Havelaar: A Robust and Efficient Reputation System for Active Peer-to-Peer Systems

Dominik Grolimund, Luzius Meisser, Stefan Schmid, Roger Wattenhofer
{grolimund@inf., meisserl@, schmiste@tik.ee., wattenhofer@tik.ee.}ethz.ch
Computer Engineering and Networks Laboratory (TIK), ETH Zurich, CH-8092 Zurich

***Abstract*— Peer-to-peer (p2p) systems have the potential to harness huge amounts of resources. Unfortunately, however, it has been shown that most of today's p2p networks suffer from a large fraction of free-riders, who consume resources without contributing much to the system themselves. This results in an overall performance degradation, and hence proper incentives are needed to encourage contributions. One interesting resource is bandwidth. Thereby, a service differentiation approach seems appropriate, where peers contributing higher upload bandwidths are rewarded with higher download bandwidths in return. Keeping track of the contribution of each peer in an open, decentralized environment, however, is a difficult task; many proposed systems are susceptible to false reports. Besides being prone to attacks, some solutions have a large communication and computation overhead, which can even be linear in the number of transactions—an unacceptable burden in practical and active systems. In this paper, we propose a reputation system which is robust to false reports and overcomes this scaling problem. Our results are promising, indicating that the mechanism is accurate and efficient especially when applied in systems where there are lots of transactions.**

I. Introduction

The power of *peer-to-peer (p2p) computing* is based on the resource contribution of the network's constituent parts, the peers. Therefore, the success of a system in practice crucially depends on its ability to cope with selfish peers which aim at consuming more than they contribute.

When faced with the task of implementing a fairness scheme for upload bandwidth for our distributed p2p storage system *Kangoo*[1], we could not find a solution which entirely fits our needs. In Kangoo, *erasure codes* are employed to achieve high data availability with moderate redundancy. Files are divided into blocks, which are encoded into many small fragments. All these fragments are then stored on a different peer. Consequently, in Kangoo, there is a large number of transactions. Our goal was to develop a scheme for this environment, where peers contributing higher upload bandwidths for longer time periods are rewarded with a higher download bandwidth in return.

This paper presents the reputation system *Havelaar* which we have implemented for Kangoo. Unlike many existing solutions, Havelaar does not rely on transitivity of trust, and achieves a high robustness to attacks by design. This is accomplished by a novel aggregation technique in which a peer u always reports directly observed or aggregated contributions to the same set of peers. These *successor* peers are determined by a hash function $h(u)$ on the identifier (e.g., the IP-address) of u. This scheme allows a successor to detect and defend against egoistic cheating (e.g., reporting too large values, or reporting too often). Hence, our solution is different from *distributed hash table (DHT)-based* approaches where a peer u benefitting from a peer v reports v's contribution value to peers determined by a hash function $h(v)$ (i.e., depending on v rather than u).

Our results are promising: Havelaar is not only robust to attacks, but also efficient and—unlike many other solutions—scales well in the number of transactions. Therefore, we believe that Havelaar is well-suited for other active p2p systems with many transactions.

The rest of this paper is organized as follows. In Section II, related work is reviewed. Section III gives background information on Kangoo and on the intended environment for Havelaar. Section IV quickly outlines how peers could be rewarded given their contribution values. The Havelaar reputation system is then presented in detail in Section V. In Section VI, the accuracy of Havelaar's approximation of the real contributions is analyzed. We report on our results concerning communication costs in Section VII. Section VIII shows how Havelaar copes with various attacks. Finally, Section IX presents some simulation results, before Section X concludes the paper.

II. Related Work

It is not hard to find evidence of selfish behavior in existing p2p systems [2], [14], and the field has already spurred a large body of research [8], [12], [18], [20], [28].

Perhaps the simplest fairness mechanism is to directly incorporate contribution monitoring into the *client software*. For instance, in the popular file-sharing system *Kazaa*, the client records the contribution of its user. However, such a solution can simply be bypassed by implementing a different client which hard-wires the contribution level of the user to the maximum, as it was the case with *Kazaa Lite*.

In systems such as *BitTorrent* [6], where peers upload to the same set of peers from which they also download, a simple *tit-for-tat mechanism* [3] may be fine. When interactions between the same pairs of peers are less frequent, however, such *barter systems* [32] fail.

