



**QLectives – Socially Intelligent Systems for Quality**  
**Project no. 231200**

**Instrument: Large-scale integrating project (IP)**  
**Programme: FP7-ICT**

**Deliverable D2.1.3**

*Empirically informed algorithm design for cooperation -  
experiments and results*

Submission date: 2012-02-17

Start date of project: 2009-03-01

Duration: 48 months

Organisation name of lead contractor for this deliverable: TUD

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination level		
<b>PU</b>	Public	<b>X</b>
<b>PP</b>	Restricted to other programme participants (including the Commission Services)	
<b>RE</b>	Restricted to a group specified by the consortium (including the Commission Services)	
<b>CO</b>	Confidential, only for members of the consortium (including the Commission Services)	



## Document information

### 1.1 Author(s)

Author	Organisation	E-mail
Tamás Vinkó	TUD	T.Vinko@tudelft.nl
Adele Jia	TUD	L.Jia@tudelft.nl
Dimitra Gkoru	TUD	d.gkorou@tudelft.nl
Rameez Rahman	TUD	rrameez@gmail.com
Nitin Chiluka	TUD	N.J.Chiluka@tudelft.nl
Dick Epema	TUD	D.H.J.Epema@tudelft.nl
Johan Pouwelse	TUD	J.A.Pouwelse@tudelft.nl

### 1.2 Other contributors

Name	Organisation	E-mail
------	--------------	--------

### 1.3 Document history

Version#	Date	Change
V0.1	December 1, 2011	First draft
V0.5	December 13, 2011	Major corrections and clarifications
V0.9	December 16, 2011	First final draft for internal consortium
V1.0	February 17, 2012	Approved version to be submitted to EC

### 1.4 Document data

Keywords	BitTorrent, design space analysis, sharing ratio enforcement, distributed reputation
Editor address data	T.Vinko@tudelft.nl
Delivery date	17 February, 2011

### 1.5 Distribution list

Date	Issue	E-mail
	Consortium members	QLECTIVES@list.surrey.ac.uk
	Project officer	Jose.FERNANDEZ-VILLACANAS@ec.europa.eu
	EC archive	INFSO-ICT-231200@ec.europa.eu

## QLectives Consortium

This document is part of a research project funded by the ICT Programme of the Commission of the European Communities as grant number ICT-2009-231200.

### **University of Surrey (Coordinator)**

Department of Sociology/Centre  
for Research in Social Simulation  
Guildford GU2 7XH  
Surrey  
United Kingdom  
Contact person: Prof. Nigel Gilbert  
E-mail: n.gilbert@surrey.ac.uk

### **Technical University of Delft**

Department of Software Technology  
Delft, 2628 CN  
Netherlands  
Contact Person: Dr Johan Pouwelse  
E-mail: j.a.pouwelse@tudelft.nl

### **ETH Zurich**

Chair of Sociology, in particular  
Modelling and Simulation  
Zurich, CH-8092  
Switzerland  
Contact person: Prof. Dirk Helbing  
E-mail: dhelbing@ethz.ch

### **University of Szeged**

MTA-SZTE Research Group on  
Artificial Intelligence  
Szeged 6720, Hungary  
Contact person: Dr Mark Jelasity  
E-mail: jelasity@inf.u-szeged.hu

### **University of Fribourg**

Department of Physics  
Fribourg 1700  
Switzerland  
Contact person: Prof. Yi-Cheng Zhang  
E-mail: yi-cheng.zhang@unifr.ch

### **University of Warsaw**

Faculty of Psychology  
Warsaw 00927  
Poland  
Contact Person: Prof. Andrzej Nowak  
E-mail: nowak@fau.edu

### **Centre National de la Recherche Scientifique, CNRS**

Paris 75006,  
France  
Contact person: Dr. Camille ROTH  
E-mail: camille.roth@polytechnique.edu

### **Institut für Rundfunktechnik GmbH**

Munich 80939  
Germany  
Contact person: Dr. Christoph Dosch  
E-mail: dosch@irt.de

## QLectives introduction

QLectives is a project bringing together top social modelers, peer-to-peer engineers and physicists to design and deploy next generation self-organising socially intelligent information systems. The project aims to combine three recent trends within information systems:

- **Social networks** - in which people link to others over the Internet to gain value and facilitate collaboration
- **Peer production** - in which people collectively produce informational products and experiences without traditional hierarchies or market incentives
- **Peer-to-Peer systems** - in which software clients running on user machines distribute media and other information without a central server or administrative control

QLectives aims to bring these together to form Quality Collectives, i.e. functional decentralised communities that self-organise and self-maintain for the benefit of the people who comprise them. We aim to generate theory at the social level, design algorithms and deploy prototypes targeted towards two application domains:

- **QMedia** - an interactive peer-to-peer media distribution system (including live streaming), providing fully distributed social filtering and recommendation for quality
- **QScience** - a distributed platform for scientists allowing them to locate or form new communities and quality reviewing mechanisms, which are transparent and promote quality

The approach of the QLectives project is unique in that it brings together a highly inter-disciplinary team applied to specific real world problems. The project applies a scientific approach to research by formulating theories, applying them to real systems and then performing detailed measurements of system and user behaviour to validate or modify our theories if necessary. The two applications will be based on two existing user communities comprising several thousand people - so-called "Living labs", media sharing community [tribler.org](http://tribler.org); and the scientific collaboration forum [EconoPhysics](http://EconoPhysics).

# Executive summary

This deliverable aims at designing algorithms for cooperation which are empirically informed. It is focusing on the QMedia application domain, particularly BitTorrent related issues. The empirical information comes from two directions: either from measurements of deployed systems from which the design of the algorithms are obtaining inspirations and getting tested; or from experiments done at emulation level on local computer clusters, in order to have realistic technological effects on the designed algorithms at their testing phase.

The main contributions of this deliverable are:

- In Chapter 2, we implement modifications to BitTorrent and with experiments on a computer cluster, analyze some protocols discovered using the Design Space Analysis (DSA) we reported in D2.1.2. We show that they yield higher system performance and robustness as compared to the reference implementation, thus demonstrating the effectiveness of DSA.
- In Chapter 3, based on measurement of BitTorrent communities, we notice the presence of oversupply, that is, there are much more supply than demand; which has two undesired negative effects: a) Peers are forced to seed for long times, even though their seeding efforts are often not very productive (in terms of low upload capacity utilization); and b) sharing ratio enforcement (SRE) discriminates against peers with low bandwidth capacities and forces them to seed for longer durations than peers with high capacities. To alleviate these problems, we propose four different strategies for SRE, which have been inspired by ideas in social sciences and economics. We evaluate these strategies through simulations.
- Finally, in Chapter 4, summarizing an ongoing work, we propose a scheme for reducing the amount of history maintained in decentralized interaction-based reputation systems based on elements such as the age of nodes, and we explore its effect on the computed reputations showing its effectiveness in both synthetic and real-world graphs.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Design Space Analysis of BitTorrent</b>	<b>3</b>
2.1	Results and Discussion . . . . .	3
2.2	Validation of DSA Results . . . . .	12
<b>3</b>	<b>The Reward and Punishment of Sharing Ratio Enforcement</b>	<b>14</b>
3.1	Support from real world observations . . . . .	15
3.2	Experimental setup . . . . .	17
3.3	The Reward and Punishment of SRE . . . . .	18
3.4	Description of proposed strategies . . . . .	22
3.5	Strategy evaluation . . . . .	25
3.6	Discussion: The change of user behavior? . . . . .	27
<b>4</b>	<b>Reducing History in Reputation Systems</b>	<b>30</b>
4.1	Motivation and Problem Statement . . . . .	31
4.2	Creating the Reduced History . . . . .	32
4.3	Datasets . . . . .	35
4.4	Computation of Reputations and Evaluation Metrics . . . . .	38
4.5	Evaluation . . . . .	39
<b>5</b>	<b>Summary</b>	<b>45</b>
<b>A</b>	<b>Introduction to Design Space Analysis</b>	<b>51</b>
A.1	Key elements of DSA . . . . .	51
A.2	The PRA quantification . . . . .	52
A.3	Applying DSA to P2P File Swarming Systems . . . . .	53

# Chapter 1

## Introduction

This deliverable is focusing on the QMedia application domain, particularly BitTorrent related issues: incentives to encourage nodes to follow the prescribed protocol, sharing ratio enforcement and reputation.

Taking inspirations from Axelrod [9] as well as from deliverables D1.1.1 and D2.1.1, we aimed to devise a method which can be used to analyze BitTorrent protocols more comprehensively. In D2.1.2 we presented a simulation-based approach called Design Space Analysis (DSA) together with some initial results from applying the DSA approach on a design space of the BitTorrent file swarming protocol. Here, in Chapter 2 we are giving full details, analysis and explanations.

Promoting seeding for media sharing is one of the main focal point of QLectives. Our previous research on the so-called sharing ratio enforcement (SRE) in private BitTorrent communities had been reported in deliverables D2.1.1 and D2.1.2. In particular, citing from D2.1.1: *“A requirement for QMedia is to produce the levels of cooperation (seeding) found in private communities without the need for a central website or administration. That is, we wish to provide a distributed incentive system of sufficient quality to support efficient downloading and video-on-demand.”* Thus, we need to further understand the mechanisms and effects of sharing ratio enforcement. In Chapter 3, using extensive simulations as well as measurements from real communities, we notice the drawbacks of using SRE. To alleviate these problems, we propose four different strategies for SRE, which have been inspired by ideas in social sciences and economics. We evaluate these strategies through simulations and discuss the possible consequences. For further research on the sharing behavior of BitTorrent users we refer to the deliverable D1.4.1 (Modeling the dynamics of quality collectives).

In D1.3.1 the role of time in recommendation and reputation systems was dis-

cussed. In Chapter 4 we propose a scheme for reducing the amount of history maintained in decentralized interaction-based reputation systems such as QMedia and its BarterCast mechanism (see deliverable D2.1.2 and D4.3.2 for further details on BarterCast). This chapter is using updated version of a dataset from D3.1.1.

All the three chapters of this current deliverable can be directly used later in the project, in particular the development of the next version of QMedia in Stream 4.

# Chapter 2

## Design Space Analysis of BitTorrent

Distributed systems without a central authority, such as peer-to-peer (P2P) systems, employ incentives to encourage nodes to follow the prescribed protocol. In order to comprehensively model incentives in complex protocols, we propose a simulation-based method, which we call *Design Space Analysis* (DSA). DSA provides a tractable analysis of competing protocol variants within a detailed design space. We apply DSA to P2P file swarming systems. With extensive simulations we analyze a wide-range of protocol variants and gain insights into their robustness and performance. To validate these results and to demonstrate the efficacy of DSA, we modify an instrumented BitTorrent client and evaluate protocols discovered using DSA. We show that they yield higher system performance and robustness relative to the reference implementation.

In D2.1.2 we have already shown some initial results from applying the DSA approach on a design space of the BitTorrent file swarming protocol. Here we are giving full details, analysis and explanations. For the reader's convenience, we summarize the technical details of our DSA approach in the Appendix of this report.

### 2.1 Results and Discussion

Figure 2.1 shows all the 3270 protocols actualized in Section A.3.2 in the Appendix, with their normalized Robustness and Performance values. Given the methodology of conducting the PRA quantification as described in Section A.3.3, this figure represents a synthesis of 107 million individual runs. For performance, each point represents the average normalized performance of a protocol  $\Pi$ , over 100 runs. The robustness values are calculated as described in Section A.3.3 of the Appendix. The maximum variance in the runs for each protocol's performance and robustness was as low as 0.0014 and

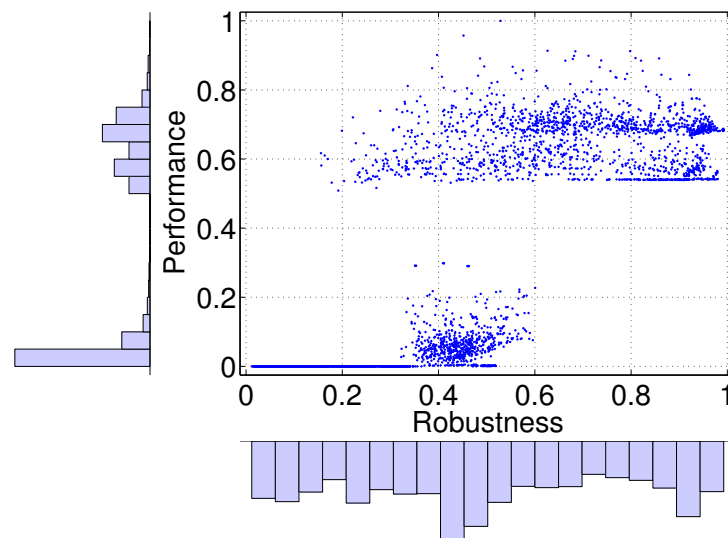


Figure 2.1: Scatter plot of all 3270 protocols in the design space with Robustness against Performance. The results presented here are a synthesis of over 107 million individual simulation runs. Histograms are also shown.

0.0206, respectively.

### 2.1.1 Performance

We shall start by looking at different regions of Figure 2.1. It can be seen that numerous protocols have both very low performance and robustness in the range  $[0, 0.2]$  (as can be seen from the large histogram bars in the bottom left hand corner). Upon inspection we discover that most of them are freeriders, although different kinds of freeriders. The freeriders with low robustness usually defect on strangers, while freeriders with low performance usually defect on their partners. The maximum performance that such freeriders get is 0.31. This can be seen in the form of two clusters in Figure 2.1, where the lower and upper clusters are below and above 0.4 performance, respectively.

The lower cluster for performance contains all freeriders that do not reciprocate with their partners. It also contains some freeriders that defect on strangers. The upper cluster for performance does not contain any freeriders that defect on their partners but interestingly does contain freeriders who defect on strangers. In fact, the protocol with the highest performance (of 1), is the one that always defects on strangers, has the *Sort Slowest* ranking function, and maintains one partner. We try to dissect why this particular protocol performs so highly. By defecting on a stranger, a protocol *in effect* uploads nothing to the stranger. Since, all peers follow the *Sort Slowest* ranking func-

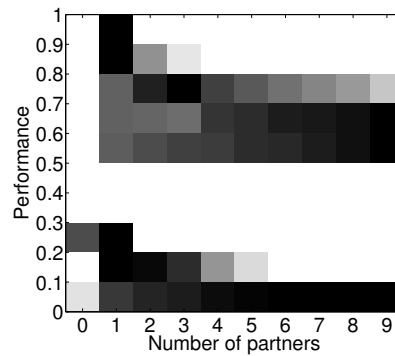


Figure 2.2: Normalized Performance histograms for different partner values. Darker squares represent high ‘partner value’ frequency for a particular Performance interval.

tion, a peer  $p_1$  on downloading nothing from another peer  $p_2$ , immediately discards its partner  $p_3$  and chooses  $p_2$  to be its partner. Peer  $p_3$  now finds itself partner-less. However, it can also quickly find a partner for itself by uploading nothing to another peer. Thus by following this protocol, peers rarely find themselves without a fully occupied partner set and can always download at full speed.

Thus, counter-intuitively, by maintaining a single partner, by not uploading anything to strangers and by employing the *Sort Slowest* ranking function, a very high performing protocol can be designed. It is imperative of course in this case that the *resource allocation* method should *not* be *Prop Share*. This is because *Prop Share* will ensure that a peer on getting nothing from another peer, also uploads nothing in response. If that happens, the entire population that follows this protocol will fail to bootstrap.

