P2P applications: Performance Improvements and Network Awareness

Univerisitá degli Studi di Torino

Dipartimento di Informatica

Dottorato di Ricerca in Informatica
XXIII ciclo

PhilosophiæDoctor (Ph.D.) thesis

Salvatore Francesco Spoto

Advisors:

Prof. Matteo Sereno

Prof. Rossano Gaeta

Prof. Marco Grangetto

Ph.D. Coordinator:

Prof. Mariangiola Dezani

Abstract

In the last years the research community has dedicated considerable attention to peer-to-peer (P2P) protocols. This manly derives from the great diffusion of P2P applications on the Internet and from the benefits of this paradigm, that are scalability, robustness and resource savings. The number of P2P software is steadily increasing, and many of them have a huge user base, sometimes made up of thousands of clients connected at the same time. Furthermore, services have evolved from the first file-sharing protocols into a wide variety, showing the flexibility of a P2P overlay. Some notable examples are IPTV, distributed portals, distributed computing and so on.

This popularity has also some drawbacks. The huge daily traffic generated by P2P has a great impact on the underlying network, and protocols optimization becomes a very critical task. Indeed the efficiency of a protocol is closely related to the user satisfaction and resource savings. The goal is to reduce the impact on the physical infrastructure while maintaining similar performance in terms of customer service. This dissertation focuses on P2P protocols, aiming to study the impact on the underlying network layer and improving their performance using coding techniques.

First, we begin our investigation with a case study to define the behaviour of a P2P overlay and evaluating its impact on the physical network. Our first analysis is based on a very popular P2P IPTV application i.e., PPLive. This is a closed source software, so we develop a methodology that is a mixture of active and passive measurements to describe the client's behaviour and the network overlay, their relationships and evolution over time.

Using this knowledge, we address the problem of protocol optimization focusing on the most popular file-sharing application: BitTorrent. With sim-

ulation tools we evaluate the potential applications of rateless codes in BitTorrent and identify specific network conditions in which the use of coding techniques is particularly useful.

We than develop a novel protocol, Toroverde, based on Luby-Transform codes. We provide a complete protocol specification and develop a fully functional protocol implementation. Its performance are compared with BitTorrent using simulation tools and a C++ prototype deployed on PlanetLab. Our results show that ToroVerde achieves better performance than BitTorrent when the overlay is in unstable conditions i.e., the peers exhibit a high dynamical behaviour characterized by high churn rates and flash crowds.

In this dissertation we also study the impact that a P2P protocol has on the network infrastructure. We propose a model for P2P traffic and several techniques for network awareness. First we focus on the Internet Service Provider (ISP) behaviour and their cooperative/non-cooperative attitudes. We propose a game theoretic framework to help the design of techniques promoting the ISP cooperation in P2P streaming platforms. Moreover, we apply ideas from Evolutionary Game Theory to the ISP game in a large ISP population. The model shows that the proposed strategies can effectively stimulate ISP cooperation aiming at the minimization of inter-ISP traffic and help to provide reliable P2P streaming service.

Finally, we use an approach based on the use of generating functions, to study analytically the link occupation and the transfer costs associated to file-sharing traffic. Our model is able to compute costs and rewards for Autonomous Systems organized in a complex network topology, considering commercial agreements like peering and transit and inter Autonomous Systems routing policies. Our model has been validated using one of the most widely used P2P simulator, PeerSim. Several experiments show how the proposed model is able to consider overlay with many peers connected through a series of links and sharing several resource.

Contents

List of Figures vii

List of Tables xi

1 Introduction 1

1.1 Motivation and context: P2P and ISPs . . . 2
1.2 Research objective and contribution . . . 4
1.3 Used methodologies . . . 7
1.4 Dissertation overview and structure . . . 8
1.5 Original contribution . . . 9

2 Literature review 11

2.1 The peer-to-peer paradigm . . . 11
2.2 BitTorrent . . . 13
2.3 IPTV . . . 15
2.3.1 BitTorrent for video-streaming . . . 15
2.4 The use of network coding in P2P applications . . . 16
2.5 The impact of P2P protocols on Internet Service Providers (ISPs) . . . 20

3 A case study: PPLive 25

3.1 The PPLive application . . . 25
3.2 Methodology . . . 26
3.3 The crawler . . . 26
3.3.1 Crawling "completeness" and passive/active measures . . . 28
3.4 Passive measurements . . . 28
3.5 Combination of active and passive measures . . . 30

3.6 Active measures and topological properties 32
3.6.1 Analysis of user sessions 34
3.7 Conclusions 35

4 Improving BitTorrent using Luby-Transform (LT) codes 41
4.1 The Luby-Transform codes 41
4.2 The modified Protocol 44
4.2.1 Modifications to BitTorrent protocol 44
4.3 Simulation methodology 46
4.3.1 Extension of the GPS simulator 46
4.4 Experimental results 47
4.4.1 A simple topology of three peers 48
4.4.2 A more complex scenario 49
4.4.3 Flash crowd scenario 52
4.5 Conclusions 54

5 P2P simulator 55
5.1 The General Purpose Simulator (GPS) 55
5.2 The transit-stub model 56
5.2.1 GT-ITM 57
5.2.2 The new file format 58
5.3 The new simulator 59
5.3.1 Network layer 59
5.3.2 Application layer 60
5.3.3 Simulation layer 60
5.4 Benchmarks and validation 60
5.4.1 Simulations validation 61
5.5 Conclusion 64

6 The Toroverde protocol 65
6.1 Protocol Description 65
6.1.1 Protocol overview and terminology 65
6.1.2 Overlay management 67
6.1.3 LT coding and decoding 68

6.1.4 Bitmap update . . . 68
6.1.5 Flow control . . . 69
6.1.6 Block scheduling policy . . . 70
6.1.7 Rarest first implementation . . . 71
6.1.8 Enabled neighbors . . . 73
6.1.9 Protocol's messages . . . 73
6.1.9.1 TRACKER_REQUEST . . . 73
6.1.9.2 TRACKER_RESPONSE . . . 74
6.1.9.3 CONNECTION_REQUEST . . . 74
6.1.9.4 CONNECTION_ACCEPT . . . 74
6.1.9.5 CONNECTION_REFUSE . . . 74
6.1.9.6 ALREADY_CONNECTED . . . 74
6.1.9.7 CLEAR_BM . . . 74
6.1.9.8 CLEAR_ACK . . . 75
6.1.9.9 KEEP_ALIVE . . . 75
6.1.9.10 TRACKER_KEEP_ALIVE . . . 75
6.1.9.11 BLOCK . . . 75
6.1.9.12 TFRC_FEEDBACK . . . 76
6.1.9.13 QUIT . . . 76
6.1.10 ToroVerde versus BitTorrent . . . 76
6.2 PlanetLab prototype and simulator description . . . 76
6.2.1 PlanetLab architecture . . . 77
6.2.2 Simulator of ToroVerde . . . 77
6.3 System architecture . . . 78
6.3.1 Outgoing queue . . . 79
6.3.2 Event list . . . 79
6.3.3 Message receiver . . . 79
6.3.4 File manager . . . 79
6.3.5 Neighborhood manager . . . 79
6.4 Experimental results . . . 79
6.4.1 System parameters and scenarios . . . 80
6.4.2 Results from PlanetLab . . . 81
6.4.3 Simulation results . . . 82

6.5 Conclusions 83

7 P2P protocols and Internet Service Providers: a game theoretical approach 85

7.1 Internet Service Provider awarness 85
7.2 P2P Streaming: Peers and ISPs 86
7.3 Game Theory Formulation 93
7.3.1 Peer Game Model 93
7.3.2 ISP Game Model 97
7.3.3 Evolutionary Analysis of Streaming Games 100
7.4 A ISP Streaming Traffic Management Algorithm 107
7.5 Conclusions 110

8 The impact of P2P protocols on ASes network topology 113

8.1 The Internet AS-level 113
8.1.1 Traffic Agreements: peering and transit 114
8.2 Modelling Autonomous Systems 115
8.3 P2P Resource Diffusion 119
8.3.1 Network Scenario 119
8.3.2 The Model 121
8.3.3 System dynamics 122
8.4 Traffic among Autonomous Systems 125
8.4.1 Different Search Policies: internal search first 126
8.4.2 Different Search Policies: weighted search 127
8.5 Considering more resources 127
8.6 Model validation 128
8.7 Experiments 130
8.7.1 Changing ASes size 130
8.7.2 Changing links' weights 131
8.7.3 Changing links' costs 131
8.7.4 Changing AS agreements 131
8.8 Conclusions 132

9 Conclusions and future work **141**

9.1 Summary 141

9.2 Topics for further investigation 143

9.2.1 Network coding and P2P overlay 143

9.2.2 Applying network coding to other P2P protocols 144

9.2.3 Extend the game theoretical framework 144

9.2.4 Improving the cooperation between Autonomous Systems and P2P protocols 144

A **147**

A.1 Proof 1 147

A.2 Proof 2 148

A.3 Proof 3 148

References **151**

List of Figures

2.1 Butterfly network 17

3.1 PPLive basic architecture 27
3.2 PPLive packet size histogram 29
3.3 Class A rates (kbps) per peer 31
3.4 Class B-C rates (kbps) per peer 32
3.5 Class C sorted rates (kbps) per peer 33
3.6 Class A rates (kbps) per peer 34
3.7 Class B-C rates (kbps) per peer 35
3.8 Number of neighbors versus time for $\Delta = 15$ seconds 36
3.9 Evolution of PPLive peer population over nine days 36
3.10 PPLive peer degree distribution 38
3.11 Peer session duration 38
3.12 Peer session duration 39

4.1 Example of decoding process 43
4.2 Three peer topology 49
4.3 Normalized distributions of the download times 51
4.4 Evolution of overlay population 54

5.1 Example of transit-stub network 56
5.2 Memory requirements comparison between GPS and the NEW simulator 62
5.3 Time requirements comparison between GPS and the NEW simulator . 62
5.4 Heap usage comparison between GPS and NEW simulators without BitTorrent clients 63

5.5 Heap usage comparison between GPS and NEW simulators with 500 BitTorrent clients . . . 63
5.6 BitTorrent simulation validation . . . 64

6.1 ToroVerde simulation validation . . . 78

7.1 I_i and O_i as function of m_i . . . 90
7.2 $I_i(\alpha_i)$ and $O_i(\alpha_i)$ as function of m_i . . . 92
7.3 Set of values that allow universal streaming and set of positive payoff values . . . 94
7.4 Values that allow universal streaming for two ISPs . . . 98
7.5 Set of values of O_1^* and O_2^* as function of I_1, and as function of m_1 . . . 99
7.6 Evolution of the replicator dynamic . . . 103
7.7 Numerical Experiment results for the game theoretical model . . . 105
7.8 Simulation traces for the game theoretical strategy . . . 106

8.1 ASes topology used for experiments . . . 116
8.2 Example topology of six ASes with the cost and reward matrices for AS_2 133
8.3 ASes topology used to validate the model . . . 134
8.4 Comparison between the model and the simulation for transferts from AS_1 to others . . . 134
8.5 Comparison between the model and the simulation for transferts from AS_2 to others . . . 135
8.6 Comparison between the model and the simulation for transferts from AS_3 to others . . . 135
8.7 Comparison between the model and the simulation in the steady state . 136
8.8 Different configurations of ASes sizes . . . 136
8.9 Rewards distribution considering several configurations of ASes size . . 137
8.10 Costs distribution considering several configurations of ASes size 137
8.11 Distribution of the difference reward-cost (net profit) considering several configurations of ASes size . . . 138
8.12 Distribution of the difference reward-cost (net profit) when all ASes increase weights to their peering links . . . 138

8.13 Distribution of the difference of the net profit respect to $w = 0$ when only AS_7 increase weights to its peering links 139
8.14 Net profit when all only AS_8 increase its trasnsit costs 139
8.15 Comparison between two different topologies 140

List of Tables

3.1 Rates (kbps) for different PPLIVE packet classes 30
3.2 Intersections between neighbors list and sniffed traces 37
3.3 Active measures on PPLive nodes 37

4.1 Dowloading times of the modified BitTorrent protocol 47
4.2 Downloading times and bitrate on a three peers topology 50
4.3 Downloading times and bitrate during a flash crowd 51
4.4 Parameters for flash crowd scenario 52
4.5 Downloading times and bitrate in a scenario with by several flash crowds 53

5.1 CPU and memory requirements comparison 61

6.1 Table of symbols for ToroVerde 67
6.2 System parameters for ToroVerde 80
6.3 Average completion times measured on PlanetLab 83
6.4 Average completion times for simulated protocols 83

7.1 Notation for the game theory model 87

8.1 Generating functions model notations 120
8.2 Parameters of simulation for generating function model validation . . . 129

Chapter 1

Introduction

A P2P network (or overlay) is a distributed application architecture that subdivides the workload among all participating peers. The peers, also called nodes of the overlay network, are equally privileged, and each one shares a fraction of its resources with others, without a central coordination. This type of architecture is suitable for many kind of applications: file sharing, video-on-demand, live-streaming and distributed computing are few examples.

Nowadays peer-to-peer (P2P) applications generate a substantial fraction of the total Internet traffic (81, 82). The popularity of these applications is rapidly growing thanks to the widespread adoption of large bandwidth access links by residential customers.

Many people use P2P software to retrieve and share contents over the Internet and the user base is constantly growing following a trend that it is not likely to stop soon. The reasons for this widespread diffusion are not difficult to understand. From the perspective of the end user, P2P applications make possible to retrieve a large amount of multimedia contents, and in turn to share media with other people. Most important, the benefit of using a P2P overlay with respect to the classical client/server paradigm is to shift the load from a central server to peers, with many advantages: saving server bandwidth, providing the service to more users with the same resources utilization, obtaining a more robust system due the error-resilient features of a distributed overlay.

On the other side, a P2P overlay exploits the Internet infrastructure (core and access networks) and generates a huge amount of traffic. This represents a problem especially for Internet Service Providers (ISPs), that are flooded by traffic generated

by P2P networks. The optimization of P2P protocols and understanding the interaction dynamics between ISPs and a P2P networks are very critical tasks to reduce the workload on the Internet infrastructure and to offer better services for end users.

1.1 Motivation and context: P2P and ISPs

A P2P protocol can be suited for many different purposes. Historically these kinds of protocols was mainly oriented to share and download content for an off-line utilization, while in recent years other applications arose. Nowadays P2P systems are used in different contexts, such as file sharing (Napster(71), Gnutella (70), KaZaa(73), eDonkey (72), BitTorrent (11, 12) to name a few), telephony applications (Skype (74)), video-on-demand and live-streaming.

In a P2P overlay, the strategies used by peers to exchange data one another can be very different, according to the application that the protocol implements. E.g. a live-streaming software needs to retrieve the video chunks that are near to be reproduced, so its strategies to share and download video pieces take in account the playback deadline. To the contrary in file-sharing applications the content can be distributed without particular order's constraints. Clearly a protocol is built and developed around its objectives. Protocol's strategies can be less or more sensitive to several factors, like peers behaviour, network conditions and end users bandwidth distribution, to name a few.

Among all P2P applications file-sharing is one of the most popular. The resources shared with this kind of application range from several kilobytes up to some gigabytes. The reference protocol is nowadays BitTorrent, and the research community has dedicated many efforts in the last years to investigate, understand and improve its strategies. Many works in the literature argue that BitTorrent can reach near-optimal performance (15, 16, 17, 18, 19, 20, 21, 52, 62, 63).

Despite its excellent performance, (52, 62, 63) show how these are strongly related with the network conditions: when the network is characterized by constant arrival rates, there are no flash crowds and many peers continue to share the already downloaded file the protocol performs well. In other words BitTorrent shows near-ideal dowloading times when the overlay is quite stable. But the peers' behaviour could be very different, especially in different P2P applications. Examples are video-on-demand

or live-streaming, characterized by higher churn rates respect to file-sharing application with a high percentage of short-stay users (13, 14) and flash crowds (81). This made the BitTorrent protocol not suitable for other purposes than file-sharing.

To increase the performance of the P2P overlay, many works concentrate their efforts on the network coding. Indeed a large body of literature has focused on the analysis of potential benefits of coding techniques for content distribution. The concept behind is to combine packet together before transmission: a node receives some packets from its neighbors, combines a subset of these together (the re-combination strategies depend on the coding technique adopted), then sends the newly generated packet to other nodes in the network. A node that has received enough coded packets is able to retrieve the original message.

Since network coding was originally introduced in information theory (51), (64), some approaches have been proposed to use it with P2P protocols. However, conclusions ranged from very optimistic (early results pointed out that throughput of the system exploiting network coding at the block level is two to three times better than the non-coding system (58, 59, 60)) to rather pessimistic (systems using network coding perform even worse than BitTorrent (69)) to inconclusive (both throughput and its theoretical upper bound overlap in coding and non coding systems (66, 67)). These results depend on the type of coding technique and on the analysis methodology employed by different researchers.

In Chapters 4 and 6 we study how several network factors impact on protocol performance, and propose some solutions to counteract drawbacks resulting by the unstable peer's behaviour, high churn rates and flash crowd. In particular we investigate how network coding techniques can improve the P2P protocol performance first modifying the BitTorrent protocol, then developing a completely new one.

The use of network coding, even when introduces improvements in the protocol, is limited to the application level. Indeed, the P2P overlay operates over a complex network made of interconnections among Autonomous Systems (ASes). So another important aspect speaking of P2P overlay is the interaction between the application level and the underlying network layer.

The main problem is that in P2P applications, the participating nodes form an overlay network which is largely agnostic of the ASes topology. This increases the cost associated with P2P traffic for individual Internet Service Providers (ISPs) which face

the problem to manage a vast amount of unnecessary inter-domain traffic. Such traffic level can cause congestion on valuable inter-ISP links and more generally an overall performance degradation. Network-aware techniques may drastically reduce the inter-ISP traffic volume. As example, in (81) a measurement study of PPLive, reported that the 70% of the observed traffic deployed in North America was originated by peers located in Asia. In this situation clearly a ISPs-aware policy (e.g. here trying to maximize connections in the same country) reduces drastically the negative effects of topology mismatch between the P2P overlay and the underlying IP network layer, improving the poor usage of the latter.

New policies that regulate the interaction between ISPs and P2P applications are necessary to alleviate these drawbacks and to lead to a better use of network resources. Model the interaction between ISPs and P2P overlay, studying how the resources distribute between the different autonomous systems and how this process impact on ISP's costs are important steps to understand what are the potentially drawbacks of policies not ISP-aware. The goal is to develop effective methods that optimize the process of resource distributions and minimize the costs for ISPs.