Inspired by real economies, some researchers have also proposed the introduction of some form of virtual money which is used for the transactions. However, these *monetary or credit based approaches* have a substantial overhead in terms of communication costs and infrastructure, and are inefficient [11], [33]. Often these systems also require market regulating mechanisms [31] to cope with inflation or deflation—a complex issue.

[1]To be released (http://www.caleido.com/kangoo).

Additionally, monetary based systems may deter users from participating [21].

If a peer makes too few direct observations to judge a peer's contribution, it has to take into account indirect observations from other peers [17]. Such systems are generally called *reputation* (or *reciprocity based*) systems, and are well-known from auctioning applications such as eBay. However, second-hand observations introduce the problem of *false reports* [4], [17]. Many proposals seek to mitigate these effects [1], [5], [9], [15], most of them relying on *transitivity of trust*, where observations are weighted by the reputation of the reporter. Additionally to the problem of false reports, an infrastructure to exchange the second-hand observations is needed [17]. In most reputation-based systems, second-hand observations are either requested before a transaction from other peers [1], [4], or they are simply flooded throughout the system. Alternatively, the contribution values can be stored in a *distributed hash table* (DHT) [25], [27], [30] ("DHT-based approach"). For systems with lots of transactions where contribution values are updated constantly, however, this results in an unacceptable communication overhead: updating and checking a peer's reputation entails costly (wide-area) DHT lookups [22], [23].

In contrast, in the Havelaar reputation system, a peer is able to compute the reputation of a requesting peer *locally*. By aggregating the contribution values of a large number of peers, our system tackles also the problem of transitivity of trust. Finally, a peer is able to defend against cheating by checking the *credibility of reports*, thus preventing many reputation attacks by design.

III. System Model

Havelaar was designed with a special application in mind: *Kangoo*, a large-scale distributed storage system for the Internet. Kangoo divides files into blocks, which are encrypted and then encoded into redundant fragments using erasure codes. These encrypted fragments are stored on different peers in a DHT. However, in Kangoo, peers fall into different categories, and fragments are only stored on so-called *storage peers* which fulfill some minimal requirements such as having long uptimes (e.g., more than 6 hours a day). How storage peers are selected among the set of all available peers, and why rational peers also have incentives to become storage peers is beyond the scope of this paper. However, in the following, we will assume that all peers store data, and that these peers generally have long uptimes.

Because of the used encoding scheme, storing and retrieving files in Kangoo results in lots of transactions with many different peers in the system (e.g., 500 transactions with different peers to store and retrieve *one* file). While this is necessary for high availability and for fast parallel downloads, it is crucial that the fairness mechanism scales well in the number of transactions—an important objective of Havelaar.

There are a number of other properties of Kangoo which influence Havelaar. Since fragments are stored in the Kangoo DHT depending on hash functions on the fragment itself, an attacker cannot determine the destination of a transaction. Furthermore, peer identifiers are securely assigned by Kangoo—i.e., externally to Havelaar—, and therefore Havelaar can rely on randomly assigned node identifiers and does not tackle Sybil attacks [7] or white-washing [9], [10] itself. However, Havelaar also minimizes incentives to create new identifiers by assigning new peers a low initial reputation.

Finally, in Kangoo, *churn* [16] is not a major concern, as (storage) peers are required to stay online on almost a daily basis and for several hours, and are also expected to remain in the system for longer time periods (months or years).[2]

IV. Rewarding Mechanism

Havelaar is mainly a *reputation system* (cf Section V) and therefore independent of a concrete rewarding mechanism; that is, given the contribution values of Havelaar, many strategies can be applied to allocate bandwidth to the peers.

However, to complete the picture, we briefly sketch the approach we have chosen for our system. We make use of a mechanism similar to the one described in [19]. But as performance is crucial in Kangoo, we only apply fairness mechanisms in situations of *contention*, i.e., when several peers want to download from the same node *concurrently*. Assuming that bandwidth is free—and lost when not used—, the maximum possible bandwidth will always be allocated to a requesting peer; no artificial limits are used. For our system, this is the desired behavior because—unlike monetary-based systems—we do not want to provide any disincentives to participate and download in the network. We only limit the resource allocation for excessive downloaders, as will be described in Section V.

V. Reputation System

In this section, we describe the main ideas behind Havelaar. The goal of Havelaar is to track the contribution of each peer in the system. Since it would be very expensive to inform each peer about *all* transactions happening in the system, we seek to provide the peers with a good *approximation* of the real contribution values. Thereby, our solution must be efficient and also resilient to cheating. Basically, Havelaar has three goals: (1) accurate estimation of the real (global) contribution values of other peers, (2) robustness against selfish peers, and (3) efficiency, i.e., scalability in the number of transactions.