In Figure 2.2 we observe that in fact many of the top performing protocols are those with one partner. In Figure 2.2, for each performance interval, darker squares represent higher relative frequency of the number of partners. The empty white spaces in Figure 2.2 reflect the empty spaces in the scatter plot and the histogram of the Performance values in Figure 2.1. All the top 15 performing protocols maintain one partner each and the overall trend in the top performing protocols shows a low number of partners. In the top hundred performing protocols only 11 maintain more than 2 partners. It can be seen from Figure 2.2, starting from the top, till the performance interval  $[0.7, 0.8]$ , the frequency of low number of partners is relatively higher than the frequency of high number of partners.

We analyze these high performing protocols with a low number of partners more closely now. We have already discussed the features of the highest performing protocol, which uses the *Sort Slowest* ranking function. However, not all high performing protocols with low number of partners have the same features. Many of them employ

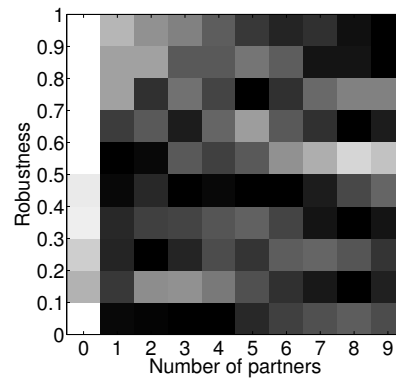


Figure 2.3: Normalized Robustness histograms for different partner values. Darker squares represent high ‘partner value’ frequency for a particular Robustness interval.

the *Sort Loyal* ranking function. With the *Sort Loyal* ranking function, it can be conjectured that peers which come to form cooperative relationships early on, stay committed in their relationships. This stability of relationships increases the performance of the system, because at no time do peers find themselves short of partners.

Apart from this, many such protocols often also use the *When needed* stranger policy. This policy also leads to more committed partnerships. With the *When needed* stranger policy, peers only cooperate with strangers when their set of partners is not full. Thus once its partner set is full, a peer will not cooperate with strangers. By not cooperating regularly with strangers, a peer protects itself from breaking relationships by avoiding temptation. This is because when a peer  $p$  defects against a stranger, it does not get back anything in response from the stranger. In this way peer  $p$  does not discover how much better or worse the stranger is than its current partners, and thus continues to stick with them.

It could be assumed that in the presence of churn, protocols with low number of partners will not perform so well. However, we ran Performance tests for the whole space under churn rates of 0.01 and 0.1 per round, and found that it was still the protocols that employed a low number of partners that performed the best. Thus, it can be claimed that having low number of partners is a desirable feature of high performing protocols, given that everyone in the population runs the same protocol.

## 2.1.2 Robustness

It is interesting to compare Figure 2.2 with Figure 2.3. While a low number of partners seems to be a good choice for high performance, the situation is reversed when it comes to robustness. In Figure 2.3, we observe that most of the highly robust protocols

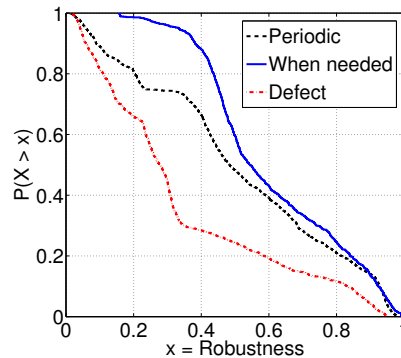


Figure 2.4: Complementary CDF plots of Robustness of different stranger policies.

have a high number of partners. This can be seen in the top right hand corner of the figure. This is intuitive, because protocols that employ a low number of partners would very likely perform worse in face of an invading protocol that employs high number of partners. This is because, in case of an invasion, peers employing a protocol that maintains a high number of partners are less likely to find themselves short of partners as compared to peers who follow a protocol that maintains a low number of partners. Hence, in the case that some partners defect, peers with a high number of partners can continue to download at high speeds while peers with a low number of partners are likely to suffer with poor speeds. To put it straightforwardly, a peer that maintains 9 partners will suffer less when one of its partners defects on it, as compared to a peer that only maintains one partner.

As we are considering the robustness of protocols, it can be seen from Figure 2.1 that there are some protocols that are highly robust with robustness values above 0.99. By analyzing these highly robust protocols, we discovered their interesting properties. Including the most robust protocol in that cluster, these protocols use a combination of the *When needed* stranger policy, the *Sort Fastest* ranking function and the *Prop Share* reciprocation function. Figure 2.4 shows that only those protocols which use the *When needed* stranger policy reach robustness levels greater than 0.99. Similarly, it can be seen from Figure 2.6 that protocols which use the *Sort Fastest* ranking function, are the most robust protocols. Figure 2.5, shows that even though the *Equal Split* resource allocation policy does quite well, it is the *Prop Share* policy that manages to get robustness values greater than 0.99.

Since this is a vital point, as it tells us about the features of protocols that are almost completely robust, we consider these properties in detail. As discussed previously, the *When needed* stranger policy only cooperates with strangers when its set of partners is not full. The *Sort Fastest* ranking function, as the name implies, ranks peers in decreas-

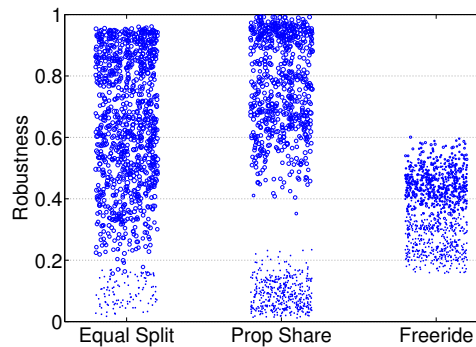


Figure 2.5: Robustness values using different resource allocation methods. Each circle is a unique protocol. Bigger circles represent better performance.

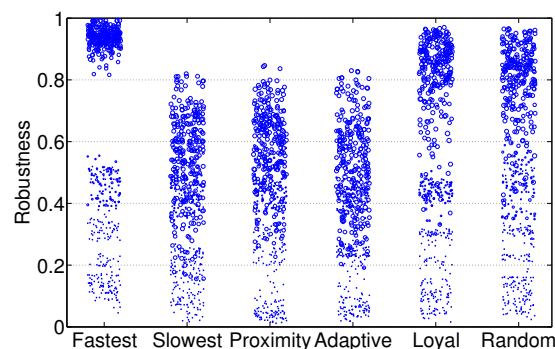


Figure 2.6: Robustness values using different ranking functions. Each circle is a unique protocol. Bigger circles represent better performance.

ing order of their speed, and finally the *Prop Share* resource allocation policy, allocates resources to peers in proportion to their contribution.

This combination, first of all, validates the claims about the robustness of the *Prop Share* mechanism [31]. However, it should be noted that, unlike their proposal, these protocols do not reciprocate with everyone. In fact, the most robust protocol maintains only 7 partners. Secondly, the combination of *When needed* with *Prop Share* is very important to note. In [31] a cryptographic bootstrapping technique was proposed to avoid exploitation by freeriding strangers. Our results suggest that the combination of the *When needed* stranger policy with the *Prop Share* resource allocation policy is a practical and lightweight alternative for designing robust protocols.

### 2.1.3 Trade-off between Performance and Robustness.

Looking at the very robust protocols, we note that most of them are not among the high performing protocols. This suggests a trade-off between performance and robustness.

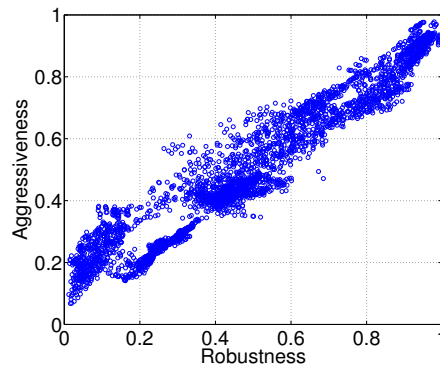


Figure 2.7: Scatter plot of robustness and aggressiveness values of the protocols. The Pearson’s correlation coefficient is 0.96.

However, looking at the top right hand corner of Figure 2.1, we can see that there are at least some protocols that are robust and also have high performance (with robustness and performance values above 0.8). On inspection we find that there are 9 such protocols and *all* of these protocols follow the *Sort Loyal* ranking function. No other dimension (such as resource allocation, stranger policy, etc.) is uniform across all 9 protocols. *Sort Loyal* cooperates with those other peers who cooperate with it for the longest durations. It could have been assumed that a ranking policy like *Sort Loyal* would not fare very high in terms of robustness. This is because of the danger that a fast peer that employs the *Sort Loyal* ranking function, could get stuck with very slow peers (who follow another protocol) that keep cooperating with it. However, it is interesting to note that the highest robustness achieved by a protocol that sorts others based on loyalty, is actually a very high 0.97.

#### 2.1.4 Aggressiveness

In Figure 2.7 we see that Robustness and Aggressiveness are linearly correlated with a Pearson’s correlation coefficient of 0.96. This suggests that robust protocols are also very aggressive and there does not seem to be a major payoff between robustness and aggressiveness. We can conclude that the results for Robustness also hold for Aggressiveness.

#### 2.1.5 Regression Analysis

We applied multiple linear regression analysis for the whole protocol design space, which is reported in Table 2.1. Values of  $h$  and  $k$  (i.e. number of strangers and partners) were treated as numerical values (in the table  $\tilde{h}$  and  $\tilde{k}$  are the standardized values of  $h$

and  $k$ , respectively), whereas the other variables are categorical, thus were substituted by dummy variables.

Table 2.1: Multiple linear regression analysis applied for the PRA measures of the whole search space. The adjusted  $R^2$  values are reported in the second row. The standard errors for all the variables are less than 0.012. Significance level is indicated as 'OK' if it was less than 0.001.

variable	Performance (adj. $R^2 = 0.68$ )			Robustness (adj. $R^2 = 0.52$ )			Aggressiveness (adj. $R^2 = 0.61$ )		
	estimate	$t$ value	sign.	estimate	$t$ value	sign.	estimate	$t$ value	sign.
(intercept)	0.661	64.372	OK	0.813	73.286	OK	0.801	96.300	OK
$\log(\tilde{k})$	-0.008	-2.487	-	0.035	10.334	OK	0.037	14.591	OK
$\log(\tilde{h})$	0.109	33.121	OK	0.115	32.287	OK	0.092	34.340	OK
B2	0.008	1.046	-	0.026	3.010	OK	0.026	4.012	OK
B3	-0.206	-26.257	OK	-0.241	-28.399	OK	-0.190	-29.778	OK
C2	-0.066	-10.567	OK	-0.045	-6.576	OK	-0.039	-7.716	OK
I2	0.023	2.151	-	-0.214	-18.186	OK	-0.212	-24.000	OK
I3	0.020	1.831	-	-0.199	-16.911	OK	-0.155	-17.503	OK
I4	0.022	1.975	-	-0.230	-19.517	OK	-0.193	-21.796	OK
I5	0.031	2.843	OK	-0.074	-6.262	OK	-0.097	-11.004	OK
I6	-0.009	-0.796	-	-0.082	-7.080	OK	-0.090	-10.259	OK
R2	-0.194	-25.260	OK	-0.093	-11.188	OK	-0.053	-8.511	OK
R3	-0.544	-70.848	OK	-0.220	-26.562	OK	-0.253	-40.697	OK

These results serve as a summary of our previous results and also examine some dimensions which were not covered in the previous section. From Table 2.1 we can conclude the following: i) Choosing *Freeride* as a resource allocation policy (R3) has the biggest negative impact on Performance and Aggressiveness and it is also has a big negative impact on Robustness. ii) Another bad choice is the *Defect* stranger policy (B3). This policy has the biggest negative effect on Robustness. On the other hand, the *When needed* policy (B2) increases Robustness and Aggressiveness, but is not statistically significant for Performance. iii) Increasing the number of strangers to cooperate with increases all three, Performance, Robustness, and Aggressiveness values and this variable has the biggest positive effect on all three measures. iv) Higher number of partners results in an increase in Robustness and Aggressiveness, but not for Performance. v) We can see that *TF2T* strategy (C2) plays a consistent negative role for all three measures. vi) The choice of the ranking function has a big effect on Robustness and Aggressiveness. This is in line with Figure 2.6. However, choice of ranking function does not have significant impact on Performance, except for the *Sort*

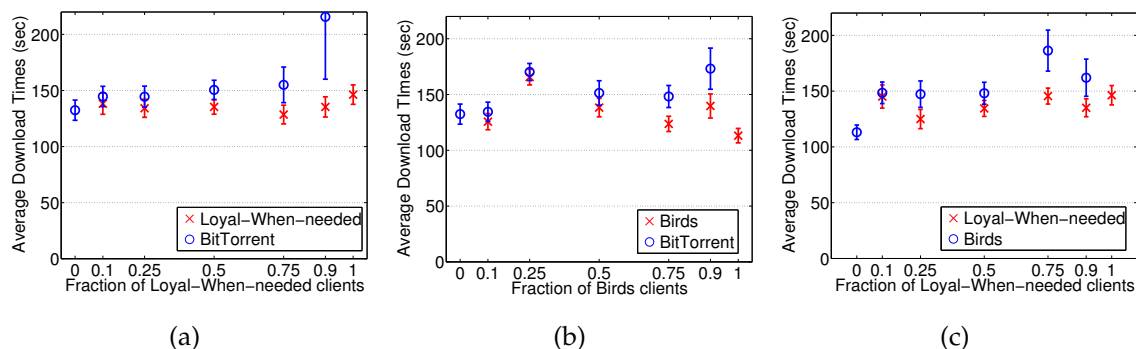


Figure 2.8: Encounters between three selected protocols.

*Loyal* ranking function (I5), which increases Performance.

### 2.1.6 Birds according to Design Space Analysis

We devised a robust variant of BitTorrent in D2.1.2, called *Birds*, using a game-theoretic analysis. Subsequently, we augmented game-theoretic analysis for protocol design with Design Space Analysis. We would now like to inspect if the results that we obtained from our game-theoretic analysis have held. How does *Birds* fare in the larger design space, under a more comprehensive and more realistic analysis?

For the Performance measure, the best *Birds* variant, i.e., a protocol that at the very least ranks other by *Proximity* and employs *Equal Split* reciprocation, does well with a Performance value of 0.83 to rank at 30 among all 3270 protocols. In Robustness, *Birds* achieves a highest value of 0.76 and ranks at 714. For Aggressiveness, *Birds* achieves a highest value of 0.74 to rank at 630 among all protocols.

### 2.1.7 Discussion

Using DSA we were able to discover protocol variants that do well by employing interesting and counter-intuitive combinations of actualized dimensions. Some combinations lead to extremely high performance, while some lead to very high robustness. We also discovered a few protocols that are both highly robust and also have high performance.

The highly robust protocols are the best candidates for usage in open distributed systems, in which protocol variants may enter; whereas the protocols that achieve very high performance are perhaps more suited to closed systems, where incentives are not required. With the regression analysis on the whole design space we were able to measure the relative impacts of the different dimensions. This also gives new insights

into practical protocol designs, and indicates the actualized dimensions that should be preferred and those that should be avoided.

Finally, we observed that Birds ranked very well in Performance and it is within the top quarter in Robustness. Given the fact that Birds was devised using a highly abstracted game-theoretic analysis, we claim to have: a) shown that a game-theoretic analysis is a useful tool which can be used to devise protocols through simple abstractions, that do reasonably well; and b) demonstrated the utility of DSA, by discovering *several* protocols that do better than Birds, in terms of Robustness and Performance.

## 2.2 Validation of DSA Results

We discovered several interesting protocols using Design Space Analysis. In this section we want to explore how well DSA can be used to design deployable protocols. In order to prove the feasibility of using DSA to produce robust and high performing protocols that can be deployed, we modified an instrumented BitTorrent client provided by [29]. From the discovered protocols we choose one that uses the *Sort Loyal* ranking function and the *When needed* stranger policy because this variant resulted in both high Performance and Robustness according to DSA. We term this protocol as ‘Loyal-When-needed’. Another variant was suggested by our Nash equilibrium analysis, which is the Birds protocol. Birds uses the *Proximity* ranking function. The third protocol type is the standard BitTorrent, which represents the baseline.

