Improved Accuracy and Coverage in the BarterCast Reputation Mechanism

R. Delaviz Aghbolagh, N.F. Andrade, D.H.J. Epema and J.A. Pouwelse

Department of Software Technology
Delft University of Technology, The Netherlands
r.delavizaghbolagh@tudelft.nl

Keywords: file sharing, reputation systems, accuracy

Abstract

In spite of a large number of proposed reputation mechanisms for P2P system, few have been investigated in real situations. BarterCast is a distributed reputation mechanism used by our Bittorent-based file-sharing client Tribler. In BarterCast, each peer uses messages received from other peers to build a weighted, directed subjective graph that represents the upload and download activity in the system. A peer calculates the reputations of other peers by applying the maxflow algorithm to its subjective graph. In this paper, we identify and assess three potential modifications to BarterCast for improving its accuracy and coverage (fraction of peers for which a reputation value can be computed). First, a peer executes maxflow from the perspective of the node with the highest betweenness centrality in its subjective graph instead of itself. Second, 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 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 the modifications on the collected data and measure accuracy and coverage.

1 Introduction

P2P systems can benefit from *reputation mechanisms* in which peers evaluate the reputations of other peers in order to identify proper service providers. Important properties of reputation mechanisms are their *accuracy*, that is, how well peers approximate "objective" reputation values when calculating the reputation values of other peers, and their *coverage*, that is, the fraction of peers for which peers are able to compute reputation values. Inaccurate or partial reputation evaluation may lead to misjudgment, poor behavior, and finally, system degradation. The BarterCast mechanism [9] is an Internet-deployed reputation mechanism that is used by the Tribler BitTorent-based file-sharing client [11] to select good bartering partners and to prevent free-riding. In this paper we evaluate the accuracy and the coverage of the BarterCast reputation mechanism, and we propose three modifications to this mechanism and evaluate them with respect to these two metrics. The evaluation is performed based on empirical data that we have collected by crawling the Tribler P2P system over a 3-month period.

In large-scale online applications such as P2P file-sharing systems, because of the large population size and high turnover, the participants often do not have previous interaction experience with each other. So, when encountering unknown parties they need a reputation mechanism to help them to choose a proper participant to interact with. The consequence of wrong decisions could be harmful; for example, in a P2P file-sharing system, peers may upload much content to free-riders without receiving anything in return. Empirical study [1] has shown that this phenomenon can seriously degrade the system welfare. The fundamental idea behind a reputation mechanism is that, usually individual behavior does not change radically over time, and past activity is a good predictor of future actions [12]. Using this idea, a typical reputation mechanism collects information on the past behavior of the participants in a system and quantifies this information into reputation values. In a distributed reputation mechanism, depending on how the information about peers behavior are disseminated or how the reputation values are computed, each participant may have different reputation values for the same participants.

We have previously designed and implemented the BarterCast reputation mechanism in our BitTorrent-based P2P client Tribler. In BarterCast, peers exchange messages about their upload and download

actions. From the BarterCast messages it receives, each peer builds a local weighted, directed graph with nodes representing peers and with edges representing amounts of transferred data. In BarterCast, a peer calculates the reputation values of other peers by applying the maxflow algorithm to its subjective graph, interpreting the edge weights as "flows."

In this paper 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 any combination of these three modifications. First, rather than have each peer execute the maxflow algorithm to compute reputations from its own perspective, it does so from the perspective of the node with the highest *betweenness centrality* [7] in its subjective 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 original BarterCast. In order to evaluate the original BarterCast reputation mechanism and our three modifications, we have crawled the Tribler P2P system for 83 days to obtain as many BarterCast records of the Tribler peers as possible. From the records obtained from each peer, we emulate its reputation computations by reconstructing its *subjective view*, and in order to execute the maxflow algorithm this view is represented in the form of a digraph (subjective graph). In this paper the terms subjective graph and subjective view are synonym and the subjective graph is a way to represent the view.