Considering the popularity of P2P application and the current trend, we expect that great research efforts will be made soon in this direction. In Chapters 7 and 8 we focus on the interaction of P2P overlay with the underlying Internet structure, proposing a model for P2P resource diffusion and several techniques for network awareness.

1.2 Research objective and contribution

In order to understand the P2P overlay dynamics, it is interesting to study a widespread diffused video-streaming application. Moreover the measurement of relevant application parameters could be exploited to build simulators or analytical models of P2P based IPTV applications and support the design and implementation of more efficient systems.

Our case study is the popular P2P IPTV application PPLive. This software permits to watch both video on demand and live streaming contents. In Chapter 3 we develop a measurement study of PPLive that combines the use of passive and active measure to derive informations on the traffic as well as on the overlay topology generated by PPLive channels. To this end we develop a distributed analysis software (i.e., a crawler)

that collects information on the topological properties of PPLive overlay networks. This active measurement experiment is correlated with the concurrent measures obtained by monitoring a peer joining the crawled overlay (passive measurement).

Our technique allow us to discuss the limits and the accuracy of the information that can be collected by using only active measurement for PPLive, and provides a better interpretation of the results presented in previous studies. We report findings that confirm the results obtained by other investigators as well as a characterization of the user behavior in the time domain. These results are useful as they enable us to understand the potential weaknesses of a p2p overlay, and work on improving these.

Indeed a significant part of our research is dedicated to improve the performance of file-sharing applications. We investigate how the introduction of application level coding techniques in a P2P protocol can improve its performance.

Our choice has fallen on LT codes (36). This is a fountain code invented by Michal Luby that use simple XOR operations to encode the message. LT codes were chosen due their simplicity and efficiency. They belong to the class of rateless erasure codes, and they have the property to generate a potentially limitless sequence of encoded symbols, differently from standard erasure codes that have a fixed number of codewords.

First we work on BitTorrent, testing some modifications to the protocol's standard algorithms. In BitTorrent no coding at all is used. The whole file is subdivided in pieces, and each piece in blocks. These latter are exchanged among peers. We introduce the LT encoding at piece level, generating coded blocks from the file's pieces. The other protocol's strategies i.e., tit-for-tat or rarest-first (12) required also several modifications to work with rateless codes.

We proved by simulations that better performance can be achieved considering file of small size and when peers show a very dynamical behaviour, characterized by high churn rates and flash crowds. In such situation our proposed protocol yields downloading times that are 10% less in all simulated scenarios, despite the decoding overhead that the LT codes introduce. These results show that using coding techniques can lead to improvements, but these are mainly related to network conditions and peers' behaviour.

Using this knowledge, we implement a completely new P2P protocol that uses rateless codes, namely ToroVerde. We defined a complete specification of the protocol and provided a detailed description of all the employed strategies. We developed both a

detailed simulator and a complete prototype that we used to perform a fair comparison between ToroVerde and BitTorrent. Experimental results suggest that the average download delay can be reduced by ToroVerde in several realistic scenarios for small-to-medium sized files in overlays composed of a few hundred peers. Experiments on PlanetLab show a clear advantage in all the scenarios presented. ToroVerde can achieve performance increase of 10%, in terms of downloading time. The simulation results confirm these benefits, in particular for overlay networks with a size less than 300 peers.

In this dissertation, we also work on models for P2P resource diffusion and strategies for network awareness trying to optimize, in terms of resources saved, the interaction between P2P protocol and Internet Service Providers. We have developed a game theory framework that can be used for the design of ISP strategies aiming at providing efficient and reliable support for P2P streaming application and a probabilistic model, based on generating functions, that describes the diffusion of a resource taking into account non-homogeneous resource popularity and peers behavior.

The game theory framework allows us to illustrate the existence of equilibrium points and the role of possible strategies to find these points. We use ideas from the evolutionary game theory to derive techniques, that a player (an ISP) of the game can use, to compute the operational points assuming only limited knowledge on the state of the other players. We develop a simulation based analysis that shows the relation among peer dynamics and the effects of these dynamics. We focus the attention on the role of the ISPs. In particular, we do not consider them as simple providers of network connectivity but as the main players for supporting P2P streaming platforms. The proposed framework helps the design of techniques promoting the ISP cooperation in P2P streaming platforms. Using the proposed models we derive strategies that can effectively stimulate ISP cooperation aiming at the minimization of inter-ISP traffic by supporting a reliable P2P streaming service.

To understand the diffusion of a resource in a network organized into independent ISPs, we also propose a probabilistic model. We have studied the diffusion of a resource in this non-homogeneous environment. Peers are grouped into Autonomous Systems (ASes), each one having its own parameters in terms of resource availability and demand. We then describe the diffusion of a resource taking into account resource popularity and peers behavior. We study the impact of P2P file sharing applications

on the overall network traffic, by considering the geographic location of peers and the communication costs. Our analytical model computes how a P2P overlay influence the distributions of costs and rewards for ASes organized in a complex network topology, considering peering and transit accords and inter-AS routing policies.

1.3 Used methodologies

Different methodologies and techniques has been employed during our investigation. Our work on PPLive network requires us to develop the software for data collection and analysis. Our goal was derive topological property and traffic patters of the protocol, so we have developed a mixed measurement technique, that combines passive and active ones. Implementing this technique requires us to program a crawler that explore the PPLive network overlay and report topological information. We have merged these active measurements with passive ones, that are collected by logging nodes activity. Using this methodology we have been able to derive many useful informations such the length of users session in the PPLive overlay, statistics on bandwidth utilization and a graph representation of the network.

We have also developed a lot of simulation tools during our study on P2P protocols for file-sharing. Using these we have tested our strategies and evaluated their effectiveness in different network conditions. The simulated protocols also permit to scale up to overlay with many peers. First we have modified the General Purpose Simulator (GPS) modifying BitTorrent. Then we have developed a totally new simulator from scratch, using the Java programming language, providing a full implementation both of BitTorrent and our new protocol, ToroVerde. To evaluate realistic scenarios we added in our software the support for the transit-stub topology model, peer's churn and heterogeneous bandwidth distribution.

In the second part of this dissertation we analyze the relationships between Internet Service Providers and P2P overlays. The goal was derive strategies that can effectively stimulate ISPs cooperation aiming at the minimization of inter-ISP traffic and help to provide reliable P2P streaming service. So we have used a game theory model to investigate the peer's interactions and analyze incentive mechanisms for P2P file-sharing systems. With this model we have discussed the Nash equilibrium, the Pareto

optimality, and fairness criterion to renew the equilibrium points. Moreover, we apply ideas from Evolutionary Game Theory to the ISP game in a large ISP population.

Then we focused the attention on the resource distribution in a network of autonomous systems. To obtain a model that handle very large networks with low computational cost we proposed a probabilistic approach that exploits the properties of the Z-Transform. We validate the proposed model implementing a simulation using PeerSim (117). Once we derived the resource diffusion we used it to compute interesting performance indexes such as the amount of P2P traffic on the links connecting the ASes and the cost required to each AS for transmitting P2P traffic.

1.4 Dissertation overview and structure

We now provide an overview of each technical chapter of this thesis. We present the background material, literature review and additional related work in **Chapter 2**. We describe the general P2P paradigm and how it is adapted to different tasks. We introduce BitTorrent and describe its dynamics. We also give an overview of IPTV applications. Here we introduce the previous attempts to conjugate P2P with some kind of network coding.

In **Chapter 3** we start our investigation focusing on a popular P2P-IPTV application, PPLive, exploiting a measurement strategy that combines both active and passive measures. We take active measurements using a crawler that explores the topology of the PPLive network, and these are combined with passive ones that are relative to the behaviour (bandwidth usage, number of open download and upload connections and so on) of a particular node of the network. We describe PPLive and the motivations of our research. We describe the crawler we developed that allows the study of the topological characteristics of the overlay.

In **Chapter 4** we first introduce some modifications in BitTorrent to enhance it with a particular class of rateless codes, i.e., the LT codes. Later on, by using a simulation based study, we compare the performance of the modified BitTorrent against a standard version of the protocol. We describe the different scenarios that we set up to prove the effectiveness of the modified protocol, producing experimental results. We describe the simulator we have developed in **Chapter 5**, its architecture and its validation with real BitTorrent data.

The ToroVerde protocol is presented in **Chapter 6**. We provide the protocol specification and describe the simulator and the complete prototype we developed for PlanetLab deployment and testing. We perform an analysis of the potential advantages of introducing rateless codes in a content dissemination protocol. We present results from PlanetLab experiments that are compared with performance of the protocol that is widely considered as the reference system for content distribution, i.e., BitTorrent. We also present a few simulation results showing the behavior of ToroVerde as the number of peers in the systems increases.

The second part of this dissertation concern about interaction between P2P protocols and Internet Service Provider. In **Chapter 7** we present the game theoretical framework we have developed while **Chapter 8** is dedicated to the probabilistic model that describes how the diffusion of resources in a P2P overlay impact on the underlying structure of autonomous systems. In this chapter we present some examples of application of this model.

Finally in **Chapter 9** the conclusions of this research are presented and recommendations for further research are made.

1.5 Original contribution

The work presented in this thesis has led to some original contributions. The measurements strategy of PPLive, and its results, described in Chapter 3 has been published in **"Analysis of PPLive through active and passive measurements"**, 23rd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2009, Rome, Italy.

In Chapters 4 we present our improvements of the BitTorrent protocol using LT codes. This work appeared in **"BitTorrent and fountain codes: friends or foes?"**, in proceedings of 24rd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2010, Atlanta, Georgia.

In Chapter 6 we extend the previous work and develop ToroVerde, a novel P2P protocol based on Luby-Transform codes. ToroVerde protocol has been published in **"Fountains vs Torrents: The P2P ToroVerde Protocol"**, 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication, MASCOTS 2010.

The game theoretical framework presented in Chapter 7 has been described in the article **"ISP and P2P: friends or foe?"**, currently submitted to journal.

The analytical P2P resource diffusion and the AS-level models described in Chapter 8 has been submitted to the journal "Performance Evaluation" in the article **"Peer to Peer transfer among Autonomous Systems"**.

Chapter 2

Literature review

In this chapter we describe the topics of this dissertation and give an overview of the related literature. First we introduce the peer-to-peer (P2P) paradigm and its applications, focusing in particular on file-sharing and video-streaming.

Then we describe the BitTorrent protocol, that is considered the state of the art for file-sharing P2P applications. We explain the strategies that it employs and the related terminology. We also provide an overview of the previous attempt to introduce network coding techniques in P2P protocols.

Finally we give a review of the previous works that have investigated the relationships between Internet Service Provider (ISP) and P2P overlays.

2.1 The peer-to-peer paradigm

Peer-to-peer (P2P) is a distributed application architecture that subdivides the workload of a specific task among all participating peers. These peers, also called nodes, connect one another in a network of relationships usually called overlay. In the overlay resources such storage, bandwidth or processing power are shared. There is no central coordination of the nodes, and their behaviour is totally defined by the protocol them implement. In a P2P overlay nodes are both suppliers and consumers, differently from the forerunner traditional client server model. To the contrary in this latter model the server provides for resource diffusion and all other nodes are only consumer. A P2P cooperative overlay distributes the workload on end-user systems, achieving scalability and error resilience. The system capability grows up following the overlay's peer

number, and this eliminates the need for central servers with great outgoing bandwidth.

In some P2P protocols the only exception to the totally decentralized architecture is the tracker peer. Indeed some P2P overlay use this particular node to keep track of the peers already present in the network. Peers query the tracker to obtain a list of other nodes to connect. The tracker does not participate in resource sharing or resource diffusion.

A P2P network overlay can be classified as structured or unstructured. In unstructured overlays peers' connections are not determined in any particular way. This means that when a peer joins the overlay network it connects to a random subset of other nodes, which addresses are provided from the tracker. Gnutella (70) and BitTorrent (11, 12) are examples of unstructured P2P network. In an unstructured P2P network a peer queries for the needed resource finding as many peers as possible. A drawback is that in unstructured protocols, due its random connection strategy, there is no warranty for efficient data allocation and message routing.

On the other side, in structured P2P networks peers are organized basing on particular rules and algorithms, forming and overlay with specific properties and topology. This overlay guarantees efficient message routing and provides upper bounds for resource finding. Distributed Hash Table (DHT) like Chord (49) and Kademlia (50) are examples of structured P2P overlay. This work manly focus on unstructured P2P overlay, so we do not deepen structured ones.

In recent years applications based on peer-to-peer protocols are experiencing a huge diffusion. The data flow originated by P2P applications characterizes most of the Internet traffic (81, 82). Indeed the P2P paradigm is well suited to implement a great variety of applications: video streaming, file sharing, distributed computing and others.

File-sharing is one of the most popular. The service provided by this application is to spread out contents such as multimedia and software. The size of resources ranges from several kilobytes up to some gigabytes. The best example of file-sharing architecture is doubtless BitTorrent. This protocol is considered today the state of art for generic content distribution and the strategies that it implements have been deeply investigated.

2.2 BitTorrent

Proposed by Bram Cohen, BitTorrent (12) is considered the state of the art for file-sharing protocol. Like every other P2P protocol its purpose is improving the dissemination of generic content among a network distributing the workload on the overlay. For the sake of clarity, before explain BitTorrent's strategies, we summarize the terminology related to the protocol.

- **MetaInfo file:** a file that contains information related to a particular document. The most relevant information is the tracker address, the file size and piece size. Usually it has ".torrent" extension.
- **Swarm:** the set of peers that cooperate to download a file.
- **Tracker:** a central server that stores the list of peers related to a particular torrent. It replies to client requests with a list of the addresses of the other peers in the swarm.
- **Seeder:** a peer that has completed the download but continues to share the file.
- **Leecher:** a peer that has not yet completed the download.
- **Piece:** the whole file is subdivided into pieces of roughly the same size, usually a piece has dimension of 2^{18} Bytes.
- **Block:** every piece is subdivided into blocks. These blocks are exchanged among clients and usually have dimension of 2^{14} Bytes.
- **Choked and Unchoked:** a peer can make a request to another one if this latter considers him in unchoked state. Peers do not reply to requests from peers in choked state. Choke and unchoke states depends on download and upload rates.

The protocol adopts a multi-part download scheme: the file is divided into several pieces, usually of about 2^{18} Bytes, that are shared among peers. These exchange blocks of 2^{14} Bytes at time.

A BitTorrent system uses two protocols: the *Tracker protocol* and the *peer-wire protocol*. The first is made by HTTP requests and is used when a client needs to connect to the overlay: a request is sent to the tracker that replies with a list of

peers that are actually in the swarm. The tracker address and a description of the file properties is stored in a metainfo file that the client has previously downloaded from the web. After a user has got the list, it tries to connect to a random subset of the obtained peers and starts to use the peer-wire protocol.

This protocol consists of many messages, but two are the main strategies employed: a request of pieces based on a local rarest first criterion and a tit-for-tat mechanism. The rarest first strategy assures an even distribution of the file pieces among the overlay forcing a peer to request the rarest pieces in his neighborhood. There are two exceptions to the rarest first strategy: at start up, when a peer has no piece at all, the piece to download is selected at random, and near the end, in the so called *endgame*, the requests for the missing blocks are sent to the entire neighborhood. This is necessary because it has been observed that the last pieces take a long time to download.

There are two different tit-for-tat mechanisms for seeders and leechers. A seeder sorts the neighbours by their download rates, unchoking the best ones. On the other hand a leecher unchokes the peers with the higher upload rates. In this way the protocol assures a good match between the download and the upload rates among peers and tries to maximize the bandwidth utilization for data dissemination. The download and upload rates are computed every *rechoke period*.

BitTorrent discovers new useful connections with a strategy named *optimistic unchoke*: every three rechoke periods one peer is unchoked at random from the neighborhood. Although quite standard, some parameters of the protocol, such as the peer list size or the length of the periods of *rechoke* and *optimistic unchoke*, can vary in different implementations, but this does not affect the overall performance of the protocol and his strategies. Using the first policy a peer always requests the rarest piece in his neighbourhood, contributing to replicate pieces that are difficult to found, while the latter assures reciprocity between downloading and uploading rates, indeed a peer sends data only to his best uploaders.

Its widespread diffusion and many research studies (see for instance (62), (63), (52)) have shown that the two main BitTorrent algorithmic solutions, namely tit-for-tat and rarest-first achieve optimal performance in stable network conditions. Also if the BitTorrent protocol was originally developed for file-sharing, some attempts have been made to adapt it for IPTV applications.

2.3 IPTV

IPTV applications aim at the provision of high-quality real-time video streaming. The popularity of these applications is rapidly growing thanks to the widespread adoption of large bandwidth access links by residential customers.

The first attempts of deploying IPTV applications were made by ISPs exploiting IP multicast functionalities in small-scale environments through proprietary solutions based on the traditional client-server model of Internet protocols and services. The limitations of this approach are many: due to the limited deployment of IP multicast in the Internet the distribution of contents is often confined to the network of each ISP only. Furthermore, poor scalability, lack of authentication and security mechanisms, and difficult integration with hierarchical routing, makes this solution unlikely to provide the infrastructure for future massively deployed IPTV.

Recently, a different approach has emerged: IPTV applications are developed based on the popular P2P paradigm. This model for application development requires each participant to send data to one another using application-level multicast on overlay topologies. This approach has proven to be successful as witnessed by the steadily increasing number of applications that are nowadays available (1).

A popular example is PPLive (2). It is a P2P streaming platform that distributes both live and pre-recorded contents. According to the PPLive web site (2) in January 2008 the PPLive application provided almost 500 channels with 1,000,000 daily users on average. The number of channels in December 2008 was reported to be equal to 1775. These data confirm the popularity of IPTV application nowadays.

2.3.1 BitTorrent for video-streaming

There are in the related literature few attempts to adapt the BitTorrent protocol for video-streaming purposes. The main difference between a file-sharing scenario and the video-streaming one is that in this latter the network behavior can be very variable. Peers join and leave the network with high rates, differently from file-sharing where the peers often remain connected to the overlay for the whole download time. In addition in live-streaming case, because the multimedia content is generated in real-time, only few pieces are available at a time. This because the oldest video parts are useless. In this case the BitTorrent's rarest first policy is strongly penalized due to

the reduced diversity of available file parts. Many works concentrate their efforts on video-on-demand applications.