**Experimental setup.** In our experiments we pitted one protocol against another, with varying proportions of the two protocols. We adopted an experimental setup similar to [31] and [37]. The total number of leechers is 50. We setup one seeder with an upload bandwidth of 128 KBps. We setup a local tracker, with peers downloading 5MB files. We used the bandwidth distribution provided by [37]. Peers leave upon completing their download. The results of these runs can be seen in Figures 2.8 and 2.9, where each point in the figures represents the average over at least 10 runs and error bars mark 95% confidence intervals. Finally, we base our decision to use a cluster on the arguments and results in [41].

**BitTorrent vs Loyal-When-needed.** Figure 2.8(a) represents competitive encounters between BitTorrent and Loyal-When-needed clients. We can see the consistent trend of the Loyal-When-needed clients regarding the average download times: it is largely independent of the composition of the swarm. Moreover, Loyal-When-needed clients never do worse than BitTorrent, and they do significantly better if they are in a majority

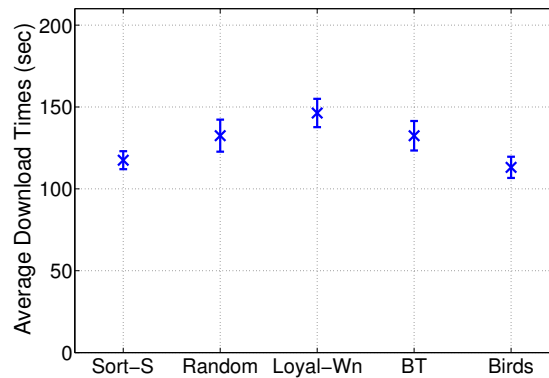


Figure 2.9: Performance of various protocols.

in the swarm.

**Birds vs BitTorrent.** Figure 2.8(b) represents a validation of our game-theoretic analysis from D2.1.2: Birds does as well or better than BitTorrent in all proportions. The difference is statistically significant if the proportion of Birds clients is more than or equal to three quarters. We can also conclude that the swarm with only Birds clients results in significantly better average download time than a swarm with only BitTorrent clients.

**Birds vs Loyal-When-needed.** Finally, in Figure 2.8(c) we compare the competitive encounters of Birds and Loyal-When-needed. We can immediately see that a swarm with only Birds clients results in better average download time. Together with the previous results, this validates our analysis using DSA, according to which Birds ranks high in performance (as discussed in Section 2.1.6). In line with our DSA results, the Loyal-When-needed protocol is more robust than Birds. The degradation in Birds performance becomes statistically significant when the two protocols compete; this is more evident when the Loyal-When-needed clients are in the majority.

**Performance of various protocols.** We now compare the performance of different protocols when all peers in the population execute the same protocol. For this, we consider two additional protocols that we discovered through DSA. Figure 2.9 shows that a protocol that we term Sort-S, together with Birds, fares the best when all peers in the swarm follow the same protocol. Sort-S is the interesting protocol that we discovered in our DSA analysis. It uses the *Sort Slowest* ranking function, defects on strangers and only has one partner. It is interesting to observe that a protocol that uses the *Sort Random* ranking function performs as well as BitTorrent. This recalls the results presented in [30]. We note that Figure 2.9 does not say anything about the Robustness of these protocols.

## Chapter 3

# The Reward and Punishment of Sharing Ratio Enforcement

BitTorrent is a popular Peer-to-Peer (P2P) protocol for file distribution. A key of its success lies in its Tit-For-Tat (TFT) incentive policy, which works reasonably well in fostering cooperation among downloading peers (also known as *leechers*). However, TFT does not provide any incentive for peers to remain in the system after the download is complete, in order to *seed* the entire file to others. Therefore, peers are free to engage in “Hit and Run” behavior, the scenario under which a peer leaves immediately upon completing a download.

In recent years, there has been a proliferation of so-called *private* BitTorrent communities aimed at incentivizing seeding. These communities employ a private tracker based method that maintains centralized accounts and records the *sharing ratio* of each peer, i.e., the ratio between its total amount of upload and download. Community administrators specify some *threshold* above which all members are required to maintain their sharing ratios. This mechanism is known as *Sharing Ratio Enforcement* (SRE). Community members whose sharing ratios drop below the threshold are warned and then banned from downloading, or even expelled from the community. In this way, it is guaranteed that each peer provides a certain level of contribution to the community.

The main motivation for implementing SRE is to close the gap between bandwidth demand and supply as observed in public communities, where there is significantly more demand than supply [34]. Thus, the basic design goal of SRE is to achieve higher system-wide downloading speeds by increasing the bandwidth supply.

Several measurement studies have shown that SRE is very effective in increasing supply [13, 32, 34, 43]. For instance, [34] reports seeder-to-leecher ratios that are at least

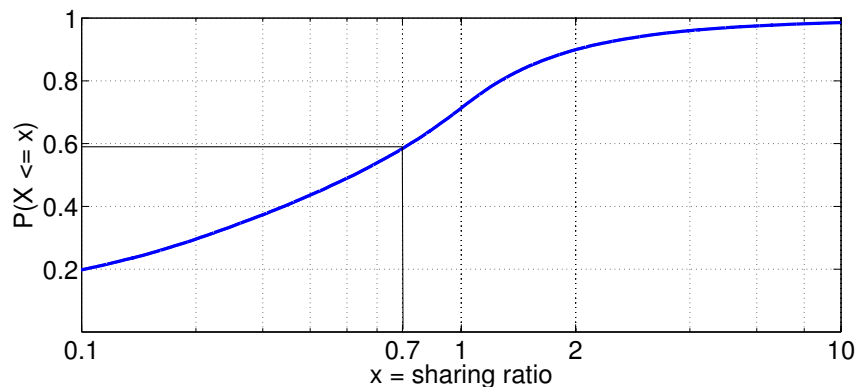


Figure 3.1: Over-seeding behavior: the CDF of the sharing ratios of peers in BitSoup.org.

9 times higher in private communities than in public ones, while downloading speeds are measured to be 3-5 times higher. Although apparently the abundant supply of bandwidth leads to high downloading speed, in this paper we argue that oversupply also has some negative effects such as excessively long seeding times that are often unproductive.

## 3.1 Support from real world observations

In order to support our model and analysis of SRE, in this section we present real world observations of a private BitTorrent community, BitSoup.org. In this community, users are required to maintain sharing ratios greater than 0.7. The measurement trace [8] reports the upload and download amount, as well as the seeding time of each user. In total, information on nearly 87,000 users and 13,000 torrents was obtained during a period of two months. For more details we refer the reader to [8].

Previous measurement studies have already shown the effectiveness of SRE in boosting the cooperation of users, in terms of high seeder-to-leecher ratios and long seeding times leading to high downloading speeds [34, 32, 43, 13]. In addition to these results, we have two further interesting observations as follows.

### 3.1.1 Existence of over-seeding behavior

Intuitively, the initial goal of users in a BitTorrent community is to download files. To achieve this, maintaining a sharing ratio close to the SRE threshold is sufficient. However, we observe that not all the users behave like this. As shown in Fig. 3.1,

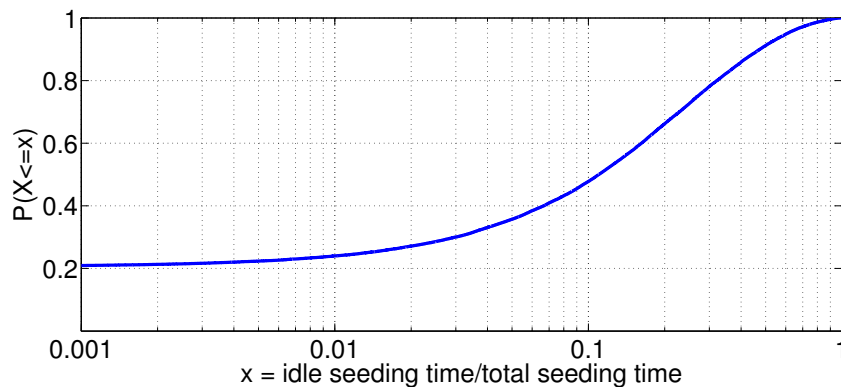


Figure 3.2: Unproductive seeding: The CDF of the fraction of idle seeding time of peers with sharing ratios smaller than 1 in BitSoup.org.

more than 40% of the users in BitSoup keep sharing ratios higher than 0.7<sup>1</sup> and more than 10% of the users keep them higher than 2. This phenomenon of peers seeding more than required and achieving sharing ratios that are (much) higher than the SRE threshold has also been observed in other communities (e.g., [32]).

From the above observation we abstract two user behaviors for our later analysis, *lazy-seeding* and *over-seeding*. Lazy-seeding peers seed the minimum amount required by SRE. They represent the users who are download-oriented, i.e., who only seed enough to maintain adequate sharing ratios to be able to start new downloads. On the other hand, over-seeding peers are deposit-oriented, and always maintain (much) higher sharing ratios than required. The behavior of such peers may be triggered by various motivations such as altruism, a desire to be part of the rich elite of the community, or a habit of storing sharing ratio for the future.

### 3.1.2 Unproductive seeding

It is clear that in order to achieve high sharing ratios, peers need to spend considerable amount of seeding time. In the case of over-seeding peers, long seeding times are to be expected. However, we observe that even many peers with considerably small sharing ratios suffer extremely long seeding times, and a significant part of their seeding time is spent idle without being able to upload anything to others. As a consequence, they have to wait for a long period until their sharing ratios are high enough to start new downloads.

<sup>1</sup>It can be observed that a significant fraction of users have sharing ratios lower than the SRE threshold. This is because in BitSoup, as well as in other private communities, users are not immediately banned after their sharing ratios drop below the threshold: they are given a certain amount of time to increase their sharing ratios.

Fig. 3.2 shows the CDF of the fraction of idle seeding time of peers with sharing ratios smaller than 1 in BitSoup. We can see that 10% of these peers spend at least half of their seeding time idle. It should be noted that Fig. 3.2 only shows the fraction of idle seeding time. It can be conjectured that the fraction of seeding time that is not completely idle yet still yields very low upload capacity utilization, would be much higher. We hypothesize that this problem of unproductive seeding is due to the oversupply under SRE.

## 3.2 Experimental setup

In our theoretical model we consider an idealized scenario of a private community where there is only one swarm, and we assume that the over-seeding peers have an infinite desired threshold for sharing ratios. Taking the insights obtained from our model, in the following sections we perform simulation-based analysis by considering more realistic scenarios. We simulate a private community that contains a number of different files (each associated to a different swarm). Each peer exhibits either one of two user behaviors: lazy-seeding or over-seeding. At any time a peer can only participate in one swarm, either as a leecher or a seeder. We do not consider parallel leechings or seedings, or a combination of these, since a peer who downloads and/or seeds  $n$  files simultaneously can be considered as being  $n$  different peers, each having  $1/n$  of the original upload capacity. In our future work, we will consider the parallel leechings and seedings in which a peer's upload/download capacity is dynamically allocated among multiple swarms.

The simulation starts with 100 initial peers. Each of them joins a random swarm and is assigned a random number between 0 and 2 as its sharing ratio, in a way that the average sharing ratio for all peers is equal to 1. A peer may start a new leeching or a new seeding process if its randomly assigned sharing ratio is above or below the threshold, which, for lazy-seeding peers is the SRE threshold, and for over-seeding peers is their desired threshold. In this way we have created a steady state for the system.

The simulation model is based on *rounds*. Each round represents a unit of time in which each peer is activated and may perform some activities, such as initiating new leeching or seeding sessions, or uploading and/or downloading data from other peers. In each round, leechers arrive according to a pre-assigned arrival rate and they join a random swarm. A new peer can start its first download freely, after which it

maintains a sharing ratio above the threshold. Each peer attempts to download all files, in random order. We run the simulation for 2000 rounds and we keep a record of those peers who finish downloading all the files by the end of the simulation. All the results represent the average of 5 runs.

We consider three performance metrics, the average downloading speed, the average upload capacity utilization, and the average seeding time. The upload capacity utilization is calculated as the ratio between the upload speed and the upload capacity of a peer. It reflects system effectiveness in utilizing the upload capacities of peers. The average seeding time is calculated separately for peers in different categories, like lazy-seeding and over-seeding peers, and high-capacity and low-capacity peers.

We always consider a bandwidth-homogeneous BitTorrent system unless otherwise indicated. The parameter settings<sup>2</sup> used in our simulation are introduced in Table 3.1.

Table 3.1: Simulation parameter settings

SRE threshold	0.7	over-seeding threshold	2
file size	10 units	no. of files (swarms)	5
piece size	1 unit	no. of initial peers	100
upload capacity	1 unit per round	simulation rounds	2000

### 3.3 The Reward and Punishment of SRE

In this section we show the performance improvement and deterioration under SRE. Based on simulations we examine the influence of several parameters and we conclude the main reasons for the reward and punishment of SRE.

#### 3.3.1 The imbalance of bandwidth supply and demand

SRE was designed to close the gap between bandwidth demand and supply as observed in public communities. However, under SRE, the presence of over-seeding peers completely reverses the situation and in private communities, swarms tend to be extremely oversupplied [34, 32, 43, 13].

<sup>2</sup>We choose 0.7 as the default value of the SRE threshold, as this value is used in many private communities [1, 2]. And we choose 2 as the default value of the desired threshold of over-seeding peers. We conjecture that for different values the tendency of this problem would be the same.

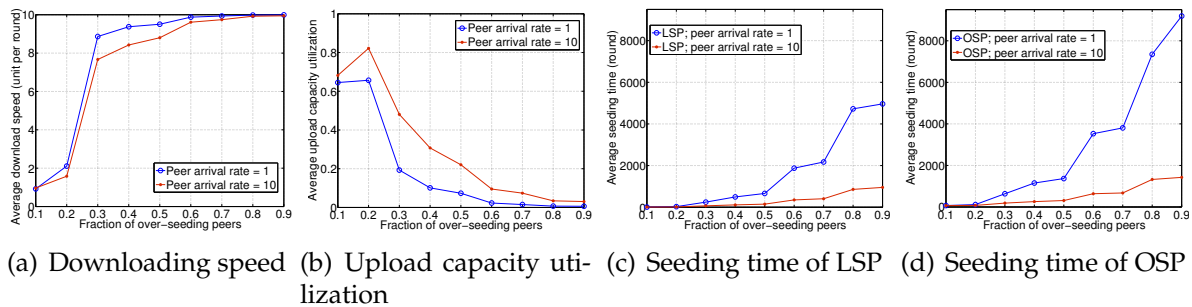


Figure 3.3: System performance under different fractions of lazy-seeding peers (LSP) and over-seeding peers (OSP), and different peer arrival rates.

In our first experiment we vary the fraction of over-seeding peers, thus generating different levels of oversupply. As shown in Fig. 3.3(a), with the fraction of over-seeding peers increasing from 0.1 to 0.9, the average downloading speed is increased nearly 10 times. However, the disadvantage of oversupply is more crucial: the average upload capacity utilization is significantly deteriorated and the seeding time is increased dramatically. With 50% over-seeding peers, on average each peer can only utilize less than 20% of its upload capacity (Fig. 3.3(b)). With this low upload capacity utilization, all peers have to stay for extremely long times (compared to their downloading times) to achieve the sharing ratio required by SRE (Figs. 3.3(c) and 3.3(d)). In our experiment with 50% over-seeding peers, the seeding time of a lazy-seeding peer is nearly 200 times more than its downloading time, and for over-seeding peers, it even increases to over 400 times.