In our system, the reputation of a peer u should be reflected by the peer's contribution value C_u, which in turn depends on the bandwidth b_u the peer provides, *and* the size s of the corresponding fragments. Hence, the total contribution value of peer u is given by $C_u = \sum_{\text{transactions t}} (b_{u,t} \cdot s_{u,t})$. Note that the contribution value will only be increased *after a complete upload* in order to reward proper transactions only.

So how does Havelaar track these contributions? Each peer u maintains a vector $\vec{o}$ (observations) of size n (number of peers in the network) to store the contributions of other peers which u has directly experienced itself. That is, after each download, u updates its vector $\vec{o}$ accordingly.

Even in active systems with lots of transactions, a peer only gets in touch with a subset of all peers. Therefore, peers need to share their private observations by sending them to other peers once in a while. To achieve this, Havelaar employs a *round-based aggregation technique*. Thereby, once in a round, each

[2]Note that in Kangoo, peers represent long-term customers.

peer u sends its observation vector $\vec{o}$ to a small number k (e.g., 7) of other peers in the system, called u's *successors* (similarly, we will refer to u as a *predecessor* of such a successor peer). The successor peers are determined by a set of k hash functions $\{h_1(u), ..., h_k(u)\}$ on u's identifier.[3] Moreover, we will assume that a *round* is roughly one week. Note that it does not matter when exactly in this time interval a peer sends its report to the successors, i.e., when a peer u cannot contact a peer v (e.g., because v is offline), it can try again later.

A peer u always informs the same set of peers, *independently* of which peers contributed resources to u. Upon receiving a vector, a peer can check whether it has been sent by a correct predecessor by verifying the hash function[4]—otherwise, it can simply drop the vector. The "observed" contribution value of the predecessor itself is not taken into account, in order to render the most attractive attack impossible by design. What is more, each peer can also ignore the vector of a peer that sends too frequently (more than once per round). Therefore, a peer can only attack the system with a false praise or accusation of another node. However, such an attack can either be detected or it will be averaged out, as we will see in Section VIII.

Unfortunately, sending direct observations to the k successors is still not sufficient in order to accurately estimate the contributions of all peers in large networks. Therefore, we extend the mechanism as follows: Upon receiving the observation vectors from its k predecessors, each peer aggregates them with its own observations, and sends the new vector to its k successors. Thus, one vector can summarize a large number of observations. However, we have to make sure that the values in the vectors do not contain too many observations from the past which do not reflect the current behavior of each peer. What is needed is a scheme where old observations can be truncated in the observation vector, but where there are still enough observations to update the contribution vector after each round.

This is accomplished as follows. In every round, a peer puts its own observations into a vector $\vec{o}_0$ (so far denoted by $\vec{o}$). After each round, it sends a message to its k successors containing its own (direct) observations from this round ($\vec{o}_0$), the aggregated observations of its k predecessors from the last round ($\vec{o}_1$), the aggregated observations of the k predecessors of its own k predecessors from the round before the last round ($\vec{o}_2$), and so forth. The message thus contains a matrix $O := [\vec{o}_0, \ldots, \vec{o}_{r-1}]$. Upon receiving the matrix $O_i := [\vec{o}_1, \ldots, \vec{o}_r]$ from predecessor $i \in [1, k]$ (note that when sending, the index runs from $0 \ldots r-1$, and when receiving, it is renamed to $1 \ldots r$), a peer aggregates all observations and updates its contribution vector $\vec{c}$ accordingly. Thus, the vectors from previous rounds aggregate an exponentially growing number of observations. The oldest observations lie r rounds in the past.

A simplified description of the Havelaar reputation system is given in Algorithm 1. Here, the algorithm is generalized by an *aging factor* $\gamma \leq 1$. However, since the aggregation vectors already include many observations from the past, using $\gamma = 0$ is fine for our purpose, but can be increased in order to account for longer absences from the system.

[3]Due to the hash function, some peers will have slightly more, others slightly less predecessors.

[4]To verify the predecessor identifier, public/private-key pairs among peers are presumed.