All (31, 32, 33, 34) share very similar approaches: the intent is to modify the rarest first policy and the tit-for-tat mechanism in order to make them suitable for video-on-demand applications. (31) and (32) modify the rarest first policy to require the pieces that are necessary in the immediate future for the playback, while the papers (33) and (34) extend the tit-for-tat mechanism; in (33) the peers are ranked on forwarding rates, i.e., the capability that a peer has to serve the other peers, in place of standard download and upload rates. Also mixed approaches has been proposed: (29, 30) maintain a conventional streaming server and to assist it with the BitTorrent overlay: a request is first sent to the peer-to-peer network then, if the request cannot be satisfied, it is forwarded to the streaming server. (30) modifies the rarest first policy in order to give higher priority to the pieces near to the playback point. Tripler (35) is a modification of the protocol proposed for live-streaming. Similar to the previous works, it proposes a modification of the rarest first strategy reducing the downloading set to an interval around the playback time.

2.4 The use of network coding in P2P applications

Network coding is a technique that combines several packets together for transmission instead of simply relaying them. Figure 2.1 show the so called butterfly network, that is a simple example that clarifies this concept. There are two sources in the top of the figure and two destination nodes in the bottom. Figure 2.1 shows how using a very simple code we can forward both A and B messages to both destinations simultaneously. Indeed we encode the $(A+B)$ message and send it to the central channel. Both recipient nodes receive this message besides the A message, for the bottom-left node, and B message, for the bottom-right one. Now both the original messages can be retrieved simply computing respectively $B = (A+B)-A$ and $A = (A+B)-B$. It is not possible achieve the same performance using unicast routing, because no routing strategy can transmit both the A and B messages to destination at the same time.

This simple example illustrates how using coding techniques it is possible to obtain better performance then simply relaying. The code used in the example is linear because both the encoding and decoding schemes use linear operations.

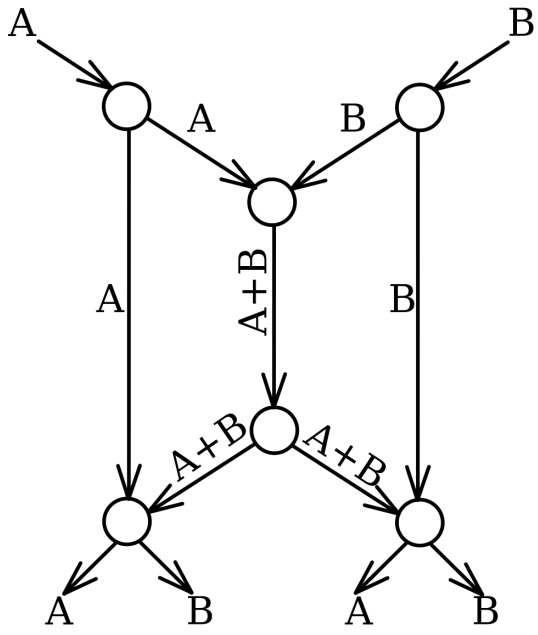


Figure 2.1: Butterfly network

The linear network coding problem can be generalized as: $X_k = \sum_{i=1}^{n} c^i M_i$ where X_k are the coded packets generated from the original messages M_i, that are combined using the c^i coefficients. These coefficients are chosen from the Galois field $GF(2^s)$. Each node in the network forwards the computed value X_k and the destination peer can retrieve the originals M_i solving $X = GM$. This is possible when $N \geq n$ packets are received.

In (75) is proposed the Random Network Coding technique. This is a linear code where nodes choose random the coefficients used to combine the original packets. These coefficients are forwarded with data and receivers are able to retrieve the original message when obtain enough linear independent combinations.

Network coding initially appeared as a potential breakthrough for application level multicasting and content distribution networks. Theoretical results showed how network coding can asymptotically achieve the maximum capacity of a network (51), (64). However, such important results until now have never become popular in practical P2P applications. At the same time, a large body of literature has focused on the analysis of potential benefits of coding techniques for content distribution. To maximize band-

width utilization and error resilience some attempts have been made to extend P2P content distribution applications with coding techniques.

In particular, network coding has been the subject of several studies: conclusions ranged from very optimistic (early results pointed out that throughput of the system exploiting network coding at the block level is two to three times better than the non-coding system (58, 59, 60)) to rather pessimistic (systems using network coding perform even worse than BitTorrent (69)) to inconclusive (both throughput and its theoretical upper bound overlap in coding and non coding systems (66, 67)) depending on the type of coding technique and on the analysis methodology employed by different researchers.

One of the most relevant results so far is represented by the Avalanche proposal (60), (59), (58) that suggests a protocol based on exclusive use of network coding. In Avalanche a file is divided into blocks and distributed by using random linear network coding. A peer sends to its neighbors a linear combinations of the original blocks using random generated coefficients. These need to be selected in a field of relevant size, e.g., 2^{16} bits, and are included in the block header to allow the receiver to solve the corresponding linear system of equations and retrieve the original file blocks. The use of coding coefficients with 2^{16} has a significant cost in terms of additional computational power. The use of coded blocks simplifies the issue reconciliation and the search of rare blocks in the overlay, since after few hops both rare and popular blocks turns to be combined making very likely hat each block is useful for a receiving peer. The original file can be recovered when enough pieces are received.

Avalanche use random network coding to generate coded blocks from the original file. In (60) simulation results show that Avalanche can achieve a 30% gain in performance respect to a generic P2P protocol. A prototype implementation and some experiments are presented in (59), where network coding is applied to different file segments independently.

However all the results presented in (59, 60) are not compared with BitTorrent, that is considered the reference protocol concerning file-sharing. Instead the generic P2P protocol used for comparison has 4-6 neighbors, while BitTorrent in the real world typically uses 30-50. Moreover the simulation does not look at other variables such rarest first policy or endgame mode. Finally in (60) all the memory and calculations required by random network coding are ignored and have no impact on protocol performance. Therefore, the results presented cannot be considered conclusive.

Many attempts have been made trying to integrate network coding also in BitTorrent; in (61), (45) the use of coding is exploited to avoid some scenarios in which the BitTorrent's local rarest first strategy fails, resulting in the peers starvation.

Some works (47, 48, 53) proposes, instead of network coding, protocols based on Luby-Transform (LT) (36) codes. LT belong with the family of erasure rateless (also called fountain) codes. These are forward error correction (FEC) code, that transform a message of n symbols into a longer message of k symbols, with $k \geq n$. The original message can be retrieved when $k^{'}$ messages are received. The ratio k/n is called the code rate and k'/k reception efficiency. The LT codes are rateless because it is possible generate a potentially infinite sequence of coded symbols. The original message can ideally be recovered from any subset of the encoding symbols of size equal to or only slightly than the number of source symbols.

Rateless codes are particular useful for erasure channels, that are noisy transmission channels, where a message could be delivered with some errors. More formally, in information theory, a Binary Erasure Channel (BEC) is a channel that transmits binary symbols, 0 or 1, and has a certain error probability. On a BEC a message is delivered with errors with a probability p and without error with a probability $1 - p$. BEC is useful to model many real world situations. Indeed files that travel across the Internet are subdivided into packets that could be delivered with errors or not delivered at all. Usually the communication methods over such channels provide constant feedback from the receiver. If some messages are lost or damaged, they are retransmitted. The drawback is that acknowledgements are wasteful specially on very noisy channels or some networks could not have a feedback channel. Thanks to the property of fountain codes in general, and LT in particular, it is possible eliminate acknowledgements using a one-way protocol. It is used only one feedback message, when the receiver has decoded successfully the original data.

In (47) is proposed rStream, a protocol for streaming video content based on LT codes. In (47) the whole file is encoded and coded blocks are exchanged among peers. A peer must wait for the recover of the original data before relay packets on the network. This introduce a delay in the content's spreading. In (48) some preliminary ideas on the distribution of LT coded blocks with application to the P2P video streaming are presented without the design of a complete protocol. Many of the cited works do not compare their performance with the BitTorrent protocol.

In (53) provides simulation analysis of a protocol based on network coding, but the results are inconclusive. Moreover, according to results presented by the authors, the overall performance is worst than BitTorrent. In (61) a study on the distribution of file's pieces that arise in the overlay using a rarest-first strategy is presented. The paper points out that this latter can lead to bottleneck when all peers try to download the same piece (i.e., the rarest piece). Then, a protocol based to use of source coding in an attempt to bridge these gaps is proposed. A similar approach is described in (45).

In conclusion, many attempts have been made trying to conjugate coding techniques and P2P protocols, but until now the results are inconclusive, and further research is needed to understand the potential benefits.

2.5 The impact of P2P protocols on Internet Service Providers (ISPs)

P2P protocols, both for file-sharing and video-streaming, may generate a huge amount of network traffic (see for instance (81) and (83)). Moreover, the participating nodes in the overlay are largely agnostic of the underlying IP network. This increases the cost associated with P2P traffic for individual Internet Service Providers which face the problem to manage a vast amount of unnecessary inter-domain traffic (82). Such traffic level can cause congestion on valuable inter-ISP links and more generally an overall performance degradation in an ISP network.

There are several negative effects that derive from the topology mismatch between the P2P overlay and the underlying IP network layer (see for instance (86)). As example a measurement study of PPLive reported that the 70% of the traffic observed by tagged peers deployed in North America was originated by peers located in Asia (81). In this situation a policy ISPs aware, that aims to maximize connections in the same country, can reduce the more expensive intercontinental traffic.

Several works argue that network awareness leads to improvements. (91) shows that by exchanging information about network topology both ISPs and P2P systems could optimize their operations: ISPs could improve the utilization of the network resources, while P2P systems could achieve better system performance. In (93) is proposed a solution that aims to minimize the AS op count for a message between source and destination. They also modify the BitTorrent system to add locality-awareness into

neighbor selection, peer choking/unchoking, and piece selection. An interesting results is that while the work clearly shows the advantage of locality-aware solutions at reducing inter-AS traffic and achieving shorter downloading time, there are also deficiency in the distribution of the peer workload respect to the traditional strategy employed by BitTorrent.

In literature many other practical proposals aim to reduce the inter-ISP traffic. The paper (80) proposes to cache P2P contents by using ISP-controlled proxy-caches. Theoretically this is an interesting solution because helps to segregate P2P traffic inside the ISP network, but due to legal risks for unauthorized exchange of copyrighted contents, this solution is not applicable.

A different approach in (76) proposes the use of an "oracle" provided by the ISP that helps a peer to locate neighbor peers. In (89) has been proposed the use of an ISP-controlled tracker that provides network related information such as topology, traffic engineering information, and so on.

The works (88) and (84) study the need to localize P2P file sharing traffic within the ISP boundaries. The paper (88) proposes an overlay structure composed by mTrackers aiming at minimizing transit ISP costs. The paper (84) describes an ISP-friendly variant of BitTorrent to reduce the cross-ISP traffic.

An approach to improve inter-AS routing is geographically informed routing . The most relevant work in this direction is the Geographically Informed Inter-domain Routing (GIRO) (120) protocol. The protocol tries to improve performance including the geographical information of the routing nodes. A similar approach is presented in (118), another inter-domain routing protocol that propose some heuristics to reduce the geographical path between ASes. However results are not conclusive, indeed in (118) the strategy proposed is compared with the Border Gateway Protocol protocol but does not demonstrate great improvements.

Some approaches during the last years, in the P2P literature, show that the game theory assumed an important role in modeling strategies used by the peers of a P2P system and several papers on this topic have been published. In (77), (78), and (79) game theory models have been used for the investigation of peer interactions and to analyze incentive mechanisms for P2P file sharing systems.

The work presented in (85) studies the interaction among peers in P2P streaming applications, illustrates the existence of Nash equilibrium points, and introduces

renewement criteria to help the selection among these equilibria.

Our opinion is that to find effective network-aware strategies, is essential to work on modeling the interaction between P2P overlays and ISPs. Many works in literature aim to model P2P traffic, but only few evaluate the P2P overlay in relation with the underlying AS topology.

Some models try to describe the resource diffusion process in the overlay, also referring to specific protocols, like BitTorrent (23),(22) or eDonkey (24). In (23) is proposed a fluid model for resource diffusion in the BitTorrent protocol. The model is able to study steady-state performance measures, such as the number of peers that have a resource and remains in the system to allow its diffusion. A similar approach is used in (28), that describes P2P dynamics through a set of second order fluid equations and derive results related to the resource distribution among peers.

One model presented in (22) uses dynamic analysis to study the variations in terms of popularity of individual files, and in terms of the number of available files at individual peers. Also this work considers how a document is spread to the requesting peers into a BitTorrent environment.

A different approach is use simulation techniques to understand protocol dynamics. (24) proposes a simulation for investigate the diffusion of a file in an e-Donkey system, as function of several parameters such as sharing probability and requests arrival rate. The main drawback related with this kind of approach is that often it do not scale very well for overlay with many peers. In (25) is presented an analytical model based on biological epidemics. Also here the model is used to predict the diffusion of single files in a P2P network.

An attempt to develop a mathematical model for topology awareness of overlay networks is described in (94). The work builds a mathematical framework capable to mach as possible the P2P overlay with the ASes topology. The main drawback is that it do not consider commercial accords and how these affect the routing paths of messages. In (92) is presented a model of interaction between P2P and ISPs. This work take into account peering and transit agreements, but the examples are limited to a topology of only two ISPs, and complex inter-AS routing is not modeled.

A great limitation of the existing P2P models is that they do not take into consideration economical and commercial factors and do not regard the underlying IP network, but mainly concentrate on the P2P overlay. Also when commercial agreements are

considered, model the complex inter-AS routing is a difficult task. We consider model interaction between ISPs and P2P overlay network and develop effective ISPs-aware strategies hot research topics nowadays, due the popularity of P2P applications and the economic implications, and we expect many research efforts will make in this direction in the near future.

Chapter 3

A case study: PPLive

In this chapter we start our investigation focusing on a popular P2P-IPTV application, PPLive. PPLive provides users with hundreds of TV channels and this made it an interesting case study in order to understand P2P protocols dynamics. We describe a passive/active measurement study regarding the characteristics of the PPLive overlay networks.

Our contribution is the refinement of the crawling strategy developed in [7] to confirm results presented therein as well as to discuss on the accuracy of active measurements aimed at the characterization of overlay network topology.

All data collected in this chapter are useful in order to understand P2P overlay dynamics, their weaknesses and identify possible protocols optimizations

3.1 The PPLive application

PPLive is a mesh-pull P2P streaming platform that distributes both live and pre-recorded contents. According to the PPLive web site (2) in January 2008 the PPLive application provided almost 500 channels with 1,000,000 daily users on average. The number of channels in December 2008 was reported to be equal to 1775. The bit rates of video programs mainly range from 250 kbps to 400 kbps with a few channels as high as 800 kbps. The PPLive is a proprietary system, which means that neither the source code nor protocols are available. The following description of the behavior of PPLive is derived by our experiments and by previous investigation that use passive and active measurements (see for instance (6), (4), and (7)).

The PPLive platform consists of multiple overlays. A single overlay corresponds to a PPLive channel. Each peer in an overlay is identified by the pair (IP address, port number).
Figure 3.1 shows the basic actions of a PPLive peer: *(1)* channel list download from the *channel list server* (via HTTP); *(2)* for the selected overlay (channel) the peer collects a small set of peers involved in the same overlay by querying the *membership servers* (via UDP); *(3)* the peer communicates with the peers in the list to obtain additional lists which it aggregates with its existing peer list (via UDP). In this manner the peer maintains a list of other peers involved in the same overlay (i.e., watching the same channel). During the streaming the peer periodically executes steps (2) and (3) to collect neighbors peers candidate.

3.2 Methodology

A common approach for measurement based investigation of P2P systems is to capture snapshots of the overlay network using a crawler, i.e., a program that collects information on the topological properties of the overlay network. The development of a crawler requires the knowledge of protocols used by the application under study. In the PPLive case this is a challenging task because this application is not open source and its details are not public. In the following we first provide a short description of what is known about the PPLive protocols and then we describe the design of our crawler.

3.3 The crawler

As discussed PPLive is a closed source application whose details are unknown. To overcome this difficult we use the technique presented (7) that allows to collect the query strings that the peers use in steps (2) and (3)[1].
Unfortunately, this is not the only problem to solve to obtain accurate data from the crawling experiments. In fact, according to the studies of the Gnutella P2P file sharing application presented in (3) and (8), the accuracy of captured overlay snapshots depends

[1] Unfortunately, the number of PPLive channels that can be investigated using the method described in (7) is decreasing, this is probably due to changes in internal protocols used by PPLive.

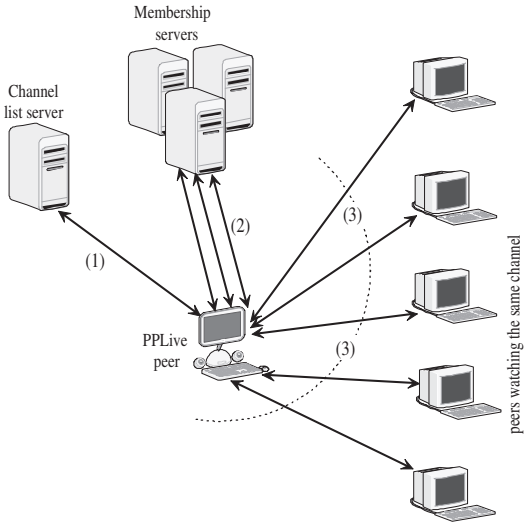


Figure 3.1: PPLive basic architecture

on the crawling speed. In principle, a perfect snapshot is captured if the crawling process is instantaneous and complete, i.e., if all peers are contacted. However, neither of these conditions is generally met for the following reasons: rapidly changing topology, and unreachable peers.

To obtain accurate overlay snapshots, we used a distributed crawler architecture, similar to the one described in (3). The PPLive crawler employs a master-slave architecture to achieve a high degree of concurrency. For our experiments, we used 5 PCs that run 1 master and 200 slaves. The master process performs step (2) of Figure 3.1 to collect the initial list of peers and then coordinates multiple slave processes. In particular, each slave receives peer addresses from the master according to the slave crawling speed and performs the step (3) of Figure 3.1. The slave processes are independent crawlers that probe different portions of the overlay. Moreover, to increase the crawling speed each slave process is composed by a set of threads that independently send queries to PPLive peers. Periodically, each slave process reports back to the master process with the peer neighbors.

Unresponsive peers make the snapshots less accurate. Peers may not respond to queries from our crawler for several reasons. If a peer is temporarily overloaded it might later respond if we adopt a high timeout value. On the other hand, a too high timeout

may introduce long delays.
We performed a set of preliminary experiments to derive a timeout value to trade-off the increase of the number of responding peers and the decrease of long delays. In our experiments we set the timeout equal to 5 sec.
A related issue concerns the difficulty in obtaining the neighbor list. In particular, as observed in (7) and confirmed in our experiments, successive queries sent to the same peer may yield slightly different list of neighbors. We performed a set of experiments to derive a value for the number of successive queries that represents a trade-off between the need to obtain the peer's neighbor list and the necessity of having short crawling time.
In our experiments we set the number of successive queries sent to the same peer equal to 10. Furthermore, we address in more detail the "completeness" of the peer's neighbor lists in Section 3.3.1.
With the optimizations we use the average duration of each crawling experiment is 106 sec, with an almost negligible standard deviation.

