Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Sybil-resistant trust mechanisms in distributed systems

A thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirements

for the degree

MASTER OF SCIENCE
in
APPLIED MATHEMATICS

by

PIM OTTE

Delft, the Netherlands
December 2016

MSc THESIS APPLIED MATHEMATICS

"Sybil-resistant trust mechanisms in distributed systems"

PIM OTTE

Delft University of Technology

Daily supervisors	**Responsible professor**
Dr. D. C. Gijswijt	Prof. dr. ir. K. Aardal
Dr. J. Pouwelse	

December 2016 Delft, the Netherlands

Abstract

In this thesis, the problem of estimating trust in distributed systems is considered. Distributed systems can be virtual or real-world systems in which multiple agents interact. One of the biggest problems in distributed systems is that they only function well if everyone contributes some resources. If agents do not participate, or try to pretend they participate, but in fact are slacking off, then the system might not achieve its desired purpose. This work presents two methods to estimate how well agents contribute to the network, while preventing cheating. Firstly, the NetFlow algorithm, which uses max-flow computations to bound the profit of cheating by Sybil attacks. Secondly, the Temporal PageRank algorithm, which uses information about the ordering of the interactions to provide a robust mechanism to determine reputation. Theoretical guarantees about several aspects of these algorithms are provided. Furthermore, these mechanisms are tested on data from a real-world distributed system: Tribler, an anonymous peer-to-peer downloading system. Finally, taking both the theoretical and practical results, the broader implications of these mechanisms and future possibilities are explored.

Preface

This thesis is the capstone to my master Applied Mathematics at TU Delft. As this final project comes to an end, I am very satisfied with the work that follows. While I do not feel this is necessarily my best mathematical work in the pure sense, I am happy to have been able to apply my mathematical knowledge to the field of distributed systems and specifically to one system which supports privacy and security. I feel that these areas are key in ensuring that the digital world remains at least as open and free as it has been in the past decade.

There are a few people who have supported me in this endeavour who I would like to thank. My daily supervisor, Johan Pouwelse. His ideas and contagious enthusiasm have kept me inspired and motivated throughout. Dion Gijswijt, who as mathematical supervisor has brought the necessary rigour and precision where desired. I thank both of them for their input and I could not have wished for more balanced pair of supervisors. I thank Pim and Ewout, fellow members of the "MultiChain team", as well as office roommates Tim and Jesse for their input during sparring sessions. Stefan and Hassan have my gratitude for their proofreading of this work. Of course, any and all remaining flaws are my responsibility alone. I thank all students and staff on the 7th floor for their creation of an enjoyable working environment, as well as pizza/gaming night. I thank all my mathematical friends who have tolerated my defection from mathematics.

Finally, my parents, Arien and Bernard, and my sister Lidwien. They have supported and encouraged me throughout the entirety of the journey that has lead me to this point. They have shown interest and encouraged me to think critically. I thank them for all of this.

Pim Otte

Contents

1 Introduction **11**

2 Research question **15**
2.1 Fairness . . . 16
2.2 Resistance to manipulation . . . 16
2.3 Efficiency of computation . . . 16

3 Base Model of Interaction Information **19**
3.1 Base model . . . 19
3.2 Base model applied to Tribler . . . 23

4 Accounting mechanisms and Sybil-proofness **25**
4.1 Accounting mechanisms and DropEdge . . . 26
4.2 Defining accounting mechanisms . . . 27
4.2.1 Properties of accounting mechanisms . . . 28
4.2.2 Defining Sybil attacks . . . 29
4.3 Bounding Sybil attack profit . . . 30
4.3.1 Improving informativeness by scaling . . . 33
4.4 Attacks on Net-Flow . . . 35
4.4.1 Propagation slowness . . . 35
4.4.2 Partial network visibility . . . 35
4.4.3 Collusion attack . . . 35
4.5 Theoretical Performance of NetFlow . . . 36

5 Considering Time **39**
5.1 Random walks in the Ordered Interaction Graph . . . 39
5.2 Computational aspects of Temporal PageRank . . . 40
5.3 Properties of Temporal PageRank . . . 41

6 Experimental Evaluation **45**
6.1 Data collection . . . 45
6.2 NetFlow evaluation . . . 45
6.3 Temporal PageRank Evaluation . . . 50
6.4 Cross-mechanism Evaluation . . . 53
6.5 Performance Evaluation . . . 55

7 Conclusion and Discussion **57**
7.1 Fairness . . . 57
7.2 Resistance to Manipulation . . . 57
7.3 Performance . . . 58

7.4 General conclusion 58
7.5 Discussion 58
7.5.1 Future Research 58
7.5.2 This Research in Broader Context 59

Chapter 1

Introduction

Nowadays, the Internet is an absolutely critical piece of infrastructure. According to the International Telecommunication Union, 47% of the world population uses the internet. In the developed world, 81% of the inhabitants use the internet [28]. The Internet is used for important communication and banking, education, but also for entertainment and leisure. Perhaps surprisingly, Internet access has not quite made it to being a basic human right. However, the United Nations Special Rapporteur on the promotion and protection of the right to freedom of opinion and expression, wrote the following on the matter in a report [25]:

> "*there should be as little restriction as possible to the flow of information via the Internet, except in few, exceptional, and limited circumstances prescribed by international human rights law.*"
>
> -Frank La Rue

This leaves little doubt that the availability of the internet is protected by Article 19 of the Universal Declaration of Human Rights "freedom of opinion and expression"[2].

A more controversial issue has been going on for longer than it is known to the general public. In 2013, Edward Snowden disclosed the first of several pieces of information that have shown digital communication, the internet included, is subject to extensive surveillance. Also known as the "Snowden Revelations", the information released consists of a series of documents and accompanying articles. The documents originate from several intelligence agencies and detail the scale and methods with which these agencies perform digital surveillance. About this, David Lyon writes the following [20]:

> "Given the reliance on western liberal legal traditions it is hardly surprising that public debate generally commences around the question of **privacy**. Understood as a human right, it underlies aspects of democratic polity, such as freedom of expression. Often understood in the post-Snowden era as relating to control of communications about oneself, it is clearly a threatened value if not according to some a forlorn hope."
>
> -David Lyon

Hence, if we value the human rights to freedom of expression and privacy, then an internet with entities performing unbounded surveillance is not desirable.

On the one hand, even back in 2001 there was a growing consensus that the internet poses a threat to the existence of authoritarian regimes [15]. On the other hand, regimes like China [8] and Iran [17] try their very best to censor content.

The solutions to ensuring availability of the internet and its services appears to have one thing in common. All systems aiming to do so leverage the concept of a *distributed system*. Distributed systems are networks of computers that all perform part of a common task. One of the advantages of distributed systems is that, if well designed, they are extremely hard to take down. One of the disadvantages of a distributed systems is that there will be difficulties in the distribution, in particular if the system is actually physically distributed.

A prime example of using a distributed system to increase availability is the BitTorrent protocol [3]. This protocol enables a technique called peer-to-peer filesharing. In this paradigm, many computers are in possession of (part of) a file. Computers that do not possess the entire file can request pieces of it from other computers (also known as peers). This protocol ensures the availability of this file. If any peer stops functioning, the rest of the network can continue to exist as is, as long as all active peers together have the entire file. The BitTorrent protocol enables availability. It provides a technical solution to technical censorship. However, in this protocol it is trivial for an entity to find out who is downloading or uploading a certain file. The whole protocol is built to find others who are downloading the same content. This means that political censorship can not be circumvented using BitTorrent, as one might be able to obtain the data, but would then be arrested afterwards.

An example of a distributed system that ensures both availability and privacy is Tor [4]. Tor uses a so-called onion routing protocol. The concept of onion routing is as follows. One obtains a list of peers in the network, at least one of which needs to be available as "exit node". If one then wants to communicate with the internet, any request that would normally be sent directly is instead routed through two or more of these peers, the last of which must be an exit node. This is done by wrapping the request in one layer per peer. These layers constitute the onion. The way this is done is such that each peer in the network knows the previous and next link in this chain, but no one except the originator knows the entirety. The exit node is the last hop, and they will send this on to the recipient on the internet. Hence, they know everything that is not encrypted about this request and response, which includes at least the recipient. See Figure 1.1 for a visual representation. The aim of using an onion protocol is to hide the identity of the browsing party. Indeed, their identity can only be revealed if too many of the peers in the chain are controlled by the same party. This protocol can be applied to any sort of web traffic, including BitTorrent traffic.

The techniques of BitTorrent and Tor have both been incorporated into another piece of software: Tribler. Since both of these techniques are quintessentially distributed, naturally Tribler itself works as a distributed system. This piece of software unifies several key pieces of technology to provide anonymous BitTorrent to its users. In addition, it provides a distributed search engine for torrent files and a built-in media player in order to stream a torrent.

While this piece of software may at first glance seem to be the ultimate tool of an internet pirate, allow us to consider it in the light of the human rights mentioned above. Due to its distributed nature, Tribler is much less susceptible to traditional filtering and censoring techniques. In particular, unless all connectivity to a peer is severed, there is very little an authoritarian regime can do once a user gets their hands on Tribler. This enables information to freely flow from peer to peer and hence enables Frank La Rues vision of as little restriction as possible. Furthermore, the other side of the human rights equation considers the human right to privacy. That Tribler offers this is not a point of discussion. What might be considered is the question of whether Tribler offers too much privacy. Does it prevent law enforcement from doing their job? Does Tribler impact the war on terror? One might argue that Tribler does in fact offer too strong of a protection. However, the legal measures law enforcement can take, like a digital wiretap specific to a person or computer, are not prevented by Tribler. What is prevented is the style of mass surveillance uncovered by the Snowden revelations.

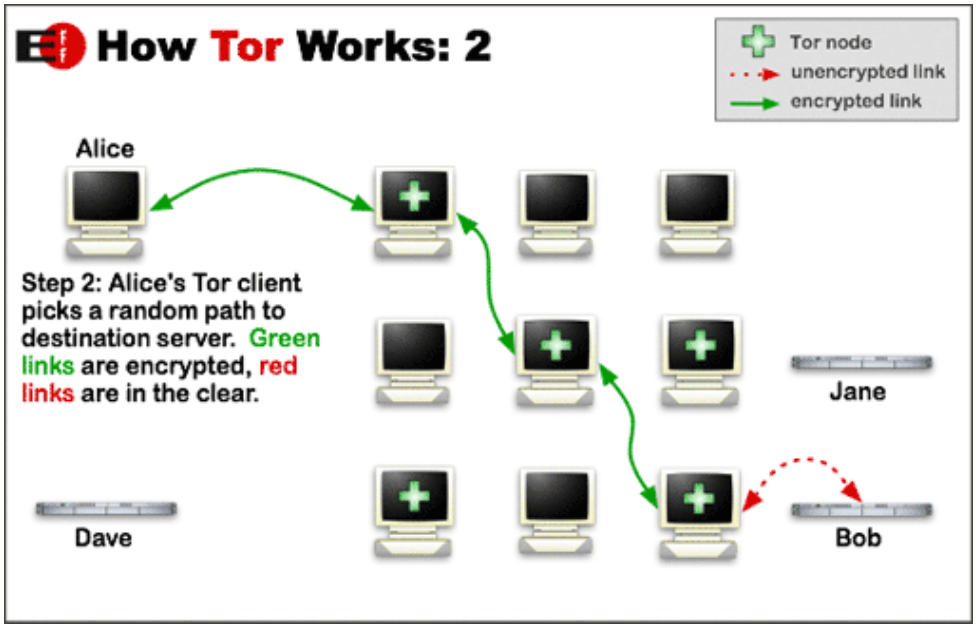


Figure 1.1: How Tor works. The Tor Project, Inc.[1] is the owner of this image. Used under the Creative Commons Attribution 3.0 United States License[2]

Distributed systems seem to be all sunshine and no rain. Protecting human rights and ensuring availability of services. However, the decentralization does pose certain problems. One problem in particular is resource management. A decentralized system like Tribler has many of the same properties as the climate change problem. Most of the world is aware climate is changing and that a change in their behaviour would help to mitigate this problem. In fact, if the world as a whole does not alter their behaviour, there might be catastrophic consequences. However, the impact a single person or household has is extraordinarily small. So small, that most people reason that their behaviour does not impact the world, so they will just continue living as they always have. This problem is known as the tragedy of the commons. Everyone would win if everyone participated, but one person can not make the difference. This concept has been studied at length in the past decades. We refer to Hardin [13] for a view of this problem in broader society. In Tribler, this concept exists in the following form: Everyone would like to use the network to download data. However, in order to do so, there needs to be at least one agent to relay this data, and then another one to request it from the internet. This means that if enough agents fulfill these functions out of their free will, and everyone does a little bit, the load will be spread and the network will function. However, it is much more enticing to not contribute and only consume.

The problem here is that since the system itself needs to maintain anonymity, the task of finding out if an agent is contributing suddenly becomes a lot harder. In particular, it becomes very hard to determine whether an agent is real, and has had interactions with others, if one can not have a direct interaction with this agent. An agent could simply exist by claim of another, including fictional interactions with the real agent. Any trust system should therefore be able to deal with such attacks. The theoretical resistance to attacks, and the performance in practice are the subjects of this thesis.

Traditionally, in BitTorrent networks, this problem has been solved for public torrents by simply relying on altruistic agents, who contribute based on nothing more than their good will.

[1] https://www.torproject.org/about/overview.html.en
[2] https://creativecommons.org/licenses/by/3.0/us/

There also exists the concept of a private community, in which a central entity records how much data is downloaded and uploaded by each agent. This central entity then does "ratio enforcement" and excludes members that drop below a certain contribution ratio. Tribler has so far mostly relied on the altruists. Recently, a movement has started to bring some sort of trust system to Tribler. In particular, if an agent considers another one to be trustworthy, in the sense that they have contributed a lot to the network, the former would give the latter priority in requests for resources. This scheme solves the tragedy of the commons by inducing agents to contribute. Contributing more would give an agent better service. By introducing a direct incentive, the tragedy of the commons is solved by the rationality of the agents involved.

In this trust system, each agent should have some indication of the reputation of each other agent with respect to their contribution and consumption in the network. If an agent contributes enough relative to their consumption to have a positive impact on the network, they have a good reputation. If not, they have a negative reputation. Reputation may be subjective, in the sense that it could be different from the perspective of different agents. However, it should lead to a ranking of the other agents, such that priority can be given to the agents with better reputation as described above.

The concrete contributions of this thesis are as follows: Chapter 2 contains the research question about the existence of a reputation system for distributed systems. Additionally, it contains a division of this question into several parts representing different aspects of the problem. Chapter 3 presents the base model for interactions in distributed systems, which will be used in the rest of the thesis to study systems that are candidates for fulfilling the research question. Chapter 4 contains our NetFlow mechanism, supported with examples and theoretical results about its properties, in particular its resistance to Sybil attacks. Next is Chapter 5, in which Temporal PageRank is introduced, a mechanism that uses the order of interactions in the system to compute reputation. Again, some theoretical properties are highlighted. Chapter 6 shows the performance and properties of above mechanisms in practice. This is done by applying the mechanisms to a dataset obtained through a real-world deployment of Tribler with the MultiChain data structure. Finally, Chapter 7 concludes the research and discusses caveats and potential for future research.

Chapter 2

Research question

Distributed systems have many advantages over traditional centralized systems, such as resilience against technical attacks as well as political attacks to take the system down, the ability to spread system load over multiple entities and physical locations and resilience against technical system failures or natural disasters. However, building and maintaining a distributed system comes with several challenges, in particular that the agents in this system will need to contribute in order for the system to function. This can easily lead to a tragedy of the commons, where nobody participates, but the result would be better if everyone pitched in a small amount of effort.