In the rest of the paper, we first explain the BarterCast mechanism in detail and we define the metrics accuracy and coverage. In Section 3, we explain the crawler and the data collecting process, and we give some statistics about the collected data. In Section 4, we state the problem of the current version of BarterCast and explain the modifications we propose. In Section 5, first we explain the experimental setup then the experimental results are presented. We present our conclusions in Section 6.

2 The BarterCast Reputation Mechanism

In this section, first we explain the BarterCast mechanism in detail and then we formulate the metrics accuracy and coverage in this mechanism.

2.1 The BarterCast Mechanism

The BarterCast mechanism is used by the Tribler Bittorent client to rank peers according to their upload and download behavior. 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. In BarterCast, when a peer exchanges content with another peer, they both log the amount of transferred data and the identity of the corresponding peer in a BarterCast record; these records store the total cumulative amounts of data transferred in both directions since the first data exchange between the peers. In BarterCast, peers regularly contact another peer in order to exchange BarterCast records. Peer sampling for selecting to whom to send BarterCast records is done through a gossip protocol called BuddyCast, which is at the basis of Tribler. In BuddyCast, peers regularly contact another peer in order to exchange lists of known peers and content with each other. Using the BarterCast message exchange mechanism, each peer creates its own current local view of the upload and download activity in the system.

To be more precise, the receiver of BarterCast records creates and gradually expands its *subjective graph*. The subjective graph of peer i is $G_i = (V_i, E, \omega)$, where V_i is the set of nodes representing the peers about whose activity i has heard through BarterCast records, and E is the set of weighted directed edges (u, v, w), with u and $v \in V_i$ and w the total amount of data transferred from u to v. Upon reception of a BarterCast record (u, v, w), peer i either adds (a) new node(s) and a new edge to its subjective graph if it did not know u and/or v, or only (a) new directed edge(s) if it did know u and v but did not know about the data transfer activity between them, or adapts the weight(s) of the existing edge(s) between u and v. If peer i receives two BarterCast records with the same sender u and the same receiver v from different peers, it keeps the record that indicates lower amounts of data transferred in order to avoid invalid reports from malicious peers that try to inflate their uploads. Furthermore, the direct experience of the peer has higher priority than received reports from others.

In order to calculate the reputation of an arbitrary peer $j \in V_i$ at some time, peer i applies the maxflow algorithm [6] to its current subjective 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 node to a destination node in a weighted graph. When applying maxflow to the subjective graph, we interpret the weights of the edges, which represent amounts of data transferred, as flows. The original maxflow algorithm by Ford-Fulkerson [6] tries all possible paths from the source to the destination, but in BarterCast, in order to limit the computation overhead, only paths of length at most 2 are considered. Another reason for setting this limit is that the majority of peers have shared relationships through popular intermediaries, and using two hops in maxflow provides sufficient data for the eval-

uation of reputations [10]. Using the values $F_2(.,.)$ as computed with the 2-hops maxflow algorithm, the subjective reputation of peer j from peer i's point of view is calculated as:

$$SR_{ij} = \frac{\arctan(F_2(j,i) - F_2(i,j))}{\pi/2}, \qquad (1)$$

and so $SR_{ij} \in [-1, +1]$. If the destination node j is more than two hops away from i, then its reputation is set to 0.

In Figure 1 a simple subjective graph is shown in which peer i as the owner of the graph evaluates the reputation of peer j. In this graph, $F_2(i,j) = 11$ and $F_2(j,i) = 5$, and so $SR_{ij} = -0.89$.

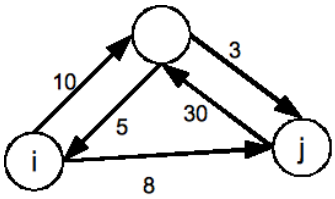

Figure 1: A sample subjective graph

Using a flow algorithm (e.g., maxflow in BarterCast) is like doing a collaborative inference where the knowledge of all involved nodes is included in the computation of the final value. Beside this, flow algorithms like maxflow are more resilient against sybil attacks that trivial operations like averaging or summation are not [5]. The BarterCast mechanism can be generalized in the form of *flow-based* mechanisms. Such mechanisms have two common features. First, the relation between participants is shown as a graph. Second, there is a function ϕ which calculates the flow of a specified metric from a node set I to a destination node set J, and the obtained flow is used to calculate the final reputation value.