3.3.1 Crawling "completeness" and passive/active measures

In Section 3.4 we provide results on the amount and type of data exchanged by a controlled PPLive client whose activity is traced by a sniffing packet tool. The data collected during these passive measurements are exploited to define the concept of neighbor and to compute the intersection between data obtained by the active crawling and passive monitoring. This latter task permits to evaluate the completeness of the active measurements.
Furthermore, active measurements allows us to report on the topological properties of sample overlay networks and on the time behavior of peers joining these channels.

3.4 Passive measurements

For the passive measurements we use a PPLive running on a non-natted LAN node. We only capture UDP traffic on the known PPLive port number. In the first experiment we capture the in/out traffic of the PPLive node watching a cartoon channel. In Figure 3.2 the packet size histogram of the captured packets during a session of 1500 s

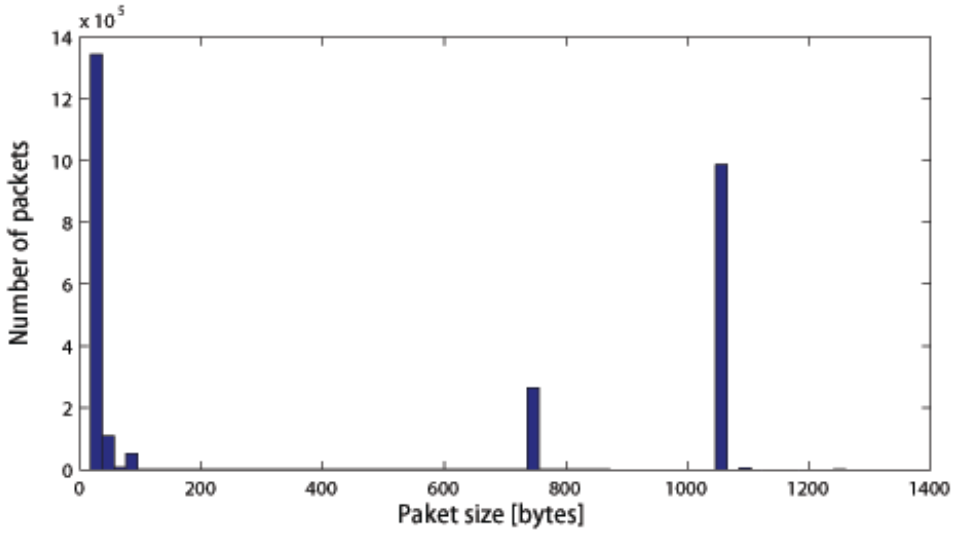


Figure 3.2: PPLive packet size histogram

is shown.

Three packet types can be identified: < 100 bytes (signaling, class A), around 800 bytes (to be identified, class B), > 1000 (video data, class C). During the same session we observe an overall download rate of 616 kbps whereas the upload rate reaches 6082 kbps. Table 3.2 shows the partial rates obtained considering the three packet classes identified in Figure 3.2. It is very likely that class C represents video data encoded at 350 kbps. Class A is probably to be referred to the signalling for the overlay network management and control (neighbor lists); it is worth pointing out that such control traffic is larger in the downlink than in the uplink. Class B deserves further attention; it can be noticed that the class B traffic represents a constant fraction of the class C rates, i.e. video packets. In particular, class B amounts to the 18% of the video traffic in both the downlink and uplink. This allows us to infer that class B carries data which are strictly correlated to the video; these are likely to be the buffer maps, which are exchanged among peers in a fixed proportion with respect to the video rate.

In Figure 3.3 and Figure 3.4 the rate from and toward the different peers is shown; each peer is mapped onto a unique positive integer identifier reported on the x-axis; on y-axis positive are used for the uplink, whereas negatives refer to the downlink rates,

Table 3.1: Rates [kbps] for the packet class A, B, C.

Packet class	A	B	C
uplink	42.79	964.98	5116.81
downlink	196.51	64.76	355.07

measured in kbps. For sake of clarity, the same data are plotted in Figure 3.6 and Figure 3.7 where upload and download bandwith are sorted in ascending order. It can be noticed that class A rates are far more uniform with respect to class B, C. This is due to the fact that control packets evenly spread around the overlay. On the contrary, video and buffer maps traffic is more peaked around few peers which the probed node is collaborating with. In the observed time window about 2000 peers are contacted by our node. Nevertheless, the number of active neighbors is rather limited; as an example, by considering only those peers exhibiting class B/C rates larger than 1 kbps, one retains only 15% of the observed 2000 peers. The same behavior is shown in Figure 3.5, where the uplink/downlink rates are sorted in descending order.

3.5 Combination of active and passive measures

The goal of crawling experiments is to obtain a snapshot of the overlay topology that could also be used to derive a graph representation of the overlay network.
To this end, peers are queried to provide the list of their neighbors. Who are these neighbors? We analyzed the trace obtained from the experiment described in Section 3.4 and we computed the number of neighbors of our controlled PPLive client. The concept of neighbor is defined as follows: a peer u is a neighbor of our controlled PPLive client if and only if in a time window starting at time t and lasting for Δ seconds at least one packet exchange is reported. Clearly, this definition yields different concepts of neighbor depending on the direction of the video packet exchange and on the class of the packet being exchanged (class A, B or C). Furthermore, this definition is dependent on both t and Δ. In Figure 3.8 the number of neighbors is reported versus time for $\Delta = 15$ seconds by considering only class C packets exchange and for all directions of packet exchange. It can be observed that after an initial transient time interval, where the cardinality of the neighborhood is rapidly growing, the measures reach a steady

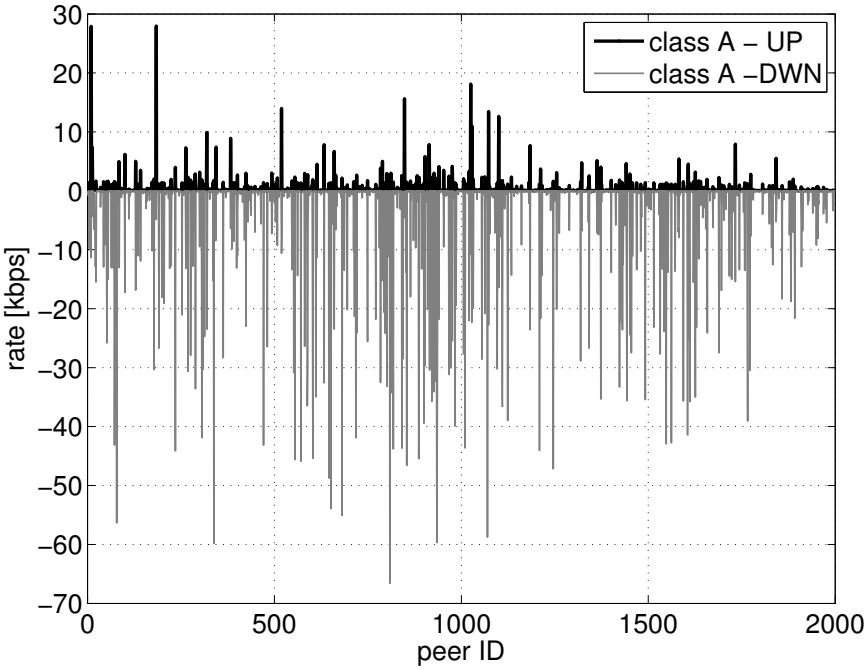


Figure 3.3: Class A rates [kbps] per peer.

state where slight oscillation around an equilibrium value are noted.
In order to compare active and passive measurement results we performed the following experiment. With the same settings described in Section 3.4 we started a crawling of the cartoon channel overlay network after a time interval that allowed our controlled PPLive client to reach the steady state. At the end of the crawling we computed the intersection between the list of neighbors obtained from the crawler for our controlled client and the list of neighbors we compute from the passive measurement. We consider different values for Δ $(10, 20$ and 30 seconds) and throttle download and upload rates of our PPLive client by means of the tool *NetLimiter* (9). We consider neighbor as defined for each of the packet classes (A, B, and C).

From the results of Table 3.2 and Table 3.3 we can observe that (i) the best overlapping among the passive and active measures occurs for packets belonging to class C; (ii) the overlapping decreases as the bandwidth increases, and in this case the upload bandwidth has a stronger impact; (iii) a measurement strategy for PPLive that uses a crawler yields only partial information about the actual overlay links. Therefore active

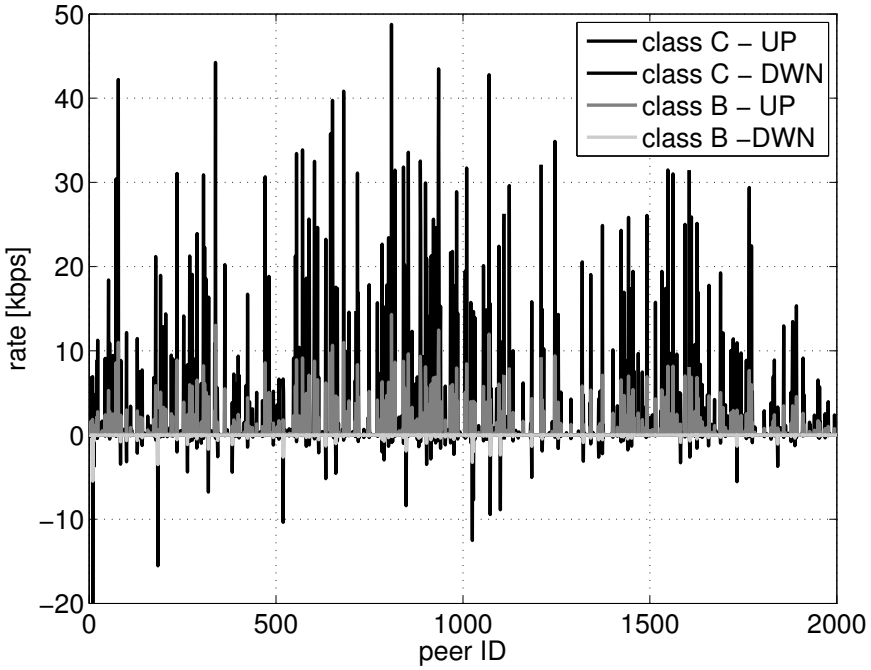


Figure 3.4: Class B-C rates [kbps] per peer.

measurements must be used with care in order to infer overlay features e.g., topological characteristics.

3.6 Active measures and topological properties

As we observed in the previous section the accuracy of the crawling experiments depends on the type of bandwidth connectivity and the quality of results decreases as the bandwidth increases (and this effect is more evident for the upload bandwidth).
On the other hand the accuracy of the experiments also depends on the percentage of non-responding peers, those whose identity has been discovered during the exploration of the overlay network (i.e., from another peer or from the membership servers) but that deny (or excessively delay) their reply to a request from the crawler.
In the following we present results obtained from a group of successive measurements of the same PPLive overlay channel (in particular we focus the attention on a cartoon channel) acquired at a constant rate. The duration of this battery of snapshots derives

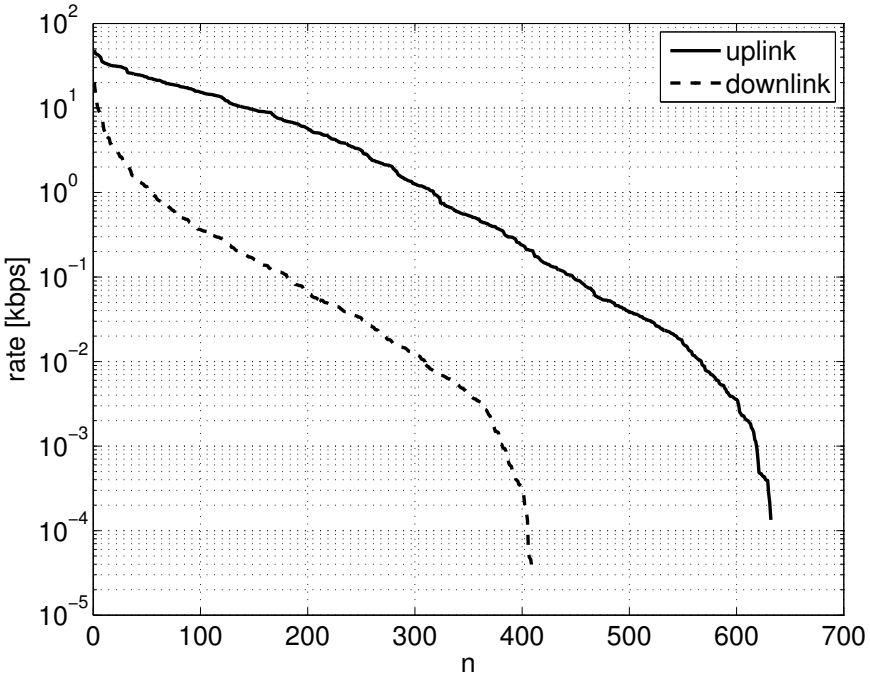


Figure 3.5: Class C sorted rates [kbps] per peer.

from the crawling time of each snapshot and from the constant time interval between successive acquisitions. We identify each snapshot with a progressive number (ID). These crawling experiments were collected between 11/02/2008 and 11/11/2008; during this period we collected 7670 snapshots of the PPLive overlay channel under study. The average duration of the crawling experiments was 106 sec (with a small standard deviation). The plots of Figure 3.9 show a typical daily behavior.

From these experiments we found that the fraction of responding peers varies from the 83% to the 52%. We cannot discriminate excessive long delay from un-reachable peers but with a time out chosen to minimize these mis-classifications we found the fraction of non-responding peers is mostly due to un-reachable peers (probably NAT-ed peers).

In Figure 3.10 we plot the degree distribution computed from the collected snapshots. In particular, we consider two PPLive channels, one the cartoon channel that we previously described (channel A), and a news channel (channel B).

To be representative of the "real" overlay PPLive topology the results depicted in the previous figure should be "corrected" and according to the considerations summa-

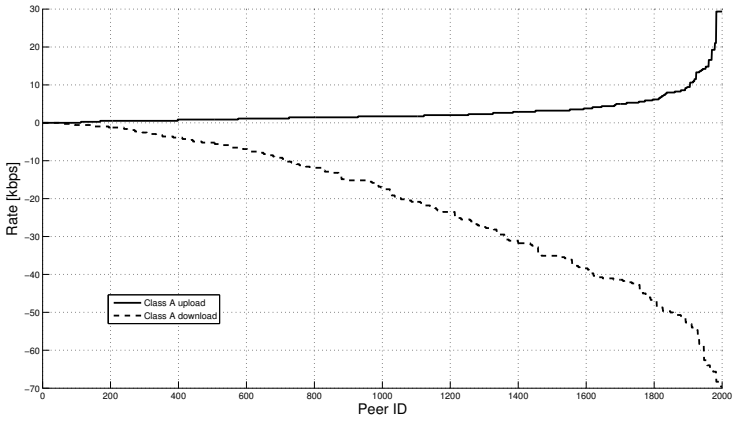


Figure 3.6: Class A sorted rates [kbps] per peer.

rized in Table 3.2 and Table 3.3 the "corrections" should be focussed on the tail of the degree distribution.

3.6.1 Analysis of user sessions

We also studied the length of users connection to PPLive overlay, taking the battery of consecutive snapshots we introduced in previous section. A continuative session is characterized by the presence of the same node, identified by his IP address and UDP port, in a set of continuative snapshots. Every snapshot has an average duration of 106 sec, in which a peer is considered active and connected. A connected session is characterized by the presence of the same IP address and UDP port in a set of consecutive snapshot. The lengths of connected sessions have been already studied by using a similar approach in (7). We obtain results similar to those presented in that paper i.e., the duration of sessions on a PPLive channel (we perform the same experiment for different channels) can be approximate by means of an exponential distribution (or geometric as reported in (7)). In particular we plot in Figure 3.11 the session length distribution computed on the battery of 7670 snapshots. In 3.12 we plot the distribution of peer session length obtained from an example snapshot on logarithmic scale.

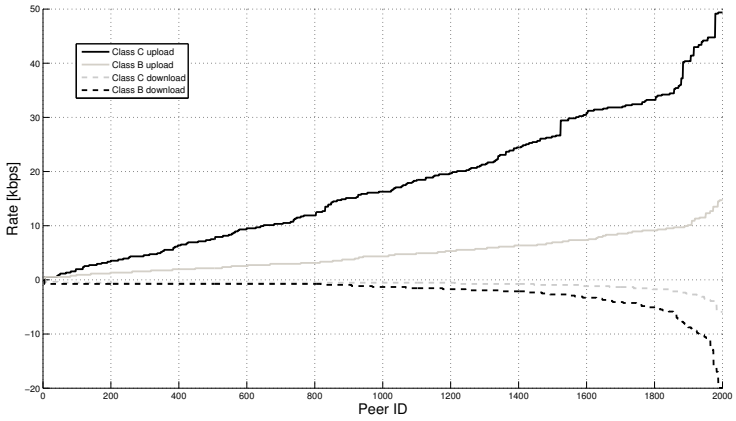


Figure 3.7: Class B-C sorted rates [kbps] per peer.

3.7 Conclusions

We produce a set of measurement useful to describe the protocol behaviour and all data collected in this chapter are useful for understanding the dynamics of P2P protocols, weaknesses and identify potential optimizations.

As a sort of validation we re-derived results on the length of users connection to PPLive channels: the preliminary results we obtained confirm those in (7) i.e., the session lengths can be represented by memoryless distributions (geometric or negative exponential). The main contribution of the analysis presented is the study of the limits and the completeness of active measurements acquired crawling the PPLive overlays.

The presented results showed that a simple crawling strategy is not able to get accurate snapshots of the PPLive. Moreover we also point out the relations between bandwidth (upload and download) and the crawling accuracy.

The lesson that we can draw is that the derivation of accurate topological information for PPLive (and possibly for similar P2P-IPTV applications) requires the setting of a more sophisticated measure methodology.

We describe peers' behaviour and overlay topology. The closed source nature of the software is a great limitation, but with our methodology we have been able to collected data useful for understanding interesting dynamics of P2P streaming protocols.