The focus of this thesis is the following research question:

Can agents in a distributed network use confirmed information about interactions and their order to judge the reputation of other agents in a fair and efficient way?

Before anything else, we need to specify what "judge the reputation" means. An agent will judge the reputation of others by assigning each of them a score. If one agent is assigned a higher score than another, the former is judged to be more reputable. These scores could then be used to determine priority in distributing resources.

Throughout this thesis we consider Tribler as an application area and experimental evaluation will be done using Tribler. In this context, a system that assigns a score to agents given information about their interactions has been dubbed an "accounting mechanism" by Seuken and Parkes [27]. This work will adhere to this terminology. The exact definition of an accounting mechanism will follow in Chapter 4.

To restrict the scope of this research, only accounting mechanisms are considered as a method of using the information about the order of interactions. The limitation of accounting mechanisms is that the reputation of each other agent must be reduced to a single number. The alternative would be a vector-valued reputation, which introduces extra complexity. Since using single-valued reputation already brings us to a case which is far from fully understood, this research is limited to that case.

This reduces the main research question to the following:

Does confirmed information about interactions and their order allow the design of an accounting mechanism that satisfies the following three properties?

- The accounting mechanism is fair.
- The accounting mechanism is manipulation-resistant.
- The accounting mechanism is efficiently computable.

These properties are not well-defined in this context, so we consider each of them.

2.1 Fairness

The meaning of fairness as in the above property is that the score assigned to an agent accurately reflects their contribution. The problem with this is that the target might vary for different application domains. For example, in the context of a social network where interactions are friendships, interactions are positive for both sides. In a system like Tribler, interactions will be positive for at least one side, but might have a negative impact on the reputation of the other side. In a distributed system like Bitcoin, we might interpret the amount of money an agent possesses as a score assigned. In this case, the transfer of money is the interaction, and it will be positive for one agent and negative for the other.

For purposes of this thesis we will zoom in on fairness in the context of Tribler. This notion would hold up for other distributed systems in case that each node can contribute or consume resources, where contributing is positive, consuming is negative, and contributing and consuming the same amount is explicitly seen as positive. Generally, in BitTorrent networks, the ratio between upload and download is seen as a fair measure of contribution. However, since in Tribler, relaying data for onion routing is a service that the network provides, uploading and downloading the same amount is seen as a positive thing. This is not reflected in the ratio, which means that we cannot take it as an accurate representation of a fair accounting mechanism.

To at least get some notion of what constitutes fair, let us consider an agent that engages in a new interaction. Then the following would be an indication of a fair mechanism:

- If the agent consumes more resources than they contribute, their score goes down.
- If the agent consumes less resources than they contribute and less than they have done historically, their score goes up.
- If the agent consumes less resources than they contribute, but more than they have done historically, their score might not change.

2.2 Resistance to manipulation

While fairness says something about the score in relation to the actions within the system, resistance to manipulation concerns actions that are not "normal" in the system. In the main research question, this property is hidden in the concept of "fair". It is implied that in a fair system, cheating should not be possible. In particular, lying of any sort, creating extra or new identities and any other actions than participating in the network should not lead to an increase in score. Attacks that involve non-existent identities are referred to as Sybil attacks, a term which was coined by Brian Zill and introduced into scientific literature by Douceur [6]. These kinds of attacks can be particularly problematic in systems where there is no control on or cost associated with the creation of a new identity. An example of a network that has been attacked in such a way is the KaZaA filesharing network, where users could rate files based on their quality. It was found that songs of which a lot of invalid copies existed in the network also had a high incidence of being falsely rated as good quality, suggesting that Sybils were used to pollute the system [19].

2.3 Efficiency of computation

As well known, the efficiency of algorithms can be compared with big-O notation. It might vary per application domain if an accounting mechanism with a certain theoretical worst-case performance is usable. It might be that some theoretically slow mechanisms turn out to be fast

in practice, or that the instance sizes are small enough to allow a higher-complexity algorithm. Hence, in addition to big-O notation, the efficiency will be studied in practice.

In the case of Tribler, it is still the subject of discussion how long computation of an accounting mechanism may take for it to be fast enough. This would also depend on how it is used. In Tribler, connections last for 10 minutes. This suggests that doing a computation every 10 minutes would be a reasonable interval. However, taking 10 minutes worth of CPU time would lead to user complaints. Anything below 30 seconds of single core computation would be acceptable. Waiting for a longer interval between computations makes a longer computation time acceptable, but this comes at a cost of relying on slightly outdated values for decision making. Performing one computation daily would make 5 minutes of single core computation time an upper bound.

Chapter 3

Base Model of Interaction Information

In order to exploit information about interactions, this information will need to be available in the first place. In order to allow general applicability of the techniques, first the general model is considered in Section 3.1, which is then shown to be relevant in the particular case of Tribler in Section 3.2.

3.1 Base model

In order to incorporate temporal information in Sybil defence mechanisms, there will be some assumptions needed about the information which is available. Hence, applications that fit the following model will be explored.

Definition 1 (Ordered interaction model)**.** An *ordered interaction model* $M = \langle P, I, a, w \rangle$ consists of two sets and two functions.

- P, a finite set of agents
- I, a finite set of interactions
- $a : I \to P \times P$, a function mapping each interaction to the participants involved
- $w : I \times P \to \mathbb{R}_{\geq 0}$, a function which describes the contribution of an agent in an interaction

Note that $w(i, p) = 0$ must hold if $p \notin a(i)$.

Agents represent entities that can interact with each other. An interaction involves two different agents, one or both performing work for each other.

Furthermore, for each $p \in P$, the following set must be completely ordered.

$$I_p = \{i \in I : p \in a(i)\}$$

The ordering on I_p is denoted by $\leq_p$.

This model can be applied to any system where agents interact and perform work for each other, where this work can be quantified in a number. In most cases, the ordering on the interactions per agent will be relatable to time. A possible application is a system in which agents lend each other money. Any kind of payment then is an interaction, where the contribution is the amount of money transferred. Other applications are systems where agents perform favours

for each other, where either the favour is measurable in a quantity, or has a monetary value that can be associated with it. This model breaks down if different agents value the same work differently. In this case an extension using game theory would provide a solution.

Definition 2 (Successor of an interaction)**.** Let $M = \langle P, I, a, w\rangle$ be an ordered interaction model. Let $p \in P$, $i, j \in I$ such that $i <_p j$. If there exists no i' such that $i <_p i' <_p j$, then j is the successor of i. This is denoted by $i \prec_p j$.

This model induces a graph that resembles traditional interaction graphs, but preserves the knowledge of the order of interactions.

In order to shed more light the concept of an ordered interaction model and other definitions that will depend on this, let us consider a running example. This example involves 3 agents: P, Q, R. The model is represented in Table 3.1

Interaction Id	Agent 1	Seq 1	Contribution 1	Agent 2	Seq 2	Contribution 2
P1Q1	P	1	5	Q	1	3
P2R1	P	2	7	R	1	1
Q2R2	Q	2	3	R	2	8
P3R3	P	3	2	R	3	3

Table 3.1: Set of interactions for examples

In Table 3.1, the attributes of the interactions are specified. For each interaction the agents involved are listed, along with their respective contributions. In addition, the sequence number ("Seq") is listed. Recall that the set of interactions of each agent must be completely ordered. The sequence number is simply the numbering of this set as prescribed by the ordering. This also gives an easy way to uniquely identify an interaction, namely by concatenating the agents with their sequence numbers. This is listed as the interaction id.

With this example in mind, allow us to consider Definition 3, one of three graphs based on an ordered interaction model.

Definition 3 (Ordered interaction graph)**.** The directed graph $G_M = (V, A)$ is defined as the *ordered interaction graph* derived from an ordered interaction model $M = \langle P, I, a, w\rangle$. Its structure is as follows:

- $V := \{v_i : i \in I\}$
- $A := \{(v_i, v_j) : i, j \in I, \text{ and } \exists p \in P \text{ s.t. } i \prec_p j\}$

The weight of an arc in G_M is as follows:

$$w_{G_M}(e) := w(i, p), \text{ where } p \in a(i) \cap a(j), e = (v_i, v_j) \in A : w$$

Example 1 (Ordered interaction graph)**.** Figure 3.1 depicts the ordered interaction graph derived from the data in Table 3.1. Note that some numbers present in the table are not in the graph, as the contributions in an interaction are weights on the arcs to the successors of that interaction. If that successor does not exist yet, those contributions have no representative in the graph. The ordered interaction graph functions mainly as a structure to consider, as the name says, the order of interactions. Note that if the ordering of the I_p represents a temporal relationship and there are no disparities between the time an agent records an interaction, the

ordered interaction graph is an acyclic graph that corresponds to the natural partial order that is induced by the total orders on the I_p. If this is not the case, this graph will contain cycles. In this case one can still derive a partial order, in which interactions that are in the same cycle are considered to be incomparable. However, this might result in the complete order of the I_p not being maintained.

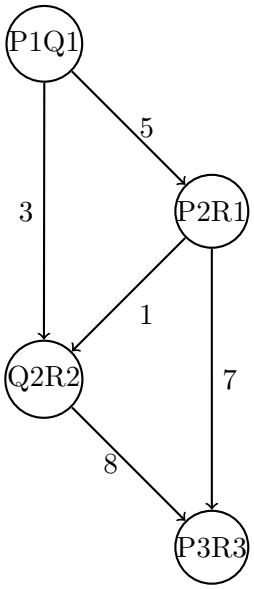


Figure 3.1: Example of an ordered interaction graph

Definition 4 (Block Graph)**.** Let $M = \langle P, I, a, w \rangle$ be an ordered interaction model.

We define $I' := I \cup \{i_p^f : p \in P\}$, where $i_p \leq i_p^f \forall i_p \in I_p$ and $a(i_p^f) = (p, p)$. Basically, we extend I with the next interaction for each agent, which only involves p.

The block graph G_M^B is defined as follows:

- $V := \{v_{p,i} : i \in I', p \in a(i)\}$
- $A := \{(v_{p,i}, v_{q,j}) : i, j \in I', p \in a(i), q \in a(j), (i \preceq_p j \vee i \preceq_q j)\}$

This graph can be weighted. Let $(v_{p,i}, v_{q,j}) \in A$, then the weight of $(v_{p,i}, v_{q,j})$ in G is:

$$w_G((v_{p,i}, v_{q,j})) := w(i, q)$$

Example 2 (Block graph)**.** In Figure 3.2 an example of a block graph can be found. This representation yields a another view on the ordered interaction model. In this graph, the fact that each agent has their own sequence of interactions is stressed, with additional arcs indicating the other party involved. In the ordered interaction graph, there were some issues with certain contributions not being reflected in the graph. This issue is not present in this graph, at the cost of a less elegant definition. The reason for this is that this graph will be used in Chapter 5 to serve as the basis for Temporal PageRank. It would be undesirable to throw the last interactions away on a pseudo-arbitrary basis.

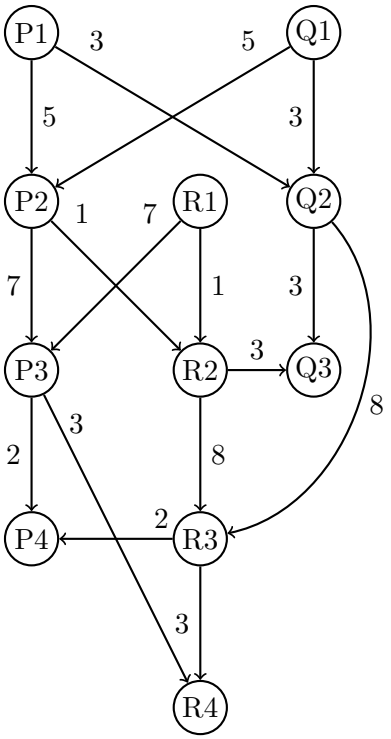


Figure 3.2: Example of a block graph

Definition 5 (Interaction graph)**.** Let $M = \langle P, I, a, w \rangle$ be an ordered interaction model. The interaction graph G_M^I is defined as follows:

- $V := \{v_p : p \in P\}$
- $A := \{(v_p, v_q) : \exists i \in I, a(i) = (p, q)\}$

This graph has weights. Let $(v_p, v_q) \in A$, then:

$$w((v_p, v_q)) := \sum_{i \in I : a(i) = (p,q)} w(i, p)$$

Example 3 (Interaction graph)**.** Figure 3.3 shows the interaction graph. Traditionally, this is the graph that is used in literature. However, when considering an ordered interaction model, this graph is the one that might be less obvious at first glance. The structure of the graph is the block graph with a contraction of all nodes per agent. The weights on the arcs are the sums of the edges in the block graph, but in the other direction. This graph does not reflect the notion of time or order in any way. It will be the basis for the NetFlow mechanism presented in Chapter 4.

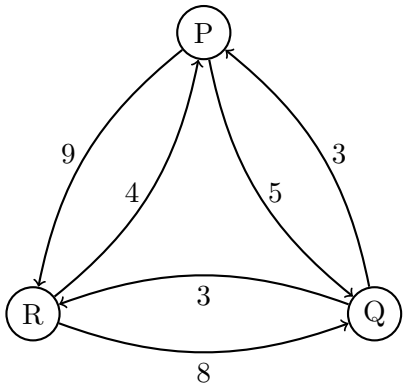


Figure 3.3: Example of an interaction graph

3.2 Base model applied to Tribler

To obtain more insight into the ordered interaction model, consider the following application.

Tribler is an anonymous peer-to-peer file sharing client. In the application of the model, the set of agents consists of Tribler clients. Each Tribler client is uniquely identifiable, but a single real world entity could have multiple agents. In Tribler an interaction consists of two agents uploading data to each other. The amount of work w_p performed by an agent p is the amount of megabytes uploaded to their peer (q).

In Tribler, these interactions are recorded in a distributed data structure, called the MultiChain. Every client saves and signs their interactions, marked with a sequence number. They will then request a digital signature from their peer, who will mark the interaction with their own sequence number. These sequence numbers induce a partial ordering, where $i < j$, if and only if there exists a sequence $(i_0, i_1, \ldots i_n)$, where $i = i_0, j = i_n$ and every pair (i_k, i_{k+1}) has a common agent, for which the sequence number in i_k is lower than in i_{k+1}. This results in a full order on I_p, since in that case (i, j) is a valid sequence if i occurred before j and both involve agent p.

The above reasoning shows that the base model as presented is compatible with Tribler as a use case. Let us further elaborate on some of the properties of the MultiChain, in order to sketch a better image. The MultiChain has first been introduced by Norberhuis [22]. For a detailed view into this structure, we refer to his work.

There are two main properties of MultiChain that are exploited in the mechanisms this work builds upon. Firstly, there is the fact that a record of an interaction can only exist if both parties agree that it is accurate. Secondly, the MultiChain is built such that there is eventual detection of lying. In particular, if an agent spreads inconsistent interactions to different agents, this will be detected. In terms of the model, if an agent spreads interactions such that his set of interactions has no total order, the network will be able to detect and prove this.

There are also properties of the MultiChain that are less desirable. In particular, it is impossible for all agents to be notified about each the interactions at the same time. In fact, when the network grows too large, agents will not even get a full view of the network. This is not necessarily problematic for the systems that we propose, but is something that should be kept in mind and these properties will eventually give way to attacks on the systems.