Algorithm 1 Simplified Havelaar Reputation System

1: **observe** $\vec{o}_0$;
2: **receive** $O_1 := [\vec{o}_1, \ldots, \vec{o}_r], \ldots, O_k$ **from** predecessors;
3: **for** $j \in [1, r]$: $\vec{o}_j = \sum_{i=1}^{k} O_{ij}$;
4: $\vec{c} = \gamma\vec{c} + (1-\gamma)(\sum_{i=0}^{r} \vec{o}_i)$;
5: **send** $O := [\vec{o}_0, \ldots, \vec{o}_{r-1}]$ **to** k successors;

In Havelaar, a peer u increases the contribution values only after downloading fragments from other peers, but it *never decreases* any contribution values if it has to provide upload bandwidth to some peer. This has the drawback that if two peers have contributed to the system equally, they will be assigned the same amount of download bandwidth, *independently* of their downloading behavior—it is questionable whether this is fair. However, as mentioned, we do not want to provide any disincentives for downloading in our network.

But the behavior of *excessive downloads* should be discouraged. Such downloads could trigger a vicious circle: because of excessive downloads, the network is congested, which in turn encourages other users to download in advance, resulting in an even more congested network. Therefore, in Havelaar, each peer u additionally maintains a second vector $\vec{d}$ (downloads). After another peer downloads from u, u will increase the download value of that peer. As opposed to the observation vector, the download vector will not be sent to other peers. Before allocating resources among competing peers, the download values will be subtracted from the respective contribution values of $\vec{c}$, and only then used to allocate the bandwidth. Since excessive downloaders are more likely to be involved in repeated interactions, they are eventually slowed down.

Finally, note that downloads could of course also be treated differently. For instance, downloads could be punished with a mechanism similar to the one used to reward uploads.

VI. Analysis

A. Overview

Assume that two peers u and v compete for the same upload bandwidth of a given peer w. In order to achieve the desired fairness, peer w should allocate the bandwidth to u and v according to their real, i.e., global, contribution values C_u^g and C_v^g, i.e., with respect to all transactions to which they have contributed. However, in Havelaar, peer w does not have precise information about C_u^g and C_v^g, but only knows the local approximations C_u^l and C_v^l (values from its contribution vector $\vec{c}$). Hence, we want to achieve a good approximation $C_u^g/C_v^g \approx C_u^l/C_v^l$ such that the peers indeed receive the corresponding share of the bandwidth.

In this section, we will analyze how many observations x are needed such that the ratios of the values in the local vectors $\vec{c}$ are an acceptable approximation of the ratios of the real contributions. Consequently, we can compute the number of rounds r that are necessary in Havelaar to get the required number of (aggregated) observations. Henceforth, let C_u and C_v denote C_u^l and C_v^l, respectively.

Our network consists of n peers, not all of which are always online. We simplify the analysis by assuming that at any time exactly $m < n$ peers are online. Furthermore, we assume that each peer downloads t fragments from other peers; the transactions are assumed to be distributed uniformly at random among the peers and over time. Hence, the probability of downloading a fragment from a given peer is $p = 1/m$.

B. Analysis

Whenever a peer downloads from peer u, it increases the contribution value C_u of peer u. Let us first assume that bandwidth and fragment sizes are equal to 1, that is, C_u is increased by 1 after each download from u. As explained before, peer u is chosen with probability p for every download. Therefore, C_u is a random variable. What is the probability distribution of C_u after x downloads?

This situation corresponds to a *balls-into-bins problem* [24], where x balls are tossed into m bins, and where $p = 1/m$ is the probability that a tossed ball lands in any given bin. For a given bin u, the ball tossing process is a sequence of x random, independent *Bernoulli trials*, each with a probability p of success. Therefore, the random variable C_u follows a *binomial distribution* $C_u \sim Bin(x, p)$, where $\mu_{C_u} = E(C_u) = x \cdot p$ and $\sigma^2_{C_u} = Var(C_u) = x \cdot p \cdot (1 - p)$.

This assumes that peers u and v are online all the time and can thus be chosen for all x transactions. In reality, however, some peer u might be online much longer than some other peer v. Therefore, u is likely to be involved in more transactions and will hence also contribute more to the network. This should be reflected in the local approximations C_u and C_v.

Let us assume that peer u is online with a fixed probability p_u, and peer v with probability p_v. Based on the assumption that the transactions are distributed uniformly over time, peer u can only be chosen for $x_u = p_u x$ transactions on average, and peer v for $x_v = p_v x$ transactions.