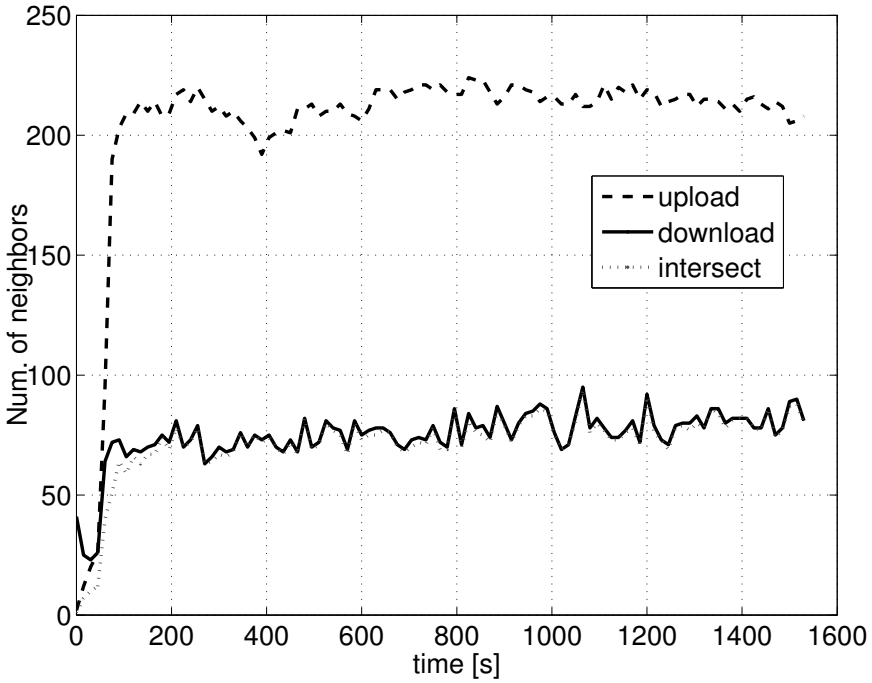


Figure 3.8: Number of neighbors versus time for $\Delta = 15$ seconds.

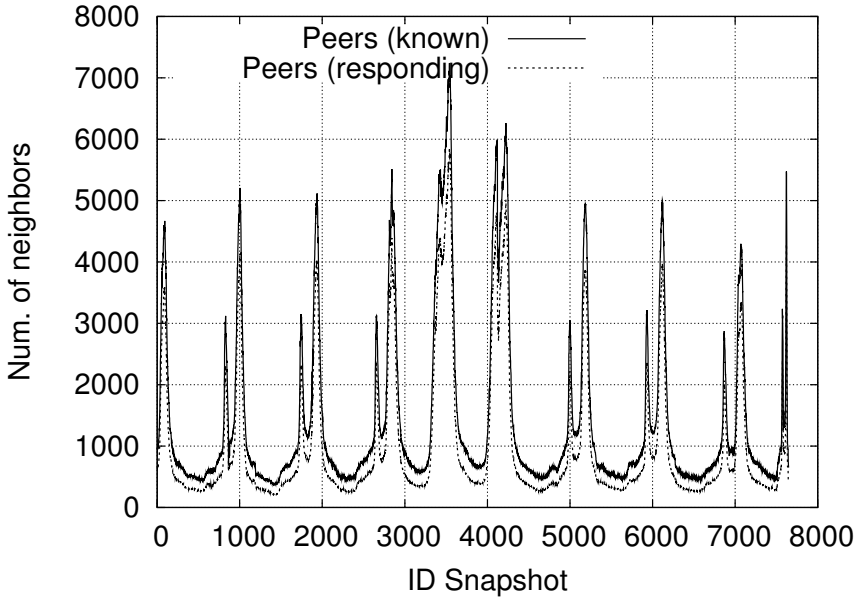


Figure 3.9: Evolution of peer population over nine days

Table 3.2: Intersections between neighbors list and sniffed traces.

Download/Upload bandwidth, Δs	Class A Intersection sniffed/crawler	Class B Intersection sniffed/crawler	Class C Intersection sniffed/crawler
1Mb/0.25Mb, 30s	7%	50%	78%
1Mb/0.25Mb, 20s	9%	50%	74%
1Mb/0.25Mb, 10s	13%	50%	73%
1Mb/0.5Mb, 30s	15%	33%	88%
1Mb/0.5Mb, 20s	23%	33%	91%
1Mb/0.5Mb, 10s	39%	33%	90%
1Mb/1Mb, 30s	5%	$< 1\%$	35%
1Mb/1Mb, 20s	7%	$< 1\%$	36%
1Mb/1Mb, 10s	9%	$< 1\%$	37%
4Mb/0.25Mb, 30s	12%	25%	61%
4Mb/0.25Mb, 20s	13%	25%	61%
4Mb/0.25Mb, 10s	20%	33%	61%
4Mb/0.5Mb, 30s	11%	10%	59%
4Mb/0.5Mb, 20s	15%	34%	60%
4Mb/0.5Mb, 10s	19%	$< 1\%$	63%
4Mb/1Mb, 30s	11%	50%	55%
4Mb/1Mb, 20s	13%	50%	54%
4Mb/1Mb, 10s	18%	50%	54%
Lan, 30s	10%	40%	40%
Lan, 20s	16%	40%	44%
Lan, 10s	12%	40%	44%

Table 3.3: Active measures on our nodes: number of neighbors with different bandwidths

Down/Up band.	Node degree
1Mb/0.25Mb	93
1Mb/0.5Mb	94
1Mb/0.5Mb	102
4Mb/0.25Mb	102
4Mb/0.5Mb	115
4Mb/1Mb	115
Lan	121%

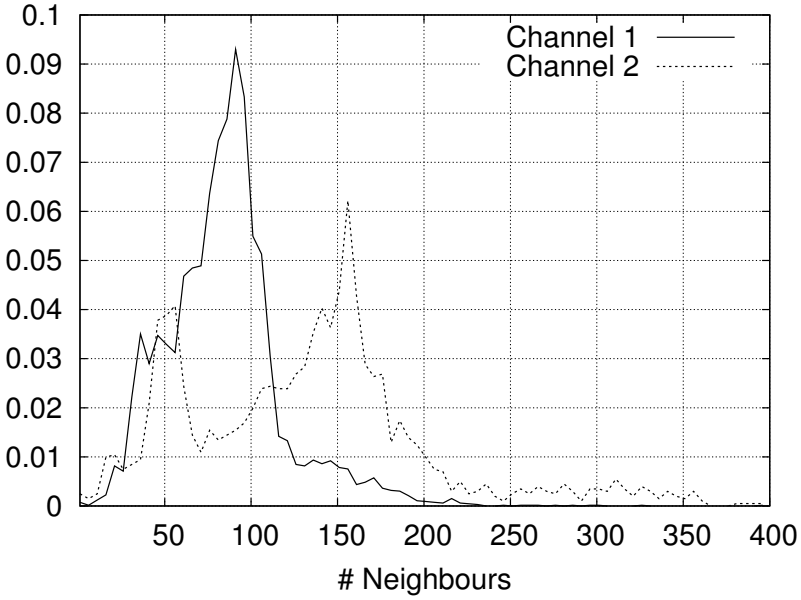


Figure 3.10: Peer degree distribution (for two PPLive channels)

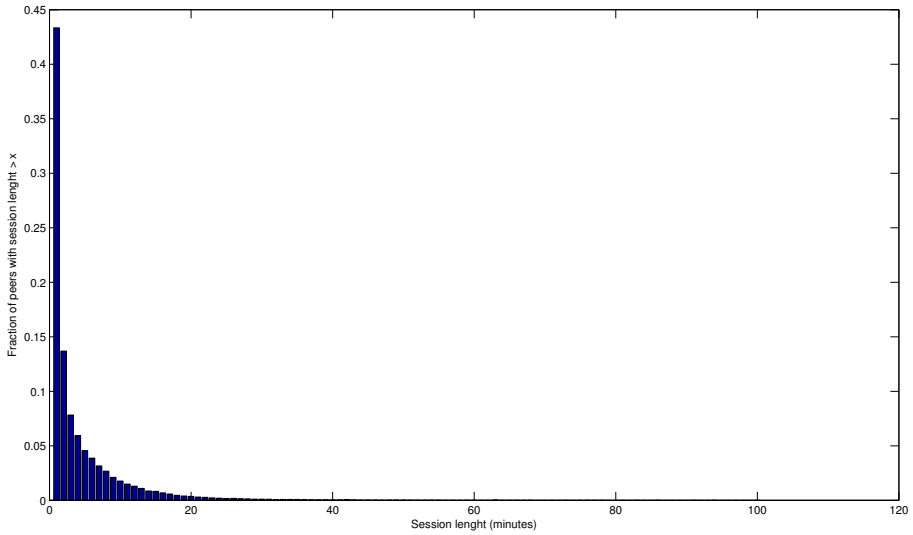


Figure 3.11: Peer session duration

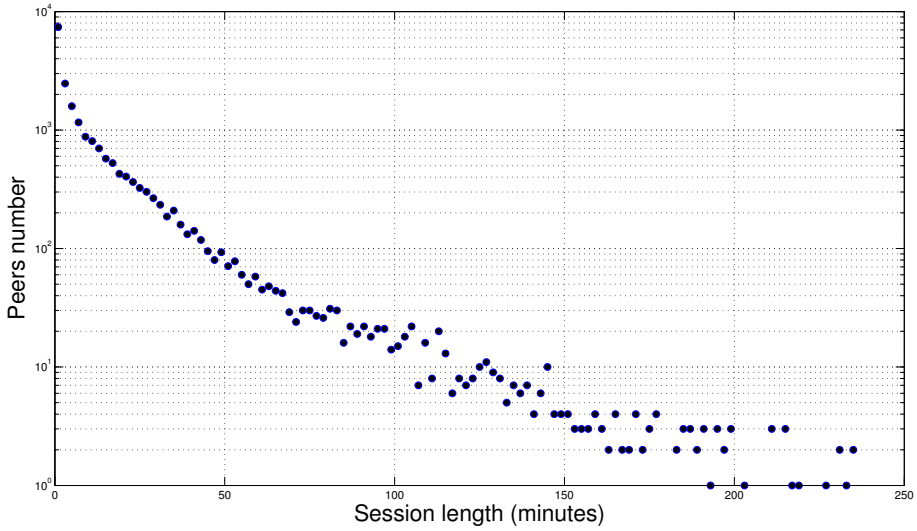


Figure 3.12: Peer session duration on log-scale

Chapter 4

Improving BitTorrent using Luby-Transform (LT) codes

In this chapter we propose a modification of the BitTorrent protocol with the purpose to enhance it. We use a particular class of rateless codes i.e., the LT codes.

Later on, by using a simulation based study, we compare the performance of the modified BitTorrent against the standard version of the protocol. Here we give a description of LT codes and describe the proposed protocol. Then we describe both simulation methodology and the different scenarios that we set up to prove its effectiveness. Finally experimental results show that in particular network conditions our protocol performs better than the original one.

4.1 The Luby-Transform codes

LT codes are a class a fountain codes invented by Michal Luby that use simple XOR operations to encode and decode the message. They belong with the class of rateless erasure codes, and they have the property to generate a potentially limitless sequence of encoded symbols, differently from standard erasure codes that have a fixed rate. An erasure code transforms a message of n symbols into a longer message of k symbols, with $k \geq n$. The original message can be retrieved when $k^{'}$ messages are received.

LT codes are particular useful for erasure channels. A message sent on an erasure channel is affected by errors with a certain probability. The standard communication methods over such channels provide constant feedback from the receiver, so lost or

damaged messages could be retransmitted. But acknowledgements could be wasteful specially on very noisy channels or some networks could not have a feedback channel. Thanks to the property of fountain codes in general, and LT in particular, it is possible eliminate acknowledgements with a one-way protocol. It is used only one feedback message, when the receiver has decoded successfully the original message.

In the following we give a description of the encoder and the decoder.

The encoding process

A coded message M_i is generated from the original M_o according to the following process:

- the original message M_o is divided into n blocks of the same size, each block is labeled with an index i, so the blocks are $S_1...S_n$.
- choose a degree d, $0 < d \leq n$, using a particular distribution, the Robust Soliton Distribution (36).
- choose randomly d blocks with uniform distribution.
- XOR together the d chosen blocks generating one coded message M_i.

The coded message M_i is sent to the destination, appending a list of blocks indexes used to generate it.

This process is repeated until the original message is decoded from the destination. The Robust Soliton Distribution used in the step 2 of the coding process minimizes the overhead required for decoding (36) (see the decoding process).

The encoder could be considered as a process that generates a bipartite graph connecting the coded packet with those used to generate(see Figure 4.1).

The decoding process

Also the decoding process uses XOR operations to recover the original message. It is based on the consideration that for a coded block M_i, $M_i \oplus M_i = 0$, where $\oplus$ is the XOR operator. So when a coded message is XOR-ed with a packet of degree 1 i.e., a decoded block, his degree is decreases by one.

If this operation is repeated for $d - 1$ times the original message is retrieved.

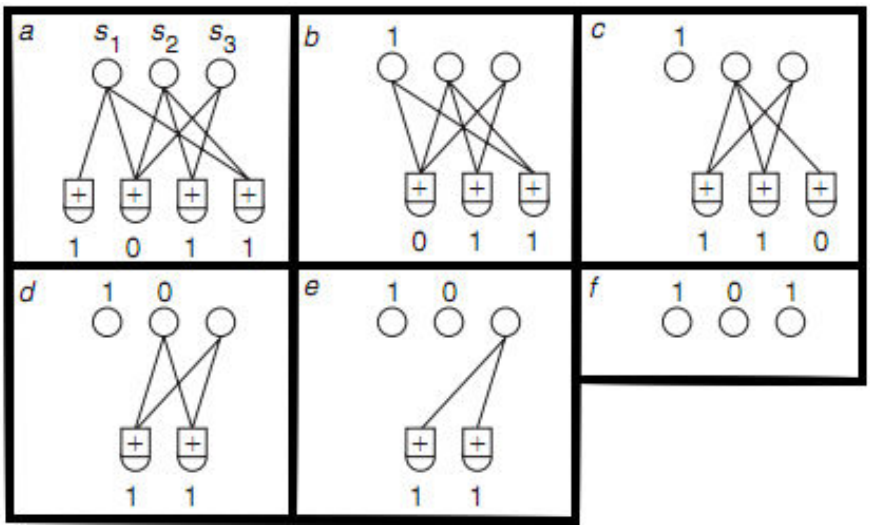


Figure 4.1: Example of decoding process for k=3 source bits and n=4 coded bits

The graph representation is particular useful to understand how the decoding process works (see Figure 4.1). Here the coded packets (showed on the bottom of the graph) are connected to the source packets (showed on the top) S_i used to generate them.

Following are the steps for decoding:

- find a coded packet M_i that is connected to only one source node, S_k. If there is no such node, the decoding fails.
- XOR S_k with all the coded packet connected to S_k.
- remove all the arcs connected to the source packet S_k.
- repeat from the first step until all packets are recovered or decoding process fails.

In Figure 4.1 is showed an example of decoding for a simple message with $k = 3$ source bits and $n = 4$ encoded bits. Here each packet is only one bit. In the top of each subfigure the nodes are the original bits S_1, S_2, S_3. In the bottom the coded packets are showed. The coded message is "1 0 1 1".

In the beginning, the coded block connected to only one source packet is the leftmost one (Figure 4.1a). According to the first step of the decoding process we can set the first source bit to 1 and discard the coded node (Figure 4.1b). Then we add the value

of S_1 (1) to all coded bits which it is connected and then disconnect S_1 from all nodes (Figure 4.1c). We repeat the process using the rightmost coded block, now the only node with degree one. So we set S_2 to 0 (Figure 4.1d) and XOR S_2 with all the coded packet which it is connected (Figure 4.1e). Finally (Figure 4.1f) is simple to obtain the value of S_3, connected with the two remaining coded nodes. Putting $S_3 = 1$ we have ended the decoding process. The source message is "1 0 1".

As we have seen from the example, this decoding process has a certain overhead (36): the recipient needs to receive a number of packets that exceed in percentage the original size by a certain amount ε. So if k is the original number of non coded blocks, the decoding process succeeds when $k(1+\varepsilon)$ packets are received.

The decoding overhead depends on the number of source packets (36). In (36) it is shown that LT codes are asymptotically optimal, i.e., the overhead $\varepsilon \to 0$ in the limit of large block sizes $k \to \infty$.

4.2 The modified Protocol

In order to introduce the LT codes we have done some modifications to the standard BitTorrent protocol. First of all the pieces are now encoded using LT codes (36) and the generated coded blocks are exchanged between peers. Consequently some modifications to the protocol's strategies must be done. The choice of LT codes has been made due their simplicity and efficiency.

4.2.1 Modifications to BitTorrent protocol

In order to evaluate the LT codes' effectiveness, we choose to start from BitTorrent then introduce the proper modifications. This choice is motivated by many reasons.
First of all it is impractical to encode the whole file, specially when it is of large size. Indeed the LT's decoding process requires that all received blocks must be maintained in memory until the original piece is recovered, and this can require a considerable amount of memory for large files. For this reason we choose to maintain the file's piece subdivision and encode the pieces' data. So, similar to BitTorrent, our protocol subdivides the file into pieces, and each one is encoded with LT codes.
The original BitTorrent's piece size is of 2^{18} Bytes and the block size is of 2^{14} Bytes.

In order to use the LT codes with efficiency the piece is now subdivided into about 1000 blocks. To obtain this ratio the block size is set to 500 Bytes with a subsequent piece size of 2^{19} Bytes, that results in roughly 1048 blocks for piece. According to the LT codes properties (36) (37), with this number of blocks the decoding process is accomplished with an overhead of about 10%. LT codes introduces some advantages with respect to the original protocol.

In BitTorrent a piece can be shared only when it is completely downloaded. Indeed, downloading of a partial piece with the standard protocol would require the introduction of a proper strategy for content reconciliation: a peer must share with others the piece's progress information, avoiding to receive duplicated information. This would result in a signalling overhead.

Instead, with LT codes it is possible to download the same piece from multiple peers. In fact, provided that the transmitting peers use asynchronous random generators for the creation of LT coded packets, every new packet is innovative for the receiver and coded blocks can be sent without information of the piece's progress.

More importantly, LT codes allow one to share partially downloaded pieces without content reconciliation. When a piece is partially downloaded it cannot be decoded at all, so the only possible action is forwarding some of the received coded blocks. The block to be forwarded is chosen randomly, so it can happen that some duplicated blocks are collected by the receiving peers. When a peer successfully recovers a piece, new encoded blocks can be generated, avoiding duplicates. In the simulation we made, we observed a reduced number of duplicates received on average. A significant percentage of duplicated packets is experienced only in the early stage of torrent distribution when the leechers own partial pieces.