Chapter 4

Accounting mechanisms and Sybil-proofness

The first reputation system that was introduced in Tribler was BarterCast [21]. BarterCast is a system designed to deter what is described as "lazy freeriders". Freeriders are agents that do participate, but purposefully contribute no or very little resources to the network, but still try to profit. In the paper introducing this mechanism, Meulpolder et al. suggest that for most practical applications, a reputation system does not need to account for malicious users, just for lazy users. This assertion is based in the observation that users classified as "die-hard freeriders", people who resort to using non-standard software to cheat the system, are relatively uncommon in real-world systems.

BarterCast is built on voluntary reporting. All agents in the system can report to other agents about their interactions with third parties. These reports contain information about how much data was exchanged between the two agents. These reports are then used in a max-flow based algorithm. The interaction graph from Chapter 3 is used. Using this graph, the subjective reputation of j from the perspective of i is the result of an increasing function applied to the difference of the max-flow from j to i minus the max-flow from i to j.

BarterCast has various desirable properties. Firstly, agents contributing more resources, or consuming less resources, have a higher reputation. Secondly, it captures the concept of transitive trust. If agent j uploads a lot of data to agent i, i will consider agents who help j reputable. Thirdly, it is more resistant to cheaters than systems that are based on agents self-reporting their own uploads and downloads. Finally, this is a system that does not require the presence of any centrality.

However, BarterCast also has some flaws. In particular, there is a disincentive for agents to report interactions in which they had a net negative contribution to the network. Even worse, there is no mechanism to verify who is lying in the case of conflicting reports. This means there exists a misreporting strategy that makes an agent seem better than they have performed. Claiming that you uploaded a lot of data to one or more agents, even if that is not the case makes other agents think you have contributed much more than you have. This is a major weakness in the protocol and a solution has been suggested by Seuken and Parkes [27]. The solution involves dropping the reports by any agents which are currently under consideration for receiving services.

In addition to the reporting disincentive, there is a more fundamental problem. BarterCast is built to transitively propagate uploads to you. In some sense, agents are more trusted if they uploaded more to you. This is a perfectly valid construction. However, the reverse also holds. If an agent has downloaded a lot, all agents that downloaded from them are transitively punished. While this seems philosophical at first, there is a Sybil attack that exploits this property.

Example 4 (Sybil attack on BarterCast)**.** Let p, s be agents. Let s contribute 1 unit of work to i. Let s create n Sybils, $s_1, s_2, \ldots, s_n$ and claim that each uploaded 1MB to s. Hence, the flow from s to p is 1, the flow from p to s is 0. Now the flow from each Sybil to s_1 is 0, the flow from each Sybil to p is 1. This situation is depicted in Figure 4.1.

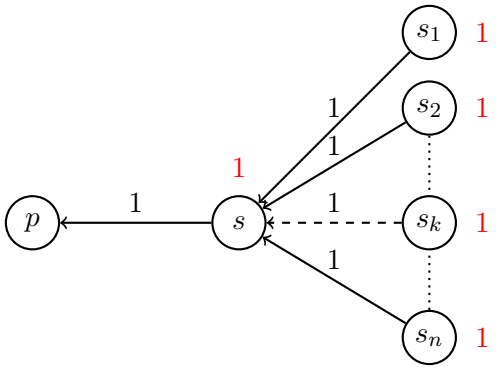


Figure 4.1: Sybil attack on BarterCast. BarterCast scores displayed in red.

Now s_1 asks for a contribution from p, and i obliges and contributes 1 unit of work. This is shown in Figure 4.2 Now the flow between p and both s and s_1 is 1 in either direction. However, the flow from p to all other Sybils is 0 and the flow from each other Sybil to p is 1. So when any of them ask for a contribution, p is likely to oblige.

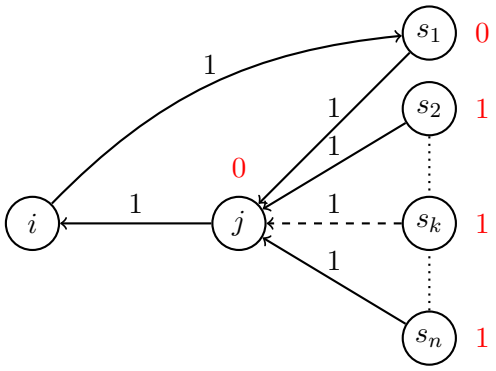


Figure 4.2: Sybil attack on BarterCast. BarterCast scores displayed in red.

4.1 Accounting mechanisms and DropEdge

In the paper mentioned above, Seuken and Parkes introduce the concept of an accounting mechanism. This formalization gives a little bit more structure to the problem at hand. It is here that the notions of subjective work graphs and choice sets are also formalized. Subjective work graphs allow for a framework in which agents do not have a full view of the network, but only have information about agents that they have interacted with and their peers. Choice sets provide a mechanism to describe which set of agents are requesting service from another agent at a fixed point in time. This concept is relevant, because the way Seuken and Parkes suggest to patch the problem with BarterCast is to simply ignore all reports of agents in the choice set and then compute the BarterCast score from the resultant graph. This solves a large part of the reporting problem. Since there is no longer a disincentive to report correctly, the assumption is that agents will cooperate and provide reports. This mechanism is called “DropEdge”. Seuken

and Parkes show that this mechanism satisfies the property of being "misreport-proof". This property states that if an agent is in the choice set, any misreport of this agent will not lead to their own score being increased, or the score of any other agent in the choice set being decreased.

Furthermore, it has been suggested that DropEdge does not get rid of too much valuable information. This is supported by theorems bounding the resulting scores and the amount of work in the subjective work graph after dropping the reports from the choice set members. While these results seem to imply not too much information is lost, the theorems surrounding this information loss all concern statistical claims assuming a uniform distribution over the choice sets. This particular choice of distribution is defensible, but in real situations the distribution over choice sets will be much less uniform, at least when monitoring over a period of time. The reason for this is that whenever an agents requests or stops requesting work, the choice set just changes with this one agent. Hence, there is a large correlation between choice sets in a real world system, yielding a non-uniform distribution.

4.2 Defining accounting mechanisms

Following this work, Seuken and Parkes [26] formulated a framework that considers Sybil attacks as well. A Sybil attack is defined by them as constructing a set of identities and interactions between that set and the Sybils real identity. Beneficial Sybil attacks are then defined as Sybil attacks that increase the Sybils score, either their real or one of their false identities, or decrease the score of another agent, such that one of the Sybils identities is given priority over other agents. We adopt most of their definitions to keep our work compatible and comparable to theirs. All definitions before the Sybil attack profit are due to Seuken and Parkes [26]. Any deviation from their work is noted and explained.

Definition 6 (Work Graph)**.** A work graph $G = (V, E, w)$ has vertices $V = \{1, \ldots, n\}$, one for each agent, and directed edges $(i, j) \in E$, for $i, j \in V$, corresponding to work performed by i for j, with weight $w(i, j) \in \mathbb{R}_{\geq 0}$ denoting the number of units of work.

The work graph is a model of the objective truth about what happened. However, in a distributed system, not everyone may have knowledge of all interactions, and there may be disagreement. To this end, agent information and subjective work are considered.

Definition 7 (Agent Information)**.** Each agent $i \in V$ keeps a private history $(w_i(i, j), w_i(j, i))$ of its interactions with other agents $j \in V$, where $w_i(i, j)$ and $w_i(j, i)$ are the work performed for j and received from j respectively.

Definition 8 (Subjective Work Graph)**.** A subjective work graph from agent i's perspective, $G_i = (V_i, E_i, w_i)$, is a set of vertices $V_i \subseteq V$ and directed edges E_i. Each edge $(j, k) \in E_i$ for which $i \notin \{j, k\}$, is labelled with one, or both, of weights $w_i^j(j, k), w_i^k(j, k)$ as known to i. For edges (i, j) and (j, i) the associated weight is $w_i^i(i, j) = w(i, j)$ and $w_i^i(j, i) = w(j, i)$ respectively.

The definition of a subjective work graph contains a lot of machinery to deal with conflicting information and already resolves some of the conflicts. In particular, if reports of others conflict with the observations of the agent itself, those reports are not considered. This model does not assume anything about the truthfulness of the reports used to constructed the subjective work graph, but it is implied that all information concerning interactions involving agent i themselves is correct. Furthermore, in the application to Tribler, the MultiChain structure enforces that any report is signed by the two parties involved. Hence, it is not possible to simply make negative reports about other parties without their consent.

Definition 9 (Choice Set)**.** We let $C_i \subseteq V \setminus \{i\}$ denote the choice set for agent i, i.e., the set of agents that are currently interested in receiving some work from i.

Definition 10 (Accounting Mechanism)**.** An accounting mechanism M takes as input a subjective work graph G_i, a choice set C_i, and determines the score $S_j^M(G_i, C_i) \in \mathbb{R}$, for any agent $j \in C_i$, as viewed by agent i.

Note that definitions 9 and 10 consider highly decentralized mechanisms. Both are centred around a single agent, who will then compute possibly subjective scores. Because each agent should be able to do this, accounting mechanisms that are to be deployed in real world systems need to be efficiently computable.

Definition 11 (Allocation Policy)**.** Given G_i, choice set C_i and an accounting mechanism M, an allocation policy A is a function that maps these three objects to an agent $j \in C_i$. This choice is denoted by $A(S^M(G_i, C_i))$. This agent is chosen to receive a unit of work from i.

This definition is not part of the original work of Seuken and Parkes, where only the following example is given.

Policy 1 (Winner-Take-All)**.** The winner-take-all allocation policy (WTA) selects the agent with the highest score, breaking any ties randomly:

$$A(S^M(G_i, C_i)) = \arg\max_{k \in C_i} S_k^M(G_i, C_i)$$

This is one the simplest allocation policies possible. Other options could involve splitting the resources or more randomization.

Definitions 6 through 11 supply the framework for accounting mechanisms. Accounting mechanisms are closely related to reputation systems. The difference between the two lies in that reputation systems are more about trust, whereas accounting mechanisms, as the name suggests, keep a (running) record. Often this is work, or service, quantified in some way. Accounting systems are applicable in cases where we are less concerned with the actual identity of users, and more concerned with the resource consumption of each agent relative to their contributions.

4.2.1 Properties of accounting mechanisms

Now that the basis for accounting mechanisms has been provided, it is possible to consider different properties that accounting mechanisms can possess. To start off with, two properties that might have been included in the definition, but are not quite so generic that one would want to do this, as the lack of these properties does not necessarily imply a bad accounting mechanism.

Property 1 (Independence of Disconnected Agents (IDA))**.** *Let M be an accounting mechanism, let $G_i = (V_i, E_i, w_i)$ be a subjective work graph, let C_i be a choice set. Agent $k \in V_i$ is a disconnected agent, if for this agent there are no edges in E_i, or for this agent all edges in E_i have zero weight. By $G_i' = (V_i', E_i', w_i')$ we denote the subjective work graph with a disconnected agent k removed.*

M satisfies "Independence of Disconnected Agents" if for any disconnected agent k, the following holds:

$$\forall j \in V_i' : S_{ij}^M(G_i, C_i) = S_{ij}^M(G_i', C_i')$$

Property 1 simply states that scores are not affected by adding or removing agents that are not connected to the rest of the network. On the one hand, this is a fairly reasonable property. In particular, it provides possible routes for proofs. On the other hand, it is possible to imagine systems in which this property does not hold. In particular, mechanisms that do assign score to independent agents, but limit the sum of these scores, might not possess this property.

Property 2 (Anonimity (ANON))**.** *Let M be an accounting mechanism, let $G_i = (V_i, E_i, w_i)$ be a subjective work graph, let C_i be a choice set and let f be a graph isomorphism such that $G'_i = f(G_i)$, $C'_i = f(C_i)$. M satisfies anonymity if the following condition holds:*

$$\forall j \in V_i \setminus \{i\} : S^M_{ij}(G_i, C_i) = S^M_{f(i)f(j)}(G'_i, C'_i)$$

Property 2 is also a common property of accounting mechanisms. It essentially states that scores can only be based on the structure of the subjective work graph. If two agents seem exactly the same with respect to the subjective work graph, they must get the same score. Again, it is possible to imagine accounting mechanisms without this property, but they will need external information about agents to assign sensible differing scores. The definition given here is slightly stronger than the original definition. In the original definition, the results may depend on which agent is doing the computation. This definition is agnostic of the actual identity of the computing agent.

Property 3 (Weak Transitive Trust)**.** *Accounting Mechanism M satisfies weak transitive trust if, for every subjective work graph $G_i = (V_i, E_i, w_i)$, there exists a $j \in V_i$, after adding node k to $G_i = (V_i, E_i, w_i)$, this leads to $G'_i = (V'_i, E_i, w_i)$ with $V'_i = V_i \cup \{k\}$, and there exists an amount of work W_j and W_k such that if j performs W_j units of work for i, and k performs W_k units of work for j, a report of which leads to a work graph G''_i representing this, then for every choice set C_k that contains k, but not j, it holds that $A(S^M(G''_i, C_k)) = k$.*

Property 3 is a prescription of the way in which the accounting mechanism provides transitivity. It is a desirable property since it ensures that some sort of network effects are possible. In simple terms, it specifies that if agent A is helped significantly by agent B, and agent B in turn was helped by agent C, then A will value C's contribution.

Seuken and Parkes also consider a property called "misreport-proofness". This property states that no advantage can be obtained by reporting interactions that did not happen, or wrongly reporting transactions that did. This property is not relevant because the MultiChain externalizes this problem.

4.2.2 Defining Sybil attacks

At this point, this work will slightly diverge from the model used by Seuken and Parkes. Seuken and Parkes define a Sybil attack from a single point. Only one identity is allowed to interact with the rest of the network, and all Sybil identities can only interact with each other and the single agent that interacts with the network. This approach is quite limiting in the sense that it is generally easy for an attacker to interact with the real network through several identities. Hence, the definition presented here includes the ability to perform a Sybil attack with multiple identities. This definition collapses to the one used by Seuken and Parkes if $|J| = 1$.

Definition 12 (Sybil attack)**.** Given a subjective work graph $G_i = (V_i, E_i, w_i)$. A Sybil attack by agents $J \subseteq V_i$ is a tuple $\sigma_J = (V_s, E_s, w_s)$ where $V_s = \{s_{J_1}, s_{J_2}, \ldots\}$ is a set of Sybils, $E_s = \{(x, y) : x, y \in V_s \cup J\}$, and w_s are the edge weights for the edges in E_s. Applying the

Sybil attack to agent i's subjective work graph $G_i = (V_i, E_i, w_i)$ results in a modified graph $G_i \downarrow \sigma_J = G'_i = (V_i \cup V_s, E_i \cup E_s, w')$, where $w'(e) = w_i(e)$ for $e \in E_i$ and $w'(e) = w_s(e)$ for $e \in E_s$.

Definition 12 deviates from the original definition by allowing a set of agents to perform an attack. This does not seem to add much to the definition of a Sybil attack. After all, an attacker can already create an arbitrary number of identities. However, their definition of a Sybil attack only allows a single point where Sybils can interact with the rest of the network, which is a pretty limited view of possible attacks. This definition does allow that, and allows for a wider range of Sybil attacks. Note that E_s may include edges to identities not controlled by Sybils. In order to include these edges, the work actually has to be performed. For edges within V_s,

Definition 13 (Sybil attack profit)**.** Let G_i be a subjective work graph. For all $j \in \mathbb{N}$, let $(\sigma_J)_j$ be a Sybil attack on $(G_i)_j$, where $(G_i)_0 := G_i$ and $(G_i)_j$ for $j > 0$ is defined by the subjective work graph that consists of $(G_i)_{j-1} \downarrow (\sigma_J)_j$ and the assignment of one unit of work to $A(S^M((G_i)_{j-1} \downarrow (\sigma_J)_j, C_i))$.