On the other hand, with a smaller peer arrival rate (which means a smaller demand) the imbalance is even worse and so is the performance. As shown in Fig. 3.3, when the peer arrival rate decreases from 10 to 1 peer per round, with the same fraction of over-seeding peers, the average upload capacity utilization is decreased 2-3 times and the average seeding time is increased 2-5 times.

Our simulation results show that, under SRE, the existence of over-seeding peers makes the swarms oversupplied. As a consequence, with a relatively large fraction of over-seeding peers and a small peer arrival rate, peers have to seed for extremely long times, though their seeding is not very productive.

### 3.3.2 The influence of the SRE threshold

In this subsection we analyze the influence of varying the SRE threshold. Fig. 3.4 shows that, which is consistent with our intuition, when the SRE threshold is increased from 0.2 to 0.9, the upload capacity utilization is decreased while the average down-

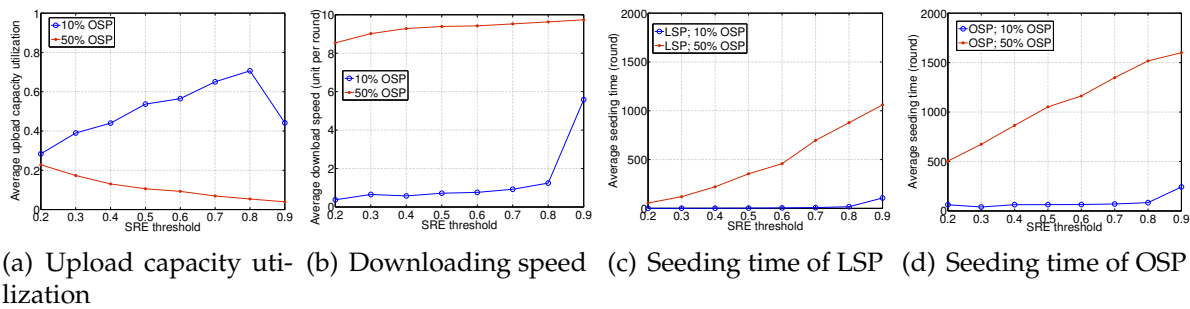


Figure 3.4: Influence of the SRE threshold under different fractions of lazy-seeding peers (LSP) and over-seeding peers (OSP).

loading speed and the average seeding time are increased. Further, the reward and punishment of SRE are limited when the fraction of over-seeding peers is small.

Surprisingly, in Fig. 3.4(a) we see that when there are 10% over-seeding peers, the upload capacity utilization is increased when the SRE threshold increases from 0.2 to 0.8, and then drops when it further increases to 0.9. We believe this is due to, what we term as, the *seeder's dilemma*: with either a very small or a very large number of seeders, peers cannot well-utilize their upload capacities. The former case is due to the piece availability problem. When there are not enough seeders, leechers have to exchange data with each other, which is not always possible since they only hold a part of the entire file. The latter case is due to the insufficient download demand. Without enough demand, even though seeders have the will, they cannot find enough leechers to upload to.

### 3.3.3 SRE's discrimination against peers with limited capacities

Until now we considered only bandwidth-homogeneous systems. However, would SRE have the same effects even in a bandwidth-heterogeneous system, and would these effects be the same on peers with different capacities? We answer these questions by extending our previous experiments to bandwidth-heterogeneous systems. More specifically, we simulate a system with two classes of peers, namely slow and fast peers. All the other settings are the same as in previous experiments, except that the upload capacity of slow peers is 1 unit per round and for fast peers it is 4 units per round.

### Discrimination exists even without over-seeding peers

We first consider a private BitTorrent system without over-seeding peers, and we change the fraction of fast peers from 0.1 to 0.9. The simulation results are shown in Fig. 3.5.

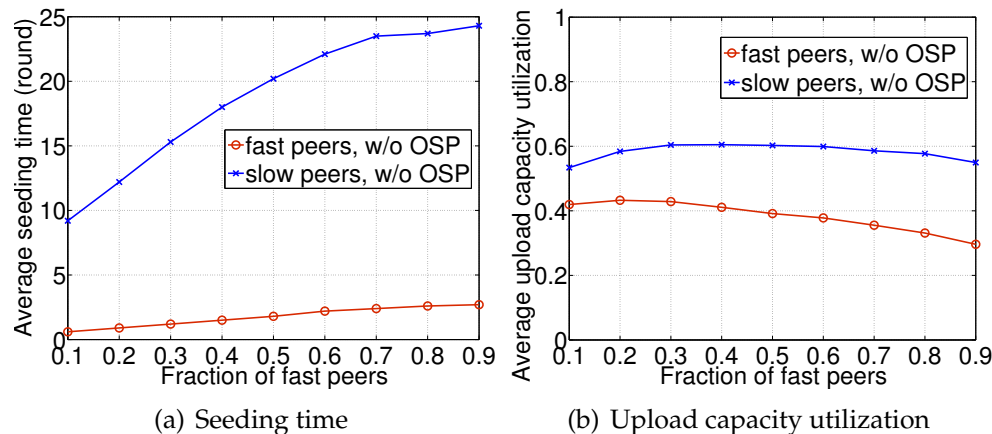


Figure 3.5: SRE's discrimination: without over-seeding peers (OSP).

We see that fast peers barely need to do any seeding work, but their existence increases the seeding times of slow peers (Fig. 3.5(a)). This result is consistent with our previous work [25] where we show that high-capacity peers manage to upload considerably more during the leeching process, and thus need to seed for shorter times. Meanwhile, consistent with our earlier model result, Fig. 3.5(b) shows that the upload capacity utilizations of both fast and slow peers do not change much with the fraction of fast peers. However, in general slow peers have better upload capacity utilizations. We believe this is due to the fact that slow peers stay as seeders longer than fast peers. Normally seeders can achieve better upload capacity utilizations, since they are not influenced by the piece availability problem.

While fast and slow peers both put all their effort in participating in the community, slow peers need to seed longer. We term this as SRE's *discrimination* against low-capacity peers. Next we show that when there are over-seeding peers, this discrimination is even more severe.

### Discrimination is more severe with over-seeding peers

Similar to the results of previous bandwidth-homogeneous experiments, Fig. 3.6 shows that the negative aspects of SRE are more severe with a higher fraction of over-seeding peers. Interestingly, with the existence of 30% over-seeding peers, fast and slow peers

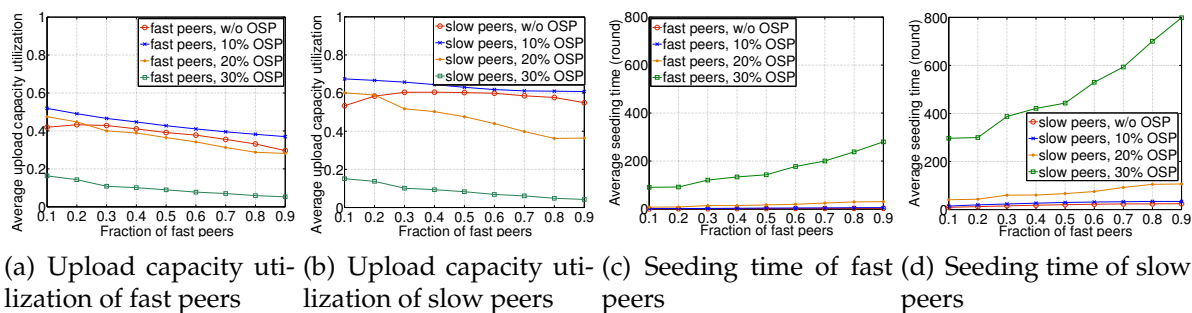


Figure 3.6: SRE’s discrimination under different fractions of over-seeding peers (OSP).

now achieve similar upload capacity utilizations (Figs. 3.6(a) and 3.6(b)). We believe this is due to the fact that both fast and slow peers need to seed. Meanwhile, as shown in Figs. 3.6(c) and 3.6(d), when the fraction of over-seeding peers is increased from 0 to 30%, slow peers need to seed 200 to 500 rounds more than fast peers, while originally they only needed to seed 20 rounds more. In general, slow peers need to seed 4 times as long as fast peers, which is the same as the ratio between the upload capacity of a fast and a slow peer.

Clearly, the long seeding time, the low upload capacity utilization, and the discrimination against low-capacity peers severely deteriorate the user experience in private communities. In the following sections, we propose several strategies to alleviate these problems.

## 3.4 Description of proposed strategies

Inspired by ideas in social sciences and economics, in this section we propose four strategies aimed at alleviating the negative effects of SRE, which require only a minor revision of the original SRE strategy.

### 3.4.1 Negative taxation

The idea of *negative taxation* is that people earning below a certain amount receive supplemental pay from the government [3]. As we have already shown, due to the keen competition in uploading, peers may seed for long times, but still achieve very low sharing ratios. We take inspiration from the concept of negative taxation and devise a new strategy in which the upload amount of a peer is calculated as its actual upload amount multiplied by coefficient  $\mathcal{T}$  defined as:

$$\mathcal{T} = \max\{\min\{1/SR, \theta\}, 1\},$$

where  $SR$  represents the sharing ratio of a peer and  $\theta > 1$  represents the *maximum negative taxation degree*.

It is easy to see that a) when  $SR \geq 1$ ,  $\mathcal{T} = 1$ , b) when  $1/\theta \leq SR < 1$ ,  $\mathcal{T} = 1/SR > 1$ , and c) when  $SR \leq 1/\theta$ ,  $\mathcal{T} = \theta > 1$ . By using this new strategy, to gain the same sharing ratio, poor peers ( $SR < 1$ ) seed less and rich peers ( $SR \geq 1$ ) seed the same amount as when using the original SRE. The maximum negative taxation degree controls the maximum negative taxation a peer can get, which alleviates the threat of free-riding.

### 3.4.2 Welfare for the rich

The term *welfare for the rich* is used to describe the bestowal of grants and tax-breaks to the wealthy [4]. Taking inspiration from this concept, we devise another strategy to alleviate the long seeding time, i.e., accelerating the seeding process of an over-seeding peer by giving welfare to it. The upload amount of a peer is calculated as its actual upload amount multiplied by coefficient  $\mathcal{W}$  defined as:

$$\mathcal{W} = \max\{\min\{SR, \varphi\}, 1\},$$

where  $\varphi > 1$  represents the *maximum welfare degree*.

By using this strategy, to gain the same sharing ratio, poor peers ( $SR < 1$ ) seed the same amount and rich peers ( $SR \geq 1$ ) seed less than when using the original SRE. The maximum welfare degree controls the maximum welfare a peer can get, to prevent the over-seeding seeders from achieving their desired sharing ratios too quickly.

In our simulation we choose  $\theta = \varphi = 2$ . We conjecture that for different values of  $\theta$  and  $\varphi$  the tendency of performance of the strategies would be the same.

### 3.4.3 Remuneration according to effort

In participatory economics, the *maxim of remuneration according to effort* has been introduced [5]. Under this scheme, people are paid according to the effort they put in rather than the amount of contribution. Taking inspiration from this concept, we propose the third strategy which takes into account the effort of users in terms of their seeding times. Previous studies have shown that the effort-based incentive policy applied in the leeching process improves the system-wide performance [40, 26]. We expect the

same improvement when this effort-based methodology is applied in a private community.

More specifically, by applying SRE with *counting seeding time*, a peer can start a new download when either it has achieved the SRE threshold or it has seeded for a sufficiently long time. In this way, peers that were stuck in long seeding process in oversupplied swarms may leave and perform further downloads. The new demand generated by these peers helps to balance the bandwidth demand and supply in the system.

Clearly, the definition of “a sufficiently long period” is quite vague. Community administrators may choose various values, like 4 hours, 10 hours, or one day. In our simulations, we simply assume that it equals the size of the shared file divided by the upload capacity of a peer. It should be noted that since over-seeding peers are deposit-oriented, they still start new downloads only when they have achieved their desired sharing ratios.

#### 3.4.4 Supply-based price

According to the law of supply and demand, if the demand remains constant and the supply increases, the price of an item decreases and vice versa. We take inspiration from this insight to devise our fourth strategy, i.e., SRE with *supply-based price*. The basic idea is that the price a downloader needs to pay for downloading one unit of data should be inversely correlated with the supply in the swarm, i.e., the higher the seeder-to-leecher ratio, the less a downloader should pay and vice versa. In this way, in an oversupplied swarm, a leecher pays less and potentially achieves a higher sharing ratio by the end of its leeching process. Hence it is less likely for it to have an insufficient sharing ratio and thus stay as a seeder, which indirectly solves the over-supply problem in this swarm. On the other hand, in an undersupplied swarm, a leecher pays more and potentially achieves a smaller sharing ratio, which makes it stay as a seeder with a higher possibility than using the original SRE. In this way, the undersupply problem is also alleviated indirectly.

We use the *seeder-to-leecher ratio* ( $SLR$ ) as a metric to decide whether a swarm is oversupplied or undersupplied. Community administrators can set different  $SLR$  values as the threshold, but we simply assume that when  $SLR \geq 1$  the swarm is oversupplied and when  $SLR < 1$  the swarm is undersupplied. The download amount of a peer is calculated as its actual download amount multiplied by coefficient  $\mathcal{P}$  defined

as:

$$\mathcal{P} = \max\{1/SLR, \phi\},$$

where  $\phi$  represents the *lowest price* for downloading one unit of data, which is used to alleviate the threat of free-riders. In our simulation we choose  $\phi = 0.1$ .

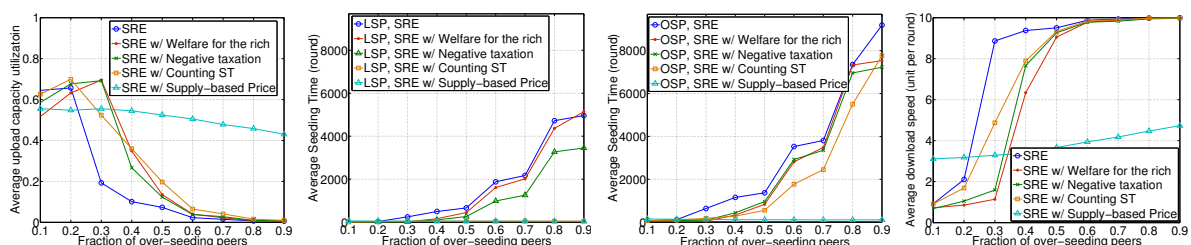
### 3.5 Strategy evaluation

In this section we evaluate the performance of the new strategies proposed in Section 3.4. The experimental setup is the same as in Section 3.3 and results are shown in Figs. 3.7 and 3.8.

#### 3.5.1 Higher upload capacity utilization and shorter seeding time

From Fig. 3.7 we see that by using any of the new strategies, peers achieve higher upload capacity utilizations, as well as smaller seeding times. As shown in Fig. 3.7(a), when there are 40% over-seeding peers the upload capacity utilization is increased 2-3 times compared to using the original SRE. While all other strategies have decreasing upload capacity utilizations with an increasing fraction of over-seeding peers, SRE with supply-based price performs stably. Given any fraction of over-seeding peers, on average peers can utilize at least 40% of their total upload capacities while for the original SRE it drops to less than 1% when there are 90% over-seeding peers.

