a lot of interest in the online world, and they are widely used in recommender and collaborative filtering systems [7, 16, 90]. Various types of similarity measures have been introduced. If entities and their relations are transformed into a graph, then we can define a new kind of similarity, called *structural similarity* [56], which is simply based on the connections between nodes in the graph. The basic premise of structural similarity is that the structure of a network reflects real information about the nodes.

In the network literature, researchers have proposed various approaches to quantify structural similarity. One of the earliest approaches is called *structural equivalence* [64]. Here, the more neighbors two nodes have in common, the more similar they are. Later, Jeh et al. [46] proposed *SimRank*, which is predicated on the idea that two nodes are similar if their neighbors are similar. Despite its elegance, *SimRank* has a number of drawbacks: nodes that are at an odd distance from each other have a similarity of zero, the edge weights are omitted from the similarity measure, and with any change in the graph all similarities have to be recalculated. Besides, *SimRank* calculates the similarity between every pair of nodes, which in some applications is unnecessary. These drawbacks limit the applicability of *SimRank* in our problem.

Antonellis et al. [11] proposed an extended version of *SimRank*, called *SimRank++*, which for similarity calculation takes into account the edge weights and an external similarity measure called *evidence*. Except for using the edge weights, *SimRank++* still suffers from the other mentioned drawbacks of *SimRank*. Considering the static and iterative nature of *SimRank*, Li et al. [60] proposed an incremental version of *SimRank*.

Based on the idea of *regular equivalence* (nodes are similar if they are connected to similar nodes), Leicht et al. [56] proposed a linear algebric method for calculating node

similarity. Their fundamental assumption is that an edge between two nodes indicates a similarity between them (similar as in BarterCast). Unlike *SimRank*, this method considers both odd and even length paths, but it is still static, computes all pairwise similarities (which is unnecessary in our case), and does not consider edge weights. In view of the limitations of the mentioned similarity methods and our specific requirements for targeted dissemination, we devise and apply our similarity methods, see Section 4.3.

4.3 Design Details

In SimilDis, the partial graph of a peer is used for two purposes: reputation calculation and similarity computation. Using its partial graph a peer builds a list of similar peers to itself, and when disseminating information the target nodes are chosen from this list. In this section, we explain the process of similarity computation.

4.3.1 Peer Similarity Requirements

Usually in distributed search techniques similarity is derived from a predefined *user-item* matrix, from which one can infer a similarity measure between users or between items [62]. Even though we do not have such fine-grained data in the partial graphs, still an edge between two peers does show their common interest in the same content, which can be used in the similarity computation process. Considering the operational requirements and the properties of partial graphs, we list the following desirable features for the similarity metric in SimilDis:

- An edge between two nodes is a sign of similarity between them and should be accounted for in the similarity calculation.
- The edge weights should be considered in the similarity calculation.
- The similarity between two nodes decreases when the distance between them increases.
- As the partial graph is growing (new nodes or edges are added and the weights of existing edges change), the similarity values should be updated dynamically.
- A peer only needs to maintain the similarities between itself and other peers in its partial graph.
- Only the relative similarity (ranks) of peers is important.

Based on these requirements we devise two algorithms to compute similarity, one is based on using a *Directed Acyclic Graph* (DAG) derived from the partial graph, and other is

based on multiple random walks in the partial graph of a peer. We use these methods since both conform to the mentioned relaxed similarity requirements, and, in our context, they are more efficient in computing similarity values than the existing solutions cited in Section 5.1.

4.3.2 DAG-based Similarity

From the point of view of similarity, the direction of a data transfer is not important. Therefore, when in the partial graph G_p there are two directed edges $u \to v$ and $v \to u$ between nodes u and v, with weights w_{uv} and w_{vu}, respectively, we replace these two edges by a single undirected edge uv with weight $w_{uv} + w_{vu}$. The new undirected graph created in this way is denoted by U_p. This graph is not used for reputation calculation; for this we still use the partial graph itself, so free-riders will not benefit from the higher edge weights in U_p.

Starting from the node p in U_p, a new labeled wieghted DAG S_p is generated, where the *label* of each node shows its similarity to p. Initially, the graph S_p only contains node p with label $s_p = 1.0$ and *level* $l_p = 0$. The label 1.0 shows the maximum similarity of p to itself, and the level of a node is the distance of the node to the source node p. For each neighbor q of p in U_p, a new edge $p \to q$ is added to S_p and the level of q is set to one higher than the level of p, so $l_q = l_p + 1$. An edge like $p \to q$ induces a parent-child relation between p and q. The weight W_{pq} of the edge $p \to q$ in S_p is obtained by relative splitting of the similarity of p among all its children:

$$W_{pq} = \frac{w_{pq}}{\sum_{i \in N_p^+} w_{pi}} \times s_p, \tag{4.1}$$

where s_p is the label of p, N_p^+ is the set of children of p, and w_{pq} is the weight of the edge pq in U_p. This process of adding nodes and edges to S_p continues with the grand children of p until all the nodes in the connected component of U_p that p belongs to have been added to S_p.

Starting with the source node p in S_p, using the similarity value of a parent node and its outgoing edge weights, we are able to calculate the similarity values of its children. The similarity of a node q to the source node p is equal to the sum of the weights of its incoming edges in S_p multiplied by a *decay* factor:

$$s_q = \theta^{l_q} \times \sum_{i \in N_q^-} W_{iq}, \tag{4.2}$$

where l_q is the level of q, θ is a predefined decay factor in $(0, 1]$, and N_q^- is the set of parents of q in S_p. Due to the factor θ^{l_q}, by going further away from the source node, the

similarities of the nodes to the source node decrease.

Using the above procedure, the graph S_p is built up level by level, but an ambiguity arises when two nodes u and v with the same level have a common edge in U_p. In such a situation it is not clear whether v is a child of u or the other way around, and which of the edges $u \to v$ or $v \to u$ should be added to S_p. If both are added, the graph S_p will not be acyclic, but if no edge is added, we lose valuable similarity-related information. We deal with this issue by having the nodes u and v exchange a fraction of their similarities and by further ignoring the edge uv in the calculation of the similarities of the lower level nodes. By this strategy, the acyclic property of S_p is preserved, and still the edge uv influences on the similarities of u, v and their children.

To calculate the amount of similarity exchange, first the edge uv is temporarily replaced by a dummy node d_{uv} and two edges from u and v to d_{uv} in S_p. This replacement is done for all the edges between nodes at the same level as u and v. Then the nodes u and v compute amounts $\eta_u(uv)$ and $\eta_v(uv)$ from their similarity and transfer them to d_{uv}, which then has similarity $\eta_u(uv) + \eta_v(uv)$. Finally, this value is equally split between the nodes u and v, and so the change in the similarity of u will be:

$$\Delta_u(uv) = \frac{\eta_v(uv) - \eta_u(uv)}{2} \ , \tag{4.3}$$

and for the node v it is $\Delta_v(uv) = -\Delta_v(uv)$. After processing all the edges ux with $l_u = l_x$, the new similarity value of u will be $s_u + \sum \Delta_u(ux)$.

In the calculation of $\eta_u(uv)$ and $\eta_v(uv)$, the nodes u and v only are allowed to play with a portion of their similarity but not with the whole—we call this limitation *parental allowance*. Parental allowance depends on the strength of the connections to the parents, the stronger the connection, the smaller the fraction of its similarity a node is allowed to give to a dummy node and vice versa. Without the parental allowance, a node highly similar to its parents may lose much of its similarity and may become very little similar to its parents. To calculate the parental allowance, if Π_u and Π_v are the sum of the weights of the edges connecting u and v to their parents, respectively, then the parental allowances of u and v will be $\pi_u = 1 - \Pi_u/(\Pi_u + \Pi_v)$ and $\pi_v = 1 - \Pi_v/(\Pi_u + \Pi_v)$. The similarity transferred to the dummy node by peer u is now:

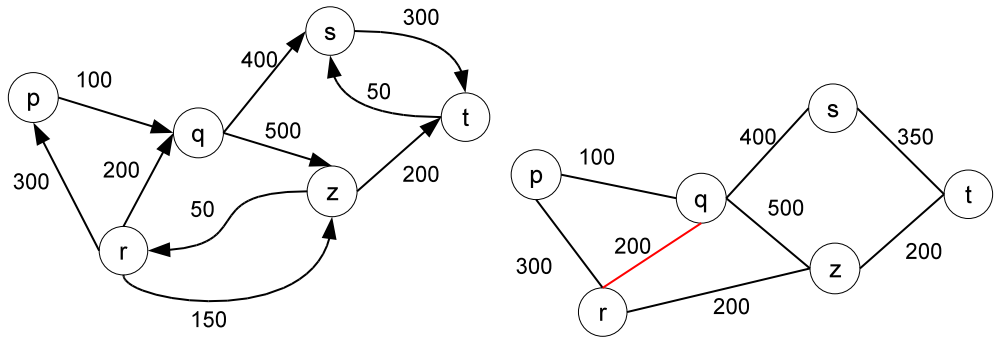
$$\eta_u(uv) = \theta^{l_u+1} \times \pi_u \times \rho_u \ , \tag{4.4}$$

where ρ_u is the ratio of $w_{uv}/2$ to the sum of all the edges connecting u to its children (including the dummy nodes).

The metaphor for this way of exchanging similarity is that because of the parental allowance, children that are strongly connected to their parents have less freedom in giving their similarity to others, and loosely connected children have more freedom, which

is natural. At the child level, the dummy node is treated like a lost child with its asset (similarity) equally divided between the parents.

As an example, now we will go through the process of similarity computation for a simple partial graph. Figure 4.2(a) shows the partial graph of p and Figure 4.2(b) shows the undirected graph U_p generated from it. In U_p, the nodes r and q are located at the same distance from the node p and they are connected, so in the graph S_p the edge rq will be replaced by a dummy node and two edges. Figure 4.3 shows the generated DAG



(a) Partial graph G_p (b) Undirected graph U_p

Figure 4.2: A sample partial graph and the associated undirected graph.

S_p along with the similarity of each node shown as a label beside it. For this example, $\theta = 0.8$, and the nodes r and s have the highest and lowest similarity to p, with $s_r = 0.54$ and $s_s = 0.07$.

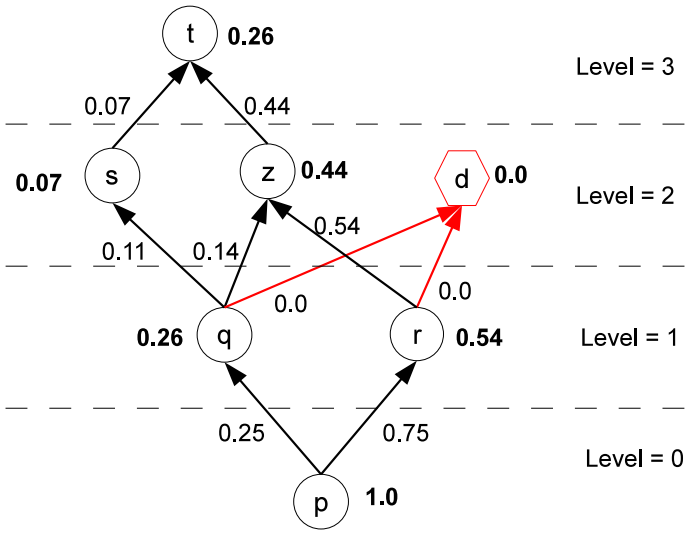


Figure 4.3: The similarity graph S_p for the example of Figure 4.2(a).

4.3.3 Random Walk Based Similarity

As a second method for computing similarity values we consider *random walks*, which have been used to compute similarity previously [98, 99]. In this method, starting from the node p we perform multiple biased random walks of length L in the partial graph G_p. These walks are non-uniform, and the choice of the next node from an arbitrary peer u is proportional to the weights of the outgoing edges of u. Besides, there is a *transport* factor $\alpha < 1.0$, which helps the walker to come back to the start point. When choosing the next node, with probability α the walker jumps to the start point p and continues the walk from there. After having performed a walk K times, the ratio of the number of times a node is visited to the total number of visits to all nodes ($K \times L$) is taken as the similarity of that node to p. As already mentioned in Section 4.3, an edge between two nodes is a sign of their interest in the same content, and the higher the edge weight, the higher the similarity between them. In a biased random walk, high-weight edges get a higher chance to be walked, and so the attaching nodes are visited more often than those connected weakly; accordingly, the hitting times of a node correlate with its similarity to the node who does the random walk. A good property of this method is that it its complexity only depends on K and L and is independent of the size of the partial graph.

Even though the RW-based method is not dynamic, in the face of changes in the partial graph we can use several heuristics to avoid recalculating the similarities for every such change and still have accurate similarity values. From the similarity point of view, the closer to the source node p, the more important are the edges. Until 2 hops the similarity update probability is the inverse of the distance from the local node p. So, if p does an upload or a download action, then the similarity values are recalculated, but if a neighbor of p does such an action then only with a probability of 0.5 the similarity calculation process is re-run. For the actions of other nodes, the similarities are only updated if the number of non-processed actions passes a threshold value $update_c$. It is possible that a peer may receive less than $update_c$ updates for a long period of time, so that the update trigger does not activate during this period. To mitigate this problem, we define a time-based similarity update trigger that after $update_t$ time of updating the partial graph (with at least one record) the similarity calculation process is re-run too.

4.3.4 Similarity Maintenance & Security

In both the DAG-based and the Random-Walk based methods, a peer p builds and maintains a similarity list of maximum size m with the top m most similar peers to itself, and in the selection of a target node for disseminating a record the peer p refers to this list.

When a peer p joins the network, its partial graph and similarity list are empty, but by doing its first upload or download it creates its first connections in its partial graph, and accordingly it gets new items in its similarity list. Later on, by receiving new records

it can update its similarity graph and similarity list. If the similarity list is full then the least similar peer is replaced by a fresher and higher similar peer. When a peer p receives a record, it first updates its partial graph G_p, then its similarity graph S_p, and finally its similarity list.

Regarding the security concerns, SimilDis carries security mechanisms against malicious acts like misreporting, sybil-attack, and white-washing. First of all, since the records are double-signed, there is no opportunity for misreporting. Second, the reputation calculation is done as in the sybil-resilient version of BarterCast [28], which is independent of how the records are disseminated. The only remaining concern is biasing the partial graph of a peer by a group of malicious peers, where they try to boost their own reputations at that peer. But this attack strategy is like the sybil-attack and the same sybil defense mechanism is effective here too.

4.4 Dynamic Similarity Update Algorithm for the DAG-based Method

In the dynamic network in which SimilDis is supposed to be executed, new peers may join the network or existing peers may perform data transfers, causing partial graphs to change. In turn, a change in the partial graph of a node may cause the similarity graph to change as well, and as a consequence, it may affect the similarity list. Creating the similarity graph from scratch for every change of the partial graph is not very efficient. In this section we will present a dynamic update algorithm for the similarity graph when the partial graph changes. In this algorithm, we use the natural *partial ordering* ($\leq$) property of a directed acyclic graph on its nodes, where $u \leq v$ if there is a path from u to v. Due to this property, a change in the similarity of a node u only affects the similarities of the nodes reachable from u. This property enables us to devise an incremental method for updating the similarity values.

