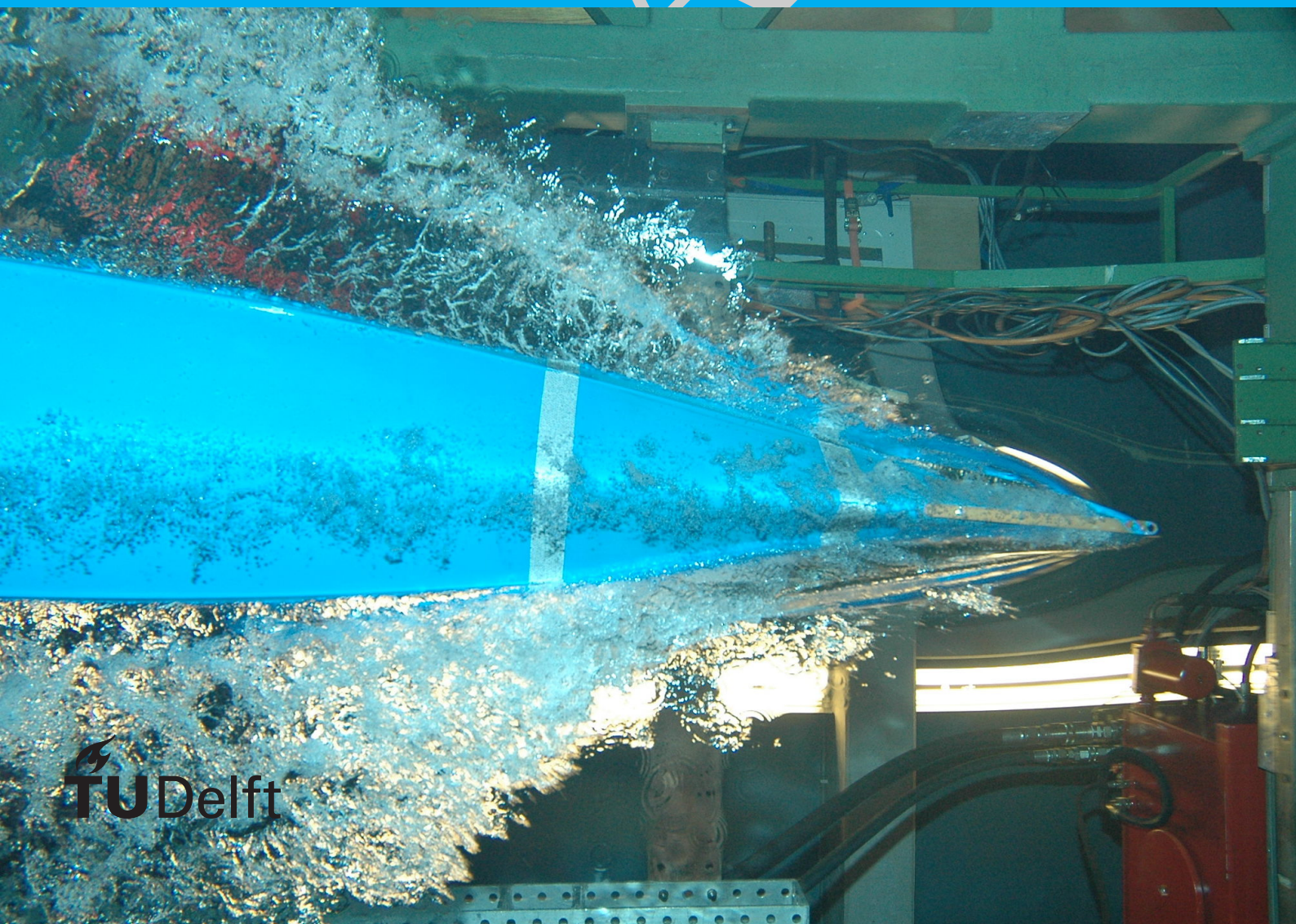
Sybil-proof Accounting Mechanisms in P2P Networks

Alexander Stannat

AET



TU Delft

Sybil-proof Accounting Mechanisms in P2P Networks

by

Alexander Stannat

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday January 1, 2013 at 10:00 AM.