2.2 Accuracy and Coverage

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, OR_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

$$OR_j = \frac{\arctan(U_j - D_j)}{\pi/2} \qquad (2)$$

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

$$error(i,j) = abs(SR_{ij} - OR_j) \qquad (3)$$

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 subjective graph G the h-hop coverage of node i is defined as

$$coverage_G(i,h) = |\{u | F_h(i,u) > 0 \, or \, F_h(u,i) > 0\}| \qquad (4)$$

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 $[0, 1]$, which makes it possible to compare this metric in graphs of different sizes.

2.3 Related Work

The BarterCast mechanism was designed by Meulpolderet al. to distinguish free-riders and cooperative peers in file-sharing environments. After the first release, Seuken et al. [**?**] 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. It is shown by Xiong et al. [13] 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. There are also some other distributed reputation mechanisms used in P2P systems, but they use different methods to calculate reputation values. EigenTrust [8] is based on summation of direct observations and indirect data and uses centralized *matrix operations* to compute the left eigen vector. The CORE system [4] uses *arithmetic weighted averaging* on historical data to calculate reputation values. The BarterCast mechanism best fits in a class of mechanisms which use flow-based reputation function as defined by Cheng et al. [5].

3 The Crawler and the Collected Data

To collect the required dataset consisting of the BarterCast records of all (or at least, many) Tribler peers for analysis, we have crawled the Tribler network for 83 days, from June 20 until September 9, 2009. Except for some slight differences, the crawler

works as an ordinary Tribler peer. Discovery of new peers is done through the BuddyCast protocol, which is the gossiping engine of the Tribler client. When a new peer is discovered with this protocol, it is added to a list, and every hour, the crawler contacts all peers on this list and asks them for the latest BarterCast records that it has not yet received from that peer by including the timestamp of the latest record it does have of each peer. From the BarterCast records received by the crawler from each peer, we can reconstruct the subjective graph of that peer in the same way the peer builds it.

The discovered peers have different ages, with 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 at the time but are no longer available. 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. So, to do smooth 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 sending at most 50 records, receiving exactly 50 records from a peer probably means that it would have had more records to send. 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 size a multiple of 50 are removed.

To make the collected records sortable and to account for the time difference with remote peers, they are asked to send their local time in a reply message to the crawler as well. When the crawler receives such a message, 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 time is:

$$t_c - t_p + t_r \tag{5}$$

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

During the crawling period, the crawler collected 547,761 BarterCast records from 2,675 different peers. After filtering out the incomplete views, 416,061 records were left, collected from 1,442 peers, which 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 paper is based only on complete views.

4 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.

4.1 Modification 1: Using Betweenness Centrality

Betweenness centrality has been introduced by Freeman [7] 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 (indegree + outdegree), which is a local quantity, betweenness centrality is a quantity across the whole graph; nodes with many connections may have a very low betweenness centrality and vice versa [2]. 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 subjective 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.

4.2 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 decreases the effect of mis-reporting attacks 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 a hop limit, so that in principle all online peers eventually receive all propagated records.

4.3 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.

5 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 subjective 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.

5.1 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* received 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.

5.2 Experiment Design

In a large scale system like the one that the BarterCast mechanism is designed for, it is not required that every peer be 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 be 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:

I) $i = s$: The peer that uploads is also the owner of the subjective graph, and it is the peer that should do the reputation evaluation.

II) $U > 0$: 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 *new-comers* or *existing peers*. The new-comers 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 new-comers, 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 new-comer 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 new-comer; otherwise it is.

Reputation evaluation for new-comers is meaningless, as without any previous information the mechanism cannot do anything. So, in the results of the accuracy and the coverage below, only the existing nodes are considered and the new-comers are ignored. 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 new-comers and existing peers are 140 and 123, respectively.

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.

5.3 Coverage

The barchart in Figure 2 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.