Let ω^n_- be the sum of the amount of the work agents in J have performed for the network after n steps, including work performed before the start of the Sybil attack. Let ω^n_+ be the amount of work that agents in J or any of their Sybils obtain from the network. Any work obtained before the start of the Sybil attack is disregarded.

The profit of this sequence of Sybil attacks is:

$$\sup\{\frac{\omega^n_+}{\omega^n_-} \ : n \in \mathbb{N}, \omega^n_- \neq 0\}$$

If this supremum is infinite, the Sybil attack is strongly beneficial.

If the supremum exists and is strictly larger than 1, the Sybil attack is profitably weakly beneficial.

If the supremum exists and is smaller than or equal to 1, the Sybil attack is unprofitably weakly beneficial. This case is also known as "contributing to the network".

Definition 13 provides a certain amount of backwards compatibility with the work of Seuken and Parkes, while still being more distinctive with respect to the different impacts Sybil attacks can have on the network. In addition, note that the gap between profitably weakly beneficial and strongly beneficial may appear to be quite small. In particular, profitably weakly beneficial attacks can be repeated by new identities for infinite profit. The difference lies in that this infinite profit then requires infinite contribution as well. This means that for practical systems allowing profitably weakly beneficial Sybil attacks may be good enough, but for most systems a strongly beneficial Sybil attack would be detrimental.

4.3 Bounding Sybil attack profit

The following section describes how an accounting mechanism can be constructed which is partially resistant to Sybil attacks, in the sense that they can be profitably weakly beneficial, with bounded profit.

Policy 2 (Strict Winner-Takes-All)**.** The strict winner-take-all allocation policy (SWTA) selects the agent with the highest score, breaking ties randomly. However, the strict winner-take-all will not select any agent when all agents have the same score.

Policy 2 defines a policy to choose an agent to receive resources given scores on resources. This is a policy which will likely not be used in practice, but is very useful for proofs. The

consequence of this is that in real world systems a little bit of good will is necessary to bootstrap a system based on trust. All results must therefore be considered with this caveat.

Definition 14 (NetFlow limited contribution)**.** The NetFlow limited contribution accounting mechanism M is defined as follows: Given a subjective work graph $G_i = (V_i, E_i, w_i)$ and choice set C_i, agent i computes the following for each agent j: $c_j := \max\{MF_{G_i}(j,i) - MF_{G_i}(i,j), 0\}$, where $MF_{G_i}(i,j)$ denotes the value of the maximum flow from i to j in G_i, where the weights are the capacities on arcs.

Let G_i^N be the graph G_i modified with c_j as node capacities for each node, except for c_i which should be infinite.

Now $S_j^M(G_i, C_i) = MF_{G_i^N}(j,i)$.

The first step of the NetFlow algorithm is exactly the BarterCast algorithm. However, as opposed to BarterCast, this step is only used to settle the consumption and production of resources. The second step determines the scores, capping it at the *net* contribution of each agent. The second step still enables transitive trust, but capped by the net contribution of the agents on the paths between the agent whose score is computed and the computing agent. This construction still preserves some amount of transitivity in trust, but also prevents it from being abused like in Example 4.

Example 5 (NetFlow computation)**.** In order to provide some insight in the behaviour of the NetFlow mechanism, let us consider a small example. Figure 4.3 depicts the interaction graph for which a computation will be done. The computation is done from the perspective of agent R.

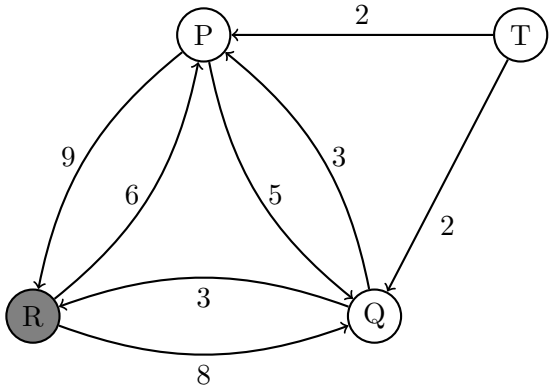


Figure 4.3: Interaction graph for a NetFlow computation

The first step in the algorithm is to compute two max-flows and subtract the flow towards the other agent from the flow coming from that agent. This first step is shown in Figure 4.4. Observe that agents P and T have a positive capacity, 3 and 4 respectively. This means they are eligible for positive scores, and that they can serve as a conduit for the purposes of transitive trust. For Q, the subtraction results in a negative number, which is then rounded to 0 to yield the capacity.

The second step (Figure 4.5) in the NetFlow algorithm involves flows from the other agents towards the calculating agent, respecting the node capacities calculated in the first step. This yields the scores for each agent. Agent P is the highest ranked agent with a score of 3, then T with a score of 2, followed by Q with a score of 0. Note that purely based on the contributions from a neutral perspective, one might rate agent T as having contributed more than agent P, but due to the structure of the interaction graph, this is not the case. Because agent T is farther away in the graph and has no direct interactions with R this is a defensible result.

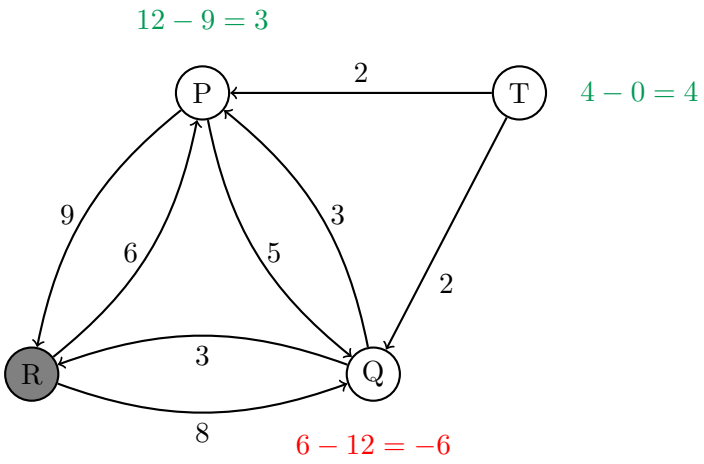


Figure 4.4: The first step of a NetFlow computation

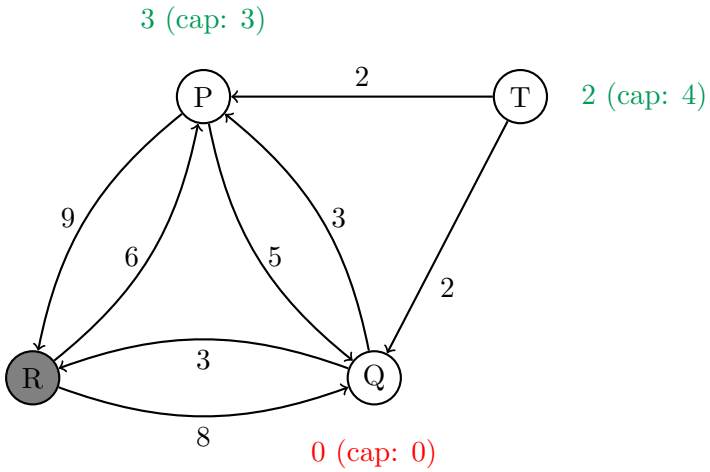


Figure 4.5: The second step of a NetFlow computation

Theorem 1 (Properties and robustness of NetFlow)**.** *NetFlow as accounting mechanism with a complete work graph, with SWTA as allocation policy satisfies IDA, ANON, weak transitive trust and is robust against profitably weakly beneficial Sybil attacks.*

Proof. IDA follows from the fact that flow to and from agents does not change if independent agents are added or removed. ANON follows from the fact that relabelling nodes will not change the flows.

Let k be the node added and let $M := \max_{j \in C_i}\{S_j^M(G_i, C_i)\}$ be the largest amount of reputation in the system. Then using $W_j = M + 1$ and $W_k = M + 1$ satisfies the requirements of weak transitive trust.

This leaves robustness against profitably weakly beneficial Sybil attacks. Consider the amount of work performed by the agents in J after n steps of the Sybil attack, ω_-^n. For each unit of work that is contributed to an agent in J during the Sybil attack, the capacity of some agent in J that has directly contributed to the network must drop by 1 unit, possibly more agents in J. This means that at most ω_-^n units of work can be contributed to agents in J after n steps. Therefore, $\omega_+^n \leq \omega_-^n$, and thus:

$$\sup\{\frac{\omega_+^n}{\omega_-^n} \ : n \in \mathbb{N}, \omega_-^n \neq 0\} \leq 1$$

Since the computing agent possesses the complete work graph, no work can leak outside of this graph. □

Note that this result does not contradict those of Seuken and Parkes. The NetFlow mechanism is still vulnerable to weakly-beneficial Sybil attacks, as originally defined in their work. This is due to the fact that the original definition is so broad that a weakly-beneficial Sybil attack does not necessarily have to be bad for the network. In particular, this is the case if a network only cares about work contribution and consumption, and not about possible other external factors. Furthermore, in contrast to Drop-Edge, NetFlow is choice set independent, which implies there is less computation for varying choice sets.

On the other hand, this algorithm computes two max-flows for every node and then one max-flow for every node that needs to be assigned a score. This means that any algorithm that simply computes these flows will run for up to $O(n^4)$ time, which is quite expensive. Furthermore, this mechanism is quite strict. There is the SWTA policy involved. However, this is mainly for the benefit of the proof. One could quite easily enable the WTA policy instead. In this case, the implication of the theorem is that Sybil attack cannot yield a profit, with the exception of those built on receiving resources by identities with a 0 score. This is an inconvenience, but not a particularly big problem.

Another problem is informativeness. In general, informativeness describes that something provides information. In this mechanism, only peers that have a strict positive contribution in terms of flow will induce network effects. In existing networks, this subset of the population tends to be fairly small. Hence, a large number of agents will have a zero score, which means that SWTA would not allocate any resources to these agents. If most agents are assigned a zero score, the mechanism does not provide any information about them. This leads us to Definition 15.

Definition 15 (Informativeness)**.** Given an accounting mechanism M and a subjective work graph G_i, the informativeness of M given G_i is the percentage of agents in G_i that have a non-zero score.

4.3.1 Improving informativeness by scaling

An idea to improve the informativeness would be to weight the work performed by the other agents higher than work consumed by them. The idea behind this scheme is that agents that perform less work than they consume are tolerated to some degree. This is a trade-off in the sense that this will allow weakly profitable Sybil attacks in exchange for increased informativeness of the mechanism. In particular, a scaling would need to guarantee that it is impossible to leverage the scaling to a strongly profitable Sybil attack.

We scale NetFlow by rating work performed by the calculating agent i lower than other work, which is equivalent to rating all other work higher.

Definition 16 (α-NetFlow limited contribution)**.** Given a subjective work graph $G_i = (V_i, E_i, w_i)$, choice set C_i, $\alpha \geq 1$, the α-scaled weights $w_{i,\alpha}$ are defined as follows:

$$w_{i,\alpha}(e) = \begin{cases} \frac{w_i(e)}{\alpha} & \text{if } e = (i,j) \text{ with } j \in V_i \\ w_i(e) & \text{otherwise} \end{cases}$$

The α-NetFlow accounting mechanism is computed as the NetFlow accounting mechanism, but on the graph $G_i' = (V_i, E_i, w_{i,\alpha})$.

As a shorthand, this mechanism is denoted by α-NetFlow.

Theorem 2 (Robustness of α-NetFlow)**.** *α-NetFlow with SWTA is robust against weakly beneficial Sybil attacks with profit more than α.*

Proof. Since NetFlow is robust against weakly beneficial Sybil attacks with profit at most 1 by Theorem 1 and the only difference between the two systems is that the contributions of agent i are decreased by a factor α, it must be the case that the profit of a Sybil attack can be at most α. After all, all altered interactions are with i and are therefore known to have actually happened. Therefore, the resource consumption of any party can be at most a factor α more than allowed by net flow, which implies an upper bound on the Sybil attack profit. □

Theorem 2 provides an interesting trade-off. For $\alpha = 1$, it reduces to one of the properties in Theorem 1. For larger α, the mechanism becomes more vulnerable to Sybil attacks, but more agents will have a positive net flow in the first computation step, meaning that more agents will have a non-zero score, increasing the informativeness of the system. Note that considering the accounting mechanism on a fixed graph as a function in α to a vector of scores results in a continuous function. Furthermore, if we let α tend to infinity, this results in an accounting mechanism where every score is the max-flow from that node to i.

There are more desirable properties of α-NetFlow. While on the surface, it seems to be about ratio's and relative usage, there actually is a subtle way in which this accounting mechanism cares about absolute amounts of work performed. This is illustrated by Example 6

Example 6 (Secondary effects of scaling)**.** In particular, consider three agents, p, q, r. Agent p has performed 100 units of work for q and q has done the same in return. In addition, p and r have performed 2 units of work for each other.

If agent p computes scores in 1-NetFlow, then both agents q and r are assigned a score of 0 units. However, in the case of 2-NetFlow, agent q will have a score of 50 units, whereas agent r will have a score of 1 unit.

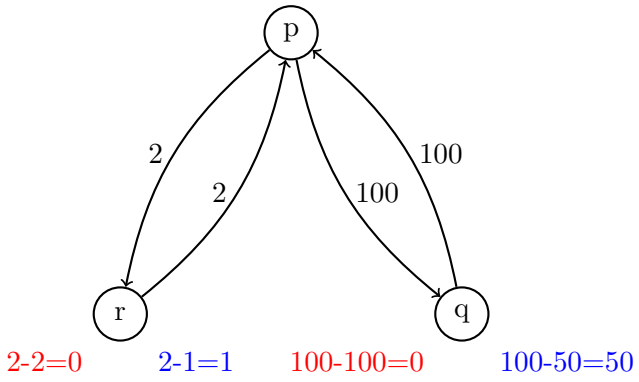


Figure 4.6: Effects of scaling NetFlow. 1-NetFlow scores in red, 2-NetFlow scores in blue.

This example illustrates that using scaling does not only increase informativeness, but also values total volume in addition to the difference between contribution and consumption.

In a broad sense, this means that choosing a higher α will increase the score of agents which have been around longer than yourself, or slightly shorter. This scheme creates a sort of trickle down system, in which the levels consist of people who have had a similar total contribution over time. Each level will prioritize contributing to agents around their level or above it. Any surplus will trickle down to the lower levels. As a huge influx of users results in an increase in the lower levels, the higher levels will not be impacted, and it is mainly the lower levels that help each other.

4.4 Attacks on Net-Flow

Theorem 2 suggests an almost absolute defence against Sybil attacks. However, due to the formulation of the model, and the properties of the underlying MultiChain data structure, there are possible angles of attack.

4.4.1 Propagation slowness

One of the properties of the MultiChain includes that any interaction is signed by two parties and then is valid immediately. There is no concept of network-wide confirmation. This means that an attacker could approach several nodes in the network simultaneously and extract more data from the multiple parties than they would have allowed if each party had known about the ongoing interactions with the other peers.