Student number:	1234567	
Project duration:	March 1, 2012 – January 1, 2013	
Thesis committee:	Prof. dr. ir. J. Doe,	TU Delft, supervisor
	Dr. E. L. Brown,	TU Delft
	Ir. A. Aaronson,	Acme Corporation

This thesis is confidential and cannot be made public until December 31, 2013.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Abstract…

Preface

Preface…

Alexander Stannat
Delft, January 2013

Contents

1

Introduction

1.1. Cooperation in Peer-to-peer Networks

1.1.1. The Evolution of Cooperation

Mechanisms for Social Cooperation

1.1.2. Reputation and Behaviour on the Internet

Reputation vs Trust

1.1.3. Distributed Computing Systems

Peer-to-peer File Sharing Networks
BitTorrent & Tit-for-Tat
Tribler

1.1.4. Sybil Attacks

Twitter, Facebook and File Sharing Examples

1.1.5. Misreport Attacks

Blockchains & Trustchain

1.1.6. Other Attacks

Eclipse, Whitewashing, DDoS, Spamming, etc.

1.2. Problem Description

1.3. Related Work

1.4. Our Contributions

1.5. Thesis Summary

2

Mathematical Framework for Accounting Mechanisms

We begin by introducing a mathematical framework for the setting in which we conduct our research, namely by rigorously formalising interactions (transactions) between nodes in the network. transaction between nodes. In [**?**] Otte et al. introduced the concept of an *ordered interaction model* from which an *ordered interaction graph* and a *block graph* are derived. While this is a very elegant definition for a set of transactions and the derivation of a work graph from it, it is directly taylored to the TrustChain architecture and lacks the possibility of misreports and counterfeit interactions. Therefore we will not adopt it here, but instead derive a slightly different and more generic definition of a transaction set, which will be our equivalent to their ordered interaction model.

2.1. Transaction Sets

We start off with the definition of a simple network transaction, or interaction, which simply denotes the transferrence of data in between two nodes.

Definition 2.1.1 (Network Transactions)**.**
Let V be the set of all agents in the network. A transaction t between two nodes $i, j \in V$ is given by a tuple (i, j, w, n), whereby $pr_1(t)$ is the contributer, while $pr_2(t)$ is the recipient of the work performed. Hence, the ordering of the two nodes in the transaction tuple is not arbitrary. Naturally, it always holds that $w \geq 0$. w corresponds to the size of the transaction, i.e. the amount of data transferred from i to j.

Note that a transaction can only ever go "one way". This means that a single transaction cannot contain the transference of data from node i to node j **and** vice versa.

As every node participates in a string of transactions in a given chronological order, we obtain a series of transactions for every node i, which we will refer to as a transaction sequence.

Definition 2.1.2 (Transaction Sequence)**.**
The transaction sequence of a node $i \in V$ is expressed as $TS_i := (t_n^i)_{n \in \mathbb{N}_{\leq T_i}}$ where t_n^i is the n-th transaction node i participated in, either as a contributor or as a consumer. As above t_n^i is given by a tuple (j, k, w) where either $j = i$ or $k = i$. T_i denotes the length of i's transaction sequence, i.e. the number of transactions i has participated in thus far. It grows as time progresses.

Note that in this definition we implicitly assume a single node can only participate in one transaction at a time. Else, the ordering of transactions would become sonsensical. Next, we define a transaction function, which will denote the size of a transaction. As time goes on, transaction sequences obtain new entries and continue to grwo, which implies that T_i is not a static value, but changes over time. We choose not to incorporate a temporal variable in this model and instead assume that a transaction sequence represents a "snapshot in time" as opposed to a dynamic variable.