Furthermore, since the ultimate goal of Havelaar is to encourage high upload bandwidth, we need to include the provided upload bandwidth into the model. Let us assume that peer u uploads fragments at a fixed bandwidth of b_u, and peer v at a bandwidth of b_v. Instead of adding 1 to the contribution value of each peer, the corresponding *bandwidth* is added.

Putting everything together, the mean of C_u is given by $\mu_{C_u} = E(C_u) = b_u \cdot p_u \cdot x \cdot p$, and the variance is $\sigma^2_{C_u} = Var(C_u) = b_u^2 \cdot p_u \cdot x \cdot p \cdot (1 - p)$. Note that the bandwidth b_u is multiplied twice in the variance (b_u^2): The variance in the contribution C_u does not increase linearly in the bandwidth, but quadratically.

Of course, our model can be extended in several ways, for example by incorporating variable fragment sizes or issues of contention. However, this would be overly exact (cf technical report [13]); in order to keep things simple, we restrict ourselves to the main factors, omitting this generalization in our analysis.

For small x, the *coefficient of variation* σ_{C_u}/μ_{C_u} (or, similarly, also the *variance-to-mean ratio* $\sigma^2_{C_u}/\mu_{C_u}$) is large. However, for $x \to \infty$, it converges to 0. This indicates that with lots of observations, relative estimates become accurate enough. We are interested in the ratio of the contribution values of two peers competing for resources at the same time. Therefore, let us introduce a random variable Z reflecting this ratio: $Z := C_u/C_v$.

What is the mean and the variance of Z? Since C_u and C_v are independent, the following approximations are reasonable [26]:

$$\mu_Z = E(Z) \approx \frac{\mu_{C_u}}{\mu_{C_v}} + \sigma^2_{C_v} \frac{\mu_{C_u}}{\mu^3_{C_v}},$$

and

$$\sigma^2_Z = Var(Z) \approx \sigma^2_{C_v} \frac{\mu^2_{C_u}}{\mu^4_{C_v}} + \frac{\sigma^2_{C_u}}{\mu^2_{C_v}}.$$

As can be seen from these formulas, for $x \to \infty$, the variance decreases quickly, and hence, for lots of observations x, Z is a good approximation of the ratio of the real contributions of peers u and v. In the following subsection, we will make use of another helpful approximation of the coefficient of variation [29]:

$$\left(\frac{\sigma_Z}{\mu_Z}\right)^2 \approx \left(\frac{\sigma_{C_u}}{\mu_{C_u}}\right)^2 + \left(\frac{\sigma_{C_v}}{\mu_{C_v}}\right)^2 \quad (1).$$

C. Results

We can now estimate the number of observations necessary for a good approximation, that is, for small coefficients of variation of Z. Plugging μ_{C_u}, σ_{C_u}, μ_{C_v}, and σ_{C_v} into (1) yields

$$\begin{aligned}\left(\frac{\sigma_Z}{\mu_Z}\right)^2 &= \left(\frac{\sqrt{b_u^2 p_u x(p - p^2)}}{b_u p_u x p}\right)^2 + \left(\frac{\sqrt{b_v^2 p_v x(p - p^2)}}{b_v p_v x p}\right)^2 \\ &= \frac{1-p}{p_u x p} + \frac{1-p}{p_v x p}.\end{aligned}$$

Solving this for x gives the following fact.

Fact 6.1: The number of observations needed in order to achieve a desired approximation (as expressed by the coefficient of variation) of the real contribution values is

$$x \approx -\frac{p_v(p-1) + p_u(p-1)}{\left(\frac{\sigma_Z}{\mu_Z}\right)^2 p_u p_v p}.$$

As an example, for a network with $n = 100,000$ peers, where at any time $m = \frac{1}{4}n = 25,000$ peers are online, and if we assume that $p_u = p_v = \frac{1}{4}$ (in Kangoo, storage peers are online for more than six hours per day), $x \approx 10^7$ observations are required for an acceptable coefficient of variation of 0.15.

Having computed the number of observations x which are approximately needed for an acceptable accuracy, we can now determine the number of aggregation rounds. Assuming that every peer makes t transactions (= observations), the number of rounds r is $r = \lceil \log_{\frac{x}{t}} \rceil$. For the above example of $x = 10^7$ observations, assuming that each peer makes $t = 5,000$ transactions and sends its observations to $k = 7$ peers, $r \approx 4$ rounds are already enough!

VII. Communication Costs

The Achilles' heel of Havelaar are the communication costs: Every peer has to send—for example, once a week—the aggregated observations to its successors. However, we believe that in many practical systems, the burden is tolerable. Moreover, as described in the technical report [13], various compression techniques—e.g., due to sparseness—can be employed to reduce the size of the messages further.