The impact of this kind of attack is limited by several factors. Firstly, there's the upload capacity of the victims. The time it takes for interactions to propagate determines the length of the attack, which means that the total amount of resources gained "unfairly" is at most this time times the average upload capacity of the victims over this period. Secondly, there is the score of the attacker. If the score of the attacker is not high enough to keep getting resources from a victim before other interactions have propagated, then this is the limiting factor in the attack.

In particular the first limiting factor is important, since this means the profit of an attack can be bounded by a constant times the number of parallel connections an attacker can make. This means it is possible to choose a lower α to account for this attack.

4.4.2 Partial network visibility

Tangentially related to the above attack is another attack that exploits the fact that if the MultiChain network grows very large, it is not expected for every agent to have a full view of the network. This is a desirable paradigm, as otherwise it would be trivial to spam the network by creating and broadcasting MultiChain blocks. However, this also implies that different agents have a different view of the network. In particular, if an attacker s participates in an interaction I_1 in which he contributes a lot of resources, and victims v_1, v_2 both know about I_1, but are not even aware of the existence of the other victim, then the attacker could exploit this and request resources from both.

This is a more difficult attack to prevent. The main hurdles for actually executing it appear to be the fact that it requires a very specific network topology. Furthermore, the attacker would need to know the view of the network that potential victims have. A possible defence against this attack is to incorporate randomness in which interactions to keep, or which agents to keep track of.

4.4.3 Collusion attack

Suppose two agents both have a positive score. If they fully trust each other, they can sign an interaction that both of them have uploaded an extremely large amount of data to each other (say 1TB). Assuming a fluid network, in which the flows are bounded by capacities of the source or sink, or the cuts around them. In such a network, the following effects would be the consequence of this attack.

- These two agents have the highest scores in each others rankings.
- The capacities of both nodes become the sum of the capacities.

This might result in increased scores from agents that have a higher absolute contribution than either of these agents, but lower than their sum. For agents that do not match this criterion, the scores will likely remain the same.

The presence of this attack vector is a consequence of the desire to fight Sybils. It should be actively undesirable to spread ones efforts over multiple identities. Since this attack effectively merges two identities into one, it is plausible this yields a positive effect, since the inverse operation must be negative, or at least neutral. At first sight, this might seem like a problematic attack. However, this attack is only strictly positive for both parties as long as they have the same ratio. Once one contributes more than the other, they will get relative negative effects. This attack can be feasible if the involved parties have complete trust in each other. In some cases it might be considered less of an attack, and more of a feature. For example, if a user has multiple machines, they can link their identities in this way without reusing it across machines. To conclude, this attack might be a risk and needs to be kept in mind in implementations, but seems relatively harmless for the network.

4.5 Theoretical Performance of NetFlow

In order to compute the score of one other node with NetFlow, up to $2n + 1$ max-flow computations are necessary, where n is the number of nodes in a subjective work graph G_i. In case one would compute the scores of all other nodes, $3n$ max-flow computations are needed. Hence, the performance of NetFlow will depend on the max-flow algorithm used and then incur another factor n of computational complexity. This could be mitigated by finding a way to compute multiple flows at the same time. In this section, possibilities to improve the performance of NetFlow will be explored. n denotes the number of nodes, m denotes the number of edges.

Well known algorithms for max-flow include Ford-Fulkerson [9], with complexity $O(m|f|)$, where $|f|$ is the magnitude of the maximum flow. This is a pseudo-polynomial algorithm, since it depends on the magnitude of numbers in the instance. The Edmonds-Karp [7] algorithm specifies the order in which augmenting paths are considered, namely by doing a breadth-first search. This allows the complexity to be pinned to $O(nm^2)$.

Dinitz' algorithm [5], (also known as Dinic's algorithm) functions in $O(n^2m)$ or $O(nm\log(n))$ if implemented with dynamic trees. Goldberg and Tarjan introduced the preflow-push algorithm [11]. This algorithm has a worst-case complexity of $O(n^2m)$. Again, this can be reduced to $O(nm\log\frac{n^2}{m})$ by using dynamic trees.

Work by King, Rao and Tarjan [18] has resulted in an algorithm of order $O(nm\log_{\frac{m}{n\log(n)}} n)$. In combination with work by Orlin [23], this results in an $O(nm)$ algorithm for max-flow. This is the current state-of-the-art when considering single source-sink max-flow.

When we consider all-pairs max-flow, one might think a speed-up is possible. Building a Gomory-Hu tree [12] yields a method for which $n - 1$ max-flow computations suffice. However, this method only works for undirected graphs. According to Ahuja, Magnati and Orlin [1], no method is known for all-pairs max-flow on directed graphs that uses less than $O(n^2)$ max-flow computations. This work is from 1993, and to our knowledge this has not changed since. Note that for the application of NetFlow, it would be necessary to compute all flows with one fixed sink or one fixed source, which is not quite the same as the all-pairs max-flow problem.

The above research implies that using state of the art algorithms, a NetFlow algorithm implemented with current state-of-the-art algorithms would be at least $O(n^2m)$ in the worst case. Our implementation uses the preflow-push algorithm, yielding a worst-case complexity of $O(n^3m)$.

In the case of Tribler, the algorithm needs to scale to at least 10.000 nodes. Anecdotal evidence suggests that this is the desired size of a distributed network that can collectively

provide enough resources to provide value to all participants. The number of pairs of agents that have interacted is the number of edges. Past measurements have shown that the number of edges in a distributed network of this sort generally observes a power law. This would result in a number of edges that is $m = O(n)$.

Chapter 5

Considering Time

NetFlow seems quite suitable from a theoretical perspective. The guarantees against Sybil attacks are exactly what we set out for. However, the computational complexity is high enough that there is reason for worry. This leads to the idea that a cheaper method that uses the temporal information in an ordered interaction model might lead to a more feasible method that is still resistant to Sybil attacks. This chapter will detail an accounting mechanism based on PageRank that will prevent "historical" Sybil attacks.

5.1 Random walks in the Ordered Interaction Graph

Random walks have been used in various strategies that combat Sybil attacks in one way or another. Gkorou, Pouwelse and Epema [10] use random walks in a pure form, more for the purposes of limiting exploration in a smart way than trying to detect Sybils. This paper includes several types of random walks and considers both the exploratory effect and the load on important nodes in a random walk.

Kamvar, Schlosser and Garcia-Molina [16] introduce the EigenTrust algorithm, which is a distributed way to calculate PageRank. PageRank is the stationary distribution of a random walk [24], where the random walk is uniform over links, with a probability to jump to a random node, specified by a distribution on those nodes. The trick in this context is to distribute the computation over all nodes, in such a way that a portion of them can be malicious without influencing the result.

Yu, Gibbions, Kaminsky and Xiao [29] present SybilLimit, which is a method that is specifically designed for Sybil detection. Again, the basis lies in random walks and this work specifies how to perform the desired algorithm in a distributed way. Again, spreading computational effort through the network.

Random walks are interesting for the purpose of Sybil detection for several reasons. Firstly, it matches certain models of behavior, as in the case of PageRank. Secondly, it is a relatively cheap technique from a computational perspective, which is especially interesting since the main drawback of the Net-Flow technique presented in Chapter 4 are the computing expenses.

We take the concept of PageRank and apply it to the Ordered Interaction Graph to yield a technique that will be referred to as Temporal PageRank

Definition 17 (Temporal PageRank)**.** Let M be an Ordered Interaction Model, with corresponding Block Graph G. Let i be the agent that performs the computation.

Consider a random walk in G, where the transition probabilities are the normalized weights of the arcs. Let this random walk have a chance of β to jump to a random node. This random

node is determined by a distribution X on nodes that represent i's half of interactions of a single agent i. By default, this is uniform over all nodes.

The score of a half-interaction is its probability in the stationary distribution of this random walk. Then Temporal PageRank is an accounting mechanism, which assigns as a score to agent j, the sum of all nodes representing j's half of interactions.

Before considering the functionality of this mechanism, let us visit the well-definedness.

Theorem 3 (Well-definedness of Temporal PageRank)**.** *Consider a random walk as described in the definition of Temporal PageRank.*

This random walk has a unique stationary distribution, so the score of an interaction is well-defined.

Proof. This random walk can be associated with a Markov chain, where the set of states is the set of blocks (half-interactions). Let W be the set of nodes/states with non-zero probability in the distribution that determines the node jumped to with chance β.

Let $\overline{W}$ be the set of nodes that can be reached by following a path in G from a node in W. Restrict the random walk to states in $\overline{W}$. Since every node in $\overline{W}$ is reachable from W and the random jump means that every node can reach all nodes in W, the Markov chain induced by this random walk, is irreducible. Furthermore, nodes in W have a self-loop, so every state is aperiodic, so this Markov chain is aperiodic. This means that the Markov chain of the random walk restricted to nodes in $\overline{W}$, has a unique stationary distribution.

Since $\overline{W}$ is closed, the unique stationary distribution with 0 probability for states outside $\overline{W}$ yields a stationary distribution for the full random walk. Since states outside $\overline{W}$ are transient. In fact, there is a 0 probability of returning to that state. This means that all states outside $\overline{W}$ must have a 0 probability in a stationary distribution, hence the stationary distribution is unique. □

5.2 Computational aspects of Temporal PageRank

Generally, stationary distributions of random walks are computed by the power method. This involves repeatedly multiplying an initial probability vector with the matrix of transition probabilities until the result converges. It is a desirable property to have quick convergence of this method.

Theorem 4 (Convergence of Temporal PageRank)**.** *The error after t iterations of the power method, for Temporal PageRank with jump probability $0 \leq \beta \leq 1$, is $O((1-\beta)^t)$*

Proof. This follows from a result of Haveliwala and Kamvar [14]. For a matrix $A = [cP + (1 - c)E]^T$, with P $n \times n$ row-stochastic and E a non-negative rank-one row-stochastic matrix, and $0 \leq c \leq 1$, the second eigenvalue of A satisfies $|\lambda_2| \leq c$.

If we take $c = 1 - \beta$, P the transition probabilities without the jump, and E the matrix representation of the distribution of the random jump, the constraints are satisfied. This means that for the matrix used in the power method, $|\lambda_2| \leq 1 - \beta$. Furthermore, since this matrix is row-stochastic, its largest eigenvalue must be $\lambda_1 = 1$.

Furthermore, the error of the power method is at most $O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^t\right)$, the proof of which follows by decomposing the matrix into Jordan Normal Form.

Combining these facts yields that the error of the power method for this particular matrix after t iterations is at most $O((1-\beta)^t)$. □

This result implies that the convergence is exponential, but perhaps more importantly, that it is bounded by a term independent of the size of the matrix.

Corollary (Worst-case performance of Temporal PageRank)**.** *Computing the Temporal PageRank with an error ϵ takes $O(m \log \frac{1}{\epsilon})$ in the worst-case, with m the number of interactions.*

Proof. The size of the matrix involved in Theorem 4 is $m \times m$. Since each interaction has at most 2 successors, the matrix P has $O(m)$ elements. This means that using a sparse implementation, an iteration of the power method will take $O(m)$ multiplications. For a certain error, the number of iterations needed is logarithmic in $\frac{1}{\epsilon}$, yielding the worst-case $O(m \log \frac{1}{\epsilon})$. □

5.3 Properties of Temporal PageRank

Now allow us to consider several aspects of Temporal PageRank. It will turn out there is not as strong of a guarantee against Sybil attacks, but historical attacks are always prevented. Furthermore, only a weaker notion of transitive trust holds, meaning that this system is not directly vulnerable to the general attack by Seuken and Parkes [26].

Proposition 1 (Properties of Temporal PageRank)**.** *Temporal PageRank satisfies IDA and ANON.*

Proof. Recall that these properties mean that scores are independent of disconnected agents, and that the scores are invariant under permutations of other agents. Both of these follow directly from the definition of PageRank, noting that in order for this to match the original definitions, we would need to consider IDA and ANON as defined on the ordered interaction graph, which can be done by considering the projection from the ordered interaction graph to the interaction graph. □

The two properties proven in Proposition 1 are quite trivial. What follows is slightly less so, and one of the motivations for considering Temporal PageRank as an accounting mechanism in the first place.

Theorem 5 (Impossibility of Historical Attacks)**.** *Consider Temporal PageRank, with W the set of nodes to which a random jump is made. Let i be an interaction.*

If there is no $j \in W$ such that $j \leq i$, then removing i with all interactions i' such that $i' \leq i$ does not change the Temporal PageRank score of any agent.

Proof. In the proof of Theorem 3, it is shown that nodes that are not in the closure of W have 0 probability in the stationary distribution. Since all interactions before i therefore cannot be in the closure either, removing all of them will not affect the stationary distribution, and hence will not affect the scores. □

This begs the question how vulnerable Temporal PageRank is to attacks in general. In particular, is it vulnerable to the style of attack that Seuken and Parkes [26] employ? The answer is no. Not because it is particularly robust, but because it simply lacks the property known as Transitive Trust.

Theorem 6 (Lack of Transitive Trust)**.** *Temporal PageRank with a uniform distribution for the jump and Winner-Takes-All does not satisfy Weak Transitive Trust.*

Proof. Consider an ordered interaction model M with agents i, j, l, where agent i has had some number N interactions, alternating with j and l, where i consumes one unit of work. Since the probability in the stationary distribution will be spread over all interactions, for large enough

N the probability on the most recent interactions will be small. Therefore, a new agent k (as in Property 3), will not be able to surpass j or l by helping the other, even if that agent provides more work to i. □

Example 7 (Lack of weak transitive trust in Temporal PageRank)**.** To further illustrate Theorem 6, a small example. Consider the block graph in Figure 5.1. Computing from perspective of p leads to the score of 0.246 for q and r.

If agent q then performs work for p, and k performs work for q, the result will be as in Figure 5.2. Here, the score of k is not surpass the score of r, which means that the definition for transistive trust is not satisfied. The result is symmetrical with respect to r and q, so the property of weak transitive trust is not satisfied.

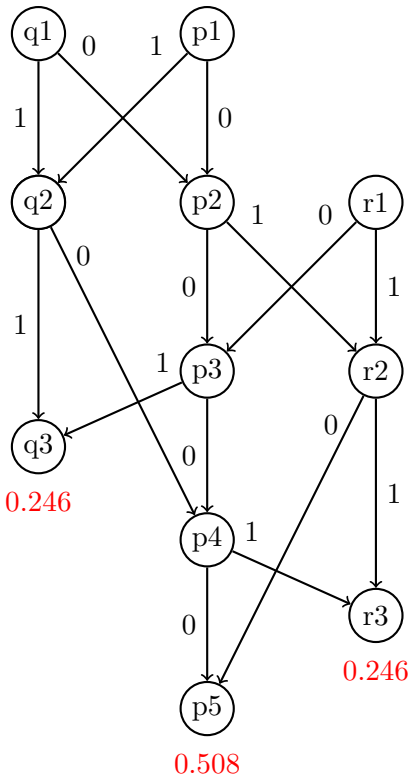


Figure 5.1: Block graph. In red, the Temporal PageRank scores of each agent, when p computes.

Theorem 6 seems to be a bad sign for Temporal PageRank. However, there still is some notion of transitivity. In particular, the score of each agent is the sum of the stationary probabilities on the interaction. In the computation, the jump probability is 1. When another interaction occurs, $1 - \beta$ of these transitions become diverted to the new interactions. Suppose agent a interacts with agent b, and b then interacts with a new agent c, where b is the only contributor in the interaction with a and c is the only contributor in the interaction with b. In this case, c will obtain a fraction $(1 - \beta)^2$ of the stationary probability of the last interaction of agent a. Furthermore, it will also obtain a fraction $(1 - \beta)$ of the stationary probability on the last interaction of agent b.