Consider a node p, and its similarity graph $S_p = (V_s, E_s)$, where V_s and E_s are the node and edge sets, respectively. In SimilDis, we do not store the undirected graph U_p (see Section 4.3.2); it was only introduced to aid the explanation, In the real implementation we have used two adjacency lists to keep the graph structures, one for in-neighbors and one for out-neighbors, and the undirected weights are computed on the fly using the partial graph itself. Initially, the graph G_p is empty and S_p only contains the node p itself, which is called the *root* of DAG. Suppose that peer p wants to update its similarity graph S_p after it has updated its partial graph G_p with the newly received record (u, v, W), indicating that peer u has uploaded an amount W of data to v. Then there are four possible scenarios:

1. $u \notin V_s$ and $v \notin V_s$.

2. $u \in V_s$, $v \in V_s$, and $u \to v \in E_s$ or $v \to u \in E_s$ (both edges can not coexist).

3. $u \in V_s$ or $v \in V_s$, but not both.

4. $u \in V_s$, $v \in V_s$, but $u \to v \notin E_s$ and $v \to u \notin E_s$.

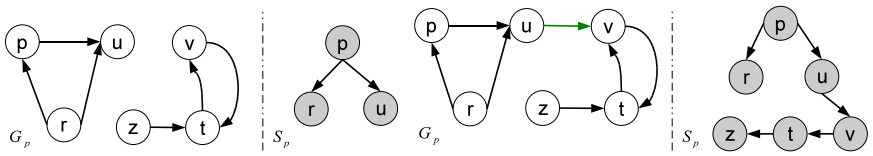
Note that the graph S_p is connected and that a node q belongs to it if and only if there is a path from p to q in G_p.

We now explain how the similarity graph is updated in each case.

Scenario 1): There is no path from p to u or to v in G_p, and u and v are not able to join S_p, so S_p does not change.

Scenario 2): Without loss of generality assume that the existing edge is $u \to v$. Due to the new data transfer between u and v, the weight of this edge is changed in E_s, and so the similarity of all the nodes reachable from u should be adapted as well. In this scenario, the graph structure S_p and nodes levels do not change. To update the similarity values, we start from the node u in S_p and using Eqs. (4.1) and (4.2), we recalculate the edge weights and the similarities of the children of u. This process continues throughout the complete subtree S_p of which u is the root.

Scenario 3): Without loss of generality assume that $u \in V_s$ and $v \notin V_s$ (the direction of the edges in G_p are irrelevant in the construction of S_p). This means that there exists a path from the root node p to u in G_p but not to v, Figure 4.4(a). In this scenario, the new edge $u \to v$ in S_p will act as a bridge that connects v and all nodes reachable from v in G_p to the similarity graph S_p. To modify S_p, before adding the edge $u \to v$ to S_p, our update algorithm treats the component of G_p that v belongs to as a standalone graph, and by starting from v it creates a sub-DAG for this component. In Figure 4.4(b), this sub-DAG is composed of the nodes v, t, and z. This new sub-DAG is then joined to the main similarity graph by the edge $u \to v$, and $l_v = l_u + 1$. After this join operation, the situation becomes like scenario 2, and starting from the node u the similarity values are updated accordingly.



(a) G_p and S_p before adding the edge $u \to v$.

(b) G_p and S_p after adding the edge $u \to v$.

Figure 4.4: An example for scenario 3, the edge $u \to v$ connects the node p to the new part of the graph.

Scenario 4): This is the most complex scenario and unlike the previous scenarios, the levels of nodes that are already present in S_p may change. As in scenario 3, first the

structure of the graph is adapted, then the similarity values for the changing nodes are recalculated. In this scenario, the current levels of the nodes u and v dictate how the graph is going to be restructured. There are three possibilities:

- $l_u - l_v = 0$. In this case the levels of u and v do not change, but to reflect the impact of the new edge on the similarity graph, using a dummy node d_{uv}, u and v amend their similarity according to Eq. (4.4). Then like in the scenario 2, starting from the nodes u and v, the similarity values of the nodes reachable from these nodes are updated, Figure 4.5 presents an example.

- $l_u - l_v = -1$. The node u is one hop closer to p than the node v, and the new edge does not change the graph structure nor the node levels. In this case only an edge is added from u to v in S_p, then like in scenario 2 starting from node u the similarity values are updated.

- $l_u - l_v < -1$. Like a domino effect, the new edge $u \rightarrow v$ causes level changes in the children and parents of v, and the changes ripple until the point that the levels of the nodes do not change any more. Unlike the previous cases, here the direction of an existing edge in S_p may change. The pseudocode of the graph rewiring algorithm for this scenario is presented as Algorithm 1. Here, the queue Q contains the nodes of which the levels are changed, and the node v with the new level $l_v = l_u + 1$ is its first item. The algorithm continues by removing an item from Q and processing it, until it becomes empty.

Algorithm 1 Scenario 4 & $l_u - l_v < -1$

```
1: Q ← {v}
2: while Q ≠ ∅ do
3:    x ← remove(Q)
4:    for m ∈ children(x) do
5:       if with the new edge the level of m changes then
6:          update l_m
7:          if it is required remove an existing dummy node
8:          if it is required add a new dummy node
9:          update the connecting edges to m
10:         add m to Q
11:      end if
12:   end for
13:   for n ∈ parents(x) do
14:      if with the new edge the level of n changes then
15:         update l_n
16:         if it is required change the direction from x to n
17:         if it is required add a dummy node, d_xn
18:         add n to Q
19:      end if
20:   end for
21: end while
```

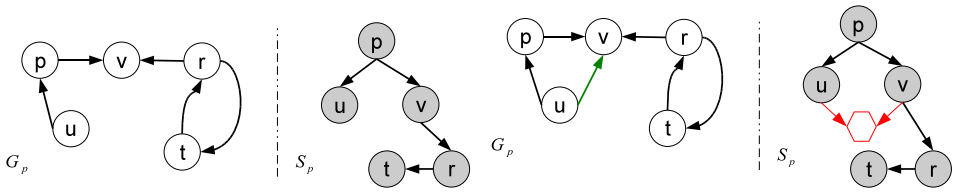
The conditions for adding/removing a dummy node, or changing the direction of an existing edge depends on the changes in the node levels. Similar to other scenarios, after rewiring S_p, by traversing it from node u, the edge weights and similarity values are updated.

Figure 4.6 shows an example, where due to the new edge $u \rightarrow v$, the similarity graph needs to be rewired. In this example, by applying Algorithm 1 the following changes happen:

- The levels of the parent and child of u (t and m) change.
- The parent-child relation between t and v is reversed.
- The nodes s and t get a dummy child.

For this example, only the similarities of the nodes r, v, m, t, s, n need to be recalculated.

The complexity of the dynamic update algorithm depends on the scenario. For scenario 1, since no graph traverse is done, the complexity is $O(1)$. For the other scenarios,



(a) G_p and S_p before adding the edge $u \to v$. (b) G_p and S_p after adding the edge $u \to v$.

Figure 4.5: An example for scenario 4 when the levels of u and v are equal.

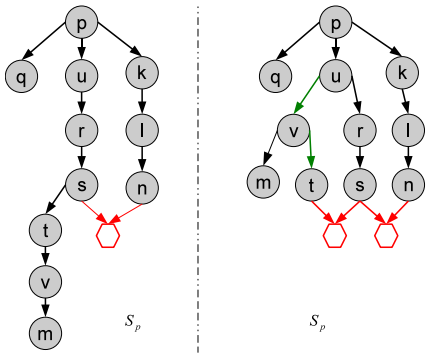


Figure 4.6: An example for scenario 4 when $l_u - l_v < -1$; similarity graph before (left), and after (right) adding the edge $u \to v$.

to update the similarity values, starting with the higher level node (see the scenarios), we traverse the sub-graph in breadth-first-search (BFS) manner, so if $|V'|$ and $|E'|$ are the numbers of nodes and edges in the traversed subgraph, then the complexity of the update process is $O(|V'| + |E'|)$. The size of a traversed subgraph depends on the structure and the growing pattern of the partial graph, and in the worst case it is equal to the DAG graph S_p.

4.5 Experimental Setup

We perform trace-driven simulations to evaluate our protocol. This section covers the simulation steps in detail.

4.5.1 SimilDis Simulation

Our experiments for evaluating SimilDis are based on a trace obtained from the Tribler network. Using a timed-ordered list of data transfer actions, we simulate the creation and dissemination of data transfer actions through SimilDis. In order to evaluate its accuracy, we compare the reputation values calculated using SimilDis with the case of having full knowledge (all records are given to all peers). The simulation is run in two phases: the *training* phase and the *testing* phase, and accordingly, the trace is split into two parts, one part for each phase. After processing 50% of the trace in the training phase, in which only dissemination is performed and the partial graphs are built up, in the testing phase peers are asked to evaluate the reputations of the peers they upload to.

In the Tribler network, a data transfer action is represented as a tuple (p, q, U, D, t), which indicates that until time t, peer p has uploaded an amount of data U to and downloaded an amount of data D from q. We sort the data transfer actions based on their dissemination time t in the network (the real data transfer time is unknown). Since the crawler may receive the same record from multiple peers, for our experiment we only keep the first occurrence of a record in the network. We filter out duplicate records, singleton nodes (nodes that are not connected to any other node), and the records in which U and D are less than 256 KB. In the final data trace that is fed to the simulator, each BarterCast record (p, q, U, D, t) is replaced by two separate records (p, q, U) and (q, p, D). Since it is not clear which of these actions has happened first, we just randomly put one before the other. Since multiple experiments with different ordering showed no meaningful effect on the outcome, we proceeded with a single random ordering of the records. Also, since the BarterCast records are processed by time and the simulator reads the trace sequentially, in the final trace the time t is irrelevant. After applying the mentioned filters we end up with a network of 11.7 K nodes and 28.1 K edges.

To simulate the record dissemination, we modify the PeerSim simulator [76], and implement SimilDis as a set of new modules over PeerSim. The simulation starts with an empty network, and by processing the trace, new nodes and edges are added to the network. Each peer keeps its own local partial graph, its similarity graph, and its similarity list, and when receiving a new record, it updates these structures accordingly. Here are the main steps that are done in each simulation cycle:

1. *Reading Trace*: In each simulation cycle, the simulator reads n_{rec} new records from the trace and injects them into the network. In our experiments n_{rec} is set to 20. To imitate reality, the peers who are involved in a data transfer action are the only receivers of a newly read record from the trace. In other words, if the simulator reads the record (p, q, U), then this record is only given to p and q. Later on, in the record sending step, they inform other peers about this record. The trace reading step is performed by the simulator, but the following steps are run by every peer in each cycle.

2. *Evaluating Reputation:* This step is done only during the testing phase, where for each record (p, q, U), the uploading peer p evaluates the reputation of q. The reputation evaluation is done before the update of the partial graph of p with (p, q, U).

3. *Updating Similarity List:* For each received record (r, s, U), the peer p first updates its partial graph G_p, then its similarity graph S_p (in the DAG-based method), and finally its similarity list.

4. *Sending and Receiving Records:* The real dissemination happens in this step, and peers actively involve in the spreading of the received records. Each peer has a buffer of size of l_{buf} which contains the candidate outgoing records. If the buffer is full then a newly received record replaces the oldest one. Also, each peer sends a record at most t_{rec} times, which is called the maximum *send-age* of a record. In each cycle, a peer forwards a maximum number of n_{msg} messages to a set of peers of size f_{out} (the *fan-out*) that consists of the top $|f_{out}|$ most similar peers to p as derived from p's similarity list.

4.5.2 Full-Dissemination Simulation

Because the views of the peers in SimilDis are only partial, the subjective reputation of a peer may differ from one evaluating peer to another. The ground truth for reputation values is obtained when peers have immediate access to all the past interactions in the network. The full graph is the ideal situation, and in effect it is like having a central server which collects all the records. The important point in informing peers about a new record is that when a record like $p \rightarrow q$ is generated, it is not clear whether in the future it

will be useful for an arbitrary peer r or not. If we would know, then the problem is already solved. In the ideal case a new record is given it everybody, and this action leads to the concept of full graph. In our previous work [27], we did experiments in using such a full graph and compared it with two other ways of improving the reputation accuracy (using a higher number of maxflow hops and computing reputation values from the perspective of the node with the highest betweenness centrality), and we observed that using the full graph is the most influential one.

To build such full knowledge, we create a special graph called G_{glob}, and after reading each record from the trace, this graph is updated with that record. The graph G_{glob} is used as the reference graph during reputation evaluations, and from a peer p's point of view, a peer q has two reputations, one obtained using G_p and the other using G_{glob}. The difference between these values shows the accuracy of SimilDis. In the real environment, the global graph is not kept by any peer, it is just used for our experiments to measure how far the nodes are from the ideal situation. Regarding the overhead of SimilDis, we compare the communication, the storage, and the computation cost against the cost of building and maintaining such a full graph by each peer. The comparison with the hypothetical scenario of Full-Dissemination measures the overheads at their extreme.

4.5.3 Parameter Setting

The SimilDis protocol has a number of parameters the values of which influence the protocol performance (see Table 4.1). In order to find an appropriate set of values for these parameters, we use *Dataset_1* (see Chapter 1.4). The filtered trace from this dataset that is fed into simulator contains 4.8 K edges and 2.7 K nodes. Using this trace, we have performed a sensitivity analysis, measuring the reputation error and costs for the combinations of parameters considered.

Since the total parameter space is too large and evaluating all combinations is impossible, we simplify the sensitivity analysis in two ways. First, we discretize continuous parameters and analyze only a subset of the feasible values. For example, for the transport factor α introduced in Section 4.3.3, we only evaluate the values $0.1, 0.2, \ldots, 0.8$. Secondly, for each parameter we perform a separate one-dimensional parameter analysis. In order to do so, we initialize each parameter to a value that gives the lowest cost (e.g., all gossiping-related parameters are set to 1), which may imply a very high error. After initializing the single changing parameter, we fix the others and do simulations with different values for the changing parameter and measure the reputation error. When the change in error between two consecutive experiments is less than the threshold value of 0.02, we fix the changing parameter and repeat this process for the next parameter. Table 4.1 contains the parameter values thus obtained that we use in the experiments in Section 4.6.