With the improved upload capacity utilization, the average seeding time is reduced significantly. As shown in Figs. 3.7(b) and 3.7(c), when there are 60% over-seeding peers, SRE with welfare for the rich reduces at least 10% of the original seeding time for both lazy-seeding and over-seeding peers. SRE with negative taxation deals with lazy-seeding peers directly, hence it achieves an even better performance in reducing



(a) Upload capacity utilization (b) Seeding time of LSP (c) Seeding time of OSP (d) Downloading speed

Figure 3.7: Strategy performance in alleviating the eternal seeding problem: under different fractions of lazy-seeding peers (LSP) and over-seeding peers (OSP).

the seeding time of lazy-seeding peers, which is a 50% improvement compared to that achieved by SRE with welfare for the rich.

SRE with counting seeding time further relieves lazy-seeding peers from the long seeding process in a more effective manner. As shown in Fig. 3.7(b), they only need to seed for a negligible time compared to when using the original SRE, or either of the above two new strategies. Interestingly, by applying SRE with counting seeding time, the seeding time of over-seeding peers is also decreased (Fig. 3.7(c)), even though they still desire the high sharing ratios as when using the original SRE. We believe this is due to the fact that with lazy-seeding peers finishing their seedings sooner, the upload competition is reduced and over-seeding peers can achieve their desired threshold more quickly. Meanwhile, when the lazy-seeding peers are released from the seeding process, they join other swarms as new leechers, which indirectly alleviates the oversupply in those swarms.

Finally, the best performance in reducing the seeding time for all peers is achieved by SRE with supply-based price. The seeding time of both lazy-seeding and over-seeding peers is reduced by three orders of magnitude. In our view the main reason for the success of SRE with supply-based price is that it adaptively adjusts the supply and demand in a swarm. When the swarm is oversupplied, the price for downloading one unit of data is lower and peers can finish downloads at less expense, which directly reduces their consequent seeding amount and hence avoids adding more seeders in this oversupplied swarm. In this way, the imbalance of bandwidth supply and demand is mitigated, and the strategy gives a way to escape out of the seeder's dilemma as described in Section 3.3.2. A similar argument can also be applied to an undersupplied swarm.

### 3.5.2 Tradeoff: slightly decreased downloading speed

By adopting any of the new strategies, while the seeding time is dramatically reduced, as a trade-off, the average downloading speed is decreased (Fig. 3.7(d)), hence the downloading time is increased. However, given that in our simulations we consider files with size equal to 10 units, the increase of the downloading time (tens of rounds) is negligible compared to the decrease of the seeding time (hundreds or even thousands of rounds).

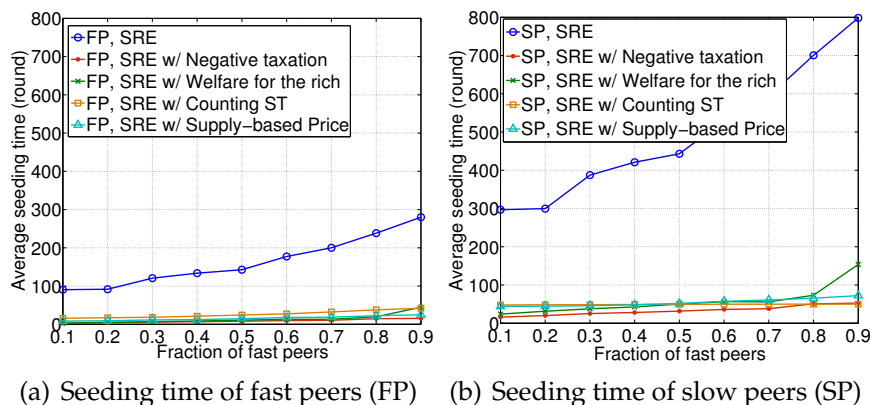


Figure 3.8: Strategy performance in alleviating discrimination: with 30% over-seeding peers (OSP).

### 3.5.3 Reduced discrimination

To examine the effectiveness of the proposed strategies in alleviating SRE's discrimination against peers with limited capacities, we repeat our experiments by further considering a bandwidth-heterogeneous system with two classes of peers, fast and slow. From Fig. 3.8 we see that *all* the proposed strategies effectively alleviate SRE's discrimination against low-capacity peers. With 30% over-seeding peers, originally slow peers need to seed 200-500 rounds more than fast peers do. By applying any of the new strategies, this difference is reduced to within tens of rounds.

## 3.6 Discussion: The change of user behavior?

It could be argued that the new strategies proposed in this paper may trigger a change in user behavior. Specifically, users from the two classes that we defined, lazy-seeding and over-seeding, might be incentivized to switch their classes under the new strategies. It could be conjectured that as a consequence, the system performance might be adversely affected.

However, we note that there is only a very small fraction of strategic users in BitTorrent communities [7]. So the likelihood of peers changing behavior is quite small. Especially regarding over-seeding peers, we argue that in general the possibility of such peers changing to lazy-seeding peers is quite low. This is because we conjecture that the behavior of over-seeding peers is motivated by either one or a combination of the following three reasons: a) Over-seeding peers always want to be relatively more well-off in terms of sharing ratio as compared to average users so that they can be

among the “rich elite” of the community and gain some potential benefits<sup>3</sup>; b) They are altruists who want to help the community as much as they possibly can; and c) They are hoarders who desire to conserve sharing ratio for “rainy days” i.e., those time periods when they feel they might engage in heavy downloading activity and might as a result be expelled from the community due to low sharing ratios.

Nevertheless, in this section we would like to analyze what happens if users do change their behavior. We consider each proposed strategy in turn and discuss the possible effects of the change in user behavior.

We emphasize that in this section we do not model or formally analyze the possible user behavioral change. The reader is referred to the deliverable D1.4.1 (Modelling the dynamics of quality collectives) for the simulation based experiments on sharing behavior of BitTorrent users.

### **SRE with negative taxation and welfare for the rich**

Under SRE with negative taxation, peers with lower sharing ratios gain sharing ratio more easily. Hence, lazy-seeding peers have no incentive to change their behaviors, while over-seeding peers may change to lazy-seeding peers. Similarly, when SRE with welfare for the rich is applied, strategic lazy-seeding peers could become over-seeding peers, while over-seeding peers will not change their behaviors.

The worst case scenario of applying either of these two new strategies is that all peers exhibit the same behavior, i.e., either all peers become lazy-seeding or all become over-seeding. In the former case every member of the community would still be forced to maintain a minimum sharing ratio required by SRE, i.e., every peer would continue to provide a certain level of contribution. This outcome would still be better than the situation in a public community where every peer has the option to leave immediately after downloading. In the latter case, i.e., when all peers are over-seeding, it is less likely for them to complain, since an over-seeding behavior, i.e., a desire for higher sharing ratios, automatically implies a long seeding time.

### **SRE with counting seeding time**

Under SRE with counting seeding time, users may set their upload speeds to zero and pretend to be seeding. However, according to the TFT policy in BitTorrent, in an undersupplied swarm the upload speed of a peer directly influences its downloading

---

<sup>3</sup>Such as priority in downloading popular files, the possibility to send invitations to others, etc. See, e.g., [1, 2].

speed. If a user sets its upload speed to zero, it would hardly be able to download at reasonable speeds in undersupplied swarms, which are normally swarms providing new and popular content.

Further, private community administrators could set another SRE threshold, which is smaller than the original one, and stipulate that users who cannot achieve the original SRE threshold must seed for a predefined time, as well as achieve the smaller SRE threshold. In this case, the potential threat of free-riders is alleviated.

### **SRE with supply-based price**

By adopting SRE with swarm-based price, the only advantage that users could gain is that they may opt to download files which have a lower price. However, this is unlikely to happen, since the choice of file to download mainly depends on user's interest in the content, rather than the price. Even if some users might be tempted to download a file simply based on its price, this would have little influence on the performance of SRE with swarm-based price, because this strategy is self-organizing and will adjust the balance of supply and demand automatically.

## Chapter 4

# Reducing History in Reputation Systems

A family of decentralized reputation systems useful in many Internet applications consists of interaction-based systems (also called content-driven systems [15]). These systems are based on algorithms analyzing all interactions among users and computing the reputations without using any explicit feedback from users, such as PageRank [36] for ranking web pages and Bartercast [35] for computing reputations of users in P2P systems. In interaction-based systems, the amount of historical information on the interactions maintained by each node affects the performance and the characteristics of the reputation mechanism. Networks such as popular online markets and social networks consist of hundreds of thousands or even millions of active users and thus, using the complete history for computing the reputation of nodes is prohibitive due to its resource requirements. Particularly in decentralized systems, such as file-sharing P2P systems, the available resources at nodes are limited and thus, only scalable solutions can be applied. Furthermore, a long-term history allows previously well-behaved nodes to exploit their good reputations by acting maliciously [33, 42, 16]. In this paper, we propose a scheme for reducing the amount of history maintained in decentralized interaction-based reputation systems. We experimentally explore its effect on the computed reputations using synthetic and real-world graphs.

In order to reduce the history of interactions, we use only a subset of the complete history to approximate reputations. We model the interactions of the *complete history* of a network as a growing graph with the nodes of the network as its vertices and the interactions between pairs of nodes as its edges, and the corresponding *reduced history* as a subgraph of the complete history. The reduced history is derived from

the complete history by deleting the least important edges and nodes. We define the importance of a node according to its age, its activity level, its reputation, and its position in the graph, while the importance of an edge is defined according to its age, its weight, and its position in the graph. Then we evaluate our approach using synthetic random and scale-free graphs, and two real-world graphs, one derived from the Bartercast reputation system of our BitTorrent-based P2P client QMedia and the other from the author-to-author Citation network of Physical Review E journal<sup>1</sup>. The main difference between the Bartercast and the Citation graphs, besides their structural properties, is that the former is derived from a deployed distributed system with personalized reputations while the latter is derived from a centralized system with global reputations.

On these networks, we apply two different computations for reputations, one based on the max-flow algorithm [14] and the other based on eigenvector centrality [12]. We evaluate our approach according to the following two observations: (i) for the vast majority of reputation systems, the rank of reputations is more important than the actual reputation values themselves; and (ii) in most cases the identification of the highest ranked nodes is enough. We demonstrate that the performance of the reduced history depends on the topology of the complete history. Furthermore, we show that the performance of the reduced history depends on the reputation algorithm. Finally, we conclude that reduced history can be applied in a large range of networks.

## 4.1 Motivation and Problem Statement

Our main motivation for reducing the history of interactions in a network is the computational cost and the storage requirements of decentralized reputation algorithms. Reputation systems, such as those of eBay or Google, cover hundreds of thousands of active nodes while reputation algorithms (e.g., Eigentrust [27], PageRank [36] and max-flow based ones [14]) have a high computational complexity.

In decentralized systems, like BarterCast, where each node stores and analyzes data locally using, e.g., the max-flow algorithm (with complexity  $O(nm^2)$  where  $n$  is the number of nodes and  $m$  and the number of edges), even much smaller graphs of  $10^6$  nodes make the computation of reputations prohibitive. Taking into account that the contributions of nodes in the computation of reputations are not equal in quality and quantity [15], we aim to delete the least important contributions and compute

---

<sup>1</sup>Data available to us after request to American Physical Society

reputations using only a subset of the complete history. In this way, we can reduce the computational cost significantly without decreasing the accuracy very much.

In addition to the computational cost, the dynamic behavior of many reputation systems makes the use of the complete history ineffective. In systems with a high population turnover such as P2P networks, only a few nodes remain for a long period in the system while the majority of nodes enters the system performing some interactions and then leaves it. Also a node behaving properly for a long time can build a good reputation and become a traitor [33] by exploiting other nodes. Preserving only short-term history forces all nodes in the system to behave continuously according to the protocol. For these reasons, several widely used reputation mechanisms, such as those of eBay and eLance, allow the use of historical information of a 1 or 6-month window. Although using a time window is useful for such feedback-based reputation systems, it is not effective in interaction-based reputation systems since important information of highly reputed nodes is deleted.

We model the interactions of a network as a directed weighted graph  $G = (V, E)$ , where the vertices  $V$  represent the nodes and the edges  $E$  the interactions among the nodes. The weight of an edge represents its importance; for instance, in Bartercast, the weight of an edge between nodes represents the amount of data transferred in the corresponding direction, and in a citation graph, it represents the number of references to an author by another. The graph is dynamically growing over time and allows not only new nodes to join but also existing nodes to create new edges. The graph  $G$  represents the *complete history* (CH) of interactions in the network.

Given the growing graph  $G$ , our target is to create a subgraph of  $G$ , denoted by  $G'$ , which preserves the highest ranked nodes in  $G$  and keeps the ranking of the reputations similar to that in  $G$ . The subgraph  $G'$  has to be dynamically maintained as the complete history grows while its size has to be almost fixed. The graph  $G'$  will be used for the computation of reputations, and represents the *reduced history* (RH) of interactions in the network.

## 4.2 Creating the Reduced History

The basic idea of creating the reduced history  $G'$  consists of removing the least important elements, either nodes or edges, from  $G$ . We use a node removal process in conjunction with edge removal. The ratio of removed nodes versus removed edges depends on the dynamics of the network. Nevertheless, edge removal implies node

removal and vice versa. More precisely, edge removal can lead to disconnecting a node from the graph and node removal results in deleting the adjacent edges of the removed node.

The parameters for **removal of a node** consist of its age, its activity level, its reputation, and its position in the graph.

The *age of node  $i$*  is expressed as  $\tau_i = t - t_i$  where  $t$  is the current time and  $t_i$  is the time instance node  $i$  joined the system. In most networks, the age of a node  $i$  affects its behavior in a non-linear way (e.g. [6, 21]). Thus, instead of its age, we consider its aging factor  $f(\tau_i)$ , where  $f$  is a decreasing function with  $f(0) = 1$  (e.g.,  $f(\tau) = e^{-b\tau}$ , where  $\tau$  represents the age of a node and  $b$  is a constant). Keeping fresh information allows the reputations system to capture the dynamic behavior of nodes.

The *activity level  $d_i$  of a node  $i$*  represents its degree. Nodes with a high activity level participated in many interactions, and so, they provide much information.

The *reputation of node  $i$*  is denoted by  $r_i$ . Our aim is to preserve the information of nodes with high reputations, since these nodes are the most reliable in the network. Moreover, allowing nodes with high reputations to contribute to the computation of reputations longer is a kind of rewarding the most trusted nodes.

For node  $i$  the *importance of its position in the graph* is expressed by its betweenness centrality (BC), denoted by  $C_B(i)$ , which measures the sum of the fractions of the numbers of shortest paths among all pairs of vertices that pass through node  $i$  [17]. Removing nodes from the graph can result in destroying its structure by creating many disconnected components and thus, we need to maintain the nodes that keep the graph connected.

The first three factors represent the behavior of node  $i$  while the fourth factor is added for preserving the structure of the graph during the deletion process. Therefore, in our method, the *priority score  $P_n(i)$*  of deleting node  $i$  is defined as

$$P_n(i) = \alpha P_A(d_i, r_i, \tau_i) + (1 - \alpha) P_B(C_B(i)), \quad (4.1)$$

where  $P_A(d_i, r_i, \tau_i)$  expresses the priority score of deleting node  $i$  based on its activity level, aging factor and reputation, and  $P_B(C_B(i))$  represents the priority score of deleting node  $i$  according to its position in the graph. The parameter  $\alpha$  takes values in  $[0, 1]$  and can be chosen according to the graph properties. We define the priority score  $P_A$  as

$$P_A(d_i, r_i, \tau_i) = \frac{n - d_i r_i f(\tau_i)}{n^2 - \sum_j d_j r_j f(\tau_j)}, \quad (4.2)$$

where  $n$  is the number of nodes in the graph, and the denominator acts as a normalization so that the sum of the priority scores sum to 1. Clearly, a node with a higher age, a lower activity level, or a lower reputation will be removed. Although the maximum value of  $d_i r_i f(\tau_i)$  is equal to  $n - 1$  (corresponding to  $d_i = n - 1, r_i = 1$  and  $f(\tau_i) = 1$ ), for simplicity, we approximate it to  $n$ . Similarly,  $P_B$  is expressed as  $P_B(C_B(i)) = (n^2 - C_B(i)) / (n^3 - \sum_j C_B(j))$ . Again, even though the maximum value of  $C_B(i)$  is equal to  $(n - 1)(n - 2)$ , we approximate it by  $n^2$ . When considering a single parameter for node removal, Eq.4.2 can be adapted in a straightforward way (similarly as  $P_B$  for parameter  $C_B(i)$ ).