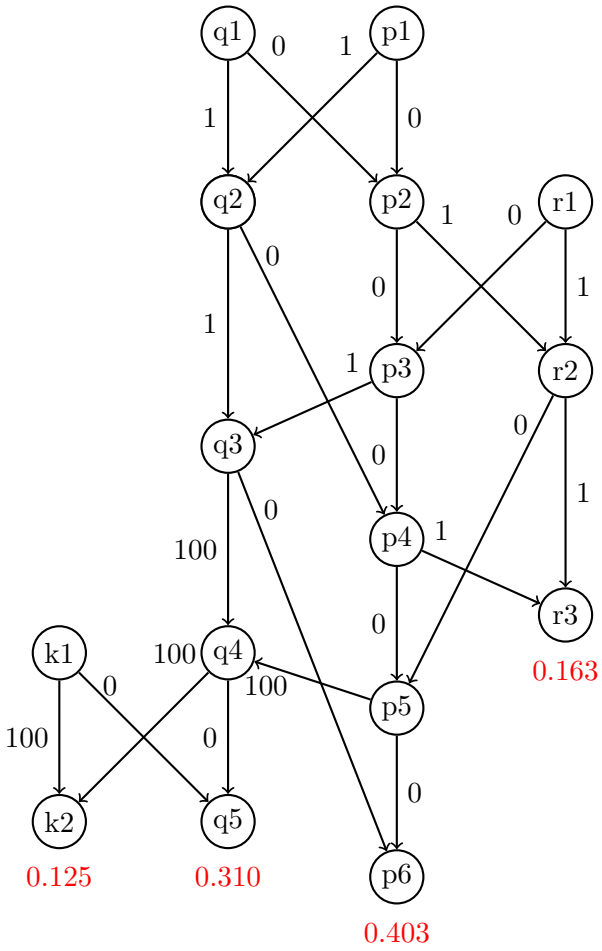


Figure 5.2: Block graph. In red, the Temporal PageRank scores of each agent, when p computes. Situation after transitive contribution.

Chapter 6

Experimental Evaluation

In Chapters 4 and 5 accounting mechanisms have been presented. In addition, theoretical properties have been proven. This chapter contains experiments on a real-world data set, showing advantages and disadvantages of each of these mechanisms.

In particular, the matter under consideration will be the properties of these mechanisms that are harder to prove theoretically. Furthermore, these experiments are used to study the behaviour of these accounting mechanisms in a real-world context.

The code to run any of these experiments is available through GitHub [1]. This code has been run on a dataset collected as below. This dataset is available upon request.

6.1 Data collection

Prior work by Norberhuis [22] includes an implementation of the MultiChain data structure into Tribler. In collaboration with Bongers and Veldhuisen, this system has been brought up to production level and released in an experimental release of Tribler, version 6.6.0[2]. This experimental release was ran by people with a close interest in Tribler. It was announced on the forums and could also be found through above GitHub page. A crawler was deployed to request MultiChain blocks from the network. A database dump from this crawler has been acquired on the 31st of August, 2016. The software was released on the 26th of July, 2016. This means that the data has been generated by users during a period that lasted a little over a month. During this period 917 identities have been observed in the network. These identities are not necessarily unique users, since users may purposefully delete their identity or users may have installed the software on multiple machines.

6.2 NetFlow evaluation

Firstly, let us consider the NetFlow mechanism. One of the first considerations will be what the influence of the parameter α is and how informative the mechanism is.

Figure 6.1 shows the NetFlow scoring mechanism for four peers, for three different values of α. These peers have been selected by sorting the list of peers by their total amount of uploaded data, and taking the peers at the 70th, 80th, 90th and 100th percentile. The peer whose point of view is taken for the computation is marked red. Each bubble is a peer whose score is computed. Note that the scale of the bubbles is different between the different pictures. This is

[1] https://github.com/pimotte/msc-experimental-code/tree/9e4eab5ea1027bb671971fab4d137c40c1c77f83
[2] https://github.com/Tribler/tribler/releases/tag/v6.6.0-exp1

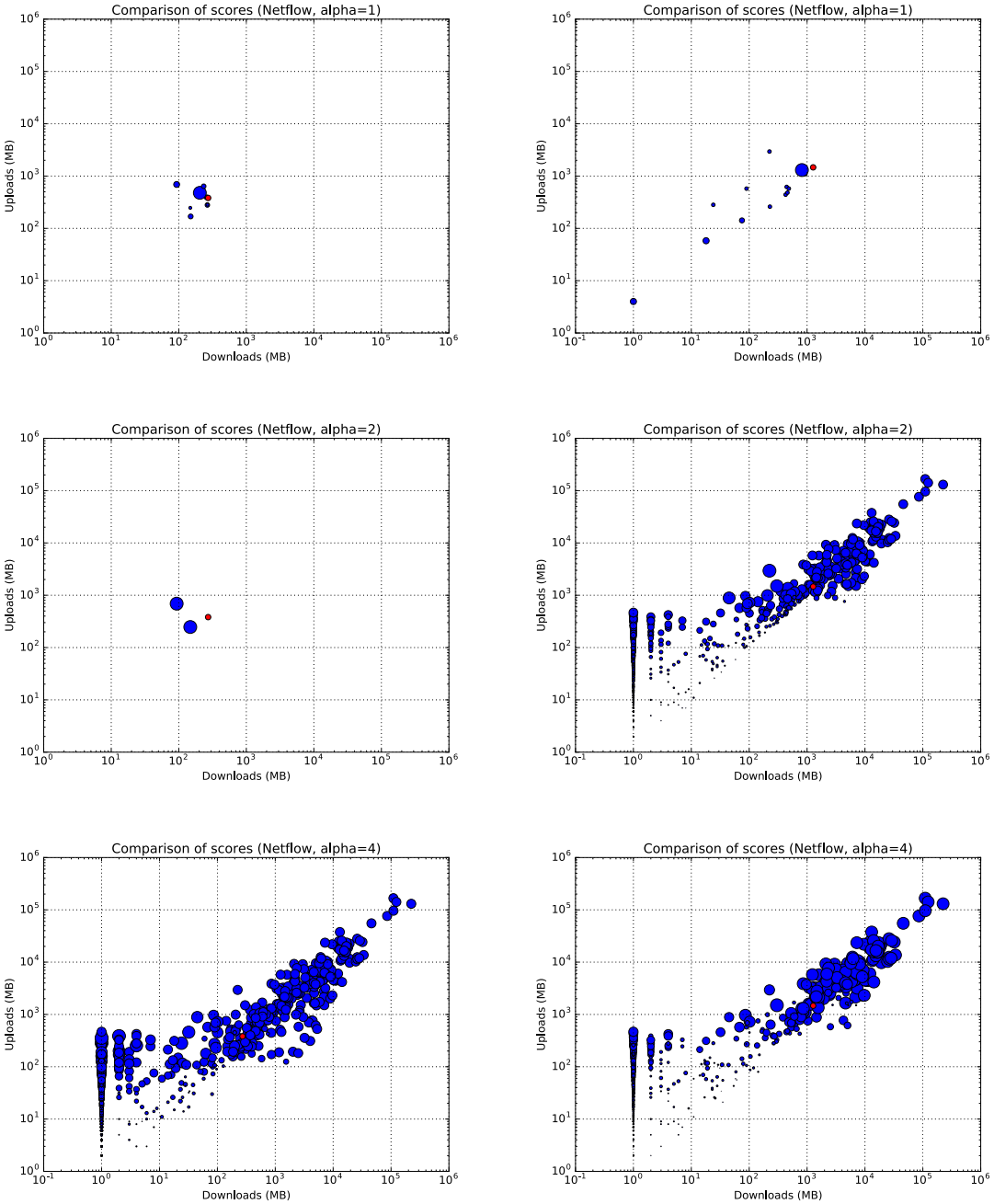
due to the fact that absolute sizing would result in indiscernible bubbles for computations from the perspective of peers with lower absolute upload and download amounts.

Upon inspection of these figures, several macro level observations come to light. The first is that increasing α does indeed increase the informativeness of the mechanism. Higher α results in non-zero scores for more peers. In particular, note that for $\alpha = 1$ a lot of the peers with higher upload/download have a zero score. This is likely due to the fact that scores of peers are not high if their contributions are limited by the consumption and contribution of the calculating peer. This effect decreases significantly. For $\alpha = 2$ and $\alpha = 4$, peers with absolute higher amounts of upload and download will often be assigned a positive score.

Increasing α does come at a cost. It can be observed that for $\alpha = 1$ no peers that have a ratio of upload/download significantly below 1, have a positive score. On the contrary, for higher α, peers that contribute less than the calculating peer might still have a non-zero score if they have contribute about the same or more in absolute terms.

Let us now consider the informativeness of this mechanism. As described in Chapter 4, one of the issues suspected was the lack of non-zero scores for nodes. If too many nodes are assigned a zero score, the mechanism does not actually yield a ranking.

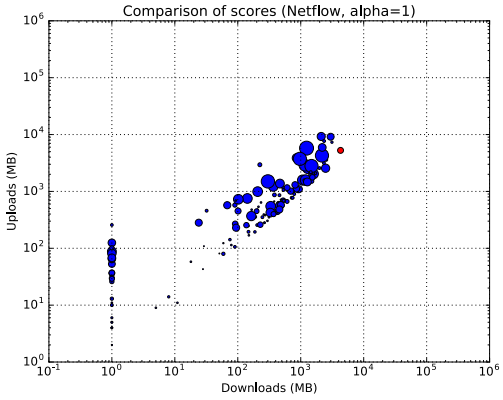
Figure 6.2 provides an informativeness curve. This is constructed as follows. For each peer, the fraction of peers that have a positive score is computed. The lines in the plot are these fractions ordered from low to high. One part of the population will never be able to increase the informativeness by scaling. Hence, Figure 6.3 shows the same data, excluding peers that have not downloaded any data. Observe that as α increases, so does the informativeness. Furthermore, for higher α the set of peers with 0 informativeness decreases in size. In addition, note that there is quite a sharp jump from 0 informativeness to around 0.7.

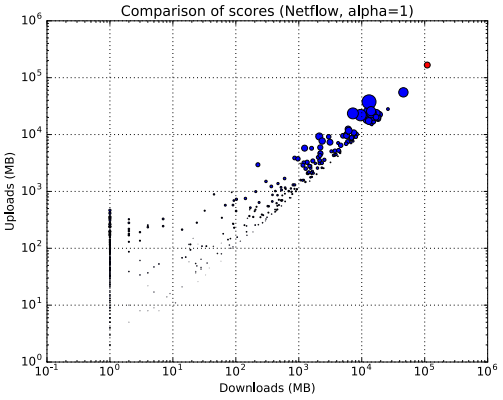


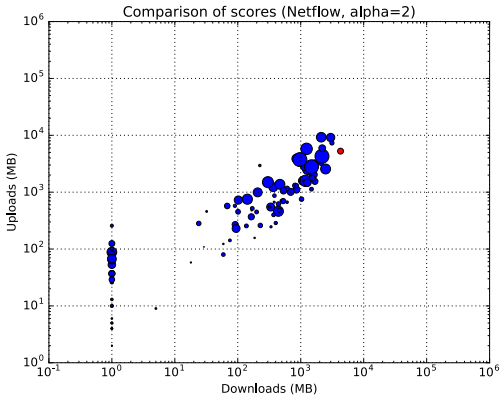
(a) Computing agent has upload and download around 120MB, for $\alpha \in \{1, 2, 4\}$

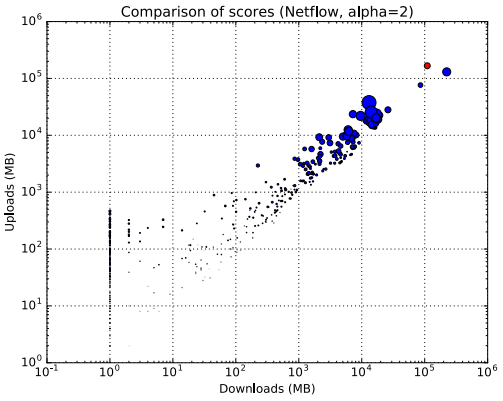
(b) Computing agent has upload and download around 1050MB, for $\alpha \in \{1, 2, 4\}$

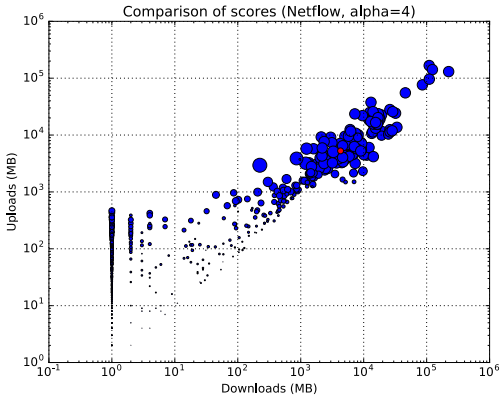
Figure 6.1: NetFlow scores from the perspective of several nodes, continues on next page... Sizes of bubbles indicate scores (different scales). Red bubble is the computing agent.



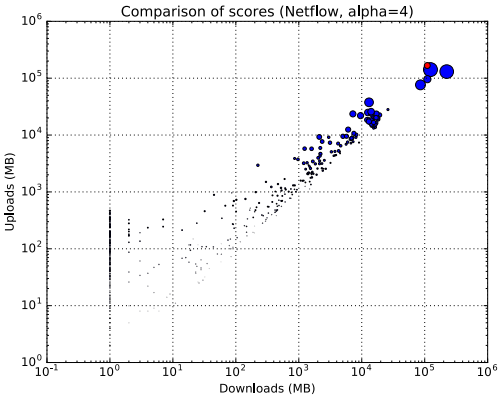








(c) Computing agent has upload and download around 1400MB, for $\alpha \in \{1, 2, 4\}$



(d) Computing agent has upload and download around 100GB, for $\alpha \in \{1, 2, 4\}$

Figure 6.1: NetFlow scores from the perspective of several nodes, continued.
Sizes of bubbles indicate scores (different scales). Red bubble is the computing agent.