Table 4.1: Simulation parameter setting

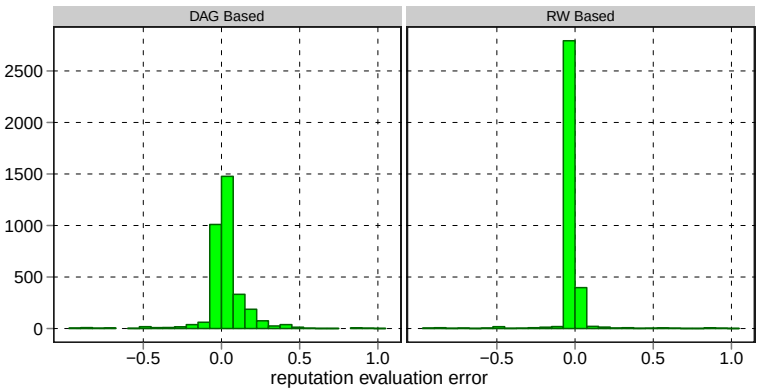
Parameter	Value
similarity list length	5
l_{buf} (size of message buffer)	10
t_{rec} (number of times to send a record)	2
f_{out} (fan-out of dissemination)	2
n_{msg} (maximum number of messages per cycle)	2
L (single random walk length)	5
K (random walk tries)	10
α (random walk transport factor)	0.4
$update_c$ (threshold for non-processed records)	20
$update_t$ (consecutive non-processed cycles)	20
θ (decay factor in the DAG-based method)	0.8

4.6 Evaluation

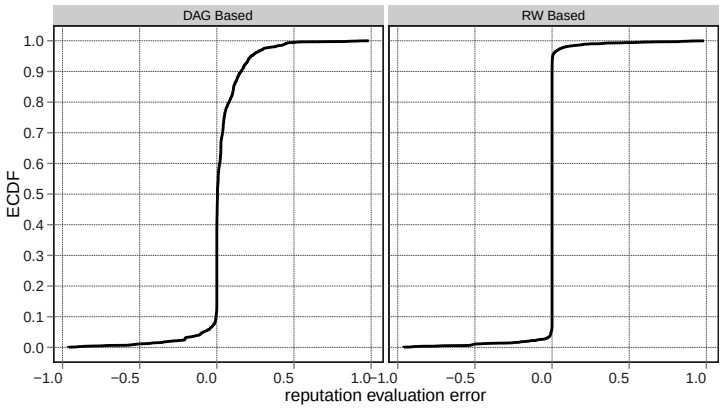
We evaluate our protocol from four angles, which are its accuracy in evaluating reputations, its communication, storage, and computation costs, the benefit of the dynamic similarity update for the DAG-based method, and its resilience in the face of churn. For these evaluations we use the crawled data from *Dataset_2*, see Chapter *sec:crawler.1*. Each result presented in this section is the average of 20 simulation runs.

4.6.1 Accuracy

Using the partial graph G_p and the global graph G_{glob} introduced in Section 4.5, when a record (p, q, U) is read from the trace, we compute the subjective and global reputations given to peer q by peer p, respectively, before updating the graphs G_p and G_{glob} with this new record. The difference between these two values shows how the restricted dissemination in SimilDis affects the reputations of peers. Figure 4.7 shows the histogram and the empirical cumulative distribution function (ECDF) of the reputation evaluation errors (subjective reputation minus global reputation) when using the DAG and Random Walk (RW) based similarity computation methods. This figure contains only the reputation evaluations for which the global reputation is non-zero, in other words, for which there is a meaningful reputation if we have full knowledge. In our previous work [27], we have shown that in terms of accuracy and computation overhead maxflow with 4 hops gives the best result, with a coverage of around 70% . As can be observed, for both methods the error values are concentrated around zero, and the standard deviation of the DAG and RW-based methods are 0.18 and 0.14, respectively. In comparison, as the ECDF plot shows, especially at the high ends, the RW-based method performs better than the DAG-



(a) Histogram(binwidth=0.065 and the intervals are left-open).



(b) ECDF (for the RW-based method the 10th and 90th percentiles are at -0.039 and 0.087, and for the DAG-based method at -0.007 and 0.201, respectively)

Figure 4.7: Histogram and Empirical CDF of the reputation evaluation errors in the DAG and RW-based methods.

based method. In general, the error values are biased toward the positive values and this positive bias may give opportunity for free-riders, but on the other hand, honest peers may also benefit from this positive bias, and they may not be rejected to get content from other peers.

4.6.2 Costs

To simulate having Full-Dissemination, we assume that there is an efficient peer discovery service that informs all peers when a peer joins the network. To mimic such a service in the simulator, when a record (p, q, U) is read from the trace, peer p is the peer who is responsible for informing all other peers and sends this record to all existing peers in the

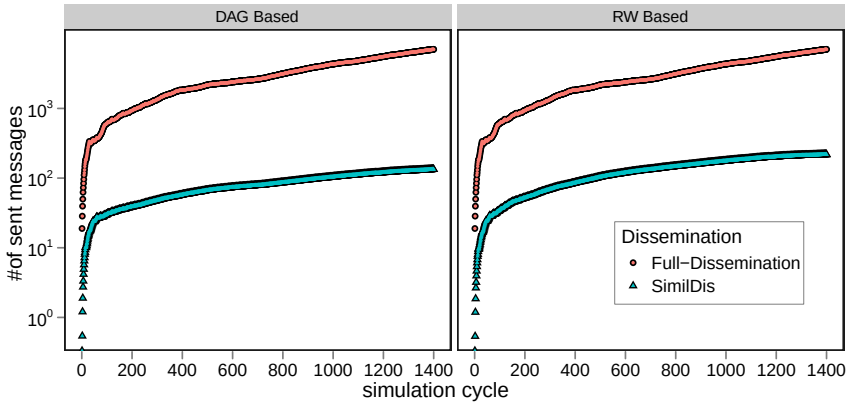


Figure 4.8: The communication cost in SimilDis vs. Full-Dissemination (the vertical axis is in log-scale).

network. When a node joins the network, all peers send their previous upload records to this peer as well. Due to the small record size, instead of sending one record in each TCP packet, peers can send multiple records in a single packet. In Tribler, the length of a BarterCast record is 48 bytes, and each TCP packet can carry around 30 records.

We evaluate the communication cost of SimilDis by counting the number of TCP packets and compare it with the case of providing peers with all records. Figure 4.8 shows the total number of messages sent in each simulation cycle. As can be seen, with both the DAG-based and the RW-based method, SimilDis sends two orders of magnitude fewer messages than in Full-Dissemination. The number of messages with RW is a bit higher than with DAG.

The two major computational costs incurred by SimilDis are the costs for *maxflow* and *similarity update*. In the case of Full-Dissemination, the maxflow computation is the only computational overhead. To compare these costs, we measure the CPU time for each of these algorithms, which are shown in Figures 4.9 and 4.10 as a function of the simulation cycle. Since the reputation evaluation happens only in the testing phase, the horizontal axis of Figure 4.9 does not start at zero. As can be observed, due to smaller partial graphs in SimilDis, the maxflow computation time in SimilDis is nearly 10 times shorter than in Full-Dissemination.

The graph in Figure 4.10 compares the similarity update time in the DAG-based versus the RW-based method. The RW-based method is around 10 times faster. Also, since the update algorithm in the DAG-based method depends on how the partial graphs grow, we observe more fluctuations in the DAG versus the RW-based method.

Finally, we consider the storage cost of SimilDis versus Full-Dissemination. We take the sizes of the partial graphs and the Full-Dissemination graph (in terms of the numbers

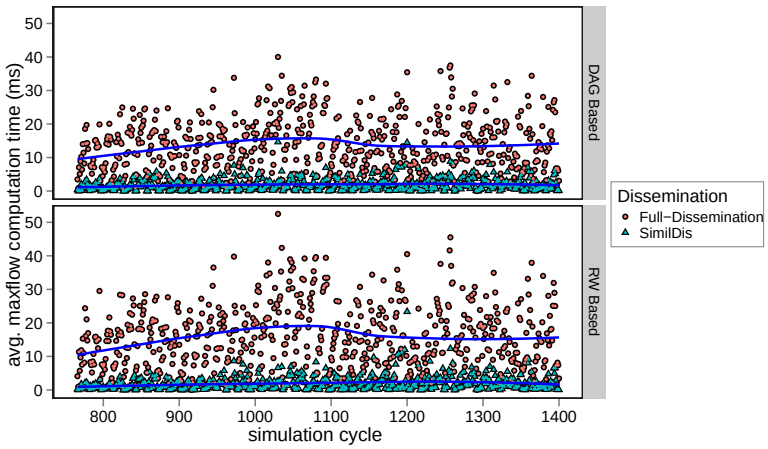


Figure 4.9: The computation cost of maxflow in SimilDis versus Full-Dissemination (the continuous lines show the averages).

of nodes and edges) as the protocol's storage cost. Similar to communication cost, in each simulation cycle we measure the sizes of the partial graphs in SimilDis and compare them with their size in Full-Dissemination, in which each peer has a full copy of the whole network. Figure 4.11 presents the average number of nodes and edges in the partial graphs of the peers. In comparison, in SimilDis the graphs are no less than approximately 100 times smaller than in Full-Dissemination.

4.6.3 Efficiency of Dynamic Similarity Update

To evaluate the efficiency of the dynamic similarity update algorithm, we rerun the experiments, but this time the similarity graph is updated in a *static* way. Here, by static we mean that for each change in the partial graph, the similarity graph is created from scratch. Even in this method, in order to speed up the similarity computation, we apply similar heuristics for when to do the similarity re-computation as we use with the RW-based method, see Section 4.3.3. Figure 4.12 shows the average dynamic and static update times. As the graph shows, even with using the same heuristics used for random walking, the static method is much slower than the dynamic one.

4.6.4 Accuracy Under Churn

To study the under churn behavior of SimilDis, we perform a set of experiments which covers different churn rates. In our experiments peers alternate between the *on* and *off* states. When a peer is *on*, it receives records and contributes to their dissemination, and in the *off* state, it is inactive and does not receive any records. In our simulation, when a peer

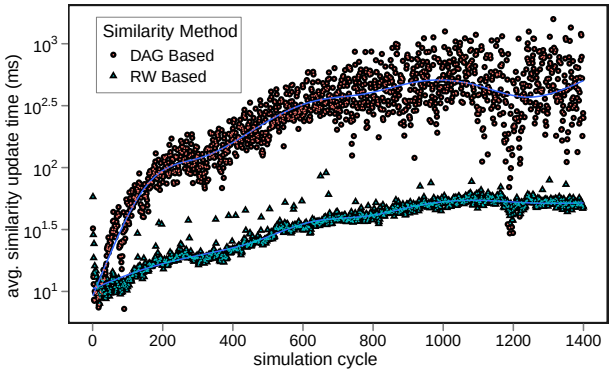


Figure 4.10: The similarity update cost in the DAG-based versus the Random Walk-based method (the vertical axis is in log-scale).

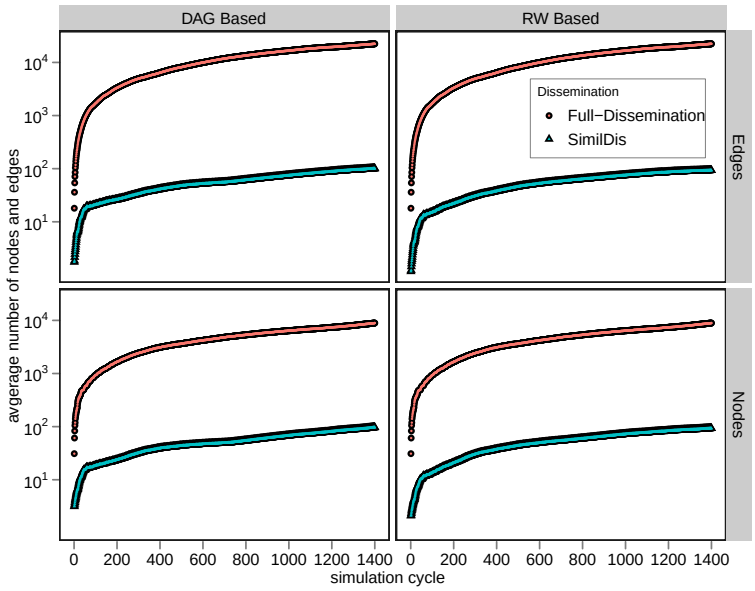


Figure 4.11: The storage cost in terms of the numbers of nodes and edges in SimilDis versus Full-Dissemination (the vertical axis is in log-scale).

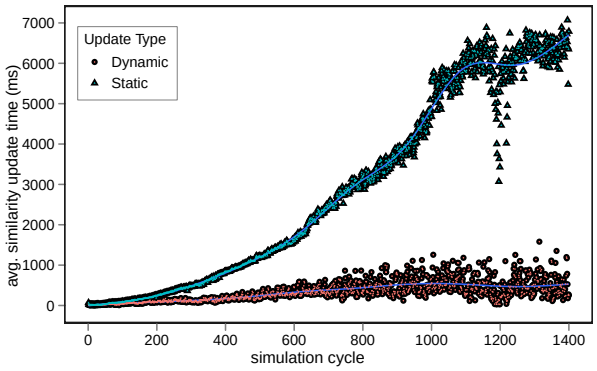


Figure 4.12: The runtime of the dynamic vs. the static similarity update algorithms in the DAG-based method.

becomes *on*, it is assigned an *on* period of a certain duration, and vice versa. However, if during an *off* period of a peer a record appears in the trace that contains the peer, then immediately it switches to the *on* state.

To study different churn intensities, we define the *online ratio* or as $\mu_{on}/(\mu_{on}+\mu_{off})$, where μ_{on} and μ_{off} are the average *on* and *off* time, respectively. We do experiments with $or = 0.1, 0.2, 0.4, 0.8$ and with $\mu_{on} = 1, 3, 5, 10, 20$. The values μ_{on} and μ_{off} are used as the means of normal distributions from which we generate the peer *on* and *off* times; the variance of these distribution is set to one-third of the mean. For each combination of or and μ_{on}, Figure 4.13 presents the average absolute reputation evaluation error (Section 4.6.1). As can be observed, even with the low online ratio of 0.1 the average reputation evaluation error is very low. The low error means that due to the targeted disseminated, even with a short online time, peers are still able to build partial graphs that lead to accurate reputation evaluations.

4.7 Conclusion

In this chapter we have introduced two methods for targeted dissemination of information in a distributed reputation mechanism, one based on building a Directed Acyclic Graph (DAG) and the other based on Random Walks (RW). The evaluation results show that both methods calculate reputations with low errors, with the RW method being slightly more accurate than the DAG-based method. In terms of communication, computation, and storage costs, both methods are dramatically more efficient than the Full-Dissemination method, in which the peers receive complete information—both the communication and the storage costs are reduced by a factor of 100. This is very important for power-constrained mobile devices. In general, the methods proposed in this chapter, which

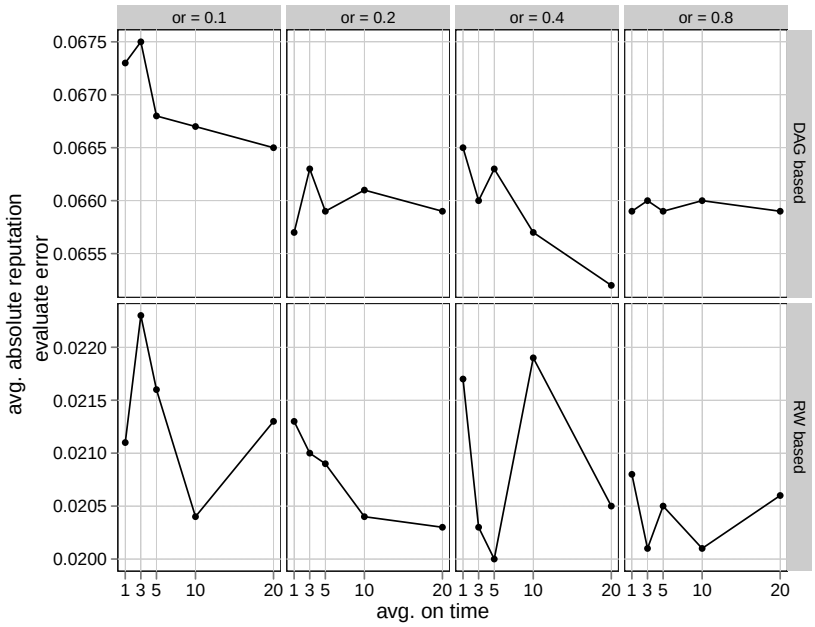


Figure 4.13: The accuracy of SimilDis under churn (the horizontal axis is μ_{on}).

aim at targeted information dissemination, are applicable in any application in which a graph among the system participants can be built. Moreover, the growth of online social networks has opened a new research area in leveraging social relations to improve security and performance of network applications [74, 106]. In such networks, for effective routing and improved security, the targeted dissemination of this chapter can be adapted and used as well.