The **removal of an edge** is determined by its age, its weight, and its position in the graph.

The *age of edge*  $e_{ij}$  connecting nodes  $i$  and  $j$ , is defined similarly to the age of a node, and is denoted by  $\tau_{ij} = t - t_{ij}$ , where  $t$  is the current time and  $t_{ij}$  is the time of its creation. The aging factor of edge  $e_{ij}$  is a decaying function  $f(\tau_{ij})$  and can be, e.g., an exponential function.

The *weight of edge*  $e_{ij}$ , denoted by  $w_{ij}$ , is one of the parameters for edge removal, since interactions with a high cost are more important for the computation of reputations, edges with high weights have to be preserved in the graph.

The *importance of the position of edge*  $e_{ij}$  in the graph is expressed by its edge betweenness centrality (BC), denoted by  $C_E(e_{ij})$ , which is defined as the sum of the ratios of shortest paths between all pairs of nodes containing this edge [17]. The aging factor and the weight of an edge represent its contribution to the computation of reputations, while its  $C_E$  helps in preserving the structure of the graph.

Similarly to node removal, we express the priority score of removing an edge  $e_{ij}$  as

$$P_e(e_{ij}) = \alpha P_S(w_{ij}, \tau_{ij}) + (1 - \alpha) P_F(C_E(e_{ij})), \quad (4.3)$$

where  $\alpha$  is the parameter used in the definition of  $P_n$  to control the topology of the derived graph. The scores  $P_S$  and  $P_F$  are defined similarly to  $P_A$  and  $P_B$ , respectively. Therefore, edges with lower age, lower weight, and lower betweenness centrality will be removed.

The basic computational components of reducing the history consist in the computation of BC (we do not distinguish between node and edge BC because the algorithm is the same). Computing the degree, the aging factor of nodes, the weight, and the aging factor of edges has a linear cost on the number of nodes and edges respectively and can be computed incrementally. However, the computational cost of BC is high

(for unweighted networks it is  $O(mn)$  where  $n$  is the number of nodes in the network and  $m$  the number of edges). The cost can be significantly reduced by using approximations [18] and exploiting the structure of the network. In particular in scale-free networks, the BC values do not have to be updated very often with the network growth [19] and in networks without community structure, the BC of a node shows a strong correlation with its degree. Note that the reputations of nodes are computed by the core reputation mechanism.

### 4.3 Datasets

In order to assess our method for creating the complete history, we consider both synthetic graphs and graphs derived from real networks. In our synthetic complete history graphs we consider two processes occur simultaneously: first, new nodes enter the system, and secondly, the already existing nodes interact, thus creating new links. Thus, we define the probability  $p_c$  which represents the probability of adding a new node at each time step to the graph, and the probability  $1 - p_c$  which represents the probability of adding new links between existing nodes. In highly dynamic systems, the appearance of new nodes is dominant, and so the value of  $p_c$  is high. In our models for synthetic graphs, we allow the occurrence of multiple edges between a pair of nodes and we consider the number of multiple edges as the weight of that edge.

For our experiments, we create the complete history  $G$  and the corresponding reduced history  $G'$  in parallel. In the complete history, we store all the new information. For the construction of the reduced history we keep its size (almost) constant to a maximum number of nodes  $n_{max}$ , which represents the computational or memory limitation of the system. We control the size of the reduced history by removing nodes or edges from the graph as new information is stored as described in the previous section. Below, we describe in detail our models for random graphs and scale-free graphs, the properties of the Bartercast and Citation graph, and the construction of the corresponding reduced histories.

A **random graph**, denoted by  $R(n, p_r)$ , is composed of  $n$  nodes, and each potential edge connecting two nodes occurs independently with probability  $p_r$ . Based on this model, we generate a growing directed random graph  $R(n_t, p_r)$  representing the complete history of interactions.

To create the graph  $R(n_t, p_r)$  with  $n_t$  nodes at time  $t$ , starting from a single node, we perform the following two operations at each time step:

- With probability  $p_c$  we add a new node with each of its potential directed edges existing with probability  $p$ , for some value of  $p$ .
- With probability  $1 - p_c$  we add  $pn_t$  new directed edges adjacent to chosen existing nodes uniformly at random.

In accordance with  $R$ , we create the reduced history graph  $R'$ . The reduced history  $R'$  is equal to  $R$  up to the maximum number of nodes  $n_{max}$ . After having reached  $n_{max}$  nodes,  $R'$  is maintained by performing the following operations at each time step:

- When a new node is added to  $R$ , we also add this node to  $R'$  along with its edges, and then we remove one node together with its edges with the highest priority score (Eq. (4.1)).
- When new edges are added to  $R$ , we add the same edges to  $R'$ . Then we remove from  $R'$  the same number of edges with the highest priority score (Eq. (4.3)).

Note that some edges in  $R$  may be adjacent to nodes that have been removed from  $R'$ ; in this case, these edges are not added to  $R'$ .

**Scale-free graphs** are characterized by their degree distribution following a power law. We create a growing directed scale-free graph based on the preferential attachment model [11]. Similarly to the procedure for random graphs, we generate two directed graphs  $S$  and  $S'$  corresponding to the complete history and the reduced-history.

We create  $S(n_t)$  by starting with a small seeding graph with  $m_0$  nodes connected by  $m_0 - 1$  edges and then performing the following steps:

- With probability  $p_c$  we add a new node with  $m$  directed edges, with  $m \leq m_0$ . Each edge is adjacent to an already existing node  $i$  with probability  $\Pi(i) = d_i / \sum_j d_j$ , where  $d_i$  is the degree of node  $i$ .
- With probability  $1 - p_c$  we add  $m$  new directed edges. Each of these edges are adjacent to an existent node  $i$  with probability  $\Pi(i)$ .

In line with  $S$ , we build the reduced history  $S'$  using the same procedure as for random graphs.

The **Bartercast graph** is derived from Bartercast [35], the distributed reputation mechanism used in our BitTorrent-based client QMedia<sup>2</sup>.

In Bartercast, when a peer exchanges content with another peer, they both store a *record* with the amount of data transferred and the identity of the corresponding

---

<sup>2</sup>This dataset is also part of the QLectives Living Lab Archive, as an updated version of the dataset reported in Deliverable D3.1.1.

Table 4.1: The average path length ( $L$ ) and the clustering coefficient ( $cc$ ) of the largest connected component of the Bartercast and Citation graph, and of the corresponding random graphs with similar average path length.

Graph	# Nodes	# Edges	$L$	$cc$	$L_{rand}$	$cc_{rand}$
Bartercast	10,634	31,624	2.64	0.00074	2.63	0.0032
Citation	15,360	365,319	3.29	0.1098	3.31	0.0012

peer. Regularly, peers contact another peer to exchange records using a gossip-like protocol. From the records it receives, every peer  $i$  dynamically creates a weighted, directed *subjective graph*, the nodes of which represent the peers about whose activity  $i$  has heard through Bartercast records, and the edges of which represent the total amounts of data transferred between two nodes in the corresponding directions.

We have crawled the QMedia system from September 1, 2010 to January 31, 2011, collecting information from 29,716 nodes. In our experimental analysis, we will assume *full-gossip* in which peers forward the records they receive from other peers, and so all peers eventually receive all the propagated records. Thus, the graph derived from Bartercast, denoted by  $B$ , can be considered as the subjective graph of all nodes which corresponds to the complete history. The graph  $B$  is not connected and so, we proceed in the analysis using its largest weakly connected component. Bartercast presents high population turnover and thus, the derived graph consists in a dense core with very few long living and active nodes and a periphery with many loosely connected nodes of low activity (small average path length and small clustering coefficient, see Table 4.1). The addition of new nodes/edges in  $B$  is based on the actual timestamps of the crawled database of Bartercast. Similarly to the procedure for random and scale-free graphs, we maintain the reduced history  $B'$  by removing nodes and edges using Eqs. (4.1) and (4.3) as new nodes and edges are added according to the timestamps.

The author-to-author **Citation graph**, denoted by  $C$ , is derived from the citation network of 21,858 papers published in Physical Review E from January 2001 to November 2011. Its vertices represent the authors of papers and edges represent the citation relationship between two authors (or coauthors). The weight of an edge indicates multiple citations from one author to another. Unlike Bartercast, the graph  $C$  is derived from a centralized system with global reputations. In Table 4.1, we can see that graph  $C$  exhibits small-world behavior with small average path length and large clustering coefficient. Its degree distribution has a power-law tail with exponent  $\gamma = 2.55$ . As described for the Bartercast graph, we create the complete history  $C$  and the corre-

sponding reduced history  $C'$  based on the actual timestamps in the database of the Citation graph.

## 4.4 Computation of Reputations and Evaluation Metrics

We consider two methods for computing reputations: the max-flow algorithm and the eigenvector centrality. However, our approach can be generalized to other methods for computing reputations as well.

The **max-flow algorithm** [14] computes the maximum flow passing between two nodes and is the core of many reputation systems (such as Bazaar [39], Bartercast [35], and the system proposed by Feldman et al. [16]) because it provides resilience to misreporting by nodes who may exaggerate their contributions to increase their reputations. In our study, we use the definition of reputation of Bartercast mechanism [19] since we use a graph derived from it for the evaluation of our approach. The reputation of a node  $j$  is computed as  $\arctan(f_{ji} - f_{ij})/(\pi/2)$ , where node  $i$  represents the node with the maximum betweenness centrality,  $f_{ji}$  represents the maximum flow from node  $j$  to node  $i$  in the network and  $f_{ij}$  is the maximum flow in the reverse direction. The function  $\arctan$  in the computation of reputations emphasizes the differences of flows close to 0 (neutral reputation), so that newcomers with only a small contribution can achieve a good reputation value and participate in the system. Every reputation value is normalized with the factor  $\pi/2$  so that it is in  $(-1, 1)$ .

**Eigenvector centrality** is a well-studied metric for the importance of a node in a network and its variants constitute the core of many reputation and recommendation mechanisms (such as EigenTrust [27], PageRank [36], TrustRank [20], FreeRank (see Deliverable D1.3.2), and many others). The basic idea of eigenvector centrality is that interactions with highly reputed nodes contribute more to the reputation of a node. In our analysis, we use PageRank computed using the power iteration:  $r_{t+1} = dAr_t + [(1-d)/N]\mathbf{1}$ , where  $A$  represents the normalized adjacency matrix of the network,  $r_t$  the ranking vector at time step  $t$ ,  $d$  the damping factor (we set it equal to its typical value 0.85 [36]),  $N$  the number of nodes, and  $\mathbf{1}$  the vector of length  $N$  containing only ones. In some networks like Bartercast, an incoming edge of a node has a negative meaning for the reputation of that node (because a weighted edge represents the amount of transferred data and so, adds to the reputation of the donator of the data). Therefore, in these networks, first we reverse the direction of links before we apply PageRank (reverse PageRank [10]).

The evaluation of our method is based on the observations that for the vast major-

ity of reputation systems, the ranking of nodes according to their reputations is more important than the actual reputation values themselves, and that in many systems the identification of the highest ranked nodes is more important than of the rest of the nodes. Therefore, we define the *ranking error* as the difference between the rankings of the nodes according to their reputations in the reduced history and the complete history. More precisely, we consider the sequences of the Unique Identifiers (UIDs) of the nodes in the reduced and the corresponding complete history of our graphs, and we compute the minimum number of inversions of consecutive elements needed in the sequence of the reduced history to get all the common nodes in their correct order in the complete history. This minimum number of inversions is then normalized over the worst case, which would occur if the ranking would be completely reversed. Furthermore, to explore the ability of the reduced history to identify the highest ranked nodes, we define a second metric called the *ranking overlap* which is defined as the fraction of nodes the sequences of the top-5%, 10% and 20% ranked nodes in the reduced history and the corresponding sequences in the complete history have in common. More precisely, we compute the ranking overlap as  $|\mathcal{U} \cap \mathcal{V}|/|\mathcal{U}|$ , where  $\mathcal{U}$  is the set of the top-5%, 10% and 20% ranked nodes in the reduced history and  $\mathcal{V}$  is the set of the top ranked nodes in the complete history of size  $|\mathcal{V}| = |\mathcal{U}|$ .

## 4.5 Evaluation

In this section, we present our experimental evaluation. First, we explore the effect of each of the parameters for node and edge removal separately and in combination. Next, we study the effect of the size of the reduced history relative to the size of the complete history. Finally, we evaluate the effect of the growth of the complete history while the size of the reduced history is constant. In our experiments, we use the synthetic and real-world graphs introduced in Section 4.3. Our synthetic graphs consist of 5,000 nodes with  $\alpha$  and  $p_c$  neutral (both equal to 0.5), unless other initializations are mentioned. We choose the other parameters for the random graph ( $p_r = 0.02$ ) and the scale-free graph ( $m = 3$  and  $\gamma = 2.2$ ) so that they roughly correspond to the Bartercast graph. For the synthetic graphs, our results presented in each plot are the average of 25 independent experiments, while for the Bartercast and Citation graphs, we conduct only one experiment since we have only one instance of these real-world graphs.

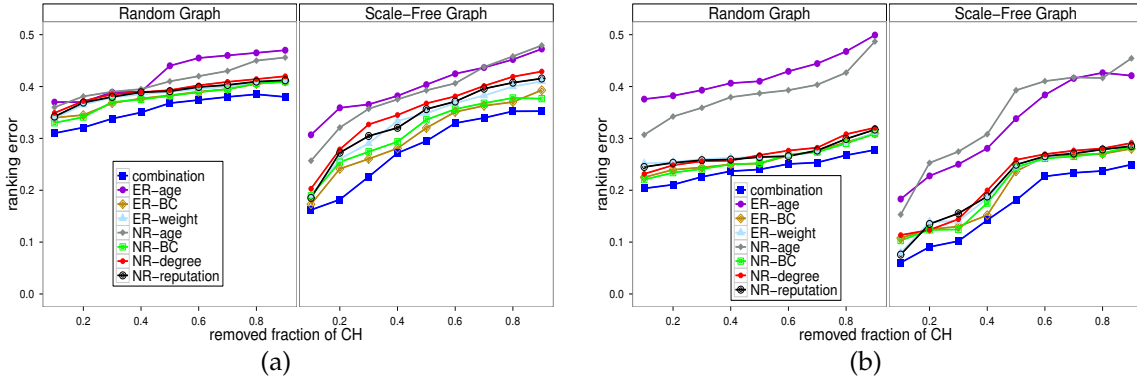


Figure 4.1: The effect of the parameters for node and edge removal when removing a fraction of the nodes and edges of CH for random and scale-free graphs when the reputation algorithm is max-flow (left) and Pagerank (right). The indication ER in the legend denotes parameters for edge removal and NR parameters for node removal.