Concretely, we have computed Havelaar's estimated message size [13] depending on the number of rounds r and the number of

transactions t in the system, and assuming that the contribution values can be encoded with 8 bits each (in a simulation, the entropy was only ≈ 7 bits). We have then compared Havelaar to solutions where the contribution value is recorded in a DHT. In this approach, the contribution value of peer u is stored at a peer which is chosen based on a hash function $h(u)$. Since the contribution values in Havelaar are only approximations and since the local vector is only updated once per round, we have compared the communication costs to a DHT-based approach with the same level of approximation and the same amount of updates. That is, instead of updating the contribution value after each download and retrieving the contribution value before each upload, peers only update the contribution value in the DHT probabilistically (with a probability resulting in the same level of approximation as Havelaar), and they only update their local vector by a look-up operation once per round.

As an example, in a network with $n = 100,000$, $t = 5,000$, $k = 7$, and $r = 4$, the communication costs are ≈ 2.2 MB per peer and week. In comparison, an approximate DHT-based approach would require ≈ 3.7 MB. Thereby, the costs for the DHT-based approach are only a lower bound: More communication would be necessary in order to store the contribution values persistently. Havelaar scales much better in the number of transactions. For the same example, but with $t = 20,000$ transactions, the communication costs of Havelaar are ≈ 2.5 MB, compared to ≈ 13.8 MB of the DHT-based approach. Finally, for a network with $n = 1,000,000$ and $t = 40,000$, the communication costs in Havelaar are ≈ 23.3 MB and the DHT-based approach ≈ 87.7 MB. We refer the reader to the technical report [13] for a more detailed comparison.

In conclusion, although the costs of Havelaar can be relatively high for a small number of transactions (e.g., up to ≈ 1.73 MB for $t < 2000$ in a network of $n = 100,000$, compared to ≈ 1.51 MB for the DHT-based approach), our system scales much better than various forms of DHT-based solutions. Moreover, we believe that the communication costs are acceptable in many practical environments.

VIII. Attacks

Havelaar is designed to cope with peers aiming at selfishly consuming larger shares of resources than other peers. The fact that every peer can send its observations only to its k successors facilitates *local defenses*.

A peer uses several defense mechanisms. A receiver checks whether a sender is one of its predecessors, and otherwise ignores the report. Moreover, it can make sure that a peer does not report contributions too often.

A peer does not take into account observations about the reporting predecessor itself. Thus, the most attractive attack is made hard *by design*: It is only possible to falsely "praise" or "accuse" *another* peer.

In addition to limiting the range of possible values, other measures are taken in our system to detect and ignore false reports. Before updating the local contribution vector $\vec{c}$ on the basis of the observation matrices, for each value the average and variance is calculated. If one value is extremely large, it is considered an outlier with respect to the other $k-1$ values, and is dropped. Then, the average of the other $k-1$ values is taken as the input to the update function.

Clearly, one can think of several further local defense mechanisms. For instance, statistical measures could be included to detect a possible false report by studying the distribution (histogram) of the observation vector, e.g., by checking whether the histogram is spiked. In any of the above cases where the successor is suspicious of an attack, it could reduce the trust value associated with each predecessor. The trust value can be used to either drop observations by misbehaving peers, or to weigh their observation values accordingly. However, it has not been necessary to make use of these techniques so far.

Besides the advantages of local defense, the robustness of Havelaar comes from the *extensive aggregation*: A single wrong value hardly influences the overall outcome. In this sense, also the damage which can be done by a small fraction of colluding peers is limited. In particular, collusion is also made difficult by the fact that successor peers are determined by hash functions, and hence becoming a predecessor of a specific peer is hard.

In summary, Havelaar's design is based on local defense and extensive accumulation, and unlike many other approaches does not rely on transitivity of trust. This renders attacking the system a difficult endeavour. Finally, recall that several other attacks such as Sybil attacks and whitewashing are tackled by mechanisms which are part of Kangoo and hence are external to Havelaar.

IX. Simulation

We have performed several simulations of Havelaar which fortify our results. In this section, we present the most interesting findings.