In comparison to the DAG-based method, the RW-based method is non-deterministic but it is easier to implement. The DAG-method has the advantage that it assigns a similarity value to all nodes in the graph, and it can be used in any similar application in which SimRank or other structural similarity methods are used, such as targeted query forwarding.

Chapter 5

Analysis of the BarterCast Network

Despite much interest in reputation systems over the last few years, there are hardly any studies of the real behavior of Internet-deployed decentralized reputation systems. In this chapter, we study the BarterCast reputation mechanism, which builds a graph structure for its operation, from a *network science* perspective, and employ many network measures to understand its behavior. In this study, using a number of datasets from a real operational environment, we build the BarterCast network of content-sharing activities. From this network, we calculate a number of appropriate measures to comprehend its structure and the operational behavior of the underlying reputation mechanism. We interpret each calculated measure in the scope of the reputation mechanism and we provide an explanation of its implication. For some of the measures we elaborate on their prospective applications for further improving the reputation mechanism.

To rebuild the network and perform our analysis, we use *Dataset_3*, see Chapter 1.4. Using the permanent identifiers of the peers we are able to correctly group the collected records from different peers and to generate a global network, which we call the *work-graph*. Moreover, in the Tribler network, there are four so-called *super peer* nodes which are used for bootstrapping. We employ the data recorded by these super-peer nodes to generate two sets of valuable information about peers. The first of these sets contains the IP addresses of the peers, which enable us to do a geospatial analysis of the network. The second contains the content swarms peers have participated in, which allows us to perform a content-based similarity analysis of neighbor peers in the network.

We perform an analysis of the topological characteristics of the work-graph, which include the *degree distribution*, the *nodes interconnectivity*, the *clustering coefficient*, the *community structure*, and *centrality* and *distance* measures. The degree analysis of the work-graph shows that, like social network graphs, it has a long-tail power law distribution. We test the hypothesis of it being obeying a power law and check it against similar distributions. The interconnectivity analysis shows that the graph has star-shape structures and is far from a scale-free graph; this result is confirmed by the low clustering coefficient of the graph. Complementary to the clustering coefficient, it is observed that the graph

has strong communities. Moreover, we observe that there is a strong correlation between the node degree and the betweenness and closeness centrality measures. This observation suggests that node degree is a good approximation for these complex measures. Finally, using the temporal graphs that we build over time, we can observe how the diameter, the average path length, and the density of the graph change over time. Complementary to the general topological analysis, we also present results on the geographical spread of nodes at the granularity of ISPs, and we evaluate the correlation of ISP co-location and having an edge (content sharing) in the graph. We present our findings about user upload and download behavior and measure the content-based similarity of neighbor nodes in the graph. These similarities are based on the complementary data that we have about the content that peers have shared with others.

The contributions of this chapter are as follows:

1. Employing the crawled data from the Tribler network, we compute various network measures such as the degree distribution, the node interconnectivity, and the community structures (Section 5.2).

2. Merging the locality and the content data from the super-peer logs, we explore the colocation of the interacting peers in BarterCast, and we measure their similarities based on their content interested (Sections 5.3 and 5.4).

3. We interpret each metric in the context of file-sharing and reputation mechanism and elaborate on its prospective application. The interpretations are presented at the end of each section.

5.1 Related Work

For years, social scientists have studied human relations and have made interesting discoveries, e.g., the small-world phenomenon by Milgram [70] and the partitioning social relations into strong and weak ties by Granovetter [40]. Recently, due to the fast growth of online social networks, researchers have put a lot of effort in studying the static and dynamic properties of these networks. Kumar et al. [54] have studied friendship relations in Flickr and Yahoo360, and they have shown that these networks have a large Strongly Connected Component (SCC). An analysis and comparison of the social networks of Flicker, YouTube, LiveJournal, and Orkut by Mislove et al. [72] confirms the power law, small-world, and scale-free properties of these networks. Recently, the Facebook network, due to its high popularity and size, has attracted many researchers. Orthogonal to other studies, which are more focused on general network properties, Viswanath et al. [106] and Wilson et al. [101] have studied this network from the user-activity and link-reliability

perspectives. They have clarified that despite the high number of links (friendship relations), only a small portion of the links of a node are reliable, meaningful, and useful for real-life applications.

Besides social networks, analyzing the structure of the Web and Internet links has led to many interesting findings as well. Analysis of the *Autonomous System* (AS) level of the Internet by Mahadevan et al. [65] has revealed that the *joint degree distribution* can characterize Internet connections. Falatous et al. [34] have shown that the Internet topology follows a power law degree distribution; a claim that has raised criticism as well [105]. Besides the study of general network structures, there are some studies on the geographical properties of the Internet infrastructure [52, 96], which study the geographical spread and node distances. Regarding the Web network, a study of the Web links by Broder et al. [17] shows that links have a "bow-tie" shape, with a large SCC and many small groups of nodes connected in one-way to SCC.

Researchers have proposed wide applications of social networks in other systems, e.g., defense mechanisms [102], recommendation systems [53], reputation systems [51], and many others. Despite many proposals only a few of them have gone beyond design into a real application, and even for those who reached that level, there is no real large-scale study of their behavior. Our study is distinguishable from similar works in twofold; First, we perform a thorough analysis of a large-scale, deployed mechanism from a network perspective. Second, despite pure social network studies, which only present a number of general measures, we look at the calculated measures from the reputation mechanism perspective and provide valuable hints for future designs.

5.2 Topological Characteristics

In this section, we study the work-graph of BarterCast from the network topology perspective, and we present a number of relevant measures that help us to understand the connectivity pattern of the nodes. For ease of reading, the terms graph and network are used interchangeably.

5.2.1 The Undirected Work-graph

In the BarterCast work-graph, an edge indicates the amount of data transferred from one peer to another, but from the interaction perspective, its direction is not important. So, for the following analysis we remove the edge directions, and we add the weights if there are edges in both directions between two nodes. The original directed graph contains of 73,201 nodes and 352,042 edges; after removing directions, the number of edges is 283,973. In Section 5.2.3, we present some measures on the edge and weight symmetry, but unless stated otherwise, our analysis is based on the undirected work-graph.

Furthermore, since most of the graph measures, like the clustering coefficient, only make sense when the underlying graph is connected, we consider the *Largest Connected Component* (LCC) of the work-graph. In total there are 939 connected components, out of which 780 contain only two nodes. Figure 5.1 plots the cumulative percentage of the nodes covered by the largest 20 connected components ranked according to decreasing size. As the plot shows, 93.55% of the nodes belong to the LCC. The number of nodes and edges in the LCC are 68,315 and 265,033, respectively. In conclusion, since the LCC is a good representative of the whole graph, we base our analysis on the LCC unless stated otherwise.

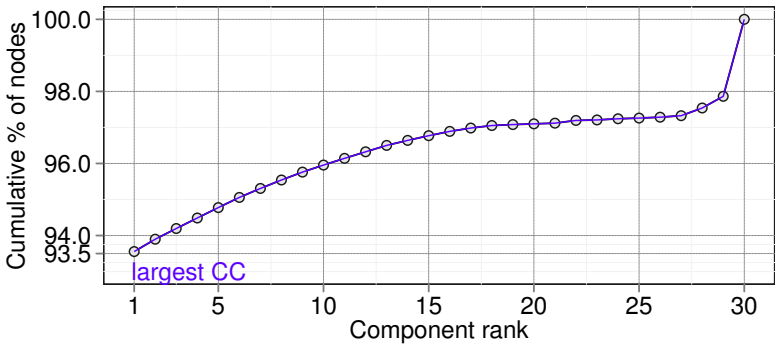


Figure 5.1: The cumulative coverage of the connected components ranked according to decreasing size.

5.2.2 Degree Distribution

The original work-graph is directed, and in such a graph a node has three types of degrees: the *in-degree*, the *out-degree*, and the *total degree*, which is the sum of in-degree and out-degree. Figure 5.2 shows the degree-frequency plot for these three types of degrees. Visually, after a threshold degree of about 30, the plot looks like a straight line. Based on this observation some researchers conclude that such a distribution follows a *power law*, and interpret the graph as a *preferential attachment* graph. But for two reasons this method is not a reliable way to conclude that a distribution follows a power law. First, due to high data (node degree) diversity, many values appear once and the frequency values are not informative. Secondly, using the frequency plot, non-power law driven data, e.g. exponential, can be misleadingly interpreted as power law [61]. Due to these limitations, using the CDF or the Complementary Cumulative Distribution Function (CCDF) is more common [8, 61, 65, 72].

Figure 5.3 shows the CCDF of the node degree in the LCC of the work-graph. In this plot, it looks as if the tail of the plot follows a power law distribution. A distribution is

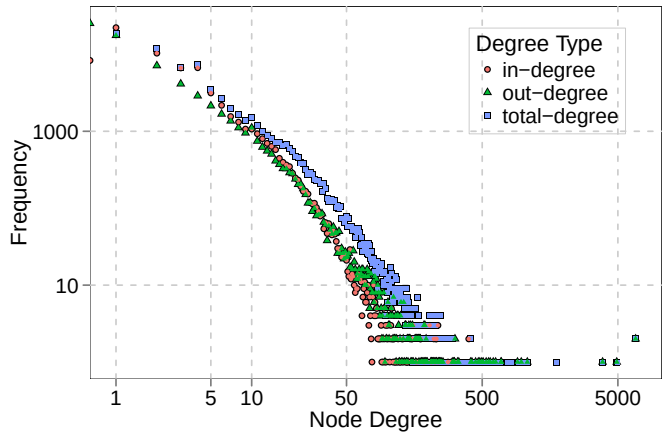


Figure 5.2: The degree frequency of the LCC of the work-graph (both axes are in log-scale).

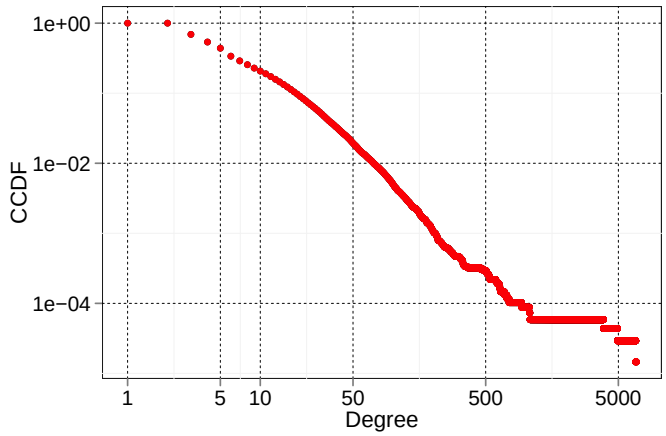


Figure 5.3: The complementary cumulative distribution function (CCDF) of the total degree in the LCC of the work-graph (both axes are in log-scale).

power law if it is driven from $p(x) \propto x^{-\alpha}$, where α is a fixed value called the *scaling* parameter. Using the method proposed by [21], we estimate the parameters α and x_{min}, where only for values larger than x_{min} the power law holds. To estimate the parameter α, first x_{min} is fixed at some value, and then using *maximum likelihood estimation* and assuming that the data are driven by a power law distribution, the value of α is estimated. To find x_{min}, starting from the lowest possible value, the Kolmogorov-Smirnov distance between the data and the estimated distribution (for the selected x_{min}) is calculated. The x_{min} that gives the lowest distance is chosen as the best value. Applying these methods we estimate $\alpha = 2.88$ and $x_{min} = 42$.

So far, we were able to fit a power law distribution to the degree values and to estimate its parameters, but whether it is a good fit or not is still a question. To evaluate the quality of the fit, using the estimated values for α and x_{min}, we calculate the *p-value* of the *goodness-of-fit* test for power law, and compare it with a threshold value. For the degree values the obtained p-value is 0.067. In the conservative approach [21], the p-value threshold for rejecting the power law hypothesis is set to 0.1, but generally in a more lenient approach it is set to 0.05. In conclusion, in the conservative approach the hypothesis that the work-graph has a power law degree distribution is rejected, but with the lenient approach this hypothesis is not rejected.

We now further analyze whether with the lenient appraoch, other types of distributions, e.g., the exponential distribution, give a better fit than the power law or not. The *likelihood ratio test* is a simple test for comparing the likelihoods of a dataset of belonging to a number of distributions. The sign of the logarithm of the ratio of two likelihoods, $\mathcal{R}$, can determine which distribution is a better representative for the given data. In practice, relying just on the sign of $\mathcal{R}$ is subject to random fluctuations around zero. To make a solid decision we use the method of Voung [103] that gives a p-value on the significant of the sign of $\mathcal{R}$, for small p-values the hypothesis that the sign of $\mathcal{R}$ is due to random fluctuations is rejected, and vice-versa. Table 5.1 presents the results of comparing the power law with four other distributions; a positive $\mathcal{R}$ indicates that the power law should be favored over the other distribution. As the table shows, the power law is reliably favored over the Poisson and Exponential distributions. For Yule, it seems that it is better than power law, but like the Powerlaw+cut off the sign of $\mathcal{R}$ is not reliable.

Table 5.1: Vuong log likelihood ratio test results

	Poisson	Log-normal	Exp.	Powerlaw+cut off	Yule
$\mathcal{R}$	+2.78	+0.008	+3.30	+9.03e-6	-0.97
p	**0.005**	0.993	**0.001**	1	0.330

Summary & Implication: Our analysis of the degree distribution of the work-graph shows that it has a long-tail distribution. Depending on the application behind the net-

work, having a high degree can have different reasons. For example, in a Web network, the popularity of a site can be the main reason for having many links to it. In our work-graph, having only a few very high-degree nodes means that a few peers are responsible for most of the content sharing in the network. Indeed, these are peers who stay online for a long time and are discovered by other peers more often. On the other hand, many low-degree nodes indicate the presence of many short-time users or even *free-riders*. A study of why there are so many short-time users will help to increase the quality of service in the whole network. Finally, a non-random structure means that the network is vulnerable to targeted strategic attacks on highly connected nodes. If an attacker provides the highly connected nodes with a contaminated content, then the content is spread very fast in the network. This is a concern that should be taken into account in future designs.