### 4.5.1 Experiments and Results

We first explore the effect of the parameters for node and edge removal defined in Section 4.2 on the ranking error. To explore the effect of the parameter  $\alpha$ , we remove 50% of the nodes and edges of the complete history according to Eqs. (4.1) and (4.3) for different values of  $\alpha$ . We find that  $\alpha$  does not affect the performance of the reduced history much. In particular, for random graphs using max-flow (or Pagerank), the ranking error starts at 0.33 (or 0.21) for  $\alpha$  equal to 0, and it slightly decreases by 0.02 (or 0.01) until  $\alpha$  is equal to 0.8. As  $\alpha$  increases further, the ranking error increases by 0.07 (or 0.06). A similar stable behavior for the ranking error is observed for the scale-free and real-world graphs. For scale-free graphs, using max-flow (Pagerank), the ranking error starts from 0.28 (0.19) and it slightly decreases by 0.01 with the increase of  $\alpha$ , until  $\alpha$  is equal to 0.8, then it increases by 0.02. For Bartercast graph, using max-flow (Pagerank), the ranking error starts from 0.26 (0.25) and it slightly decreases by 0.02 with the increase of  $\alpha$ , until  $\alpha$  is equal to 0.8, then it increases by 0.02. For Citation graph, using max-flow (Pagerank), the ranking error starts from 0.25 (0.17) and it slightly decreases by 0.02 with the increase of  $\alpha$ , until  $\alpha$  is equal to 0.8, then it increases by 0.04. Since  $\alpha$  does not affect the performance of the reduced history much we take it as neutral, equal to 0.5, for all the following experiments.

Next, we explore the effect of the parameters for node and edge removal separately, and their combination as defined by Eqs. (4.1) and (4.3). For the parameters for node or edge removal, we remove fractions nodes or edges of the complete history using only one parameter at a time. The effect of these parameters on the ranking

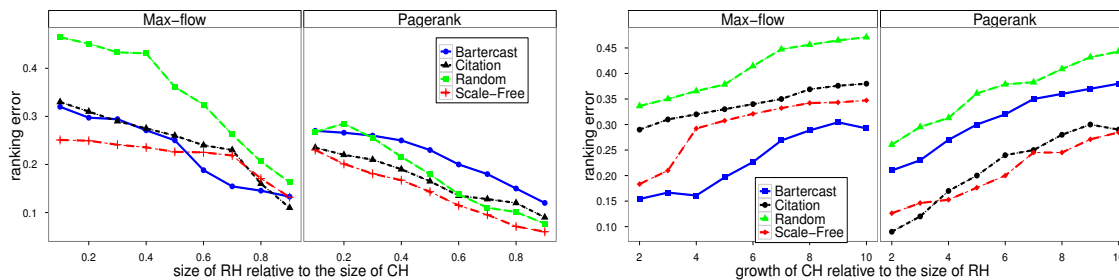


Figure 4.2: The effect of the size of RH relative to the size of CH (left) and the effect of the growth of CH relative to the size of RH (right).

error is plotted in Fig. 4.1 for the random and scale-free networks. We observe that creating the reduced history using only node removal results in similar performance as edge removal for the corresponding parameters. This is to be expected as there is a correlation between these parameters: in general, an edge with high BC is adjacent to nodes with high BC, an old edge is attached to old nodes, and an edge with a large weight is adjacent to a node with high reputation. Furthermore, the combination of all parameters in Eqs. (4.1) and (4.3) results in the smallest ranking error. The largest ranking error occurs when we remove nodes based on their age. The reputation of a node depends on the period it participates in the system and thus, when only new nodes with low reputations participate in the reduced history, the ranking error is high. All the other parameters cause quite similar ranking errors because they exhibit correlations in graphs without strong community structure, such as the random and scale-free graphs. In the real-world graphs, the parameters for node and edge removal and their combination exhibit similar relative performance as in the scale-free graphs. Since the combination of the parameters for node and edge removal achieves the lowest ranking error, we use it to create the reduced history for all the following experiments.

We next evaluate the effect of the size of the reduced history relative to the size of the complete history on the ranking error and the ranking overlap. For this purpose, we construct reduced histories of different sizes for a complete history of fixed size as described in Section 4.3. Fig. 4.2 (left) plots the ranking error for different relative sizes of the reduced history. We observe that when using max-flow, the scale-free, Bartercast and Citation graphs exhibit much smaller ranking error than the random graphs. For all the graphs using Pagerank, the reduced history exhibits smaller ranking error than using max-flow. Fig. 4.3 plots the ranking overlap for different relative sizes of the reduced history. The scale-free and Bartercast graphs exhibit much higher ranking

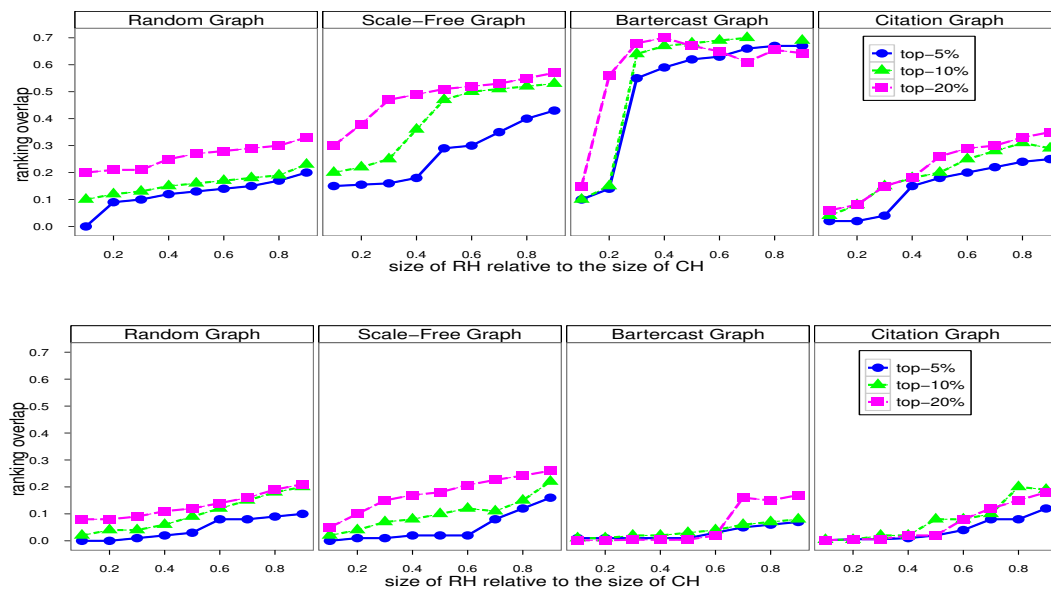


Figure 4.3: The effect of the size of RH relative to the size of CH for max-flow (top) and Pagerank (bottom).

overlap than the random and Citation graphs when using the max-flow based algorithm. Particularly, in these networks the ranking overlap decreases quite slowly with the decrease of the size of the reduced history, until the size of the reduced history is about 0.4 of the complete history. The reason is that these networks have a large amount of redundant information for approximating the highest ranked nodes when using the max-flow algorithm. When the size of the reduced history is smaller than 0.3 of the complete history, the ranking overlap degrades quickly. With Pagerank, the reduced history exhibits very low ranking overlap for all the graphs.

Finally, we evaluate the effect of the growth of the complete history while the reduced history is of constant size on the ranking error and the ranking overlap. For the synthetic graphs, we let the complete history grow from 500 to 5,000 nodes while we keep the size of the reduced history constant at 500 nodes. For the real-world graphs, using the available temporal information, we have the Bartercast graph grow from 1,063 to 10,634 nodes with the reduced history constant at 1,063, and the Citation graph from 1,536 to 15,360 nodes with the reduced history at 1,536. Fig. 4.2 (right) plots the ranking error and Fig. 4.4 plots the ranking overlap for different relative growths of the complete history. We observe again that Pagerank achieves a smaller ranking error while the max-flow based algorithm achieves a better ranking overlap, specially for the scale-free and real-world graphs.

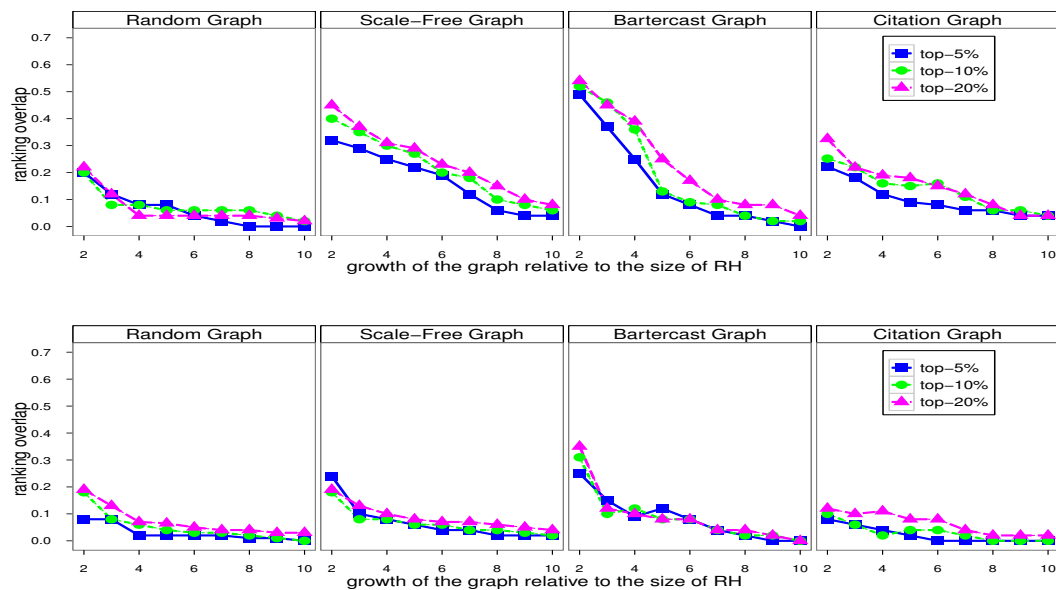


Figure 4.4: The effect of the growth of CH relative to the size of RH for max-flow (top) and Pagerank (bottom).

## 4.5.2 Discussion

The observations arising from our experiments indicate that the reduced history can give a good approximation of the ranking of nodes according to their reputations when the complete history exhibits a particular structure. In this subsection, we explain and discuss our main observations.

First, we observe that constructing the reduced history using the combination of all the parameters for node and edge removal results in the lowest ranking error. Considering only parameters such as degree and reputation gives priority for removal to the newest nodes and so, new nodes will not participate in the reduced history. On the other hand, considering only the age as parameter for removal results in high ranking error because then, only new nodes participate in the reduced history and information of old important nodes has been removed. Therefore, for good performance of the reduced history, it is required to use a combination of these parameters as defined by Eqs. (4.1) and (4.3).

Secondly, the performance of the reduced history depends on the topology of the graph, and is better in the scale-free, Bartercast and Citation graphs than in the random graphs. The scale-free and our real-world graphs have only a few well connected nodes accumulating the majority of links, while the vast majority of nodes has a very low connectivity. In the reduced history, the highly connected nodes are preserved keeping their good ranking position, while most of the loosely connected nodes have

been removed. In contrast, in random graphs all nodes have stochastically similar connectivity properties. Since most real networks exhibit heterogeneity in the connectivity properties of their nodes [6], we can conclude that the reduced history can be applied in a large range of networks.

Finally, the performance of the reduced history depends on the reputation algorithm used. In particular, it causes a lower ranking error when using Pagerank, while it achieves a higher ranking overlap when using max-flow. Pagerank computes the reputation of a node by aggregating the interactions of all nodes participating in a graph. The aggregative computation of centrality by Pagerank achieves lower ranking error even if the reduced history has a relatively small size. Unlike Pagerank, the max-flow based algorithm computes the reputation of a node taking into account only the interactions between that node and the most central node. Since both the most central and the highest ranked nodes are considered as important, they are preserved in the reduced history. Therefore, we achieve a high ranking overlap when using the max-flow based algorithm.

# Chapter 5

## Summary

In D2.1.2 we have presented Design Space Analysis (DSA), a simulation based approach that complements game-theoretic analysis of incentives in distributed protocols. DSA emphasizes the specification and analysis of a design space, rather than proposing a single protocol. In Chapter 2 of the current deliverable we have shown that DSA can be used to gain an in-depth analysis of the properties of protocol variants, and it can be used for designing deployable protocols. In conclusion, we note that DSA is a general method, which is practical and easy to apply, and we have demonstrated its merits by applying it successfully to P2P file swarming systems.

In Chapter 3, with the support of real-world observations, we have provided an analytical model that captures the essence of SRE adopted by private communities. Given the existence of over-seeding user behavior, our simulation results show that by adopting SRE, swarms tend to be extremely oversupplied. Under this oversupply, peers achieve high downloading speeds but with significant tradeoffs, which include low upload capacity utilizations and extremely long seeding times. Under certain scenarios, a peer may seed over 1000 times as long as its downloading time, while on average only utilizing 1% of its upload capacity. Further, SRE discriminates against peers with low capacities and forces them to seed for even longer durations. To alleviate these problems, we propose four strategies and the simulation results show that they are all very effective. Particularly, SRE with supply-based price, while maintaining a system-wide high downloading speed, achieves very stable high upload capacity utilization and reduces seeding durations by three orders of magnitude as compared to the original SRE.

Concluding Chapter 4, our observations demonstrate the effectiveness of the reduced history in approximating the ranking of nodes with Pagerank and in identify-

ing the highest ranked nodes with the max-flow based algorithm. This implies that the reduced history can approximate with reasonably accuracy the complete history in real world graphs, while it has much smaller resource requirements. This result is valuable especially for decentralized systems, such as QMedia, because of the limited resources available at each node.

# Bibliography

- [1] <http://www.bitsoup.org/>
- [2] <http://hdchina.org/>
- [3] [http://en.wikipedia.org/wiki/Negative\\_income\\_tax](http://en.wikipedia.org/wiki/Negative_income_tax)
- [4] [http://en.wikipedia.org/wiki/Welfare\\_for\\_the\\_rich](http://en.wikipedia.org/wiki/Welfare_for_the_rich)
- [5] [http://en.wikipedia.org/wiki/Participatory\\_economics](http://en.wikipedia.org/wiki/Participatory_economics)
- [6] L. A. N. Amaral, A. Scala, M. Barthélemy, and H. E. Stanley. Classes of small-world networks. *Proc. Natl. Acad. Sci. U.S.A.*, 2000.
- [7] K. Anagnostakis, F. Harmantzis, S. Ioannidis, and M. Zghaibeh. On the impact of practical p2p incentive mechanisms on user behavior. *NET Institute Working Paper*, 2006.
- [8] N. Andrade, E. Santos-Neto, F. Brasileiro, and M. Ripeanu. Resource demand and supply in bittorrent content-sharing communities. *Computer Networks*, 53, 2008.
- [9] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.
- [10] Z. Bar-Yossef and L.-T. Mashiach. Local approximation of pagerank and reverse pagerank. *SIGIR*, 2008.
- [11] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 1999.
- [12] P. Bonacich. Eigenvector-like measures of centrality for asymmetric relations. *Social Networks*, 2001.
- [13] X. Chen and X. Chu. Measurements, analysis and modeling of private trackers. In *Proceeding of IEEE P2P*, 2010.