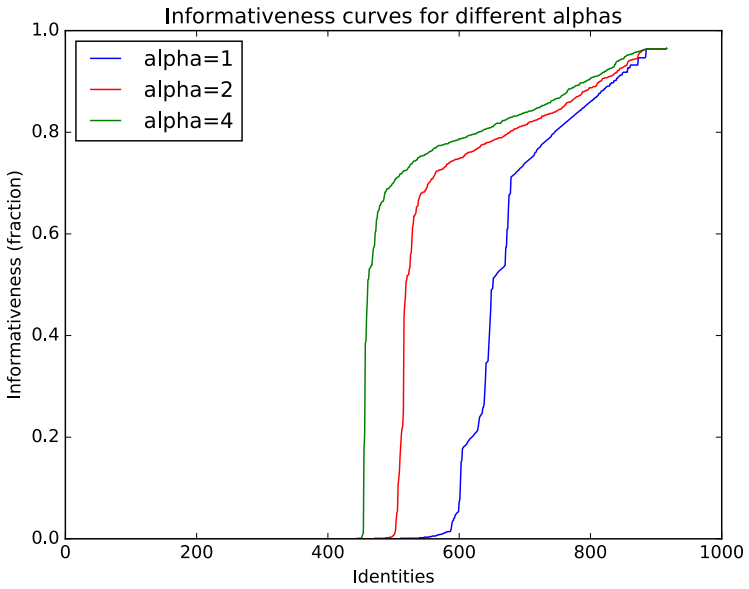


Figure 6.2: Informativeness curves for different α

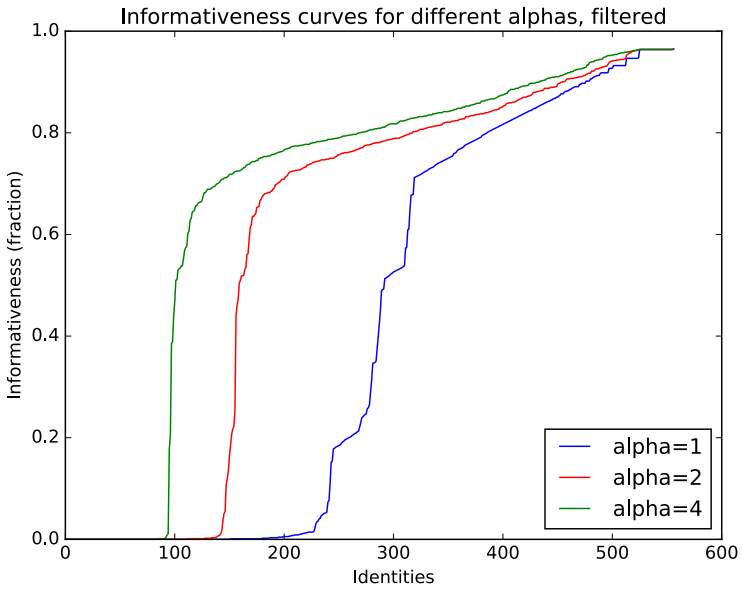


Figure 6.3: Informativeness curves for different α, excluding peers without downloads

6.3 Temporal PageRank Evaluation

Let us turn our attention to Temporal PageRank, starting with several examples of results of score computations from the perspective of a variety of peers, selected in the same way as for NetFlow, but including some of the lower quantile representatives.

Looking at Figure 6.4 leads to several insights. Firstly, note that in contrast with NetFlow, these figures are all on the same scale, and that with the exception of a few nodes, most nodes are assigned somewhat the same score, regardless of the performance of the computing node. The general pattern is that nodes with both higher upload and download get a higher score and that the absolute amount matters more than the ratio between the two. However, there appears to be a slight overall bias towards peers that have uploaded more than that they have downloaded.

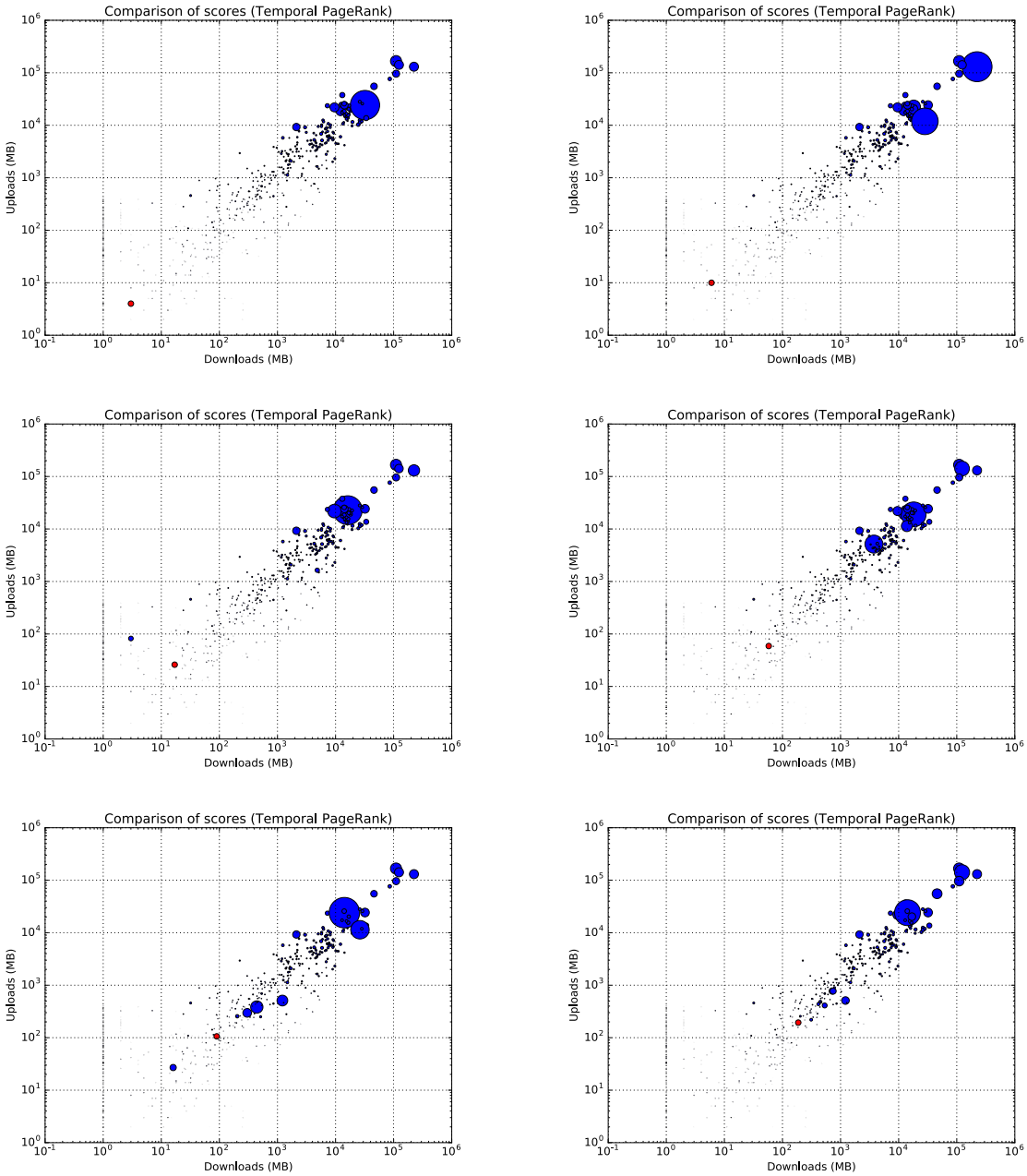


Figure 6.4: Temporal PageRank computed by several peers, continues on next page...
Size of bubbles indicate scores. The scale is the same for all graphs. Red bubble is the computing agent.

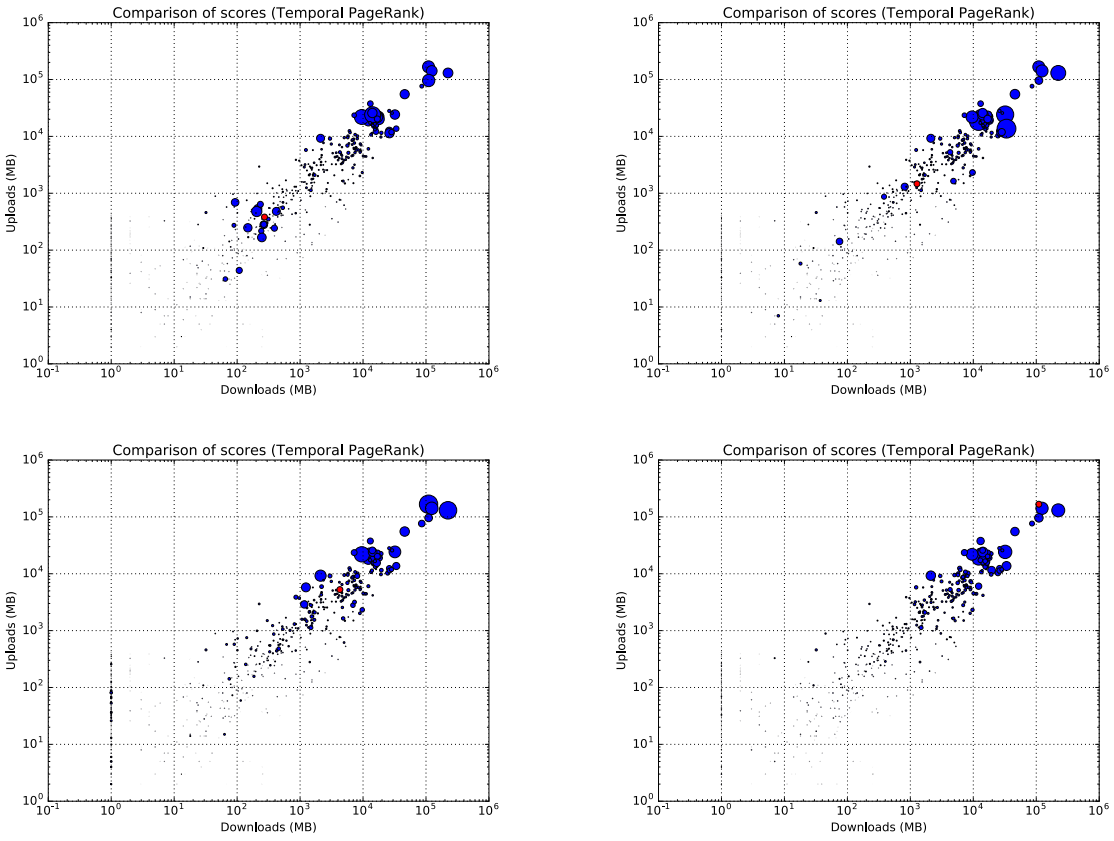


Figure 6.4: Temporal PageRank computed by several peers, continued.
Size of bubbles indicate scores. The scale is the same for all graphs. Red bubble is the computing agent.

6.4 Cross-mechanism Evaluation

Since the end result of an accounting mechanism will often be used as a ranking, let us consider the common peers in top fractions of rankings. It makes sense to compare each of NetFlow and Temporal PageRank with the mechanisms that inspired them: BarterCast and PageRank. Furthermore, it might be fruitful to inspect the same comparison for NetFlow and Temporal PageRank in order to see if they approximate each other well.

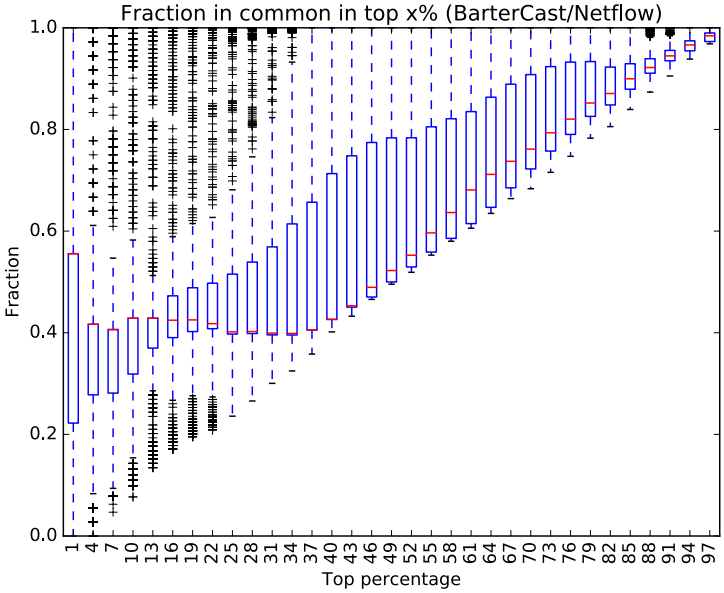


Figure 6.5: Box plots of fraction of common peers in rankings for BarterCast and NetFlow

Figure 6.5 shows the following: On the x-axis, we have the part of the ranking that is under consideration. For example, 10 means we compare the top 10% for both accounting mechanisms. The corresponding box plot then depicts the data set obtained if this comparison is done for every calculating peer. It shows the fraction in common between the two ranking methods. A simple calculation results in the observation that if this were completely random, the medians would lie on the line $y = x$. In this case, for the rankings that include up to 40% of participants the median is about 40% in common between rankings. While this is above the baseline, it is not an extreme overlap. This implies that there a difference, but it looks very structured. Once the rankings get larger, there is relatively less overlap, which can be attributed to NetFlows characteristic of cutting off scores at 0, leading to a lot of ties.

The same comparison can be found in Figure 6.6 for the mechanisms of PageRank and Temporal PageRank. Here the correlation is abundantly clear. Only for the top 1% the median fraction in common is below 0.7. This is not unexpected, as both of these methods have the same underlying structure. PageRank uses less information, but there is no particular reason Temporal PageRank would result in different rankings, since there is no reason to assume the test set has any sort of malicious activity.

Finally, consider Figure 6.7. In this figure, the same metric is depicted, but for NetFlow and Temporal PageRank. Note that the medians lie almost exactly on the expectation if the rankings were random. While the outliers are clearly biased towards having more in common, there is still very little that these rankings have in common.

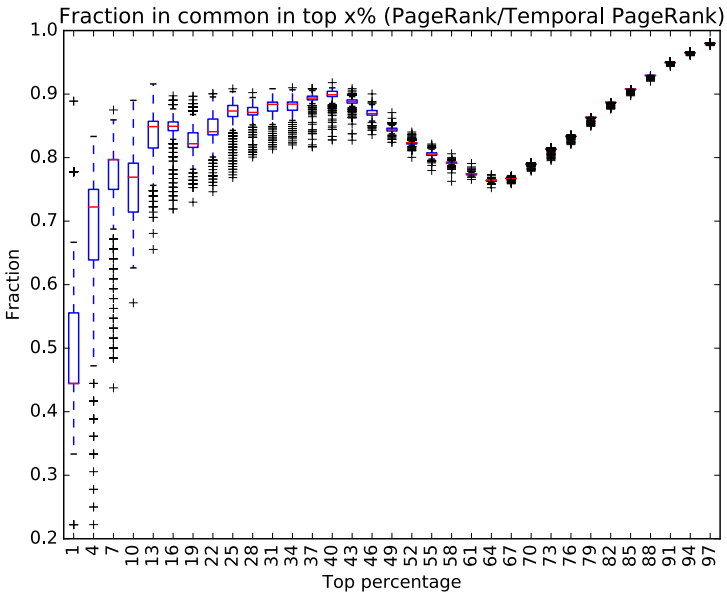


Figure 6.6: Box plots of fraction of common peers in rankings for PageRank and Temporal PageRank

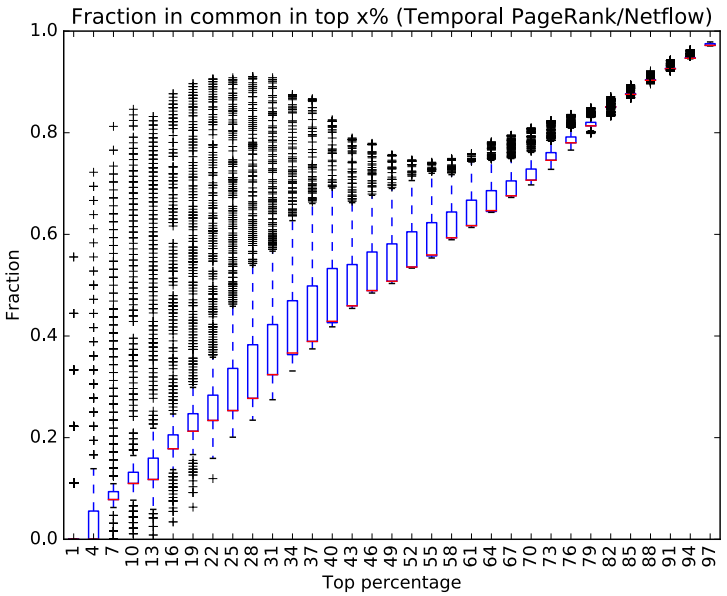


Figure 6.7: Box plots of fraction of common peers in rankings for NetFlow and Temporal PageRank

6.5 Performance Evaluation

In order to evaluate the performance of both NetFlow and Temporal PageRank, the dataset used for other experiments was again leveraged. In order to study the behaviour with increasing an increasing number of agents and interactions, the data set was replayed, and the computation time was measured for 1 computation for several points in the history. These points are chosen such that both for the number of agents and the number of interactions there is a good spread of sample points. NetFlow is considered with $\alpha = 2$, Temporal PageRank with a uniform distribution after jumping.