In Figure 1, the real ratio of the contribution values of two peers is compared to the ratio from the local approximation in a network of size $n = 100,000$, with $k = 7$ successors and $r = 4$ rounds. In each round, the peers change their upload bandwidth. In the first round, for instance, peer u contributed exactly three times more than peer v. Note that the approximation is shifted to the right; this is due to the fact that contribution values are only updated once a round, based on observations from the past. Note also that the standard deviation of the approximation is generally low—being higher when peers change their behavior abruptly.

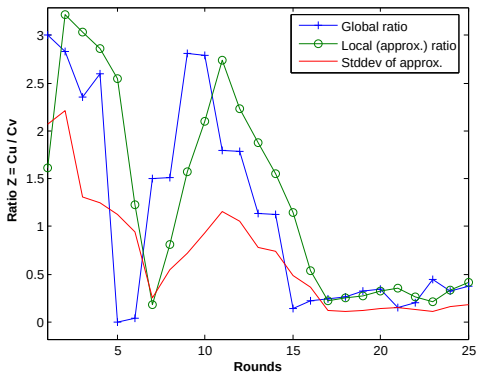

Fig. 1. Local approximation vs. global (real) contribution value of two peers in several rounds. Note that after a short bootstrapping phase in the beginning, the approximation becomes good quickly.

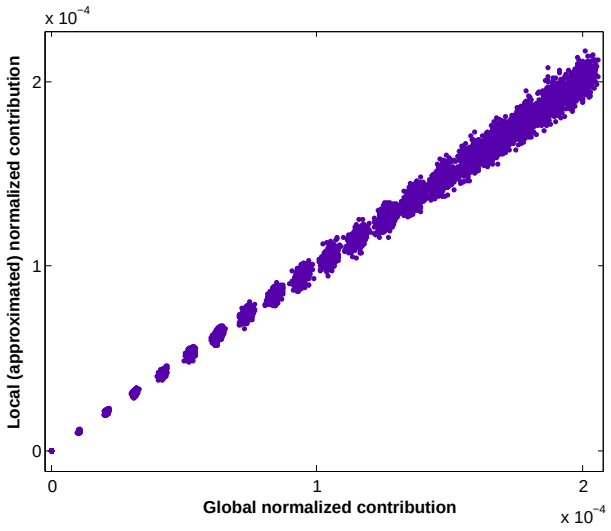

Fig. 2. Global normalized contribution value plotted against the locally approximated one.

The figure reveals that the bootstrapping process is quite fast, which implies that newly joining peers—or peers which return after a long period of absence—are up-to-date soon. Of course, this delay may still be unacceptably large for many systems where appropriate solutions which further speed up the process would be needed. However, this is not the case in Kangoo, as peers are expected to remain in the system for months or even years.

Figure 2 plots the local contribution vector for the same network against the global (real) contribution vector. Both vectors are normalized by dividing each entry by the sum of the whole vector. That is, each contribution value reflects the proportional contribution of the entire network. Again, the local approximation reflects the real contribution accurately (almost a straight line). For peers with higher contribution, however, the variance becomes larger. This is due to the fact that the variance is multiplied by the square of the bandwidth, as described in Section VI.[5]

X. Conclusions

The main goals of the Havelaar reputation system are (1) accurate estimation of the real contribution values of other peers, (2) robustness to selfish peers, and (3) efficiency, i.e., scalability in the number of transactions. This is achieved by a novel aggregation technique where peers always report the observed contributions values to the same set of peers. This allows for a local control of a peer's behavior. Encouraged by our results, we have integrated Havelaar in our distributed storage system Kangoo. We believe that Havelaar is a good choice for many active p2p systems requiring a fairness mechanism.

[5]Note, however, that the coefficient of variation is constant for every peer since it does not depend on the bandwidth.

References

[1] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proc. of the 10th Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 310–317, 2001.

[2] E. Adar and B. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), 2000.

[3] R. Axelrod. The Evolution of Cooperation. *Science*, 211(4489):1390-6, 1981.

[4] D. Banerjee, S. Saha, S. Sen, and P. Dasgupta. Reciprocal Resource Sharing in P2P Environments. In *Proc. 4th AAMAS*, 2005.

[5] S. Buchegger and J.-Y. L. Boudec. A Robust Reputation System for P2P and Mobile Ad-hoc Networks. In *Proc. 2nd Workshop on the Economics of Peer-to-Peer Systems*, 2004.

[6] B. Cohen. Incentives Build Robustness in BitTorrent. In *Proc. Workshop on Economics of Peer-to-Peer Systems*, 2003.