This transient behavior disappears as soon as the peers decode some pieces, starting to generate new blocks. The introduction of encoding at piece level requires a new policy to select the next piece to download.

BitTorrent chooses a piece according to the rarest first strategy, selecting it only among the completely downloaded ones. The possibility to share partially downloaded pieces extends the set of possible parts to download. So we modified the rarest first in order to consider also partial pieces: a peer tries first to download the rarest complete piece in its neighborhood, but when useful pieces are not available, it selects a partial rarest one. In this way we try to minimize the negative impact of duplicates. The tit-for-tat

does not require modifications, indeed it performs in the same way in both protocols. In the following we summarize the modifications of the BitTorrent protocol we made:

- the dimensions of a block is changed from the original 2^{14} Bytes to 500 Bytes for an efficient use of LT codes. Every piece is thus composed of 1048 blocks;
- a chunk is shared to the other peers in the overlay also when it is partially downloaded. Now a peer sends the *HAVE* message, that in the original protocol inform other peers that the source own a piece, as soon as one coded block has been received. In this way a peer can start to share content sooner with respect to the standard BitTorrent.
- the local rarest strategy takes in account also the partially downloaded pieces This increase the availability of downloadable content, and reduce the start-up period.
- a piece can be downloaded from many peer at the same time.

All the other protocol parameters, such as the length of the *rechoke* period or the maximum number of unchoked peers are the same as in the original BitTorrent.

4.3 Simulation methodology

To evaluate the performance of the modified protocol we use the GPS (38) simulator. GPS includes a complete implementation of the standard BitTorrent, that is, at the best of our knowledge, a complete and accurate implementation. We use this implementation as a starting point to develop and test our changes. We generated different scenarios implementing high churn rates and flash crowds, and compare the performance of both protocols.

4.3.1 Extension of the GPS simulator

We made some changes to the simulator in order to define more complex scenarios. By modifying the simulator classes we add the possibility to generate heterogeneous networks, in which the peers can have different connection speeds. In the original simulator, it was not possible to define peer departures, but only joins. We extend GPS adding the possibility for a peer to leave the network overlay at certain instant in time,

Table 4.1: Downloading times in a common file sharing scenario.

File size	BTd [1]	LTd [2]
1 MBytes	52s	59s
2 MBytes	74s	103s
4 MBytes	132s	183s
8 MBytes	219s	313s
100 MBytes	1900 s	2100s
200 MBytes	3300 s	4100s

[1] BitTorrent standard protocol download's times

[2] Modified BitTorrent protocol download's times

independently from the progress of its download. Then we implemented the modified protocol introducing the modification described above starting from the original BitTorrent. Finally, we developed the classes to collect and analyze simulation results and statistics.

4.4 Experimental results

First we conduct some experiments in a common file sharing scenario: 30 peers join the swarm to download a file; we use in these tests different file size, from 1MByte to 200MByte. The peers do not leave the network after the download completion, but start to act as seeders. In Table 4.1 the results of these simulations are reported: the modified protocol has achieved the worst performance. This is mainly due to the overhead introduced by the use of LT codes.

We can assume here that rarest first and tit-for-tat are enough to assure diversity and a fair distribution of the content in the overlay. In this condition LT codes do not introduce any performance improvement.

Then we instrumented the simulator to conduct some experiments in scenarios characterized by high rates of peer arrivals and departures, using files with low number of

pieces. According to (13, 14) these conditions could be compared to live-streaming or video-on-demand. Indeed in these applications the pieces are spread on the overlay as they are generated, and have a short life time. Moreover in such kind of applications generic users can join the channel for a short time, and this results in high churn rates. Also flash crowds phenomena (46) can be an issue for the peer-to-peer overlay. This corresponds, for example, to some particular moments of the channel schedule, as the start or the end of a particular TV program. So a desirable property for a protocol is to be resilient to such kind of events. In all scenarios we simulate also the possibility that BitTorrent shares and spreads a document with only a single piece. This limitation derives from the unavailability of more pieces due the real-time nature of the content distribution. All the results presented are the average of ten simulation runs.

4.4.1 A simple topology of three peers

We start our investigation by defining a simple toy topology of three peers as shown in Figure 4.2. In this topology, using the BitTorrent protocol, the leechers cannot exchange data between them until a piece is completely downloaded. We want to verify if the modified protocol can achieve better performance by relaying incomplete pieces. The arrows show the flow of data between peers. Peer 1 is the seeder, that replies to requests made from the leechers, peers 2 and 3. We show that in such a simple scenario our modified protocol performs better compared to BitTorrent whether the shared file has a single or multiple chunks.

In Table 4.2 we report the results that compare the download times and the resulting achievable download rates of the standard BitTorrent protocol and the one modified with LT codes. From Table 4.2 we observe that the modified protocol achieves better performance for all the considered file size. The columns **BTd** and **BTb** show respectively the download times and achievable bitrate of the standard BitTorrent protocol while the columns **LTd** and **LTb** show respectively the download times and achievable bitrate of the modified protocol that uses LT codes The column **Gain** report the gain expressed in percentage.

There is always a gain using the modified protocol, and it is particular evident for file dimensions of 1 MByte and 2 MBytes. This derive from the possibility to start to

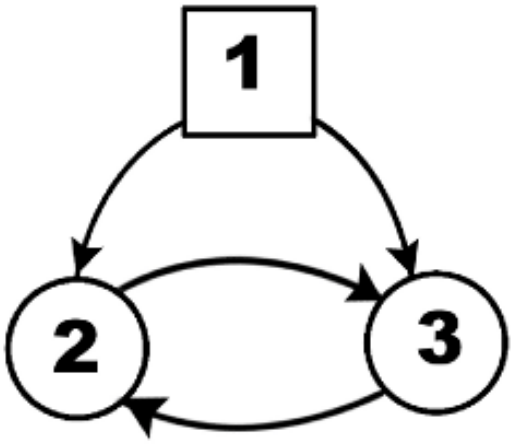


Figure 4.2: Simple three peers topology, peer 1 is the seeder while peer 2 and 3 are leecher

download a piece immediately, because the leechers can share data between them even if these latter do not belong to a completed piece.

4.4.2 A more complex scenario

After having verified that in the previous simple scenario the modified protocol has some benefit over the standard BitTorrent, we built a simulation scenario that represents a flash crowd in a network overlay: to this end we generate with GT-ITM (56) a network topology and define over it a network with a 10% of nodes with a bandwidth of 10 Mbps and a 90% of 1 Mbps for a total of 50 nodes.
The peers with the larger capacity represent some institutional nodes whereas the slower ones represent standard ADSL connections. Moreover, this scenario is characterized by massive arrivals and departures of peers. These events are scheduled in this way:

- Time 0: a flash crowd of 50 peers occurs
- Time 50: 20 random selected peers leave the network, regardless of the state of the download

Table 4.2: Downloading times and achievable bitrate observed on a simple topology of three peers.

File size	BTd [1]	LTd [2]	BTb[1]	LTb[2]	Gain (%)
1 MBytes	38s	35s	210Kbps	228Kbps	8%
2 MBytes	50s	48s	320Kbps	333Kbps	4%
4 MBytes	77s	75s	415Kbps	426Kbps	2.6%

[1] the columns **BTd** and **BTb** show respectively the download times and achievable bitrate of the standart BitTorrent protocol

[2] the columns **LTd** and **LTb** show respectively the download times and achievable bitrate of the modified protocol that use the LT codes

Also in this scenario we use a file of dimension of 1 MByte. The results are shown in Table 4.3: the first row compares the performance in the case of a file with a single chunk and the second for a file subdivided into more chunks.

Also in this case our protocol performs better, with the downloading times and the resulting achievable bitrate that show a gain of 66% if we consider files with only one piece. The gain decreases to 25% if consider the file subdivide into more pieces. When the file is divided into more parts, these are a dimension of 2^{18} Bytes for the standard protocol and of 2^{19} Bytes for the modified one. This result proves the increased resilience of the codes to flash-crowd. The resulting overlay is more robust to a single massive arrivals and departures of peers. In Figure 4.3 we report the normalized distributions of the download times.

The distribution on the left refers to the modified protocol while the one on the right describes the downloading times of the standard BitTorrent.

It is clear that the BitTorrent that uses the LT codes performs better in both situations, showing shorter download times. Indeed the modified protocol's time distribution is centered around 20 seconds, and show values from 5 to 40 seconds. Instead standard BitTorrent donwload's times are between 10 and 120 seconds, with a peak aroud 60 seconds.

Table 4.3: Downloading times and achievable bitrate observed in a single flash crowd scenario.

File size	PS[1]	BTd[2]	LTd[3]	BTb[2]	LTb[3]	Gain
1 MByte	S	53.1s	18.1s	151 Kbps	441 Kbps	66%
1 MByte	M	23.9s	17.1s	334 Kbps	467 Kbps	26%

[1] the file subdivision, single (**S**) or multiple (**M**) pieces

[2] the columns **BTd** and **BTb** show respectively the download times and achievable bitrate of the standart BitTorrent protocol

[3] the columns **LTd** and **LTb** show respectively the download times and achievable bitrate of the modified protocol that use the LT codes

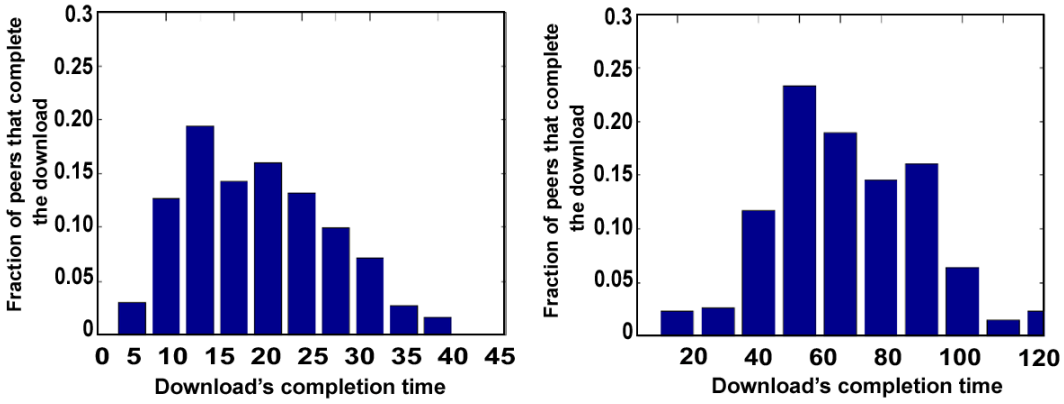


Figure 4.3: Normalized distributions of the download times: on the left the modified protocol, on the right the original BitTorrent protocol

Table 4.4: Parameters used for generate flash crowd scenario

Parameter	Description
a	Time between a flash-crowd and the next
b	Intensity of the flash-crowd
c	Flash crowd duration
d	Time between a massive departure of peers and the next
e	Intensity of the massive departure of peers
f	Flash crowd duration for massive departure
g	Inter-arrival times parameter distribution
h	Leave times parameter distribution

4.4.3 Flash crowd scenario

Now we generate a more complex scenario that presents multiple flash crowd phenomenon and massive departure of peers to test the overlay robustness. To describe a flash-crowd we use three parameters: a, b, c are the exponents of three exponential distribution, from which the values are random sampled to model a flash-crowd.
Values derived from the distribution with parameter a model the times between two consecutive flash-crowds, b is used to model the intensity, expressed as the number of the peers that join the network in a reduced interval time, and the third parameter describe the flash crowd duration. Short values sampled from the distribution with parameter c describe a more critical flash-crowd, because the peers join the network in a reduced amount of times. The total number of peers that join the network is distributed linearly among this time interval.
Massive departures of peers are modeled in the same way, but using three different parameter: d, e, f In addition in the simulated scenarios arrival and departures of peers at fixed rates are simulated independently from flash-crowds. This rates are modeled using two additional exponential distribution that use parameters g and h. Table 4.4 summarize all the used parameters.

We conducted some experiments building complex scenarios that cripple the BitTorrent performance. The experiment has the subsequent parameters: $a = 0.025$, $b = 0.05$, $c = 0.5$, $d = 0.025$, $e = 0.05$, $f = 0.5$, with a total simulation length of 1500

Table 4.5: Downloading times and achievable bitrate observed in a scenario affected by multiple flash crwod and massive departure of peers.

File size	BTd[1]	LTd[2]	BTb[1]	LTb[2]	Gain
4 MBytes	65 s	56 s	492 Kbps	571 Kbps	14%
8 MBytes	145 s	129 s	441 Kbps	496 Kbps	12%

[1] the columns **BTd** and **BTb** show respectively the download times and achievable bitrate of the standart BitTorrent protocol

[2] the columns **LTd** and **LTb** show respectively the download times and achievable bitrate of the modified protocol that use the LT codes

seconds. The unit measure of the variables is s^{-1}, while respective exponential distribution have mean: $mean_a = 40s$, $mean_b = 20s$, $mean_c = 2s$, $mean_d = 40s$, $mean_e = 20s$, $mean_f = 2s$. The results are averaged over 20 runs of 2 different randomly generated scenarios.

In Figure 4.4 one scenario generated with the above parameters is shown. It is affected by many flash-crowds and massive departure of peers, as expected. The simulated network has 1000 nodes, with 10% of 10 Mbps of bandwidth and 90% represent ADSL connections with 1 Mbps of bandwidth. In this case we conduct two simulations using file of 4 MBytes and 8 MBytes, that are subdivided into multiple pieces.

In this way the original BitTorrent protocol can use his standard local rarest first strategy and tit-for-tat at his best. Table 4.5 reports the results of such simulations compared to BitTorrent in the same scenario and network conditions: there is a clear gain of 14% for the file of 4 MBytes and 12% for dimensions of 8 MBytes.

Also in this case there is a clear advantage that derives from the introduction of LT codes, that can spread content with limited file size and in precence of high churn rates in a faster way. The gain mostly derives from the capacity to share partially downloaded pieces. These advantages, in case of file of small size and in high dynamic network conditions, exceed the drawback derived from the overhead introduced from the use of the LT codes.

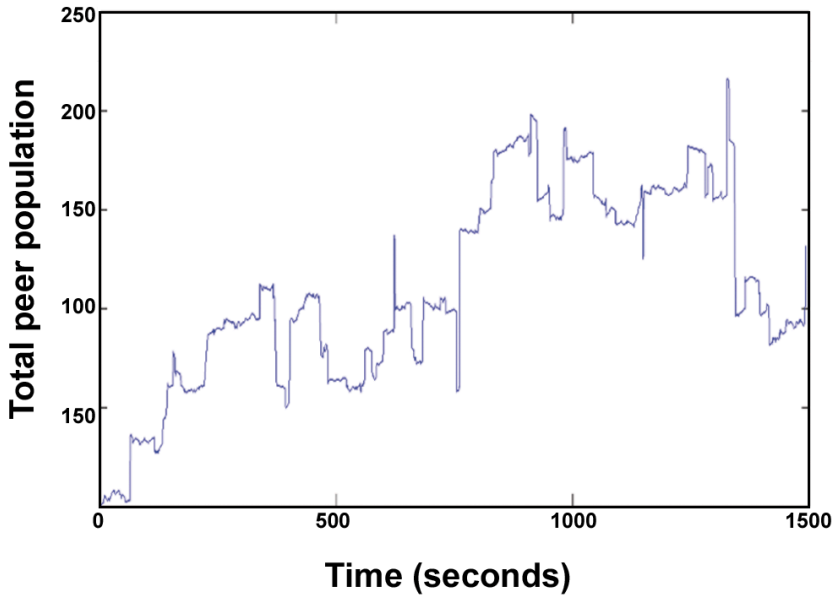


Figure 4.4: The evolution of overlay population in a sample scenario generated with parameters $a = 0.025$, $b = 0.05$, $c = 0.5$, $d = 0.025$, $e = 0.05$, $f = 0.5$

4.5 Conclusions

We proved by simulations that the protocol we developed can achieve better performance considering file of small size, high churn rate and flash crowd. This conditions represent real-world applications like live-streaming, in which only a little fraction of the total file is available at time and peer join and leave the network with high rates. In such situation our proposed protocol yields a gain that is above the 10% in all simulated scenarios, despite the decoding overhead that the LT codes introduce.

This results show that the introduction of some kind of network coding, in our case LT codes, can lead to improvements, but these are mainly related to network conditions and peers' behaviour.

Chapter 5

P2P simulator

In this chapter we describe the discrete-events simulator we develop. We implement a new simulator to obtain a more efficient tool than General Purpose Simulator (GPS), previously used in Chapter 3.

Fist we give a brief introduction to GPS. Then we describe the transit-stub model, used to represent the network layer in both simulators and explain the new simulator's architecture. We present benchmarks that show how our tool is more efficient than GPS, both for memory and time requirements, and validate the BitTorrent protocol with data obtained from real clients deployed on PlanetLab.

5.1 The General Purpose Simulator (GPS)

The General Purpose Simulator (38) is a discrete-event simulator written using the Java programming language. It has been developed with the specific task of implementing the BitTorrent protocol. The simulator architecture can be subdivided into two layers: network layer and application layer.

GPS models the network layer according to the transit-stub model. Such model is explained more in depth in the following section. The available bandwidth of a link in the network is computed according to the macro-model proposed in (41), that is used to simulate the TCP/IP overhead. In addition GPS uses a flow model to assign bandwidth to a connection: the available bandwidth is equally shared among all active connections on a certain link. Because a message is routed from source to destination

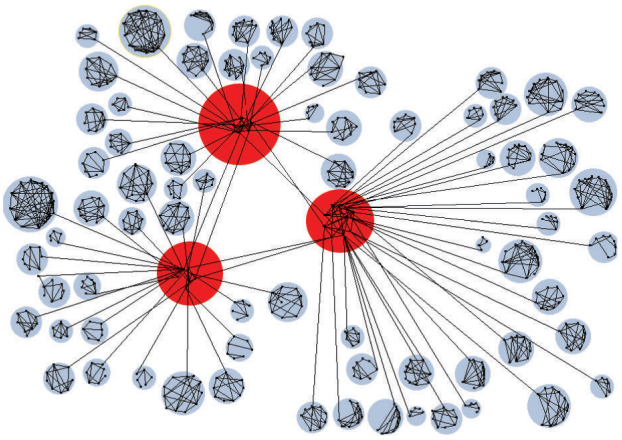


Figure 5.1: Example transit-stub network with three transit domains (in red) and many stub domains (in blue)

across a links chain, the maximum available bandwidth corresponds to the bottleneck link between source and destination.