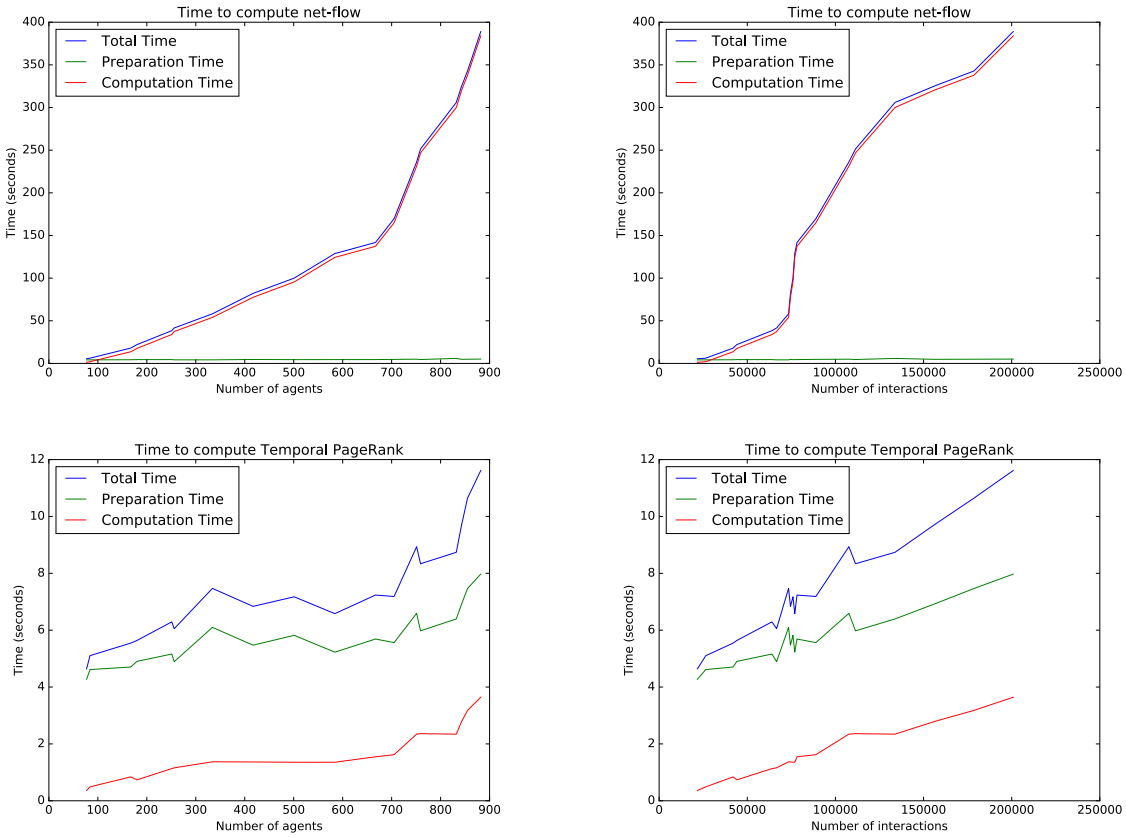


Figure 6.8: Computation times for NetFlow and Temporal PageRank

In Figure 6.8 the computation times are shown. For both NetFlow and Temporal PageRank, consider the times with the number of agents and the number of interactions on the x-axis. Also note that for each we have split the times into build time, which is the construction of the data structure on which the algorithm is executed and the computation time of the algorithm itself. Note the sharp increase in computation time for NetFlow as a function of the number of interactions. This can be explained by the fact that around that moment in time, there was a sharp jump in the number of agents, see Figure 6.9. Note that both implementations use the Networkx library, but Temporal PageRank is one library call, whereas the NetFlow algorithm involves $O(n)$ calls to a max-flow subroutine. This leads to the suspicion that not only algorithmic optimizations might be possible for NetFlow, as mentioned at the end of Chapter 4, but there is also room for programmatic optimization, in particular in the reuse of data structures. That being said, it is clear that Temporal PageRank is much easier to compute than NetFlow for the testing data, which matches the theoretical worst-case analysis.

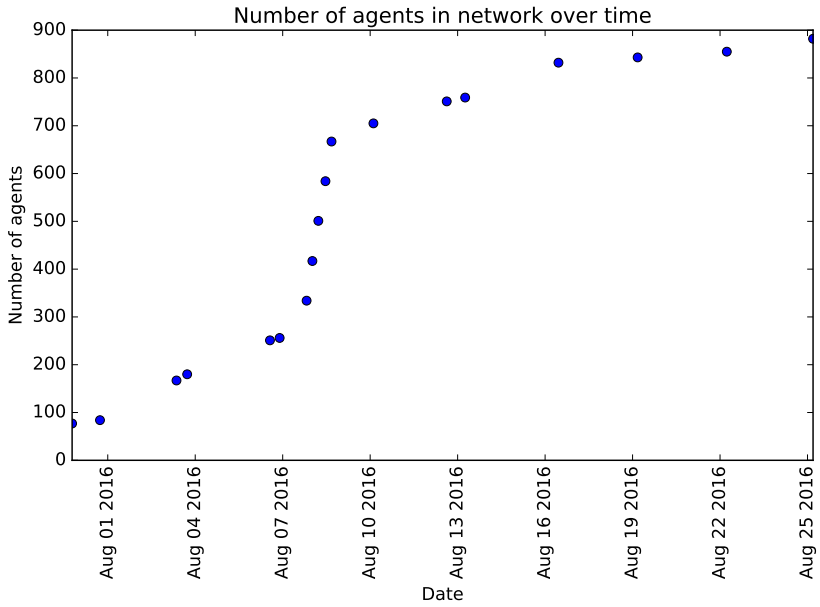


Figure 6.9: Number of agents in the network over time

Chapter 7

Conclusion and Discussion

In this thesis, we have considered a distributed system and proposed two accounting mechanisms in order to compute trust and reputation in said systems. We consider the fairness, resistance to manipulation and computability of these mechanisms.

7.1 Fairness

As mentioned, the contribution of peers to a BitTorrent system have traditionally been judged by their upload/download ratio. In the context of a system like Tribler, this is not an accurate measure of contribution, because the act of relaying would need to be rewarded. When elaborating on the research question, fairness kept vague. However, with the results presented in Chapter 6, in particular Figures 6.1 and 6.4, this notion becomes a little more clear.

Using NetFlow with $\alpha = 1$ is decidedly unfair. Agents that are not the biggest contributors scores with an unlogical pattern. Agents that have contributed and consumed more, but still have a net positive ratio are often assigned zero valued scores. For higher α, a clear pattern emerges, where agents that contribute and consume more or similar amounts are assigned the highest scores in a system. For agents that contribute less, there is a clearly visible pattern that matches with the concept of contributing more giving a higher score. In addition, peers with significantly lower absolute contribution and an upload/download ratio below 1 are assigned a zero score. These patterns are consistent with a fair ranking.

Likewise, Temporal PageRank provides a somewhat fair ranking. While the patterns here may not be as monotonous as for NetFlow, but the general trend of contributing more resulting in a higher score is still present. What is different from NetFlow is that for Temporal PageRank, the overall scoring of the agents is the same, with the exception of up to some 10 peers. The agents that are the exception have a higher than average score, likely because they have had direct interaction with the calculating agent. Also note that as the calculating peer contributes more, this effect is spread over more agents, and the magnitude of the difference drops.

In conclusion, both NetFlow for $\alpha = 2$, $\alpha = 4$ and Temporal PageRank provide fair mechanisms, whereas Net-Flow with $\alpha = 1$ does not.

7.2 Resistance to Manipulation

One of the major advantages of NetFlow would have to be that it is resistant to profitable Sybil attacks, as detailed in Theorem 1. This becomes a little less impressive if we take into account that $\alpha = 1$ does not yield a fair accounting mechanism. That being said, taking $\alpha > 1$ will still yield an accounting mechanism with a well-established bound on the profits a Sybil

attack. Temporal PageRank, by contrast, does not have such a particular guarantee. It is not susceptible to the particular attack of Seuken and Parkes' impossibility result. However, it is vulnerable to some of the known attacks on normal PageRank. Temporal PageRank, however does have the resistance against historical attacks, which implies that only interactions after the first attack edge are considered.

7.3 Performance

In Figure 6.8 the performance of both NetFlow and Temporal PageRank are shown. Recall the theoretical analysis, in which the NetFlow is shown to have worst case computational cost of $O(n^3m)$ and Temporal PageRank $O(m \log \frac{1}{\epsilon})$. For a constant error in the PageRank computation, this comes down to a difference of factor of n^3. This certainly shows in the performance. For Temporal PageRank, the time to build the data structure is in fact comparable to the actual computation time, whereas in the case of NetFlow, the computation time quickly eclipses the time to build the data structure. In comparison, Temporal PageRank is much more feasible to compute. It might depend on the actual system whether these algorithms are fast enough.

In the case of Tribler, the performance of Temporal PageRank for the size of the test set is good enough for the applications since it is at most 12 seconds, where 30 seconds is the established boundary. For actual application it is not quite performant enough still, since it is desirable for the network to be able to scale more. The test set contained a little over 900 agents, where scaling to 10.000 would be the goal. Also note that the complexity depends on the number of interactions, which will always keep growing over time, even if the number of agents is stable. NetFlow can take over 6 minutes, which is too long for practical application in Tribler. Computational tricks would be needed to ensure Temporal PageRank scales to the desired levels. In case of NetFlow, just computational tricks are unlikely to make it fast enough for practical application in Tribler and the algorithm itself needs reconsideration.

7.4 General conclusion

The research question asked if an accounting mechanism could use confirmed information about interactions and their order allow the design of a fair, manipulation-resistant and efficient reputation system. The answer these two mechanisms and their study leads to is a tentative "yes". Temporal PageRank is fair, and without question performant. It also has some desirable manipulation-resistant properties. NetFlow, with $\alpha > 1$ looks to yield a fair accounting mechanism. Furthermore, it has some of the strongest Sybil-resistant properties of known accounting mechanisms. However, the fact that a state-of-the-art implementation would still take $O(nm^2)$ makes it not performant enough for some applications.

7.5 Discussion

In this discussion, we consider two directions of future work. One is the possibility of future research on the mechanisms themselves, the other is the broader perspective in which this research can be placed, including other application areas.

7.5.1 Future Research

Both the mechanisms presented have areas where improvement can be obtained. In the case of Temporal PageRank, a more precise study should be done towards its fairness. Furthermore, since its resistance against Sybil attacks is not quite as strong as NetFlows, more research should

be done into hardening the theoretical resistance against Sybil attacks, or to study what Sybil attacks are possible and determine the risks involved.

As for NetFlow, most of the improvement is to be gained in performance. A smart way to compute more flows in one go, or a slightly different mechanism that is easier to compute, but has the same theoretical resistance against Sybil-attacks is desirable. The reference implementation can also be improved to see if programming more efficiently would make this method suitable for more systems.

7.5.2 This Research in Broader Context

The accounting mechanisms presented here have been designed with the use case of Tribler in mind. In this case, a resource is exchanged (data), but it is not of extreme importance that there is an exact balance. In other distributed systems, a currency might be involved. In this case, there is still a resource involved, but the balance is important. In other applications, like social networks, there is no resource involved at all, and "trusting someone" is often a symmetric relationship.

Systems like NetFlow and Temporal PageRank could be applicable in other distributed systems. Depending on the exact properties of said system, one of them could be applied with little alteration, but in other cases it might require non-trivial customizations. One example is the application to a peer-to-peer lending system. If the interactions represent real money, the attacks on NetFlow from Section 4.4 present a very real risk.

Bibliography

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows: theory, algorithms, and applications. 1993.

[2] UN General Assembly. Universal declaration of human rights. *UN General Assembly*, 1948.

[3] Bram Cohen. The bittorrent protocol specification, 2008.

[4] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[5] Yefim Dinitz. Dinitz' algorithm: The original version and Even's version. In *Theoretical computer science*, pages 218–240. Springer, 2006.

[6] John R Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[7] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[8] Assafa Endeshaw. Internet regulation in China: the never-ending cat and mouse game 1. *Information & Communications Technology Law*, 13(1):41–57, 2004.

[9] Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[10] Dimitra Gkorou, Johan Pouwelse, and Dick Epema. Trust-based collection of information in distributed reputation networks. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 2312–2319. ACM, 2015.

[11] Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[12] Ralph E Gomory and Tien Chung Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[13] Garrett Hardin. The tragedy of the commons. *Journal of Natural Resources Policy Research*, 1(3):243–253, 2009.

[14] Taher Haveliwala and Sepandar Kamvar. The second eigenvalue of the Google matrix. *Stanford University Technical Report*, 2003.

[15] Shanthi Kalathil and Taylor C Boas. The internet and state control in authoritarian regimes: China, Cuba and the counterrevolution. *First Monday*, 6(8), 2001.

[16] Sepandar D Kamvar, Mario T Schlosser, and Hector 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, 2003.

[17] Kai Kimppa and Farid Shirazi. The emancipatory role of information and communication technology: A case study of internet content filtering within Iran. *Journal of Information, Communication and Ethics in Society*, 8(1):57–84, 2010.

[18] Valerie King, Satish Rao, and Rorbert Tarjan. A faster deterministic maximum flow algorithm. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 157–164. Society for Industrial and Applied Mathematics, 1992.

[19] Jian Liang, Rakesh Kumar, Yongjian Xi, and Keith W Ross. Pollution in p2p file sharing systems. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 2, pages 1174–1185. IEEE, 2005.

[20] David Lyon. Surveillance, Snowden, and big data: Capacities, consequences, critique. *Big Data & Society*, 1(2):2053951714541861, 2014.

[21] Michel Meulpolder, Johan A Pouwelse, Dick HJ Epema, and Henk J Sips. Bartercast: A practical approach to prevent lazy freeriding in p2p networks. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8. IEEE, 2009.

[22] Steffan D Norberhuis. Multichain: A cybercurrency for cooperation, 2015.

[23] James B Orlin. Max flows in o (nm) time, or better. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 765–774. ACM, 2013.

[24] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: bringing order to the web. 1999.

[25] Frank La Rue. Report of the special rapporteur on the promotion and protection of the right to freedom of opinion and expression. Technical report, Human Rights Council of the United Nations, 2011.

[26] Sven Seuken and David C Parkes. Sybil-proof accounting mechanisms with transitive trust. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 205–212. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[27] Sven Seuken, Jie Tang, and David C Parkes. Accounting mechanisms for distributed work systems. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. AAAI Press, 2010.

[28] International Telecommunications Union. Ict facts and figures 2016. Technical report, ITU, 2016.

[29] Haifeng Yu, Phillip B Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *2008 IEEE Symposium on Security and Privacy*, pages 3–17. IEEE, 2008.