5.2.3 Node Interconnectivity

The degree distribution provides information on the individual connectivity of the nodes but it does not provide information on the relation between the degrees of neighboring nodes. In this section we provide some results on one-hop connectivity of the nodes as captured by the *Average Neighborhood Degree*, the *Assortativity*, and the *Rich Club Community* (RCC) metrics.

Consider the $k \times k$ Joint-Degree Distribiution (JDD) matrix $M = (m_{ij})$, where k is the largest node degree and m_{ij} is the number of edges that connect nodes with degree i to nodes with degree j. Dividing M by the total number of edges gives the probabilities that a randomly selected edge connects nodes with degrees i and j. For large and sparse graphs, the JDD matrix is highly sparse and not very informative. Instead, the average neighbor degree of the nodes of degree x, $k_{nn}k(x)$, is a more informative statistic for sparse graphs. An increasing $k_{nn}k(x)$ is an indication of the tendency of higher degree nodes to connect to other higher degree nodes and vice versa. We plot $k_{nn}k(x)$ in Figure 5.4, where due to its decreasing trend it seems that higher degree nodes tend to connect to lower degree nodes. A similar but more summarized metric than $k_{nn}k(x)$ is the *degree assortativity* of the graph, which takes values between -1 and +1, values close to +1 indicating the tendency of similar degree nodes to connect to each other and vice versa. For our graph, the degree assortativity is -0.062.

The last metric for evaluating the connectivity pattern of the nodes is *densely connected core* or *rich club community* [65]. A core is defined as a small group of well connected nodes that connect the remaining nodes. In order to understand the importance of the core nodes, we do a similar experiment as Mislove et al. [72]. In this experiment, we remove a number of the highest-degree nodes (a rich club) from the LCC and count the resulting number of disconnected components; the higher this number, the higher the importance of the removed nodes. Figure 5.5 presents the fragmentation results of removing

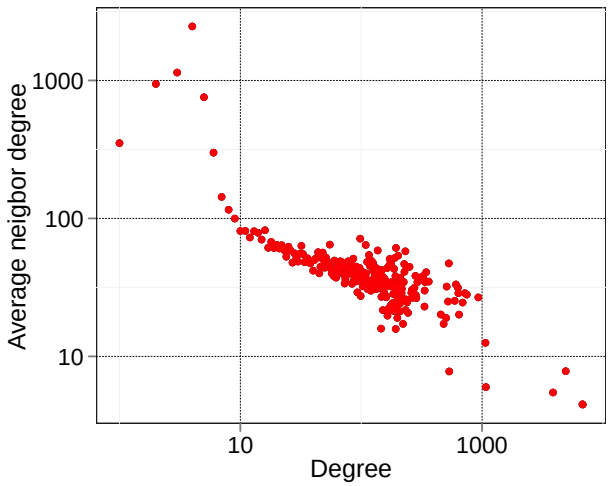


Figure 5.4: The average neighbor degree of the graph (both axes are in log-scale).

different fractions of the high-degree nodes; it shows the cumulative percentage of nodes included in the components ranked according to decreasing size, with components of the same size having the same rank (and counted multiple times in the coverage). Especially for small sizes, there may be multiple components. In this figure the right-most point of each curve represents the single-node components. As can be observed, for every removal ratio, almost all nodes either are part of the LCC or they become single-node components. Such a phenomenon occurs when there is a high number of star-shape structures with the removal of the central, high-degree node leaving many single-node components.

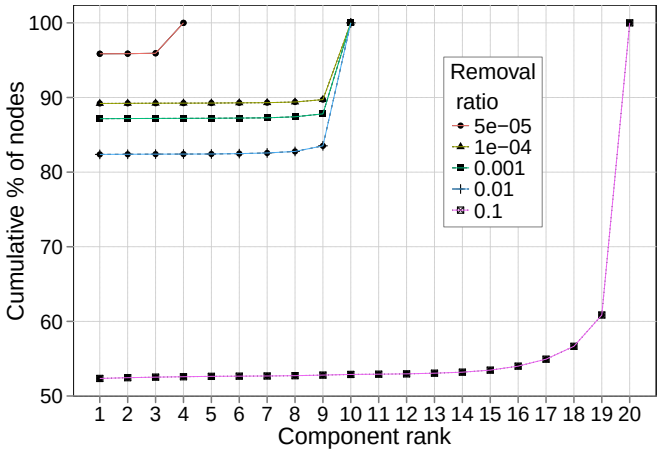


Figure 5.5: The rich club community removal effect.

Summary & Implication: The $k_{nn}k$ measure has a decreasing trend, which means that lower degree nodes and new comers tend to connect to nodes that have many links. This trend is similar to user connectivity in YouTube [72], which according to the authors is due to the "celebrity" effect, where popular users have many followers. A similar interpretation holds for the Tribler network as well, since users with many links have more content to share with others and they are more often discovered by those who look for content. Also, a decreasing $k_{nn}k$ means a low likelihood of having a *scale-free* graph [61]. This finding is confirmed by the degree assortativity which similarly to the degree assortativity of the Internet and Web networks is negative [78]. Notice that every scale-free graph is power law but not vice versa.

Finally, the RCC analysis shows a connectivity of the network that is very resilient against the removal of high-degree nodes. For example, in the extreme case of removing 10% of the highest-degree nodes, still more than half of the nodes remain in the LCC, which is in contrast to hub-like graphs where highly connected nodes play a critical role in connecting nodes. In conclusion, it seems that there are some strong community structures in the network, and many nodes are gathered around a few nodes.

5.2.4 Clustering and Communities

The degree distribution of a graph indicates the local connectivity of nodes, and the average neighbor degree $k_{nn}k$ gives information on the connectivity of similar degree nodes, but neither gives information on how the neighbors of a node are connected among themselves. In this section we provide results on the *local clustering coefficients* and the *global clustering coefficient*, which indicate whether the neighbors of nodes are tightly connected or not. Figure 5.6 shows the average clustering coefficients of nodes with the same degrees in the whole graph. The global clustering coefficient of the graph, which is the average of all clustering coefficients, is 0.0066.

Besides the clustering coefficient, we can look for communities in the graph. A *community* is simply a group of nodes with high internal and low external connectivities [80]. Depending on the application behind the network, there can be different reasons for the formation of communities, for example, geographical locality, similar taste, and etc. Since the number and the structure of communities are not known in advance, we need a way to evaluate the quality of a group of communities. Newman et al. [79] have introduced the concept of *modularity*, which quantifies the quality of partitioning a graph into communities. This measure is defined as:

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j), \tag{5.1}$$

where m is the number of edges, A is the adjacency matrix, k_i is the degree of node i, c_i is

the community id of i, and δ is the Kronecker delta function. For modularity values close to zero, the partitioning is meaningless, and values between 0.3 and 0.7 are reasonable for quality partitioning [20].

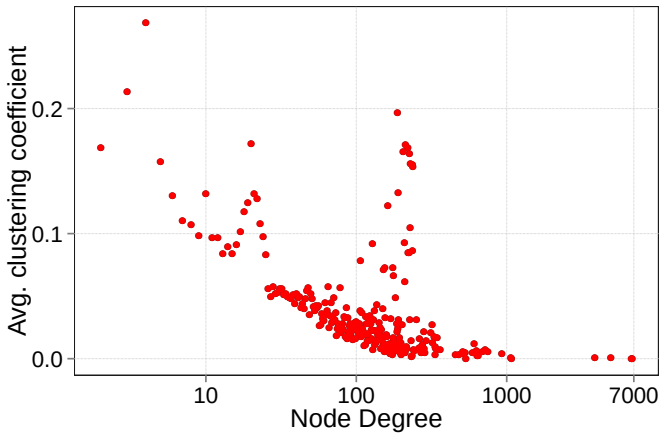


Figure 5.6: The average clustering coefficient vs. the node degree (horizontal axis is in log-scale).

Finding community structures is closely related to the notion of graph partitioning and hierarchal clustering, and there are numerous algorithms for detecting communities [36, 58]. The proposed algorithms mainly vary in their computational complexities, and only a few of them are appropriate for graphs of the size of our work-graph. Here we use four algorithms for detecting communities:

- *Fast Gready* (FG) [20], which is an efficient implementation of the hierarchal edge-betweenness community algorithm of Newman et al. [80].
- *Multi Level* (ML), which is based on local optimization of the modularity measure around a node [14].
- *Spin Glass* (SG), which is a simulated annealing heuristic method to optimize the modularity measure [88].
- *Label Propagation* (LP), which is a near-linear algorithm. First, it uniquely labels the nodes, then updates them by majority voting among the neighbors of a node [85].

Figure 7, presents the induced community graphs along with the number of communities and modularity values, obtained by applying the above algorithms. In these graphs, each node represents a community and an edge exists between two nodes if there is at

least one inter-community edge, i.e., an edge between nodes in the work-graph from either community. The size of the nodes and the width of the edges correlate with number of nodes in each community and the number of inter-community edges, respectively. As can be observed, the FG algorithm gives the highest modularity measure but the number of communities is too high. Considering both modularity and the number of communities, the ML method gives the best partitioning.

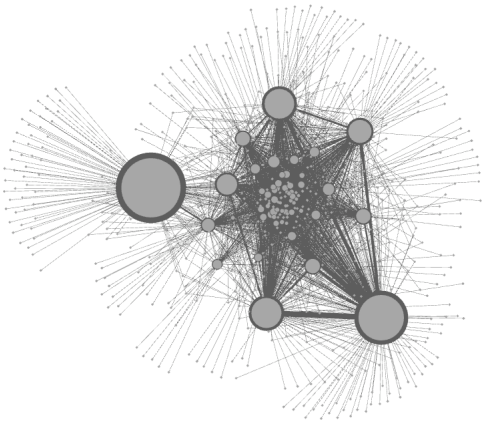
Summary & Implication: The decreasing trend of the clustering coefficient in Figure 5.6 confirms the previous finding of decreasing $k_{nn}k$, Section 5.2.3, that high-degree nodes play a crucial role in connecting many of their neighbors. Due to this phenomenon, the data exchange and presence of the high degree nodes is important for the operation of the system.

Regarding the community structures, the main reason for the formation of communities is that the work-graph grows in time, and the nodes that belong to a community are those nodes that were active in a specific period of time. When the time passes, most of the peers leave the network, but a few of them continue on sharing content with the new peers. In network terminology, these peers act as a bridge between one community to the next one(s), and since the number of such long term active peers is not high they are not strong enough to merge communities.

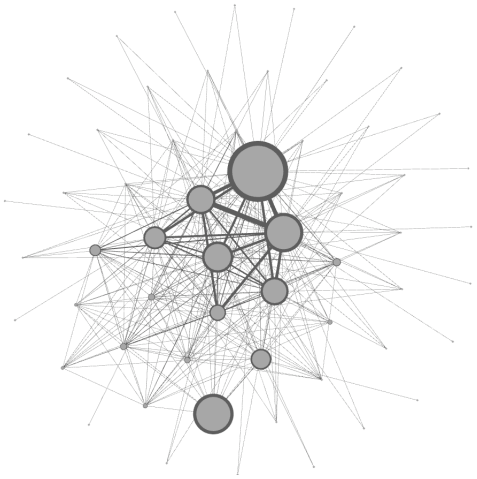
5.2.5 Distance Properties

We will now investigate the distance characteristics of *average path length* and *diameter* of the work-graph. Figure 5.8 presents the probability density of the lengths of shortest paths. It has a mean of 4.83 and 5.52 for the undirected and directed graphs, respectively. In order to understand how these measures change over time, we build temporal graphs based on edges appearing in the network and calculate these measures for them. A temporal study help us to understand how these measures change over time and whether the concept of *densification* [57] holds or not.

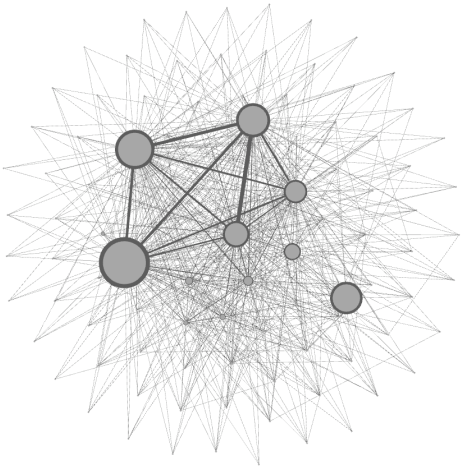
To measure the temporal features, starting from the first day of crawling we build temporal graphs in different periods, where the nodes and edges discovered in period n are added to the graph of period $n-1$. In our experiment, for the total crawling period of two years we divide the records in 52 sets of biweekly periods, and for each period we build a graph. Figure 5.9 presents the diameter, the average path length, and the density of the directed and undirected versions of the temporal work-graphs, which are plotted against the number of nodes. The density of a graph is the ratio of the number of edges to the number of possible edges. As can be observed, despite minor fluctuations, all these measures show a decreasing trend over the long term. Besides, the undirected version of each temporal graph has a lower diameter and average path length than the directed version, but such a difference is not visible for the density. The reason for this



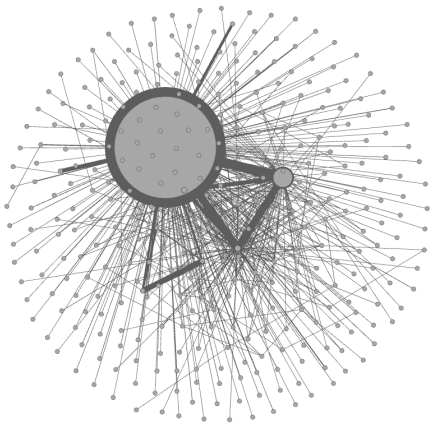
(a) Fast Greedy (modularity= 0.84, #communities= 600)



(b) Multi Level (modularity= 0.68, #communities= 58)



(c) Spin Glass (modularity= 0.59, #communities= 100)



(d) Label Propagation (modularity= 0.23, #communities= 335)

Figure 5.7: The community-induced graphs obtained through applying different community detection algorithms (nodes represent communities and edges indicate inter-community edges in the work-graph).

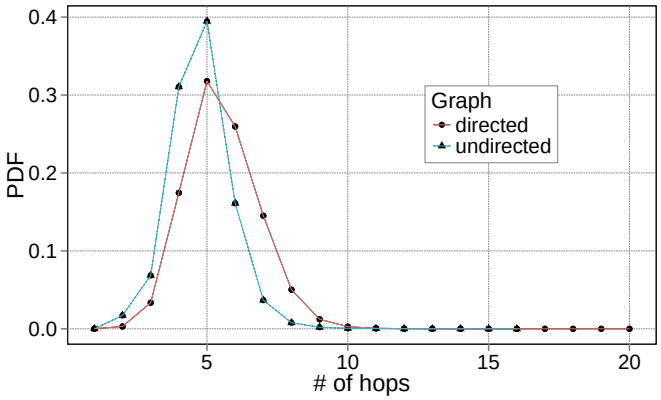


Figure 5.8: The density of the length of the shortest paths.