Definition 2.1.3 (Transaction Function)**.**
For every node $i \in V$ we define a transaction function t^i, given by

$$t^i : \mathbb{N}_{\leq T_i} \times V \to \mathbb{R},$$

where $t^i(m, j)$ corresponds to the amount of work node i has leeched from or contributed to node j in its m-th transaction, i.e.

$$t^i(m,j) = \begin{cases} pr_3(t^i_m), & \text{if} \quad pr_2(t^i_m) = j \\ -pr_3(t^i_m), & \text{if} \quad pr_1(t^i_m) = j \\ 0, & \text{else} \end{cases}$$

Note that it holds

$$t^i(m,j) > 0 \text{ if } t^i_m = (i, j, w)$$

and

$$t^i(m,j) < 0 \text{ if } t^i_m = (j, i, w)$$

Remark 2.1.1.
In theory it should always hold for any pair of nodes $i, j \in V$:

$$\forall i, j \in V, n \in \mathbb{N},\ t^i(n,j) = w \neq 0 : \exists m \in \mathbb{N},\ t^j(m,i) = -w$$

What this means is that any transaction between nodes i and j that is contained in the transaction sequence of i must also be contained in the transaction sequence of node j. This is quite trivially true.

Finally, we introduce the set containing all transactions that have transpired in the network, denoted by

$$TS := \{TS_i \mid i \in V\}$$

This set contains all transaction sequences of all nodes in the network. Based on our remark 2.1.1 we see that TS must contain every transaction exactly twice.

2.2. Work Graphs

Given the set of all transactions TS, one can transform the transaction sequences into a *work graph*. A work graph is a directed network graph visualising the interaction between nodes. It may be unidirectional or even a multigraph. The idea is that edges between vertices correspond to overall seed-leech relationships of nodes in the network.

Definition 2.2.1 (Work Graph)**.**
A work graph is given by the tuple $G = (V, E, w)$, whereby V is the set of vertices, i.e. agents in the network and E is a set of directed edges between the community agents. An edge $(i, j) \in E$ pointing from node i to node j represents node j performing work for node i.
The function $w : V \times V \to \mathbb{R}_{\geq 0}$ denotes the weight of the edges, i.e. $w(i, j)$ represents the total amount of work performed by node j for node i. If two nodes i and j are not connected then we set the edge weights $w(i, j) = w(j, i) = 0$. We choose the set of edges $E \subset V \times V$ such that $(i, j) \in E$ if and only if $w(i, j) > 0$.

There are a number of different ways transactions in TS can be aggregated to form edges in the work graph. In the *unidirectional, single-edge* case of the work graph the edges of the graph can be derived from the set of all transactions TS by

$$w(i,j) = \max\left\{\sum_{n\in\mathbb{N}} t^j(n,i), 0\right\} = \max\left\{-\sum_{n\in\mathbb{N}} t^i(n,j), 0\right\}$$

and consequently,

$$w(j,i) = \max\left\{\sum_{n\in\mathbb{N}} t^i(n,j), 0\right\} = \max\left\{-\sum_{n\in\mathbb{N}} t^j(n,i), 0\right\}$$

Here the weight of the edges corresponds to the net data flow in between two nodes. The edge is directed toward the node that has a positive deficit in the bilateral relationship. Note that there can only be a single edge connecting two nodes, which points from one to the other, i.e. for any pair of nodes $i, j \in V$ it holds $w(i,j) > 0 \Rightarrow w(j,i) = 0$. This type of work graph is quite useful as it nicely captures the overall net contributions nodes have made to the network, epsecially when keeping definition (lazy freeriding) in mind.

There are some advantages to this type of graph such as

Think of advantage single-edge graph.. plicity for one

Note that there is one drawback to this approach, which lies in the fact that the single-edge graph neglects certain contributions made to the network. For instance, if two agents have donated the same amount of resources to one another then the weight of the edge connecting them is zero. Hence, the edge has "disappeared" and the nodes appear to be Think of example where this is really problematic disconnected again
This will prove problematic at times and will enable a certain type of attack which we call a serial sybil attack in **??**.