The BitTorrent protocol is implemented on the network layer. The protocol's simulation is quite accurate, providing an almost complete implementation of BitTorrent's strategies and network messages, except for the 'endgame' scheme, that is not simulated.

Drawbacks of GPS are the huge memory and CPU requirements that limit the scalability. Moreover it is not possible to define complex network scenarios, taking into account churn, disconnections or peers failures. Finally the simulator's validation has been made in a very simple set up of only five BitTorrent clients connected to a hub.

5.2 The transit-stub model

An example of transit-stub network is shown in Figure 5.1. The topology is represented in a hierarchical way: a peer is a node in a stub domain (blue in Figure 5.1). Each stub domain is linked to a transit node. Transit nodes are organized into domains (red in Figure 5.1) building an internal network that represents the router sub-net. Both stub and transit domains are random graphs (40). Specifying the properties of these i.e., the average number of nodes or the number of connections among different domains, it is possible to generate many different topologies. For example the network depicted in

Figure 5.1 has three transit domains composed by several transit nodes and each one of these is connected to a stub domain.

A network message between two peers leave a stub, travel across a set of transit nodes, and finally is delivered to the destination stub node. The links' bandwidth among transit nodes is defined to be greater than the transit-stub ones. Connections among different stub domains are not allowed. To generate topologies that can be used in our simulator, we modify GT-ITM, a software used to build transit-stub graphs, developed at the Georgia University.

5.2.1 GT-ITM

Georgia Tech Internetwork Topology Model (GT-ITM) is a C program that allows to define topologies according to the transit-stub model. GT-ITM loads parameters from a config file, generating in output the graph. Here we explain what are the most important settings used to define a valid transit-stub graph that can be used as network topology. The format of the config file is:

$[\#comment - line]$
$[method - keyword]\,[number - of - graphs]\,[initial - seed]$
$[method - dependent - parameter - lines]$

$[method - keyword]$ can be "geo", that generates a flat random graph, or "ts", for a transit-stub. $[number - of - graphs]$ specifies how many graphs will be generated. Several parameters that depend on the method keyword specified are specified in $[method - params]$. In the following we report an example of the config file, with some comments to the most relevant lines.

```
#Type and number of graphs to be generated
ts 10 47
#Average number of stub domains for each transit domain
3 0 0
#Average number of transit domains
1 20 3 1.0
#Average number of nodes in a transit domain
4 20 3 0.6
#Average number of nodes in a stub domain
```

```
8 10 3 0.42
```

With this file, GT-ITM generates ten transit-stub graphs, using as random seed 47. Each graph will have three stub domains per transit node (line 2), with no extra transit-stub or stub-stub edges. Next line (line 3) creates one transit domain. Line 4 specifies that transit domains have (on average) four nodes and an edge between each pair of nodes with probability 0.6. The last line says that each stub domain will have (on average) eight nodes, and edge probability 0.42. Resuming this config file generates a graph with one transit domain composed by four transit nodes, each one with three stub domains of eighth nodes. The total number of nodes in the graph is $1 \cdot 4 \cdot (1 + 3 \cdot 8) = 100$.

5.2.2 The new file format

The original graph format generated from GT-ITM is not practical to be loaded from an application in real time, because the graph representation is redundant and not well structured. We have done some modification to GT-ITM software to generate a more practical file format. With the new file format we have been able to reduce the time and memory requirements in the network loading stage. The modified GT-ITM software compiles both under Linux and Cygwin32 on Windows. This latter version requires libgb-linux instead of libgb.

The new version of GT-ITM generate both the original file with ".gp" extentions and the new .alt2 format that is used in our simulator. Here we illustrate the ".alt2" file format. Each line starts with a tag, that specifies the meanings of the other parameters on the same line. In the following is reported the semantic of each tag.

- **I [number of transit domains]:** specify the number of transit domains;
- **T [transit domain id] [number of transit nodes] [list of link pairs]:** creates a new transit domain with a new id, number of nodes, and a list of pairs representing the links between nodes;
- **S [id of the parent transit domain] [parent transit node] [list of link pairs in the stub]:** creates a new stub domain and its parent transit node;

- **GWS [transit domain] [stub domain] [start stub node] [destination transit node]:** add a stub gateway i.e., a link between a stub and a transit domains;
- **GWT [transit domain source] [transit domain destination] [start transit node] [destination transit node]:** add a transit gateway i.e., a link between two transit nodes in different domains;

5.3 The new simulator

Using the Java programming language we developed a new discrete-event simulator and used it to implement the BitTorrent protocol and the ToroVerde protocol presented in Chapter 6. The simulator's architecture is organized in three logical layers: network layer, application layer and simulation layer. Here we report a brief description of each layer.

5.3.1 Network layer

The network layer is realized according to the transit-stub model (56) previously explained. Direct stub-stub connections are not allowed. End-systems are attached to stub nodes, that are logically placed at the bottom of the hierarchy, and in our case these are P2P clients. Also the tracker is attached to a stub node. A unique address is assigned to each peer and the network messages travel form source to destination via transit domains. Indeed these represent the router sub-net.

The network topology is loaded from the ".alt2" file generated using GT-ITM software (56). The nodes in the topology are connected with links that have fixed capacity. We assign a bandwidth of 100 Mbps to links among transit nodes and 1 Mbps to stub-transit ones.
A network message is routed from the source peer to destination across many interconnected links. The avaible link's bandwidth is equally shared among all the active connections.

The GPS simulator uses the Floyd-Warshall algorithm to compute shortest path between all node pairs in the network. This algorithm has $o(n^3)$ time complexity and $o(n^2)$ space complexity. We modify the original algorithm to perform the Floyd-Warshall only at local level i.e., we compute all pairs shortest path only for nodes in

the same domain, storing these paths in local routing tables. To route a message from a domain to another we build global routing tables that save the shortest path among different domains. Using both routing tables the path between any pair of nodes can be built in a top-down way. As showed in the benchmark sections this leads to a great improvements in performance, both for memory and for CPU saving. Allocation and deallocation of network resources on links occur for each network message that travels across the network.

5.3.2 Application layer

The protocols are implemented in the application layer. The application level messages are passed to the underlying network layer. Here they are encapsulated in network messages, then delivered to the recipient. The BitTorrent implementation conforms to official protocol specification described in (12). Peers and tracker only rely on local information, replicating the behavior of the real network clients.

5.3.3 Simulation layer

Finally the simulation layer loads the system parameters, builds the network, assigns bandwidth to links and allocates all peers in memory. Each peer is assigned a protocol (e.g. BitTorrent) and it is associated with a unique stub node.
Protocols are initialized with parameters and simulation started. The simulation layer has access to peer's internal state, gathering information and building statistics.

5.4 Benchmarks and validation

First we conduct some test for the simulator's network layer. We use different topologies up to 512000 nodes. In Table 5.1 are reported the results of our benchmarks. The file *ts[number of nodes in the network]* is the topology file used. For example the file t1728 specify a topology with 1728 nodes. All the benchmarks have been conducted on an Intel core2 Duo CPU T250 @ 2.00 GHz with 2GB RAM. The Java heap size is set to 1400MB.

Both for execution time and the memory requirements our simulator performs much better than GPS. Concerning loading time our algorithm is more efficient than Floyd-Warshall, while for memory requirements GPS is able to load graph only up 1728

Table 5.1: Comparison for CPU and memory requirements between GPS and the NEW simulator. OOME stand for "Out of memory error"

Topology	Time GPS	Memory GPS	Time NEW	Memory NEW
ts16	1.85 Ms	13 MB	1.96 ms	64 MB
ts125	91.5 Ms	78 MB	0.417 ms	60 MB
ts512	4887 Ms	126 MB	1.22 ms	61 MB
ts1728	161 s	854 MB	3 ms	63 MB
ts3375	OOME	OOME	6.14 ms	66 MB
ts8000	OOME	OOME	14.7 ms	71 MB
ts27000	OOME	OOME	41.6 ms	81 MB
ts125000	OOME	OOME	247 ms	153 MB
ts512000	OOME	OOME	1322ms	428MB

nodes. Indeed the simulation load the graph, but crashes when start to run simulation throwing a OutOfMemory exception.

To the contrary using our simulator we are able to load topology up to 512000 nodes. In Table 5.1 we show the memory usage during a simulation run. This is more evident in Figures 5.2 and 5.3.

Previous results refer only to the loading process of the network layer. We also examine how the memory requirements evolve during the simulation. In Figure 5.4 we depict the heap usage by GPS and our simulator for a network with 1725 nodes, without P2P clients, and in Figure 5.4 is showed the same scenario but with 500 simulated BitTorrent clients. For both scenarios simulator's requirements remain constant during the entire running time, but also here our tools use much less heap memory than GPS.

5.4.1 Simulations validation

We validate the BitTorrent simulation comparing the simulated downloading times with data obtained from BitTorrent client deployed on PlanetLab. In these tests the overlay size was up to 60, and the peers join and leave the network; they alternate periods where are connected (ON) to periods when are not (OFF). Both ON and OFF periods

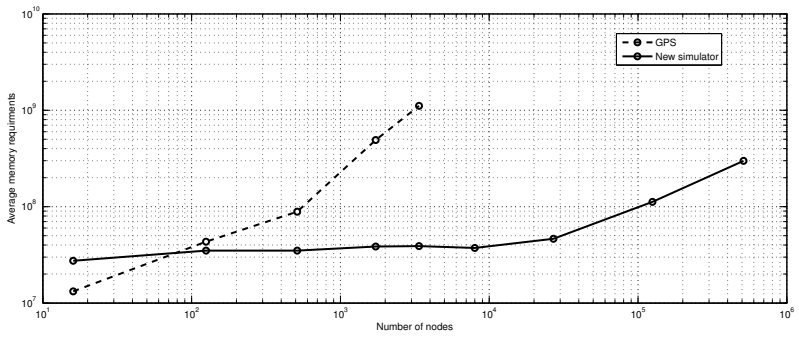


Figure 5.2: Memory requirements (in bytes) comparison between GPS and the NEW simulator

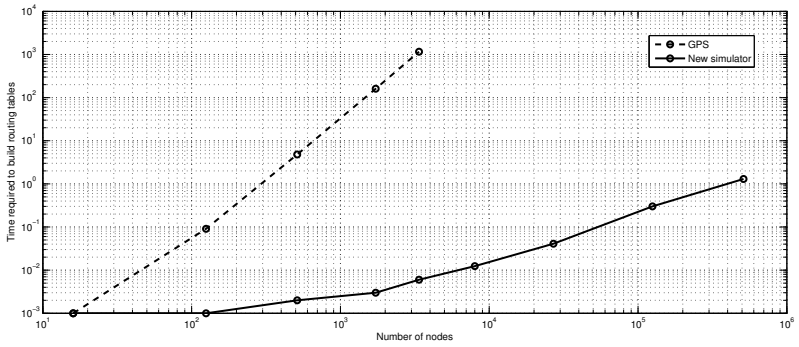


Figure 5.3: Time requirements (in seconds) comparison between GPS and the NEW simulator.

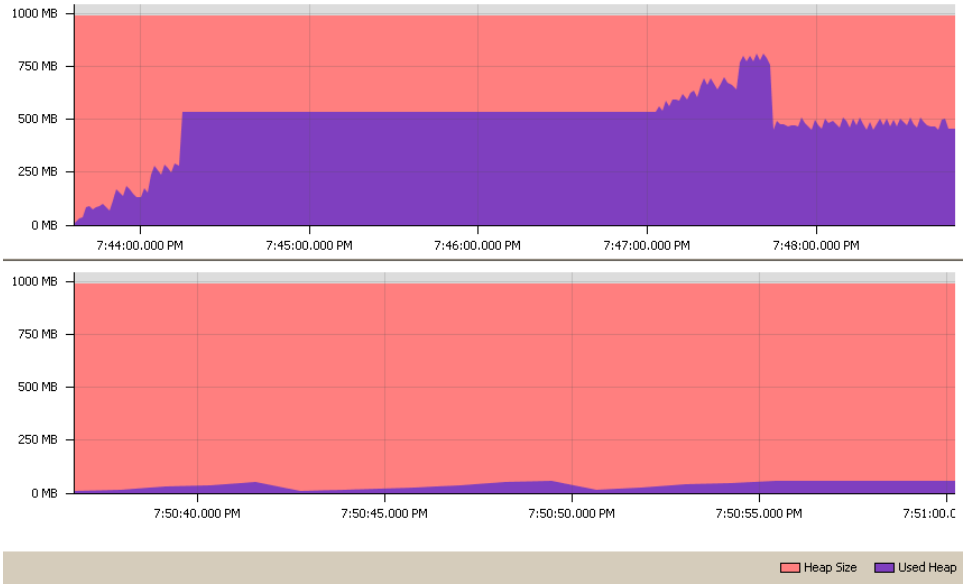


Figure 5.4: Heap usage comparison between GPS (up) and NEW (bottom) for a network of 1725 nodes without BT clients

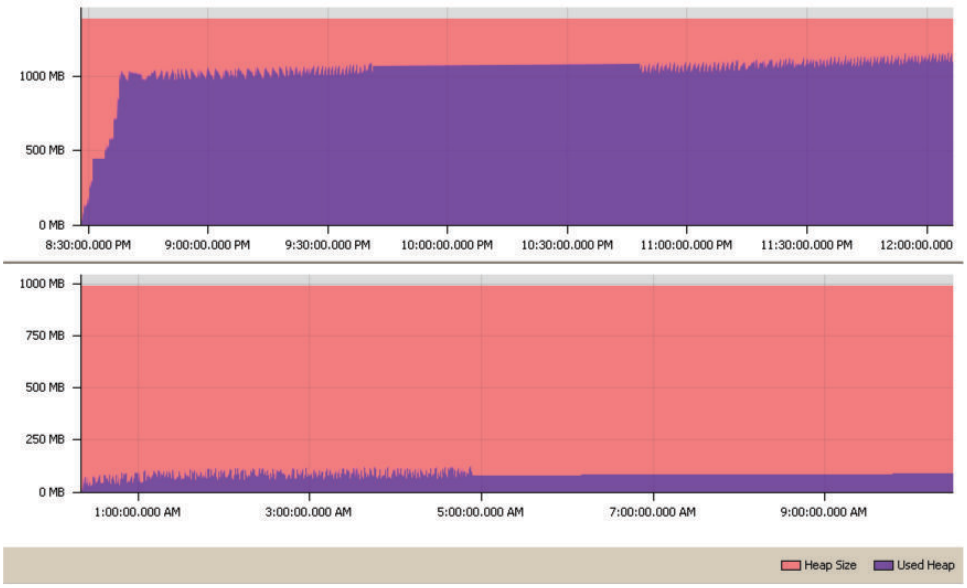


Figure 5.5: Heap usage comparison between GPS (up) and NEW (bottom) for a network of 1725 nodes with 500 BT clients

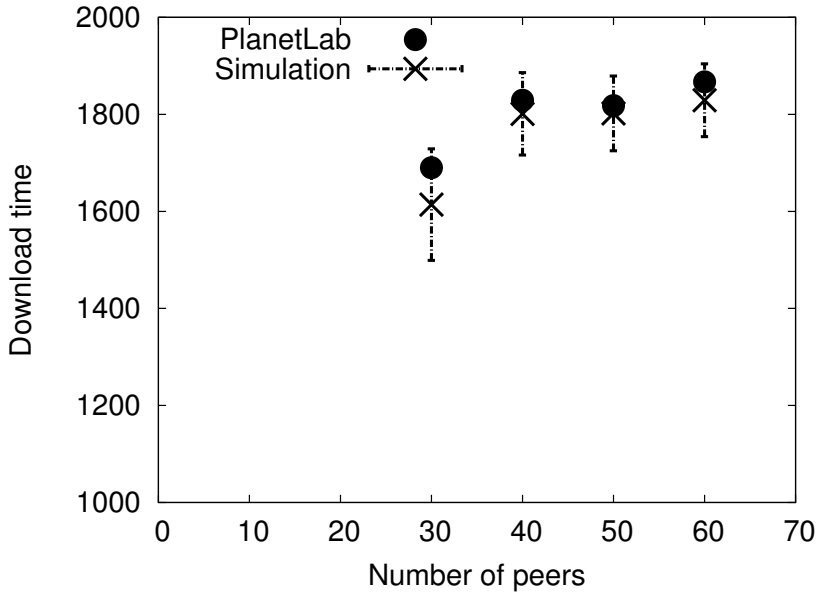


Figure 5.6: Dowloading times comparison between BitTorrent clients deployed on PlanetLab and simulated ones

are negative exponentially distributed with mean of 100 seconds. Each peer starts with an OFF period. The file size is 32 Megabyte.

Figure 5.6 shows the comparison between the simulated downloading times and real ones. We plot the mean and the confidence interval of 95% for simulation results and compare these with the PlanetLab results. For network of 20, 30, 40 and 60 peers we can see how the simulated downloading times mach with the real ones. Indeed these latter are always included in the confidence intervals of those simulated.

5.5 Conclusion

In this chapter we have presented our simulator. We have described several improvements that we have implemented respect to GPS, used in Chapter 3. We have presented several benchmarks that show how our tool is better for both CPU and memory requirements. Finally we validated the BitTorrent simulation in complex network conditions using data obtained from real clients deployed on PlanetLab.

Chapter 6

The Toroverde protocol

In this chapter we extend the work presented in Chapter 3 and present ToroVerde, a novel push-based P2P content distribution application exploiting the digital fountain concept through the use of rateless codes.

We provide the protocol specification and describe the simulator and the prototype we developed for Planetlab deployment and testing. We present results from PlanetLab experiments that are compared with the protocol that is widely considered as the reference system for content distribution i.e., BitTorrent. Preliminary results show that ToroVerde has the potential to improve the BitTorrent performance up to 20% in several scenarios for small-to-medium sized files in overlays composed of a few hundred peers.

6.1 Protocol Description

In this section we describe the main ideas and features of the ToroVerde protocol. In particular, the section is divided in sub-sections that address issues such as the used overlay management criteria, the bitmap management, the flow control, the block scheduling process, and the implementation of the rarest first block criterion.

6.1.1 Protocol overview and terminology

Since ToroVerde shares with BitTorrent several architectural characteristics (i.e., single file centric architecture, the tracker, and the use of a torrent file for instance) we adopt the classical BitTorrent terminology. In particular, a *peer* is a host that wants to

download the shared file. The peers (or the peer) that own the complete file are called *seed*s.

The *tracker* is the host that organizes the overlay topology as described in Section 6.1.2. Upon opening a torrent, a peer contacts the tracker and receives a list of peers to connect to.

The *torrent* is a small metadata file which contains information about the file to download, not the data itself.