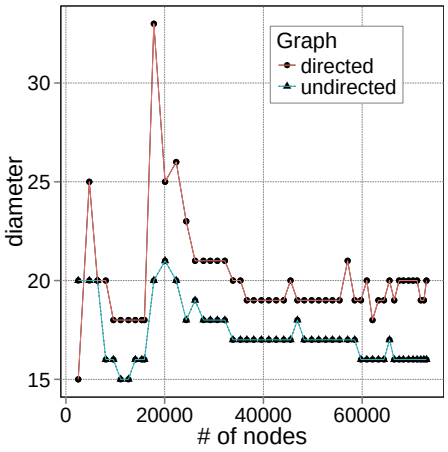
phenomenon is that there are relatively few double edges between a pair of nodes than the total number of edges, and when the direction is removed, the density is not much affected.

Summary & Implication: The average shortest path of the work-graph is very close to those of online social networks [8,72] and the AS-level Internet network [65]. Knowing the average path length is useful, since in the reputation evaluation process of BarterCast, the number of hops number in the Maxflow algorithm can be based on it. In this case, since in the final graph the directed average path length is 5.52, in the Maxflow algorithm the number of hops can be set to 5 or 6.

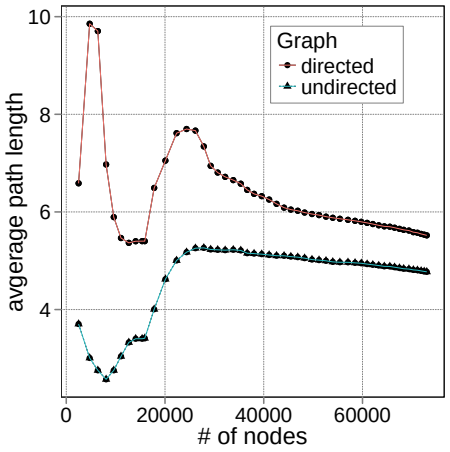
Like the graphs studies by Lescovec et al. [57], where they have discovered a *densification effect* in the studied graphs, we also observe a similar pattern in our graph. Figure 5.9(c), plots the number of nodes versus the number of edges in the temporal graphs in log-log scale. As can be observed, there is linear correlation and a linear regression fit shows a slope of 1.26. Since the slope is greater than 1, it means that the average degree of the graph increases over time. Moreover, like the graphs by Lescovec et al. [57], we observe that the diameter and the average path length have decreasing trends. In Figure 5.9(c), since the difference between the number of edges is not high we observer similar shapes

5.2.6 Betweenness

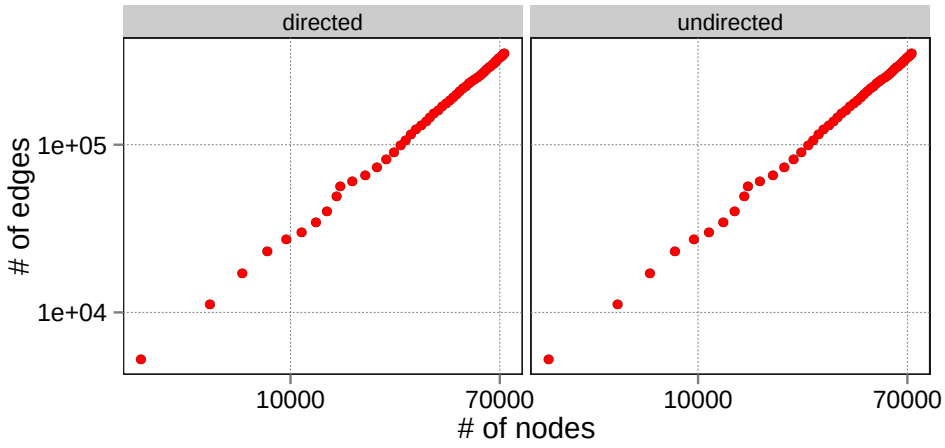
The last general topological measures that we evaluate are the *betweenness* and *closeness* centralities. The betweenness centrality of a node is the number of shortest paths between every pair of nodes that passes through the node, and it measures how important the node is in connecting other nodes. The closeness centrality is a measure of how a node is



(a) Diameter.



(b) Average path length.

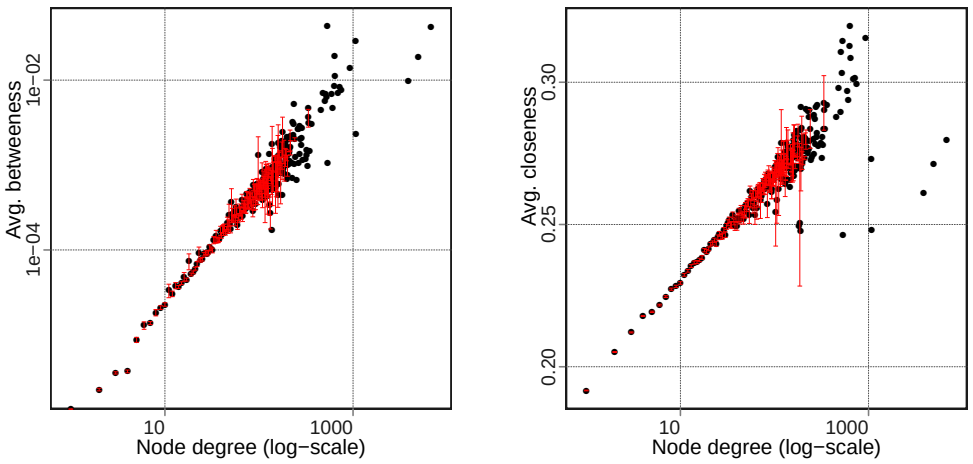


(c) Log-log plot of the number of nodes vs. number of edges.

Figure 5.9: The diameter (a), the average path length (b), and the number of edges vs. the number of nodes (c) in the temporal BarterCast graphs.

located closely to other nodes. Figure 5.10 plots the average normalized betweenness and closeness centrality measures against the node degree. As can be seen, there is a strong linear logarithmic correlation between the node degree and these centrality measures, which makes the node degree a viable approximation for the betweenness and closeness indices.

Summary & Implication: In our previous work [27], we have shown that using the node with the highest betweenness centrality as the start or end point in the Maxflow algorithm can improve the reputation accuracy. A problem associated with using this most central node is the high complexity ($O(|V||E|)$) for computing betweenness centrality in unweighted graphs [15]. Although there are approximation algorithms for this measure, but due to the high correlation between the node degree and betweenness centrality, during reputation evaluation process in BarterCast, we can easily use the highest-degree node instead of the most central node.



(a) Average betweenness centrality (both axes are in log-scale).

(b) Average closeness centrality (horizontal axis is in log-scale).

Figure 5.10: Average betweenness and closeness centrality vs. node degree.

5.3 Geographical Characteristics

In this section we consider the nodes in the work-graph from the perspectives of *Autonomous Region* (AR) and *Internet Service Provider* (ISP), and investigate the correlation of the AR and the ISP of neighbor nodes. An autonomous region is a country or a geographical region that according to the IP-to-location mapping is considered as an independent body, e.g., ”Virgin Islands of British”. To obtain the required locality data we

use the information collected by *super-peer* nodes in Tribler. When a Tribler client starts, it contacts one of the super-peer machines and gets a set of nodes to contact; the contacted super-peer logs the peer information. Using the super-peer logged data we are able to find the IP addresses of the peers and locate them at the granularity of ISP. For mapping IPs to locations, we use the IP-to-location service provided by MaxMind[1]. In total, we were able to detect nearly 75% of the peer locations. Using the location information of the peers, we compute the ratio of the data exchanges that happen inside the ARs or ISPs to the total traffic in the network, which are presented in Table 5.2. Besides, we determine the AR and ISP assortativity measures in the work-graph, which show a tendency of the peers to connect to peers in the same AR or ISP. For the work-graph, the AR and ISP assortativity values are 0.0085 and 0.039, respectively.

Table 5.2: Country and ISP level traffic information.

# AR	intra-AR traffic ratio	# ISPs	intra-ISP traffic ratio
184	0.30	459	0.20

Summary & Implication: Considering the traffic ratios presented in Table 5.2, it seems that there is a tendency toward having intra-AR and intra-ISP traffic. By further investigation we observed that in a few countries like the USA, the UK, and the Netherlands, the population of Tribler users is so high that statistically encountering a peer in these countries high enough to bias the ratio values. This argument applies to ISP ratio as well, and some huge ISPs cover many peers. Therefore, using the available traffic data we cannot confirm that there is a strong correlation between being in the same AR or ISP and doing a content exchange. Nevertheless, the positive AR and ISP assortativity values indicate a tendency of peers to connect to peers in the same AR or ISP, even though it is not strong.

5.4 Peer Behavior and Similarity

In this section, we complement our previous findings about the work-graph by analyzing it from the perspective of the activity of peers. Moreover, using our complementary dataset from the super-peers, we investigate the content-based similarity of neighbor peers in the graph.

In the work-graph the directions and the weights of the edges show the direction and the amount of the content sent and received by a peer. Consequently, the sum of the

[1] http://www.maxmind.com

weights of the outgoing edges of a node shows its contribution to the network, and dividing this value by the total amount of content sent and received gives the *sharing ratio* of the node. Figure 5.11 shows the CDF of the sharing-ratio values, which vary between the extreme situations of no uploading and only uploading.

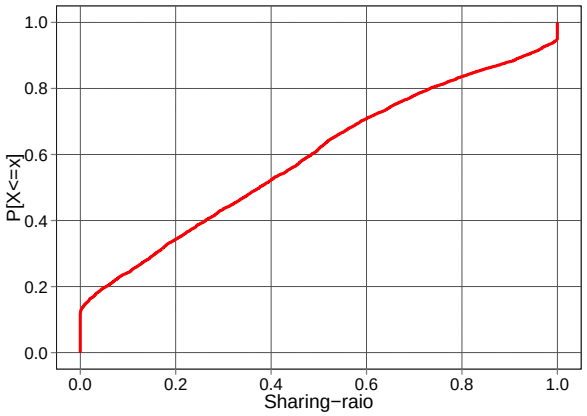


Figure 5.11: The CDF of the sharing ratio in the work-graph.

By using a complementary dataset about the activity of peers, which contains the BitTorrent swarms peers have participated in, we can derive the similarity between nodes in the work-graph. These data are logged by the super-peers. Like the locality information, we do not have the whole activity of every peer, but we are able to extract this information for nearly 80% of the nodes.

Based on the set of swarms of each peer and using the Cosine similarity method [16], we study the correlation between having an edge in the graph and participating in a common swarm. To investigate the relation between similarity and having a common edge, we do a number of experiments where we compare the similarity of neighbor nodes in the work-graph with the similarity of neighbor nodes in random isomorphs of this graph. Random isomorphs leave the structure of the work-graph untouched, and nodes have different, but the same number of, neighbors in each isomorph. Like for the original graph, for each isomorph we calculate the similarity of each node to its neighbors and average over each edge. Figure 5.12 presents the comparison of the CDFs of the similarity values in the original work-graph and the average similarity values obtained through 100 random isomorphs.

Summary & Implication: From Figure 5.11 we see that nearly 12% of the nodes are purely passive (sharing-ratio = 0) and do not perform any work for the system, and that nearly 4% are purely active (sharing-ratio = 1). The remaining nodes are divided nearly equally among passive (sharing-ratio < 0.5) and active peers (sharing-ratio > 0.5). Concerning the content-based similarity values, for nearly 80% of the edges the Cosine

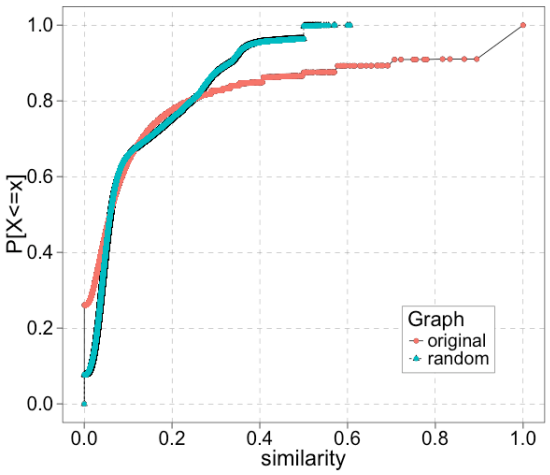


Figure 5.12: The empirical CDF of the similarity of neighbor nodes in the work-graph vs. the average similarities in the random isomorphs of the work-graph.

similarity is roughly equal in the original and the random graphs. For the remaining pairs of neighbors, it is observed that the neighbor similarity in the original graph is higher than the average similarity in the random isomorph graphs. This means that in Tribler, peers have a tendency to connect to similar peers.

5.5 Conclusion

In this chapter, through studying the BarterCast reputation mechanism from the network science perspective, we presented a number of useful insights. In relation to the reputation calculation process in BarterCast, we conclude that if peers apply the Maxflow algorithm with 5 or 6 hops, they can reach a significant portion of the nodes in the network. Besides, instead of using the node with the highest betweenness centrality value [27], which is expensive to compute, peers can use the highest-degree node as a replacement, as these metrics tend to be closely related.

Concerning the structure of the network, our measures show that it has a power law degree distribution, a relatively low diameter, and a strong community structure. Besides, the temporal study of the BarterCast graphs shows that the diameter and the average path length are decreasing, but we see some kind of densification effect, where the relative growth of edges is higher than nodes. Moreover, we observe that in the network there is a positive tendency toward interaction among peers with similar tastes.

Chapter 6

Conclusion and Future Work

In this thesis we have investigated various operational aspects of distributed reputation mechanisms for P2P file-sharing systems. Through employing the BarterCast reputation mechanism [68], as a deployed mechanism in this field, we built a robust mechanism for such applications. We have done do by improving BarterCast in the three important aspects of reputation accuracy, security, and scalability. Moreover, by using a dataset collected by crawling the Tribler BitTorrent network for two years, we have performed a thorough network-based analysis of this mechanism and we have obtained a number of insights for further improving. In this chapter, we present our conclusions that answer the research questions that was have raised in the introduction of this thesis, and we propose suggestions for future work for interested researchers in this field.

6.1 Conclusion

Based on the research conducted in this thesis we can state the following conclusions:

1. Considering the network-based structure of the BarterCast mechanism, we identify three improvements to boost the accuracy and coverage of this mechanism. The proposed modifications are increasing the number of hops in the Maxflow algorithm [24], employing the node with the highest betweenness centrality in the Maxflow algorithm, and increasing the volume of information at each peer through full gossiping. Our analysis shows that the reputation improvement gained through the first two improvements are significant if they are combined with full gossiping (Chapter 2). Applying the proposed improvements leads to the problems of low scalability of full gossiping and the high complexity of calculating the betweenness centrality. We address these issues in Chapters 4 and 5, respectively.

2. The BarterCast mechanism was designed with the assumption that everybody follows the protocol and there is no adversary to subvert it. In a distributed and open

environment these assumptions are unrealistic and countermeasures are inevitable. We identified three possible types of attacks in BarterCast: sybil attacks, whitewashing, and miss-reporting. To withstand the sybil attack, we introduced a new accounting mechanism, where each upload and download action the weights of the edges involved are adjusted and the attacker loses her reputation fast. For whitewashing, we modified the reputation so that it leaves no point for performing whitewashing. Finally, instead of sending plain BarterCast records, the peers who are involved in the data transfer disseminate double-signed records, so no one can disseminate false records. We use the crawled data from the Tribler network to evaluate the proposed solutions, especially the sybil attack, and the obtained results show that by employing the new protocol, the reputations of the sybil attackers diminish quickly (Chapter 3).