Alternatively, in the case of a multigraph we can derive the edge weights as follows.

$$w(i,j) = \sum_{n\in\mathbb{N}} \max\left\{t^j(n,i), 0\right\} = \sum_{n\in\mathbb{N}} \max\left\{-t^i(n,j), 0\right\}$$

and

$$w(j,i) = \sum_{n\in\mathbb{N}} \max\left\{t^i(n,j), 0\right\} = \sum_{n\in\mathbb{N}} \max\left\{-t^j(n,i), 0\right\}$$

In this particular type of graph an edge (i,j) corresponds to the gross data flow from j to i without subtracting the work i has done for j. A positive attribute of this this type of graph is that it's generally more informative as it doesn't reduce edge weights to net data flow. **However, there are some disadvantages this type of graph may have. A particular example of this is given by a node i that has contributed x units of data to another node j and then leeches the same amount from j later. The work graph in this scenario satisfies $w(i,j) = x = w(j,i)$. In a later chapter, we will discuss that in order for accounting mechanisms to be sybil-proof they must satisfy two , which satisfy...**

It may be somewhat counterintuitive for edges to be pointing from the recipient to the contributor. Note that we can invert the edges as well, such that an edge (i,j) pointing from i to j corresponds to work performed by i for j, in that case we obtain.

$$w(i,j) = \sum_{n\in\mathbb{N}} \max\left\{t^i(n,j), 0\right\} = \sum_{n\in\mathbb{N}} \max\left\{-t^j(n,i), 0\right\}.$$

However, we choose to stick with the former direction with a particular set of accounting mechanisms in mind. Although, this is quite irrelevant, as it does not change the nature of the problem at all. **We chose to work with the ■ work graph as it is the simplest one that does not bring about any problems later on.**

Remark 2.2.1.
Note that in our case we introduce the work graph with regard to peer-to-peer filesharing keeping the ap-

Add 3 Example gra each scenario

plication of the *Tribler* network in mind. This means that the work performed, i.e. the weight of the edges corresponds to the amount of data transferred from one node to another, by seeding and leeching respectively. Recall our examples of social network, such as Facebook and Twitter in subsection 1.1.4. We would like to extend our model to entail these networks and any kind of p2p network in general. In the case of these social networks we can apply the exact same concepts, however the transactions between agent would then correspond to "follow" or "friendship" relations, etc. The weights of these edges could then, for instance, be determined by the number of likes and/or retweets a user receives from a follower/friend.

For instance a follower relationship on *Twitter* will correspond to an edge pointing from the follower to the followed, while the edge weight may correspond to the number of tweets that have been liked or retweeted, etc. In this particular application a bidirectional graph will be more reasonable than a unidirectional one. In the case of *Facebook* a friendship may then correspond to a bidirectional edge connecting two vertices, while the amount of likes and/or mentions these people receive from one another, could be represented by another pair of edges connecting the two.

2.3. Accounting Mechanisms & Allocation Policies

2.4. Misbehaviour & Attacks

3

Sybil Attack Gain

3.1. Cost & Profit in Terms of Work

3.1.1. Interaction Model

Allocation Policies

3.1.2. Experimental Evaluation

3.2. Cost & Profit in Terms of Accounting Values

4

Representativeness of Accounting Mechanisms

4.1. Sybil-proofness of Accounting Mechanisms
4.2. Defining Representativeness

5

On the Impossibility of Sybil-Proofness

6

Sybil-Proofness of Accounting Mechanisms

6.1. Characterising Sybil Attacks
6.2. Requirements for Sybil-Proofness

7

Sybil Resistance Based on Physical Proximity

7.1. A Mathematical Framework for Physical Proximity
7.2. A Two-layered Trust Model

8

Conclusion

A

Appendix

A.1. Reputation Dynamics in Indirect Reciprocity

A.1.1. Requirements for Reputation Mechanisms

A.1.2. Leading 8