- [14] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.
- [15] L. De Alfaro, A. Kulshreshtha, I. Pye, and B. T. Adler. Reputation systems for open collaboration. *Commun. ACM*, 2011.
- [16] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *ACM EC*, 2004.
- [17] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 1977.
- [18] R. Geisberger, P. Sanders, and D. Schultes. Better approximation of betweenness centrality. In *ALLENEX*, 2008.
- [19] D. Gkorou, J. Pouwelse, and D. Epema. Betweenness centrality approximations for an internet deployed p2p reputation system. In *IEEE IPDPSW (HotP2P)*, 2011.
- [20] Z. Gyongyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. 2004.
- [21] K. Hajra and P. Sen. Aging in citation networks. *Physica A: Statistical Mechanics and its Applications*, 2005.
- [22] T. Hoßfeld, F. Lehrieder, D. Hock, S. Oechsner, Z. Despotovic, W. Kellerer, and M. Michel. Characterization of BitTorrent swarms and their distribution in the Internet. *Computer Networks*, <http://dx.doi.org/10.1016/j.comnet.2010.11.011>, 2010.
- [23] D. Hruschka and J. Henrich. Friendship, cliquishness, and the emergence of cooperation. *Journal of Theoretical Biology*, 239:1–15, 2006.
- [24] R. Izhak-Ratzin. Collaboration in BitTorrent systems. In *IFIP-TC Networking*, pages 338–351, 2009.
- [25] A. Jia, L. D’Acunto, M. Meulpolder, and J. Pouwelse. Modeling and analysis of sharing ratio enforcement in private bittorrent networks. In *Proceeding of IEEE ICC*, 2011.
- [26] A. Jia, L. D’Acunto, M. Meulpolder, J. Pouwelse, and D. Epema. Bittorrents dilemma: Enhancing reciprocity or reducing inequity. In *Proceeding of IEEE CCNC*, 2011.

- [27] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *12th WWW conference*, New York, 2003.
- [28] A.-M. Kermarrec and M. van Steen, editors. *ACM SIGOPS Operating Systems Review 41, Special Issue on Gossip-Based Networking*. 2007.
- [29] A. Legout, N. Liogkas, E. Kohler, and L. Zhang. Clustering and sharing incentives in bittorrent systems. In *ACM SIGMETRICS*, pages 301–312, 2007.
- [30] B. Leong, Y. Wang, S. Wen, C. Carbutaru, Y. Teo, C. Chang, and T. Ho. Improving peer-to-peer file distribution: winner doesn't have to take all. In *ACM APSys*, pages 55–60, 2010.
- [31] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee. BitTorrent is an auction: analyzing and improving BitTorrent's incentives. In *ACM SIGCOMM*, pages 243–254, 2008.
- [32] Z. Liu, P. Dhungel, D. Wu, C. Zhang, and K. Ross. Understanding and improving incentives in private p2p communities. In *Proceeding of ICDCS*, 2010.
- [33] S. Marti and H. Garcia-molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 2006.
- [34] M. Meulpolder, L. D'Acunto, M. Capota, M. Wojciechowski, J. Pouwelse, D. Epema, and H. Sips. Public and private bittorrent communities: A measurement study. In *Proceeding of IPTPS*, 2010.
- [35] M. Meulpolder, J. Pouwelse, D. Epema, and H. Sips. Bartercast: A practical approach to prevent lazy freeriding in p2p networks. In *IEEE IPDPS (Hot-P2P)*, 2009.
- [36] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999.
- [37] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in BitTorrent. In *NSDI*, 2007.
- [38] M. Posch. Win-Stay, Lose-Shift Strategies for Repeated Games—Memory Length, Aspiration Levels and Noise. *Journal of Theoretical Biology*, 198:183–195, 1999.

- [39] A. Post, V. Shah, and A. Mislove. Bazaar: Strengthening user reputations in online marketplaces. In *NSDI*, 2011.
- [40] R. Rahman, M. Meulpolder, D. Hales, J. Pouwelse, D. Epema, and H. Sips. Improving efficiency and fairness in p2p systems with effort-based incentives. In *Proceeding of IEEE ICC*, 2010.
- [41] A. Rao, A. Legout, and W. Dabbous. Can Realistic BitTorrent Experiments Be Performed on Clusters? In *IEEE P2P*, 2010.
- [42] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE TKDE*, 2004.
- [43] C. Zhang, P. Dhungel, Z. L. Di Wu, and K. Ross. Bittorrent darknets. In *Proceeding of IEEE INFOCOM*, 2010.

# Appendix A

## Introduction to Design Space Analysis

We wish to design distributed protocols that maximize performance of the system under the assumption that protocol variants may enter the system. We present Design Space Analysis (DSA), a simulation based method, which emphasizes the specification and analysis of a design space, rather than proposing a single protocol. First, we list the key elements of DSA. Then we present the *Performance, Robustness, Aggressiveness* (PRA) quantification, a solution concept within DSA.

### A.1 Key elements of DSA

We consider the elements that are integral to Design Space Analysis. We compare and contrast each of these with the corresponding elements in traditional game-theoretic analysis.

**Flexible behavioral assumptions.** In DSA, we relax behavioral assumptions. Specifically, unlike traditional game-theoretic analysis, we do not limit ourselves to the rational framework, where nodes are supposed to be self-interested. By foregoing the assumptions entailed in this framework, we consider a great variety of protocols, which may not necessarily be rational. Protocols may, in the words of Axelrod [9], “simply reflect standard operating procedures, rules of thumb, instincts, habits, or imitation”.

**Specification of design space.** In DSA, keeping in view that complex protocols have many elements that can be gamed by strategic nodes, a design space should encompass relevant details that can affect the incentive structure. Design space specification occurs at two levels: i) *Parameterization*, which involves determining the salient dimensions of the design space, and ii) *Actualization*, which involves specifying a host of actual values for every individual dimension.

The specification of the design space can be inspired by consulting the relevant literature and analyzing existing systems. As an example, the Parameterization phase of the design space for *Gossip Protocols* [28] could result in the following salient dimensions: i) Selection function for choosing partners for exchanging data, ii) Periodicity of data exchange iii) Filtering function for determining data to exchange, iv) Record maintenance policy in local database.

The Actualization of this example design space for gossip protocols could be: For Selection Function, following policies could be used : 1) *Random*: Choose partners randomly; 2) *Best*: Choose partners who have given the best service; 3) *Loyal*: Choose most loyal partners; 4) *Similarity*: Choose partners based on similarity; etc. Similarly, different values could be chosen for each of the other dimensions.

In comparison, often in game-theoretic analysis only a single point in the design space is taken into consideration, e.g., peer selection in BitTorrent.

An example of specifying a design space, with both the parameterization and actualization phases, will be described in detail in Sections A.3.1 and A.3.2 when we apply DSA to P2P file swarming systems.

**Systematic analysis of the design space.** In DSA, a desired feature of all solution concepts is a systematic exploration of the design space. This exploration could either follow an exhaustive approach, e.g., a parameter sweep, or a heuristic based approach. By a thorough scan of the space, DSA solution concepts can anticipate strategic variants and predict their effects. Heuristic based approaches can provide partial solutions relatively fast, however, without any guarantees on the level of goodness of the measures.

Game theoretic analyses generally use solution concepts, which involve finding the equilibrium strategy. Such concepts usually require high levels of abstractions and can become intractable when applied to the large design space for complex protocols.

## A.2 The PRA quantification

We now present the PRA quantification, a solution concept within DSA. We note that other solution concepts within DSA could also be devised. Using PRA, we can characterize any protocol, from a given design space, over three measures (or dimensions). For a given protocol  $\Pi$ , these three particular measures, are:

- Performance - the overall performance of the system when all peers execute  $\Pi$  (where performance is determined by the application);

- Robustness - the ability of a majority of the population executing  $\Pi$  to outperform a minority executing a protocol other than  $\Pi$ ;
- Aggressiveness - the ability of a minority of the population executing  $\Pi$  to outperform a majority executing a protocol other than  $\Pi$ .

We formulate a way to assign values to each of the three measures normalized into the range  $[0, 1]$ . Hence the properties of any given  $\Pi$  can be characterized as a point within a three-dimensional Performance, Robustness, Aggressiveness (PRA) space.

It is desirable, in open systems in which strategic variants can enter, to design protocols which maximize all three measures. However, it can be conjectured that there will often be a tradeoff between them. For example, one may design protocols with high performance but low robustness or conversely high robustness and low performance.

We now define more precisely how we can map a given protocol  $\Pi$ , which can be expressed as a point in the design space, to a point in the PRA space; formally, we define a function  $\mathcal{S} : D \rightarrow [0, 1]^3$ , where  $D$  is the design space.

We assume that for each peer in a system of peers we can calculate a utility which quantifies individual performance. The measure of performance is application specific, such as download speed in P2P file swarming systems. Given this we define the performance  $P$  of protocol  $\Pi$  as the sum of all individual utilities in a population of peers executing  $\Pi$  normalized over the entire protocol design space. Hence,  $P = 1$  indicates the best performance obtained from any protocol in the design space.

We define the Robustness  $R$  for protocol  $\Pi$  as the proportion of all other protocols from the design space that do not outperform  $\Pi$  in a *tournament*. A tournament consists of multiple *encounters* in which protocol  $\Pi$  plays with all other protocols. An encounter is a mixed population of peers executing one of two protocols. The winning protocol is that which obtains the higher average utility for the peers executing it.

Aggressiveness  $A$  for protocol  $\Pi$  is defined in the same way as Robustness, but here  $\Pi$  is in the minority.

### A.3 Applying DSA to P2P File Swarming Systems

In this section, we describe our methodology for applying DSA to P2P file swarming systems. First, we Parameterize a generic P2P design space. Next, based on this generic space, we Actualize a specific file swarming design space. Subsequently, we

apply the PRA quantification on this space. Finally, we present the results of our analysis.

### A.3.1 Parameterization of a Generic P2P Protocol Design Space

We have identified the following salient dimensions applicable to a large variety of P2P systems.

**Peer Discovery:** In order to perform productive peer interactions, it is necessary to find other partners. For example, when a peer is new in the system, looking for better matching partners or existing partners are unresponsive. The timing and nature of the peer discovery policy are the important aspects of this dimension.

**Stranger Policy:** When interacting with an unknown peer (stranger), past history cannot be used to inform actions. It is therefore necessary to apply a stranger policy. The way peers allocate resources to strangers is an important aspect of this dimension.

**Selection Function:** When a peer requires interaction with others this function determines which of the known peers should be selected. This could include, for example, past behavior (through direct experience or reputation system), service availability and liveness criteria.

**Resource Allocation:** During peer interactions resources must be allocated to the selected peers (given by the selection function). The way a peer divides its resources among the selected peers, defines the Resource Allocation Policy.

### A.3.2 Actualization of a Specific P2P Protocol Design Space

We define some specific actualizations of a BitTorrent-like file swarming system based on the general design space of Section A.3.1. The ideas behind these actualizations have been taken directly from, or inspired by, various works on cooperation done in P2P and also some works done in biology and social sciences in general [9, 23, 38]. We were motivated to take inspiration from other fields, because eliciting cooperation in decentralized settings is a general problem that has been well-studied.

For the **Stranger Policy**, we define three different actualizations and a value  $h$  for the number of strangers to cooperate with:

B1) *Periodic*: Give resources to up to a certain number of strangers periodically.

- B2) *When needed*: Only give resources to strangers when set of regular partners is not full. This particular implementation has been inspired by [24].
- B3) *Defect*: Always defect on strangers, i.e., give nothing to strangers.

We set  $h$ , the number of strangers to cooperate with at any given time, to be in the range  $[1, 3]$ . This gives  $3 \times 3 = 9$  different stranger policies. We further add one more stranger policy, where the number of strangers is zero. This gives a total of 10 different stranger policies.

We sub-divide the **Selection Function** into three parts: a candidate list, a ranking function over that candidate list, and finally a value  $k$  for the number of peers to select from the ranked candidate list.

For the 'Candidate List' we define two actualizations:

- C1) *TFT*, used by default in BitTorrent, using which a peer only places those peers in the candidate list who reciprocated to it in the last round.
- C2) *TF2T*, using which a peer places those peers in the candidate list who reciprocated to it in either of the last two rounds. TF2T has been taken from [9].

For the 'Ranking Function', we define six different actualizations:

- I1) *Sort Fastest*, ranks peers in order of fastest first.
- I2) *Sort Slowest*, ranks peers in order of slowest first.
- I3) *Sort Based on Proximity*, ranks peers in order of proximity to one's own upload bandwidth, as in Birds.
- I4) *Sort Adaptive*, ranks peers in order of proximity to an *aspiration level*, which is adaptive and changes based on a peer's evaluation of its performance. This has been inspired from [38].
- I5) *Sort Loyal*, ranks peers in order of those who have cooperated with the peer for the longest durations. This has been inspired by [23].
- I6) *Random*, does not rank peers and chooses them randomly. This has been inspired by [30].

After applying the ranking function, a peer chooses the  $k$  top peers. Currently, we set  $k$  to be in the range  $[1, 9]$ . This results in  $2 \times 6 \times 9 = 108$  different possibilities for the selection function. We further add one more protocol, where the number of selected peers is zero. This gives a total of 109 different stranger policies.

For **Resource Allocation**, we define three actualizations:

- R1) *Equal Split*, gives all selected peers equal resources (upload bandwidth).

R2) *Prop Share*, gives others proportional to what they gave in the past. This has been inspired by [31].

R3) *Freeride*, gives nothing to partners.

Based on the above, the total number of unique protocols comes to  $10 \times 109 \times 3 = 3270$ . We note that this number can be larger or smaller based on what, and how many, specific implementations in the space, designers want to explore<sup>1</sup>. Our purpose here is to show the practicality of the DSA analysis by analyzing a considerable space of unique protocols.

### A.3.3 Conducting the PRA quantification

First we describe our simulation model. Then we discuss our methodology for conducting the PRA quantification on the design space described in Section A.3.2.

#### Simulation Model

We use a cycle-based simulation model, in which time consists of *rounds*. In each round, a peer decides to upload to a given number of peers based on some *selection* criterion. It uses its *resource allocation* policy to decide how much to give to each of the selected partners. Furthermore, it decides to cooperate with strangers based on its *stranger policy*. A peer also maintains a short history of actions by others. At the same time a peer also has some rate of requesting services from other peers that depends on specific actualizations. This is the basic model on top of which we explore the design space of Section A.3.2. We run our simulation experiments with 50 peers, which is a good approximation of an average BitTorrent swarm-size [22]. These peers interact with each other for 500 rounds. We use a cluster for running our experiments. In order to lend realism to our experiments, we initialize the peers using the bandwidth distribution provided by Piatek et al. [37]. We assume that all peers always have data that others are interested in.

#### Methodology

Based on the PRA quantification, as described in Section A.2, we first measure the Performance of each protocol in the space. For each protocol  $\Pi$ , we run simulations in which all peers execute  $\Pi$  and measure the average performance of the population.

---

<sup>1</sup>For instance, we do not consider variants for resource allocation to strangers. Also, we do not consider Peer Discovery.

We perform 100 runs for each protocol. In these experiments we define average performance as throughput of the population.

Next we run Robustness experiments. We run simulations, where each protocol plays against every other protocol. We refer to a competition in which two protocols are pitted against each other as an *encounter*. For each encounter, the peer population is split up into two equal halves where half the peers execute  $\Pi$  and the other half executes another protocol. We chose 50% because this is the highest number that an invading protocol can have. Anything higher than 50% means that the invading protocol actually becomes the majority protocol. We hypothesize that if a protocol is robust when 50% of the population executes another protocol, then it will be robust against small invading populations. To verify this hypothesis, we also conduct simulations with the population split up into 75-25, where 75% of the peers follow protocol  $\Pi$ , while 25% execute other protocols in the space, and observe similar results. We do 10 runs for each particular encounter between two protocols. This means that a protocol  $\Pi$  plays against the same protocol ten times. For each run, we compare the average performance of  $\Pi$  with the average performance of the other protocol. If the performance of  $\Pi$  is greater than the performance of the other protocol, we mark it as a *Win* for  $\Pi$ , otherwise we mark it as a *Loss* for  $\Pi$ . The robustness value for  $\Pi$  is calculated by number of games that it wins against all opponents in all runs divided by the total number of games that it plays, which is constant for all protocols.

In Chapter 2 we present results of applying the PRA quantification over the design space described in Section A.3.2 of this Appendix. We note here that the entire series of simulations was computationally tractable.