3. In Chapter 2, we showed the importance about providing peers with complete information of uploads and downloads in the network. In practice, such a provision is costly and maybe unnecessary. A solution for this problem is to provide peers only with the information that they need. To realize this, we devised a similarity-based mechanism in which during gossiping, similar peers are prioritized to receive the BarterCast records. The similarity measure is based only on the structure of the partial graphs of the peers. We employ two methods to calculate the similarity, one is based on building and employing a directed-acyclic-graph (DAG), and the other one is based on nonuniform random-walk. We use crawled data from the Tribler network to assess both methods. Our results show that they both perform very well, with the random-walk based method being slightly better than the DAG-based method (Chapter 4).

4. In Chapters 2, 3, and 4 we focused on improving the BarterCast mechanism from the accuracy, the security, and the scalability angles. To complete our study, we do a thorough study of this mechanism from the perspective of network science and investigate the crawled network from various aspects. Our study reveals that the BarterCast graph has a long-tail degree distribution that resembles that of power law graphs. We found that this graph contains many star-shape structures where some of the nodes play a critical role in connecting others. Besides, we find out that the BarterCast graph has strong community structures. Further more, by employing the locality and content information we measured the correlation between having an edge in the BarterCast graph and being in the same ISP or geographical region. The results show only a weak correlation. We did a similar study on the correlation between sharing the same content and being connected by an edge in the graph. The results indicate a positive tendency of peers to interact with similar peers (Chapter 5).

6.2 Suggestions For Future Work

In this thesis we focused on the design of a robust reputation mechanism for P2P file-sharing applications. The findings in this thesis can be extended and applied in similar applications as well. Below we highlight some of the prospective directions for future research.

1. In Chapter 2, we introduced the use of betweenness centrality for the reputation evaluation, but regarding this proposition there are some open questions. First, for what type of graphs does this modification give better results, and why? Secondly, what happens if an attacker tries to control the node(s) with the highest betweenness centrality values, and how many nodes does he need to control in order to achieve this goal? This is a broad concern for every mechanism that relies on this centrality measure.

2. In Chapter 3, we introduced a sybil-resistant version of the BarterCast mechanism. In fact, our proposed method can be generalized and applied in similar applications which are based on the Maxflow algorithm. Especially the edge level accounting part of the method can be easily modified for other applications as well.

3. To make the BarterCast mechanism scalable, we introduced similarity-based information dissemination, which is indeed a way of doing personalized gossiping. The proposed method can be applied in any similar protocol which is based on gossiping and in which the only information about the participants selects to their interactions with each other. In particular, we introduced the DAG-based similarity method. Comparing this method with other structural similarity methods like SimRank [46] is a direction that can be taken by interested researchers.

Bibliography

[1] Bittorrent protocol, http://www.bittorrent.org, February 2013.

[2] Eztv, http://eztv.it, February 2013.

[3] Tvtorrent, http://www.tvtorrents.com, February 2013.

[4] K. Aberer. P-grid: A self-organizing access structure for p2p information systems. In *Journal of Cooperative Information Systems*, volume 2172, pages 179–194. Springer, 2001.

[5] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *ACM International Conference on Information and Knowledge Management (CIKM)*, 2001.

[6] E. Adar and B.A. Huberman. Free riding on gnutella. *In First Monday*, 5(10-2), 2000.

[7] G. Adomavicius and A. Tuzhilin. 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*, 17(6):734–749, 2005.

[8] Y.Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of topological characteristics of huge online social networking services. In *ACM International Conference on World Wide Web (WWW)*, 2007.

[9] A. Allavena, A. Demers, and J.E. Hopcroft. Correctness of a gossip based membership protocol. In *ACM symposium on Principles of Distributed Computing (PODC)*, 2005.

[10] K.G. Anagnostakis and M.B. Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2004.

[11] I. Antonellis, H.G. Molina, and C.C. Chang. Simrank++: Query rewriting through link analysis of the click graph. In *Very Large Data Base Endowment (VLDB)*, 2008.

[12] R. Aringhieri, E. Damiani, D. Vimercati, S. De Capitani, S. Paraboschi, and P. Samarati. Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems. *Journal of the American Society for Information Science and Technology*, 57(4):528–537, 2006.

[13] M. Barthélemy. Betweenness centrality in large complex networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):163–168, 2004.

[14] V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:10008–12, 2008.

[15] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[16] J.S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 1998.

[17] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Journal of Computer Networks*, 33(1):309–320, 2000.

[18] S. Buchegger and J.Y. Le Boudec. A robust reputation system for mobile ad-hoc networks. *ACM Workshop on the Economics of Peer-to-Peer Systems (P2PECON)*, 2004.

[19] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In *ACM Workshop on the Economics of Peer-to-Peer Systems (P2PECON)*, 2005.

[20] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111–6, 2004.

[21] A. Clauset, C. Rohilla Shalizi, and M.E.J. Newman. Power-law distributions in empirical data. *Society for Industrial and Applied Mathematics Review*, 51:661–703, 2009.

[22] B. Cohen. Incentives build robustness in bittorrent. In *ACM Workshop on the Economics of Peer-to-Peer Systems (P2PECON)*, 2003.

[23] G. Conforti, G. Ghelli, P. Manghi, and C. Sartiani. Scalable query dissemination in xpeer. In *IEEE Database Engineering and Applications Symposium (IDEAS)*, 2007.

[24] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, second edition, 2001.

[25] E. Damiani, D.C. Di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *ACM Conference on Computer and Communications Security (CCS)*, 2002.

[26] G. Danezis and P. Mittal. Sybilinfer: Detecting sybil nodes using social networks. *Annual Network and Distributed System Security Symposium (NDSS)*, 2009.

[27] R. Delaviz, N. Andrade, and J.A. Pouwelse. Improving Accuracy and Coverage in an Internet-Deployed Reputation Mechanism. In *IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2010.

[28] R. Delaviz, N. Andrade, J.A. Pouwelse, and D.H.J Epema. SybilRes: A sybil-resilient flow-based distributed reputation mechanism. In *International Conference on Distributed Computing Systems (ICDCS)*, 2012.

[29] R. Delaviz, J.A. Pouwelse, and D.H.J. Epema. Targeted and scalable information dissemination in a distributed reputation mechanism. In *ACM Workshop on Scalable Trusted Computing (STC)*, 2012.

[30] R. Delaviz, N. Zeilemaker, J.A. Pouwelse, and D.H.J. Epema. A network science perspective of a distributed reputation mechanism. In *IFIP Networking Conference*, 2013.

[31] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *ACM Symposium on Principles of Distributed Computing (PODC)*, 1987.

[32] T. Dimitriou, G. Karame, and I. Christou. Supertrust–a secure and efficient framework for handling trust in super peer networks. *International Conference on Distributed Computing and Networking (ICDCN)*, 2008.

[33] P.T. Eugster, R. Guerraoui, A.M. Kermarrec, and L. Massoulié. Epidemic information dissemination in distributed systems. *Computer*, 37(5):60–67, 2004.

[34] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *ACM SIGCOMM Computer Communication Review (CCR)*, 1999.

[35] R. Farmer and B. Glass. *Building web reputation systems*. Yahoo Press, 2010.

[36] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[37] L.C. Freeman. A set of measures of centrality based on betweenness. *Journal of Sociometry*, 40(1):35–41, 1977.

[38] F. Garcia and J.H. Hoepman. Off-line karma: A decentralized currency for peer-to-peer and grid applications. In *International Conference on Applied Cryptography and Network Security (ACNS)*, 2005.

[39] F. Gómez Mármol and G. Martínez Pérez. Towards pre-standardization of trust and reputation models for distributed and heterogeneous systems. *Computer Standards & Interfaces*, 32(4):185–196, 2010.

[40] M.S. Granovetter. The strength of weak ties. *American Journal of Sociology*, pages 1360–1380, 1973.

[41] I. Gupta, A.M. Kermarrec, and A.J. Ganesh. Efficient epidemic-style protocols for reliable and scalable multicast. In *IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2002.

[42] M.J. Ham and G. Agha. Ara: A robust audit to prevent free-riding in p2p networks. In *IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2005.

[43] D. Hausheer, N.C. Liebau, A. Mauthe, R. Steinmetz, and B. Stiller. Token-based accounting and distributed pricing to introduce market mechanisms in a peer-to-peer file sharing scenario. In *IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2003.

[44] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys*, 42(1):1–31, 2009.

[45] D. Houser and J. Wooders. Reputation in auctions: Theory, and evidence from ebay. *Journal of Economics and Management Strategy*, 15(2):353–369, 2006.

[46] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *ACM International Conference on Knowledge Discovery in Data Mining (SIGKDD)*, 2008.

[47] M. Jelasity and O. Babaoglu. T-man: Gossip-based overlay topology management. *Engineering Self-Organising Systems (ESOA)*, 2006.

[48] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *Journal of Transactions on Computer Systems*, 23(3):219–252, 2005.

[49] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Journal of Decision Support Systems*, 43(2):618–644, 2007.

[50] P.M. Jost and H.S. Sandhu. *The hawala alternative remittance system and its role in money laundering*. Interpol, 2003.

[51] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *ACM International Conference on World Wide Web (WWW)*, 2003.

[52] S.P. Kasiviswanathan, S. Eidenbenz, and G. Yan. Geography-based analysis of the internet infrastructure. In *IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

[53] I. Konstas, V. Stathopoulos, and J.M. Jose. On social networks and collaborative recommendation. In *ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, 2009.

[54] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Link Mining: Models, Algorithms, and Applications*, pages 337–357. Springer, 2010.

[55] R. Landa, D. Griffin, R.G. Clegg, E. Mykoniati, and M. Rio. A sybilproof indirect reciprocity mechanism for peer-to-peer networks. In *IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[56] E.A. Leicht, P. Holme, and M.E.J. Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120–10, 2006.

[57] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *ACM International Conference on Knowledge Discovery in Data Mining (SIGKDD)*, 2005.

[58] J. Leskovec, K.J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *ACM International Conference on World Wide Web (WWW)*, 2010.

[59] R. Levien. Attack-resistant trust metrics. In *Computing with Social Trust*, pages 121–132. Springer, 2009.

[60] Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, and Tianyi Wu. Fast computation of simrank for static and dynamic information networks. In *ACM International Conference on Extending Database Technology (EDBT)*, 2010.

[61] L. Li, D. Alderson, J.C. Doyle, and W. Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4):431–523, 2005.

[62] M. Li, W.C. Lee, and A. Sivasubramaniam. Semantic small world: An overlay network for peer-to-peer search. In *IEEE International Conference on Network Protocols (ICNP)*, 2004.

[63] K. Lin, H. Lu, T. Yu, and C. Tai. A reputation and trust management broker framework for web applications. In *IEEE International Conference on e-Technology, e-Commerce and e-Service (CEC/EEE)*, 2005.

[64] F. Lorrain and H.C. White. Structural equivalence of individuals in social networks. *The Journal of Mathematical Sociology*, 1(1):49–80, 1971.

[65] P. Mahadevan, D. Krioukov, M. Fomenkov, B. Huffaker, X. Dimitropoulos, k. claffy, and A. Vahdat. The Internet AS-Level Topology: Three Data Sources and One Definitive Metric. *ACM SIGCOMM Computer Communication Review (CCR)*, 36:17–26, 2006.

[66] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50(4):472–484, 2006.

[67] M. Meulpolder. *Managing Supply and Demand of Bandwidth in Peer-to-Peer Communities*. PhD thesis, Delft University of Technology, The Netherlands, 2011.

[68] M. Meulpolder, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips. BarterCast: A practical approach to prevent lazy freeriding in P2P networks. In *International Workshop on Hot Topics in P2P Systems (HOTP2P)*, 2009.

[69] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation. In *IFIP Communication and Multimedia Security (CMS)*, 2002.

[70] S. Milgram. The small world problem. *Psychology Today*, 78:1360–1380, 1967.

[71] M. Minoux. Flots equilibrés et flots avec sécurite. *EDF-Bulletin de la Direction des Études et Recherches Série C-Mathématiques. Informatique*, 1976.

[72] A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Internet Measurement (IMC)*, 2007.

[73] A. Mislove, A. Post, P. Druschel, and K.P. Gummadi. Ostra: Leveraging trust to thwart unwanted communication. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

[74] A. Mohaisen, N. Hopper, and Y. Kim. Keep your friends close: Incorporating trust into social network-based sybil defenses. In *IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

[75] J.J.D. Mol, J.A. Pouwelse, M. Meulpolder, D.H.J. Epema, and H.J. Sips. Give-to-get: An algorithm for p2p video-on-demand. In *Multimedia Computing and Networking (MMCN)*, 2008.

[76] A. Montresor and M. Jelasity. Peersim: A scalable p2p simulator. In *IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2009.

[77] A. Nandi, T. Ngan, A. Singh, P. Druschel, and D. Wallach. Scrivener: Providing incentives in cooperative content distribution systems. *International Middleware Conference*, 2005.

[78] M.E.J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, 2003.

[79] M.E.J. Newman. *Networks: an introduction*. Oxford University Press, Inc., 2010.

[80] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[81] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson. One hop reputations for peer to peer file sharing workloads. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

[82] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M. Van Steen, and H.J. Sips. Tribler: A social-based peer-to-peer system. *Concurrency and Computation: Practice and Experience*, 2008.

[83] J.A. Pouwelse, J. Yang, M. Meulpolder, D.H.J. Epema, H.J. Sips, G.J.M. Smit, and M. Lew. BuddyCast: an operational peer-to-peer epidemic protocol stack. In *Annual Conference of Advanced School for Computing and Imaging (ASCI)*, 2008.

[84] D. Quercia, S. Hailes, and L. Capra. Lightweight distributed trust propagation. In *IEEE International Conference on Data Mining (ICDM) 2007*, 2007.

[85] U.N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Journal of Physical Review E*, 76(3):036106–11, 2007.

[86] R. Rahman, D. Hales, M. Meulpolder, V. Heinink, J.A. Pouwelse, and H.J. Sips. Robust vote sampling in a P2P media distribution system. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, 2009.

[87] R. Rahman, T. Vinkó, D. Hales, J.A. Pouwelse, and H.J. Sips. Design space analysis for modeling incentives in distributed systems. In *Special Interest Group on Data Communication (SIGCOMM)*, 2011.

[88] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110–14, 2006.

[89] M.K. Reiter and S.G. Stubblebine. Path independence for authentication in large-scale systems. In *ACM Conference on Computer and Communications Security (CCS)*, 1997.

[90] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. *Recommender Systems Handbook*, pages 1–35, 2011.

[91] E. Riviere and S. Voulgaris. Gossip-based networking for internet-scale distributed systems. *E-Technologies: Transformation in Connected World (MCETECH)*, 2011.

[92] J.L. Sachs, U.G. Mueller, T.P. Wilcox, and J.J. Bull. The evolution of cooperation. *Journal of The Quarterly Review of Biology*, 79(2):135–160, 2004.

[93] S. Seuken, J. Tang, and D.C. Parkes. Accounting Mechanisms for Distributed Work Systems. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2010.