[7] J. R. Douceur. The Sybil Attack. In *Proc. 1st Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pages 251–260. Lecture Notes in Computer Science (LNCS), Springer, 2002.

[8] M. Feldman and J. Chuang. Overcoming Free-Riding Behavior in Peer-to-Peer Systems. *ACM Sigecom Exchanges*, 6, 2005.

[9] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust Incentive Techniques for Peer-to-Peer Networks. In *Proc. ACM Conf. on Electronic Commerce*, 2004.

[10] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and Whitewashing in Peer-to-Peer Systems. In *Proc. ACM SIGCOMM Workshop PINS*, 2004.

[11] F. D. Garcia and J.-H. Hoepman. Off-Line Karma: A Decentralized Currency for Peer-to-Peer and Grid Applications. In *Proc. 3rd Applied Cryptography and Network Security (ACNS)*.

[12] P. Golle, K. Leyton-Brown, and I. Mironov. Incentives in Peer-to-Peer File Sharing. In *Proc. 3rd ACM Conf. on Electronic Commerce (EC)*, 2001.

[13] D. Grolimund, L. Meisser, S. Schmid, and R. Wattenhofer. Havelaar: A Robust and Efficient Reputation System for Active Peer-to-Peer Systems. Technical report, TIK Report 246, available at http://www.tik.ee.ethz.ch/. ETH Zurich, Switzerland, 2006.

[14] D. Hughes, G. Coulson, and J. Walkerdine. Free Riding on Gnutella Revisited: The Bell Tolls? *IEEE Distributed Systems Online*, 6(6), 2005.

[15] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *Proc. WWW*, pages 640–651, 2003.

[16] F. Kuhn, S. Schmid, and R. Wattenhofer. A Self-Repairing Peer-to-Peer System Resilient to Dynamic Adversarial Churn. In *Proc. 4th Int. Workshop on Peer-To-Peer Systems (IPTPS), Ithaca, New York, USA*, February 2005.

[17] K. Lai, M. Feldman, J. Chuang, and I. Stoica. Incentives for Cooperation in Peer-to-Peer Networks. In *Proc. Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, 2003.

[18] Q. Lianz, Y. Pengx, M. Yangx, Z. Zhangy, Y. Daix, and X. Li. Robust Incentives via Multi-level Tit-for-Tat. In *Proc. 5th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.

[19] R. B. Ma, S. M. Lee, J. S. Lui, and D. Y. Yau. A Game Theoretic Approach to Provide Incentive and Service Differentiation in P2P Networks. In *SIGMETRICS*, 2004.

[20] S. J. Nielson, S. Crosby, and D. S. Wallach. A Taxonomy of Rational Attacks. In *Proc. 4th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pages 36–46, 2005.

[21] A. M. Odlyzko. The Case Against Micropayments. In *Financial Cryptography*, pages 77–83, 2003.

[22] T. G. Papaioannou and G. D. Stamoulis. Effective Use of Reputation of Peer-to-Peer Environments. In *Proc. IEEE/ACM CCGRID 2004, GP2PC Workshop*, 2004.

[23] T. G. Papaioannou and G. D. Stamoulis. Reputation-based Policies that Provide the Right Incentives in Peer-to-Peer Environments. *Computer Networks*, 50(4):563–578, 2006.

[24] M. Raab and A. Steger. "Balls into Bins" - A Simple and Tight Analysis. In *Proc. 2nd Int. Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 159–170. Springer-Verlag, 1998.

[25] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content Addressable Network. In *Proc. of ACM SIGCOMM 2001*, 2001.

[26] J. A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, 1995.

[27] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. 18th IFIP/ACM Int. Conf. on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[28] J. Shneidman and D. C. Parkes. Rationality and Self-Interest in Peer to Peer Networks. In *Proc. 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[29] W. Stahel. *Statistische Datenanalyse. Eine Einfuehrung fuer Naturwissenschaftler*. Vieweg, Braunschweig, 2000.

[30] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. ACM SIGCOMM Conference*, 2001.

[31] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. KARMA: A Secure Economic Framework for P2P Resource Sharing. In *Proc. P2PEcon*, 2003.

[32] W. Wang and B. Li. Trust Based Incentive in P2P Network. In *Proc. Int. IEEE Conf. on E-Commerce Technology for Dynamic E-Business (CEC-East)*, pages 302–305, 2004.

[33] W. Wang and B. Li. Market-driven Bandwidth Allocation in Selfish Overlay Networks. In *Proc. IEEE INFOCOM*, 2005.