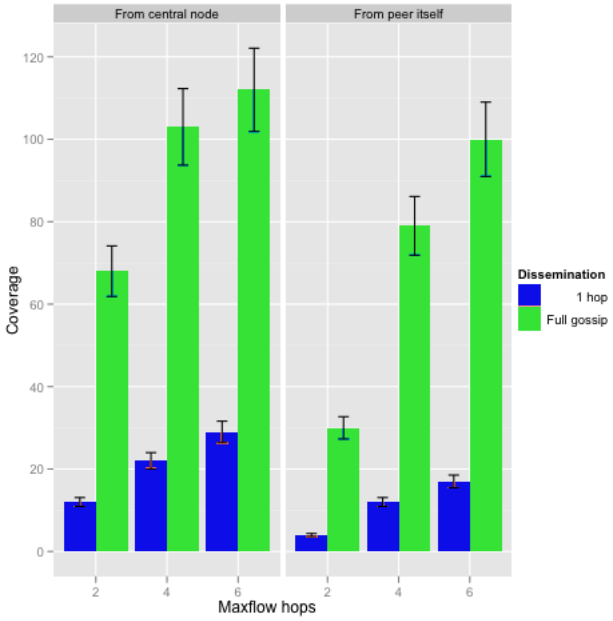

Figure 2: The coverage of the BarterCast mechanism in different scenarios. (Error bars show the standard error of mean.)

5.4 Accuracy

In Figure 3 we show the fractions of nodes for which either the central node in the subjective graph or the local peer 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.

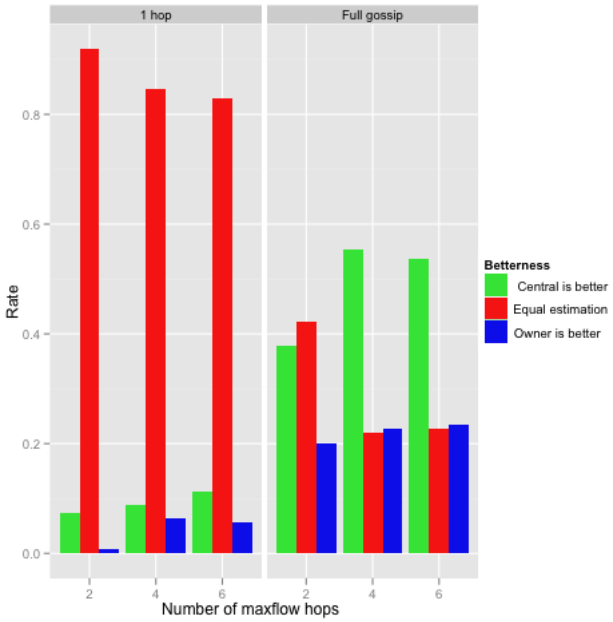

Figure 3: Comparing the accuracy the of central node against the view owner in the BarterCast mechanism.

Figure 3 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 4 shows the mean and its standard error for all combinations of the modifications. As the graphs show, changing only 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, we compare also the median of estimation errors in different scenarios. Figure 5 shows the median of the error in various situations. As can be seen, the influence of using the central node is higher than that of the other factors. When all improvements are applied, the median is only 0.087.

6 Conclusion and future work

In this paper 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

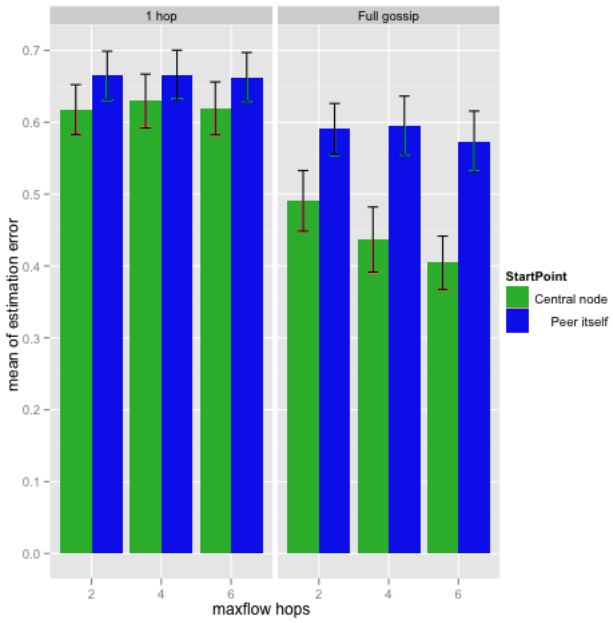

Figure 4: The mean of the estimation error in the BarterCast mechanism.(Error bars show the standard error of mean)