[94] M. Spear, X. Lu, N. Matloff, and S.F. Wu. Karmanet: Leveraging trusted social paths to create judicious forwarders. In *IEEE International Conference on Future Information Networks (ICFCN)*, 2009.

[95] M. Srivatsa, L. Xiong, and L. Liu. Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In *ACM International Conference on World Wide Web (WWW)*, 2005.

[96] L. Subramanian, V.N. Padmanabhan, and R.H. Katz. Geographic properties of internet routing. In *USENIX Annual Technical Conference (USENIX ATC)*, 2002.

[97] G. Swamynathan, K.C. Almeroth, and B.Y. Zhao. The design of a reliable reputation system. *Journal of Electronic Commerce Research*, 10(3-4):239–270, 2010.

[98] H. Tong, C. Faloutsos, and J.Y. Pan. Random walk with restart: fast solutions and applications. *Knowledge and Information Systems*, 14:327–346, 2008.

[99] H. Tong, S. Papadimitriou, P. Yu, and C. Faloutsos. Proximity tracking on time-evolving bipartite graphs. In *SIAM International Conference on Data Mining (SDM)*, 2008.

[100] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[101] B. Viswanath, A. Mislove, M. Cha, and K.P. Gummadi. On the evolution of user interaction in facebook. In *ACM Workshop on Online Social Networks (WOSN)*, 2009.

[102] B. Viswanath, A. Post, K.P. Gummadi, and A. Mislove. An analysis of social network-based sybil defenses. In *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2010.

[103] Q.H. Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica: Journal of Econometric Society*, 57(2):307–333, 1989.

[104] K. Walsh and E.G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *USENIX Networked Systems Design and Implementation (NSDI)*, 2006.

[105] W. Willinger, D. Alderson, and J.C. Doyle. Mathematics and the internet: A source of enormous confusion and great potential. *America Mathematical Society*, 56(5):586–599, 2009.

[106] C. Wilson, B. Boe, A. Sala, K.P.N. Puttaswamy, and B.Y. Zhao. User interactions in social networks and their implications. In *European Conference on Computer Systems (EuroSys)*, 2009.

[107] L. Xiong, L. Liu, and M. Ahamad. Countering feedback sparsity and manipulation in reputation systems. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing (COLLABORATECOM)*, 2007.

[108] H. Yu, P.B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy (S&P)*, 2008.

[109] H. Yu, M. Kaminsky, P.B. Gibbons, and A. Flaxman. Sybilguard: Defending against sybil attacks via social networks. In *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2006.

Summary

A Robust Distributed Reputation Mechanism for Peer-to-Peer Systems

In P2P content distribution networks, incentive mechanisms are required for sustainable operation of the network. In general, there are two classes of incentive mechanisms, direct and indirect. In direct mechanisms, a content consumer compensates the work of a supplier by providing him some content in return. In indirect mechanisms, the work of a supplier can be rewarded by any peer in the future. A sub-class of indirect incentive mechanisms is based on the concept of *reputation*. In such a mechanism, based on its past behavior, a peer is assigned a reputation value and based on this value the other peers decide whether to collaborate with him or not. Designing an effective reputation mechanism comes with many challenges such as the accuracy of the reputation values, the security, and the scalability of the mechanism. In this thesis, we investigate these aspects of distributed reputation mechanisms and we incorporate our techniques and algorithms into the *BarterCast* reputation mechanism, which is an Internet-deployed mechanism in the field of content sharing. In BarterCast, peers collect information about the upload and download activities in the network through a gossiping protocol and compute reputations of other peers using the Maxflow algorithm in the BarterCast graph built upon the BarterCast records they have received.

In Chapter 2, we focus on the *accuracy* and the *coverage* aspects of the BarterCast reputation mechanism and propose three modifications to improve these metrics. First, instead of using themselves as the start or end point in the Maxflow algorithm, peers use the node with the highest betweenness centrality. Secondly, instead of applying the limited-hop gossiping protocol, peers run a full gossiping protocol for BarterCast record dissemination. Thirdly, the number of hops in the Maxflow algorithm is lifted to 4 and 6 hops. For evaluation, employing data collected from the Tribler network that uses BarterCast, we emulate these modifications and measure the accuracy and coverage metrics. Our findings show that the full gossiping modification is the most influential and that the other two are effective if they are combined with full gossiping.

In Chapter 3, we study the security aspects of the BarterCast mechanism and make it resilient against sybil attacks, white-washing and mis-reporting. In a sybil attack, a malicious peer creates fake identities and uses them to subvert the mechanism. In white-washing and mis-reporting attacks, a malicious peer clears its bad reputation and sends

false records about other peers, respectively. To withstand sybil attacks, we introduce an accounting schema on the edges in the BarterCast graph, where based on the upload or download actions, the weights of the edges are increased or decreased. By this schema, the reputations of the sybil peers degrades very fast. To resist white-washing, we modify the formulation part of the mechanism, so a white-washer receives the lowest reputation value. Finally, to stop mis-reporting, instead of plain records, prior sending a record, the involved peers doubly sign it. To evaluate the proposed accounting schema for sybil attacks, we again employ the crawled data from the Tribler network and compare the reputations of the malicious and honest peers in the sybil-resistant and the original versions of the mechanism. Our findings show that with minimal effect on the reputations of the honest peers, the new mechanism degrades the reputations of the sybils.

In Chapter 4, through modifying the record dissemination component of the mechanism we improve its scalability. The problem is that if all the generated BarterCast records are sent and stored by all the peers, the communication and storage overhead grows very fast. On the other hand, not all the records are needed by all the peers, and a peer just needs a small fraction of all the records for reputation evaluation. To incorporate scalability, when sending a record, peers give higher priority to the peers similar to themselves. As a consequence, without compromising the accuracy, records are sent to the peers who may need them in the future. To find the similar peers we use two methods, one is based on creating a directed acyclic graph, and the other method is based non-uniform random walks. To measure the communication, storage, and computation costs, we use the crawled data from the Tribler network. In the obtained results, compared to full gossiping, the communication and storage costs are decreased by a factor of 100, while we observe minimal a effect on the reputation values.

In Chapter 5, accumulating the crawled data from the Tribler network and employing complementary data from other sources, we perform a comprehensive analysis of the BarterCast graph from the network science perspective. First, we investigate the degree distribution of the network and fit various distributions. Our finding shows that the distribution is very close to a power-law distribution. Furthermore, we study the network from the node interconnectivity, the clustering, and the community aspects. The results indicate that low degree nodes tend to connect to high degree nodes, and we observe community structures around peers that stay online for a long time. Moreover, our study shows that the shortest path distribution is similar to the one from online social networks and we observe a densification effect as the graph grows. Finally, we consider the graph from the geographical aspect, and investigate the correlation, for pairs of nodes, between being located in the same ISP or country and having an edge in the graph. The results show that even if there a positive correlation, it is weak.

Samenvatting

Een Robuust Gedistribueerd Reputatie-mechanisme voor Peer-to-Peer Systemen

n P2P-netwerken voor het verspreiden van *content* zijn *incentive*-mechanismen nodig voor de goede werking van het netwerk. In het algemeen bestaan er twee klassen van *incentive*-mechanismen, namelijk direct en indirecte. In directe mechanismen compenseert een *peer* die *content* verkrijgt de *peer* die die *content* verschaft door iets terug te geven. In indirecte mechanismen kan het werk van een verschaffer op een later tijdstip worden beloond door een willekeurige andere *peer*. Een deelklasse van de indirecte mechanismen is gebaseerd op het concept van *reputaties*. In zo'n mechanisme krijgt een *peer* een reputatie-waarde gebaseerd op zijn gedrag in het verleden, en op grond van deze waarde beslissen de andere *peers* of ze al dan niet met hem samenwerken. Het ontwerp van een effectief reputatiemechanisme brengt vele uitdagingen met zich mee zoals de nauwkeurigheid, de veiligheid, en de schaalbaarheid van het mechanisme. In dit proefschrift onderzoeken we deze aspecten van gedistribueerde reputatiemechanismen en integreren we onze technieken en algorithmen in het *BarterCast* reputatiemechanisme, een mechanisme voor *content*-distributie dat in het Internet in gebruik is. In BarterCast verzamelen *peers* informatie over de *upload*- en *download*-acties in het netwerk via een epidemisch protocol, en berekenen ze de reputaties van andere *peers* met behulp van het Maxflow-algorithme in het BarterCast-netwerk dat ze creëren op basis van de BarterCast-*records* die ze ontvangen hebben.

In Hoofdstuk 2 concentreren we ons op de aspecten van de nauwkeurigheid en de dekking van het BarterCast reputatiemechanisme en stellen we drie wijzigingen voor om deze metrieken te verbeteren. Ten eerste gebruiken *peers* in het Maxflow-algorithme niet zichzelf maar de *peer* met de grootste *betweenness centrality* als start- en eindpunt. Ten tweede gebruiken *peers* volledige verspreiding van BarterCast *records* in plaats van die verspreiding tot één stap te beperken. En ten derde vergroten we de lengte van de paden die Maxflow gebruikt tot 4 of 6 stappen. Voor de evaluatie van de nauwkeurigheid en de dekking van deze drie wijzigingen emuleren we ze met behulp van gegevens verzameld in het Tribler-netwerk dat BarterCast gebruikt. Het blijkt dat de volledige verspreiding de belangrijkste wijziging is en dat de andere twee effectief zijn in combinatie daarmee.

In Hoofdstuk 3 bestuderen we de veiligheidsaspecten van het BarterCast-mechanisme en maken we het bestand tegen *sybil attacks*, *white-washing*, en onjuist rapporteren. In

een *sybil attack* creëren kwaadaardige *peers* namaak-identiteiten en gebruiken die om het mechanisme te ondermijnen. In *white-washing* en onjuist rapporteren komen *peers* van hun slechte reputatie af respectievelijk verspreiden ze onjuiste informatie over de acties van andere *peers*. Om *sybil attacks* te weerstaan introduceren we een *accounting*-schema op de connecties in het BarterCast-netwerk waarin als gevolg van *upload*- en *download*-acties de gewichten van de connecties verhoogd of verlaagd worden. Met dit schema worden de reputaties van de *sybils* snel gereduceerd. Om *white washing* te weerstaan wijzigen we het formuleringsdeel van het mechanisme zodat *white washers* de laagste reputatiewaarde krijgen. Tenslotte voorkomen *peers* onjuist rapporteren door de te verspreiden *records* tweevoudig te ondertekenen. Voor de evaluatie van het *accounting*-schema voor *sybil attacks* gebruiken we wederom de gegevens verkregen van het Tribler-netwerk en vergelijken we de reputaties van eerlijke en kwaadaardige *peers* in het gewijzigde en het originele mechanisme. Het blijkt dat in het gewijzigde mechanisme de reputaties van de *sybils* veel lager zijn met een minimaal effect op de reputaties van de eerlijke *peers*.

In Hoofdstuk 4 verbeteren we de schaalbaarheid van het BarterCast-mechanisme door de manier van verspreiden van *records* aan te passen. Het probleem is dat als alle BarterCast-*records* worden verstuurd en opgeslagen door alle *peers*, de kosten voor de communicatie en opslag superexponentieel groeien. Aan de andere kant zijn ook niet alle *records* nodig voor alle *peers*, een *peer* heeft slechts een klein deel ervan nodig voor de berekening van reputaties. Om schaalbaarheid te bereiken geven *peers* bij het versturen van *records* prioriteit aan *peers* die lijken op zichzelf. Daardoor worden *records* verstuurd naar *peers* die ze in de toekomst nodig kunnen hebben zonder aan nauwkeurigheid in te boeten. Om gelijkelijke *peers* te bepalen gebruiken we twee methoden, één gebaseerd op gerichte acyclische grafen en één gebaseerd op niet-uniforme *random walks*. Om de kosten van communicatie, opslag en berekening te bepalen gebruiken we gegevens uit het Tribler-netwerk. Onze resultaten geven aan dat in vergelijking met volledige verspreiding de commnicatie- en opslagkosten met een factor 100 worden teruggebracht, terwijl we een minimaal effect op de reputatiewaardes waarnemen.

In Hoofdstuk 5 voeren we een uitgebreide analyse uit op het BarterCast-netwerk gecreëerd met behulp van gegevens verkregen uit het Tribler-netwerk en aanvullende bronnen. Eerst onderzoeken we de verdeling van de graad in het netwerk en gaan we voor verscheidene distributies na of die passen. We laten zien dat de verdeling dicht bij een *power law* ligt. Bovendien onderzoeken we het netwerk op zijn interconnectiviteit, clustering, en *community*-aspecten. De resultaten laten zien dat knopen van lage graad vaak verbonden zijn met knopen met hoge graad, en we vinden *community*-structuren die gevormd zijn rondom knopen die lang in het systeem aanwezig zijn. Bovenzien laat onze studie zien dat de distributie van kortste paden lijkt op die in *online social networks*, en dat er een verdichtingseffect optreedt als het netwerk groeit. Tenslotte beschouwen we het netwerk vanuit geografisch perspectief en onderzoeken we de correlatie voor paren

knopen tussen het gelocaliseerd zijn in dezelfde ISP of hetzelfde land en het onderling met elkaar verbonden zijn. De resultaten laten zien dat er een positieve correlatie is, zij het dat deze zwak is.

Curriculum vitae

Rahim Delaviz was born in the northwestern city of Ahar, Iran, on 19 February 1978. When he was 3 years old his family moved to Tabriz, the central city of East-Azerbaijna province of Iran. After receiving his high school diploma in Physics & Mathematics he moved to Tehran to study Software Engineering at Kharazmi University (former Tarbiat Moallem University). He finished his BSc in 2000 and started an MSc program in Mathematical and Economical Systems Engineering at Sharif University of Technology. After graduation, he started working in an energy and then in an electronic payment company. In 2007 he left for Stockholm, Sweden, and followed an MSc program in distributed systems at the Royal Institute of Technology (KTH). In January 2009 he defended his MSc thesis and 3 months later, in March 2009, he began as a PhD candidate at Delft University of Technology, the Netherlands. His PhD track was focused on distributed reputation mechanisms in content-sharing networks, which resulted in this thesis.

List of publications:

- R. Delaviz, N. Zeilemaker, J.A. Pouwelse, and D.H.J. Epema. A network science perspective of a distributed reputation mechanism. *In IFIP Networking Conference*, 2013.
- R. Delaviz, J.A. Pouwelse, and D.H.J. Epema. Targeted and scalable information dissemination in a distributed reputation mechanism. *In ACM Workshop on Scalable Trusted Computing (STC)*, 2012.
- R. Delaviz, N. Andrade, J.A. Pouwelse, and D.H.J Epema. SybilRes: A sybilresilient flow-based distributed reputation mechanism. *In International Conference on Distributed Computing Systems (ICDCS)*, 2012.
- R. Delaviz, N. Andrade, and J.A. Pouwelse. Improving accuracy and coverage in an Internet-deployed reputation mechanism. *In IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2010.
- R. Delaviz and K. Eshghi. An Efficient Algorithm for a Class of Capacitated Transportation Problems. *In International Conference on Operations and Quantitative Management (ICOQM)*, 2004