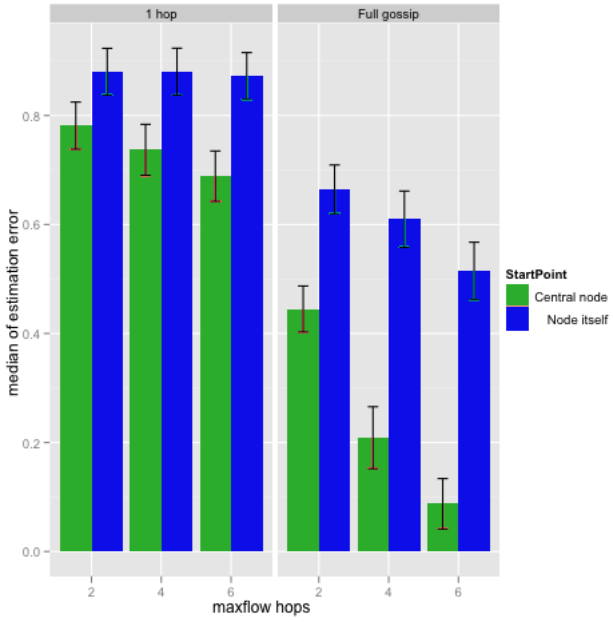

Figure 5: The median of the estimation error in the BarterCast mechanism.(Error bars show the standard error of median)

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 addressed. Also full gossiping increases the dissemination performance, but it is more vulnerable to misreporting attacks, and the indirect reports should be treated carefully. A possible solution for this problem could be to put the indirect reports 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.

The second problem related to the proposed solutions is the performance of finding the node with the highest betweenness centrality in the subjective graphs. The complexity for this calculation is considerable, $O(nm)$ for unweighted and $O(mn+n^2 \log n)$ for weighted graphs [3], where n and m are the number of nodes and edges respectively. Our results are based on unweighted version and even with $O(nm)$ complexity the usage patterns observed so far hint at its practical feasibility. Our preliminary analysis suggests that the subjective graphs in BarterCast have a scale-free property, and the highest central node in them does not change very often. In this setting, the computation of the central node can be done periodically and in relatively long periods, reducing the amount of computation overhead in the system.

Finally, we believe that our proposed solution and especially the betweenness centrality can be applied not only in BarterCast, but in any flow-based reputation mechanism that has similar features as described in Section 2.1. Both addressing the open questions in implementing the proposed improvements and evaluating the generality of the use of betweenness centrality are challenging topics for future research.

References

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

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

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

[4] S. Buchegger and J.Y. Le Boudec. A robust reputation system for mobile ad-hoc networks. *Proceedings of P2PEcon, June*, 2004.

[5] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, page 132. ACM, 2005.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*, pages 651–664. MIT Press and McGraw-Hill, second edition, 2001.

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

[8] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM New York, NY, USA, 2003.

[9] M. Meulpolder, JA Pouwelse, DHJ Epema, and HJ Sips. BarterCast: A practical approach to prevent lazy freeriding in P2P networks. 2009.

[10] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson. One hop reputations for peer to peer file sharing workloads. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 1–14. USENIX Association, 2008.

[11] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, DH Epema, M. Reinders, MR Van Steen, and H.J. Sips. Tribler: A social-based peer-to-peer system. *Concurrency and Computation–Practice and Experience*, 20(2):127–138, 2008.

[12] P. Resnick and R. Zeckhauser. Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. *Advances in Applied Microeconomics: A Research Annual*, 11:127–157, 2002.

[13] L. Xiong, L. Liu, and M. Ahamad. Countering feedback sparsity and manipulation in reputation systems. In *Proceedings of the 2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 203–212. Citeseer, 2007.