The shared file is divided into *pieces* (also called chunks). Every piece is further subdivided in k *blocks*. ToroVerde introduces linear coding at the block level. In particular, LT codes (36), an asymptotically optimal family of rateless codes, are used on every piece independently and the peers exchange coded blocks.

The use of coding naturally implies the adoption of a push mechanism for data distribution simplifying the protocol operations for content retrieval and reconciliation.

On average, a piece can be completely retrieved when any set of $k(1+\epsilon)$ coded blocks, where ϵ is the overhead introduced by coding, has been received. The details on LT coding/decoding, that are relevant for the protocol description are discussed in Section 6.1.3.

As opposed to network coding where a peer keeps combining the received coded blocks and forwards such novel combinations, with LT codes a peer is required to have already downloaded a piece to be able to create new coded blocks. In such a case, given the rateless property of the code, an arbitrary large number of coded blocks can be pushed to the neighbors.

Nevertheless, a peer can push coded blocks from incomplete pieces using the algorithm described in Section 6.1.6. This algorithm prevents the same coded block to be sent to the same destination more than once.

This approach aims at achieving a fast dissemination of the coded blocks by means of a practical coding approach exploiting both the advantage of network coding and the efficiency of the LT codes with respect to random linear coding used in the past. The kernel of ToroVerde, constituted by a push approach for the dissemination of the coded blocks, is completed by a set of solutions that have been derived from the BitTorrent experience but that need to be carefully reconsidered when moving from a pull-based to a push-based protocol.

Table 6.1: Table of symbols

Symbol	Description
P	a generic peer
K	number of blocks per piece
N	network size
$IN(P)$	bitmask representing the pieces needed by P
$OUT(P)$	bitmask representing the pieces offered by P
$N_{classes}$	number of classes used by the rarest first algorithm
N_{EN}	number of enabled neighbors

Bitmaps with a new semantic with respect to BitTorrent are used to inform the neighborhood about the pieces required by each peer and allowed us to implement the rarest first policy for the distribution of the file pieces. Each peer selects the destinations of the coded block with an algorithm aiming at maximizing the reciprocity and limiting the effects of free riders.

Another remarkable feature of ToroVerde is the use UDP with TCP Friendly Rate Control (TFRC) (57). With the use of rateless codes there is no need to resend lost blocks (like for instance TCP does) and hence UDP is the natural choice since reliability is provided by the use of the linear coding.

6.1.2 Overlay management

Peers are organized in a non structured overlay network, whose management is performed by the tracker. A peer joining the swarm contacts the tracker to receive a list of other peer addresses within the swarm. Moreover, a peer can ask the tracker for new addresses when the number of their neighbors falls below a threshold. The tracker maintains the list of the participant peers, that must also send it a periodical keep-alive message to signal their presence and to allow it to detect silent departures.

A peer establishes application level bi-directional links[1] with their new neighbors by using a two ways handshake and maintains these links by means of keep alive messages.

[1] Note that we avoid the term connection because we use UDP and not TCP.

6.1.3 LT coding and decoding

Seeds and peers that have completed the download of a piece are capable of creating new LT coded blocks for such piece. The LT encoder forms random linear combinations of the k blocks using binary coefficients as follows.
First, the number of blocks d to be combined is randomly selected according to Robust Soliton Distribution RSD(c, δ), that depends on two parameters c and δ; then d random blocks out of the k available are combined through the XOR operation obtaining a coded block.
The seed of the random number generator used to produce the LT coded block is sent along with any coded block so as that the receiving peer is able to repeat the random selection process according to the RSD. This avoids the explicit signalling of the block indexes used to form the coded blocks.
Clearly, the adopted solution being independent of k is the most compact for large values of k.
The receiving peers use the recent *On the Fly Gaussian Elimination* (54) algorithm to solve the linear system of equations imposed by LT coding. This technique guarantees low overhead ϵ i.e., percentage of coded blocks to be received in excess of k to get the original content, and exhibits a limited computational cost per received block.
In Chapter 3 LT code have been explained more in depth.

6.1.4 Bitmap update

For each peer P a certain piece can be in three possible states:

- *empty* if P has not received any block of the piece;
- *decoded* if P has received enough coded blocks to accomplish LT decoding on the piece;
- *downloading* if P has received some encoded blocks, but not enough to complete LT decoding.

To identify the state of all pieces for a generic peer we use two bitmaps, namely the IN and OUT bitmaps. The bit $IN(i)$ is set to 1 if and only if the peer has not yet decoded the i-th piece, whereas the bit $OUT(i)$ is set to 1 if and only if the peer has received at least one coded block of the i-th piece. Note that, the IN bitmap contains

information about the pieces that P is looking for, while the OUT bitmap represents the set of pieces that P can potentially share with the neighborhood.

In the ToroVerde protocol the only information exchanged between peers are by their own IN bitmaps. The IN bitmap is initially exchanged during the connection handshake and is updated every time a peer decodes a new piece to inform its neighbors that it does not need more coded blocks for that piece.
Since these update messages can get lost because all the protocol messages are communicated via UDP, the receiver must acknowledge the update. An update message is sent every time a peer accomplish LT decoding of a piece and at periodical intervals; this message contains the differences between the new IN and the last acknowledged bitmap.
The protocol can choose whether to send the list of bits to be updated or the whole bitmap, depending on what representation is the most compact.

6.1.5 Flow control

The use of UDP together with a push mechanism requires the implementation of rate adaptation mechanism to control the amount of coded blocks forwarded to every peer. To this aim we adopt the classical TFRC. With this mechanism, the sender explicitly adjusts its upload rate toward a receiver as a function of the measured rate of loss events, where a loss event consists of one or more blocks dropped within a single round-trip time.
According to the specification of (57) for the computation of the appropriate rate, the sender uses the following equation:

$$X = \frac{s}{\mathrm{RTT}\sqrt{\frac{2p}{3}} + t_{\mathrm{RTO}}\left(3\sqrt{\frac{3p}{8}}\right) p\,(1 + 32p^2)},$$

where s is the block size in bytes, RTT is the round-trip time in seconds, p is the loss event rate, t_{RTO} is the retransmission timeout value in seconds (with $t_{\mathrm{RTO}} = 4 \cdot \mathrm{RTT}$).

Clearly, TFRC statistics must be kept updated for every active P2P connection. In our implementation every data message contains a simple TFRC header, with a sequence number and the estimated RTT. Feedback messages are sent every time a data message is received by a neighbor. These messages contain the timestamp of the

last received data block, the amount of time elapsed between the receipt of the data block, and the generation of the feedback report. Further details can be found in (57).

6.1.6 Block scheduling policy

The block scheduling process includes several algorithms. The first operation is constituted by the selection of what we call the set of *enabled neighbors*, to whom a peer will send coded blocks. This set changes over time, as explained later on in Section 6.1.8. For each enabled neighbor e the sender computes, by using TFRC algorithm, the number of blocks that can be sent to e, denoted as $b(e)$.

For each enabled peer e, blocks are selected from the pieces available in the OUT bitmap of the sender and required by e according to its IN bitmap.

The ToroVerde protocol is based on a simple push protocol without reconciliation at the receiver; clearly, the transmission of duplicated information must be avoided on the sender side in order to save the precious upload resources of the peers.
To this end every peer can forward a coded block only *once*. This constraint is not enough to guarantee that any coded block is received only once by any peer in presence of cycles in the overlay. To avoid the propagation of the same coded block along a cycle in the overlay we include a Bloom filters (55) in the packet header so as to compactly represent the path already covered by any block. The ID of any destination peer is hashed and accumulated in the Bloom filter bitmask. When selecting a new destination the Bloom filter is used to detect if a given destination has already been added to the filter. In the case of completed pieces the Bloom filter is put to all 0s, since new coded blocks for all interested receivers can be generated on the fly according to the random LT coding process.

We use a 64 bits Bloom filter with 3 hash functions. This ensures that a block is never sent twice to the same peer. The Bloom filter permits a very compact representation of the path followed by any coded block even if, being a probabilistic data structure, it can generate false positives when querying for added destinations.

As a marginal note, the Bloom filter properties can be used to speed up the search for useful blocks from pieces that a peer has partially downloaded. Given a certain piece, a peer computes the Bloom filter of the intersection of all the coded blocks of that piece by a simple AND operation among all the corresponding Bloom filters. The intersection Bloom filters is used to check if a destination peer already owns all the

blocks that the sending peer has for a given piece, thus speeding up the search process of a useful piece.

6.1.7 Rarest first implementation

The choice of what piece to forward first is very critical, because some pieces are more useful than others since more peers need them.
A peer can share with others only pieces represented by its OUT bitmap. These pieces are further classified according to their popularity depending on the number of neighbors that require them.
We define the popularity class of a piece as

$$class = \left\lfloor \frac{needing}{N_{\text{neigh}}} \cdot (N_{\text{classes}} - 1) \right\rfloor$$

where *needing* represents the number of neighbors that need a piece and N_{neigh} is the current number of neighbors. For the computation of the value *needing* for each piece the IN bitmaps of the neighbors are used. The use of the floor function implies that the highest popularity class is populated only by pieces requested by all neighbors.
This is particularly useful for seeds, since a piece has the highest priority only if there are no other replicas in the overlay except the original copy.

Another important aspect for an efficient distribution of the content is the ability to increase the number of completed pieces in the overlay as quickly as possible. To this end we introduce the idea of preferential piece; every peer associates to every of its enabled neighbors a preferential piece and, whenever possible it tries to send to a neighbor always blocks belonging to the same preferential piece.
For each enabled peer e a peer selects one of the most requested i.e., rare pieces in its neighborhood, and sends as many coded blocks as possible according to the TFRC estimate $b(e)$, trying to make e complete the piece as quickly as possible according to the preferential piece selection criterion.

The piece selection algorithm is sketched in the following, where $PREF$ is the bitmap containing the indexes of all the preferential pieces of the neighbors.

For any enabled peer e with a preferential piece *pref* and bitmaps IN do:

1. Initialize the set of blocks $blocks(e)$ to be sent to e as the empty set.

2. Initialize priority $class = (N_{\text{classes}} - 1)$.

3. Create a bitmap PC, that contains the indexes of the pieces available from OUT with priority $class$.

4. Evaluate the bitmap of the pieces with priority $class$ required by e: $PIECES = PC \wedge IN$.

5. If $PIECES$ is empty and $class = 0$, then return the set $blocks(e)$.

6. If $PIECES$ is empty, then set $class = class - 1$ and go to 4

7. If $pref$ piece belongs to $PIECES$ select it.

8. Else if the intersection $PIECES \wedge PREF$ is not empty select a random piece from such intersection set.

9. Else select a random piece from $PIECES$.

10. Add blocks from the selected piece to $blocks(e)$ until it has $b(e)$ blocks.

11. Clear the bit corresponding to the selected piece in $PIECES$.

12. Update the value of $PREF$, indicating the selected piece as the new preferential piece for e.

13. If $blocks(e)$ contains less than $b(e)$ blocks go to 5.

14. Else return the set $blocks(e)$.

It is worth noting that the previous algorithm does not explicitly discriminate between pieces in downloading and decoded state. Nevertheless, the use of the preferential selection strategy implicitly advantages decoded pieces. This happens because once a decoded piece is selected from the previous algorithm, the sender continues to send blocks of this piece until the recipient decodes it, unless the piece looses popularity.

6.1.8 Enabled neighbors

Peers and seeds use different criteria to create the list of N_{EN} enabled neighbors. The peers use a mechanism that is derived from chocke/unchoke mechanism of BitTorrent. In the ToroVerde case we compare the neighbors in terms of the number of blocks they pushed to a given peer in the last 60 seconds. The list of enabled neighbors is refreshed every s seconds (in our experiments we set $s = 10$) as follows: the worst enabled neighbor is replaced with a random disabled peers. This strategy mimics the BitTorrent's optimistic unchoking.
In addition, the best disabled peer is swapped with the next to the worst enabled one (if the first is is better than the last). This refresh mechanism aims at preferring collaborative peers giving them more chances to receive blocks and complete the download. When an enabled neighbor quits, another one takes its place.

On the other hand, the seeds use a different policy and refresh the enabled neighbor set less frequently. When an enabled peer completes its preferential piece, that it is removed from the enabled set, and the one that has been disabled for the longest time takes its place.

Each peer or seed can increase the number of enabled neighbors N_{EN} dynamically when for some time it is not able to provide to the neighbors as many blocks as it should (according to the TFRC estimates) or its outgoing buffer is getting empty. On the other hand, when the outgoing buffer is full for some time, it can decrease N_{EN}. It is worth pointing out that enabling and disabling decisions are local operations that do not imply the sending of any messages in the ToroVerde push approach.

6.1.9 Protocol's messages

Here are summarized the protocol's messages.

6.1.9.1 TRACKER_REQUEST

This is the first message, that a peer sends to the tracker, it contains:

- the hash of the file that the peer wants to download
- the ID of the peer, if it has already one

- *IN* bitmask of the peer (it will be used by tracker heuristics to choose the neighbors of the peer)
- *OUT* bitmask of that peer

6.1.9.2 TRACKER_RESPONSE

It contains a list of addresses of connected peers for that file and a new *ID* for peers that have not one yet.

6.1.9.3 CONNECTION_REQUEST

This is the first message of the application level handshake, it is sent by the active opener peer to the passive one. It contains the *IN* bitmask of the active opener. After the elapsing of a timeout, the active opener can resend it to the other peer.

6.1.9.4 CONNECTION_ACCEPT

It is the positive answer of the passive opener, and contains its *IN* bitmask. If it is lost the passive peers considers that connection established, while the active one do not. This anomaly will be resolved after some seconds because the passive opener will drop this connection thanks to a keep-alive policy that will be explicated later.

6.1.9.5 CONNECTION_REFUSE

This is sent in response to a CONNECTION_REQUEST when the passive peer has reached its maximum connection number so refuses a new one.

6.1.9.6 ALREADY_CONNECTED

This is sent in response to a doubled CONNECTION_REQUEST and means that the passive peer considers that connection already on, so it sends its *IN* bitmask, and the other peer can complete the handshake.

6.1.9.7 CLEAR_BM

IN bitmasks have initially all the bits set, because a peer needs all the pieces. Every time it decodes a piece, it clears the corresponding bit. This message contains the bits to be cleared. It is sent when a peer decodes a pieces, when it keeps receiving blocks

for a decoded pieces. There is a periodical check that can send again this message if a neighbor has not confirmed to have cleared that bits. It can be sent as a list of bits or as a bitmask, depending to which representation requires less bytes.

6.1.9.8 CLEAR_ACK

When a peer receives a CLEAR_BM message, it must inform the sender that it has successfully cleared those bits, so the sender keep track of it and avoids sending useless CLEAR_BM messages.

6.1.9.9 KEEP_ALIVE

These messages are exchanged by peers to confirm their presence in the network and to make the receiver reset the neighbor timeout for the sender. When a neighbor timeout elapses, that peer is considered disconnected.

6.1.9.10 TRACKER_KEEP_ALIVE

This message have the same use of the previous, but it is addressed to the tracker, with lower frequency. It can also be used to require new peers' address when the number of neighbors of a peer falls below a certain threshold.

6.1.9.11 BLOCK

This message contains a header and one encoded block. The header is made by these fields:

- Piece
  The piece this block belongs to.
- Seed
  It is the seed used by the sender to initialize the random generator to encode the block. So the receiver can do the same and obtain the same values without explicitly receiving them from the sender.
- Bloom filter
  Contains in a compressed format, the list of the peers that have already seen this packet. This is used to avoid to send twice the same block to the same peer.

6.1.9.12 TFRC_FEEDBACK

Since we use UDP, we use a rate control algorithm, namely the TFRC. So when a peer is receiving blocks from another one, it have to send it feedbacks to limit the sending rate.

6.1.9.13 QUIT

This message is sent when a peer leaves the network, and it informs others to delete it from their neighbor lists. The elapsing of the neighbor timeout have the same effect of the receiving of a QUIT message, and a QUIT message is sent to the elapsed peer, in the case it is already in the network.

6.1.10 ToroVerde versus BitTorrent

To sum up here we summarize the main differences between ToroVerde and the standard BitTorrent protocol. Clearly, the most remarkable difference is the introduction of rateless codes in a push based protocol. Using rateless codes peers are able to share partially downloaded pieces, while BitTorrent can starve waiting for completion of a piece. We have introduced the concept of priority class in the rarest first policy, avoiding bottlenecks that can arise when all peers try to download the same piece, i.e. the rarest, at the same time.
Both protocols implement a tit-for-tat strategies in order to prevent free riding and spreading the content more efficiently, but the number of connections opened by ToroVerde can scale according to the bandwidth estimates obtained by TFRC.

6.2 PlanetLab prototype and simulator description

In this section we describe implementation details for the C++ prototype we developed for deployment and testing on PlanetLab. We also present the software architecture for the simulator we developed. The simulator fully reproduces the protocol specifications and it is thus very accurate.

6.2.1 PlanetLab architecture

Here we briefly describe the ToroVerde's implementation architecture, a C++ prototype composed by a tracker and several peers. The tracker keeps track of connected peers and gives lists of addresses to the joining peers. The addresses returned by the tracker are selected at random from the whole list of connected peers. The architecture of the peer client is organized in several modules that carry out the various tasks. In the following we give a brief description of the most important modules.
We implement an outgoing queue that stores the messages sent to neighbors. This is an intermediate layer over UDP transport that organizes messages in 2 queues with different priority. This allows us to give a higher priority to small control messages with respect to larger coded data packets. We employ this intermediate layer as a mean to control the upload rate as well.
All planned actions are scheduled using an events list with corresponding timeouts. A dedicated thread is automatically executed each time a scheduled event occurs. Events can also be delayed or removed. Several functions of the protocol like keep alive messaging, retry policies and the refresh of *IN* bitmap towards neighbors are managed by means of scheduled events.
All the incoming messages are received by a thread that listens to the UDP sockets, parses the message payload and schedules the corresponding events. Input/output disk operations are all performed from the file manager module, that handles temporary files and updates the *IN* and *OUT* bitmaps. It maintains all received blocks and it is responsible for LT decoding. It also updates the popularity classes used by the rarest first algorithm. Finally, the set of known neighbors is maintained by the neighborhood manager. It is responsible for the choice of how many and which peers to enable.

6.2.2 Simulator of ToroVerde

We use the discrete-event simulator described in Chapter 5 for implements both BitTorrent and ToroVerde protocols. The simulation of ToroVerde is accurate and implements all protocol's strategies.

We verify the simulation's validity comparing these with the tests carried out on the PlanetLab in the same way we used for BitTorrent in Chapter 5. In these tests

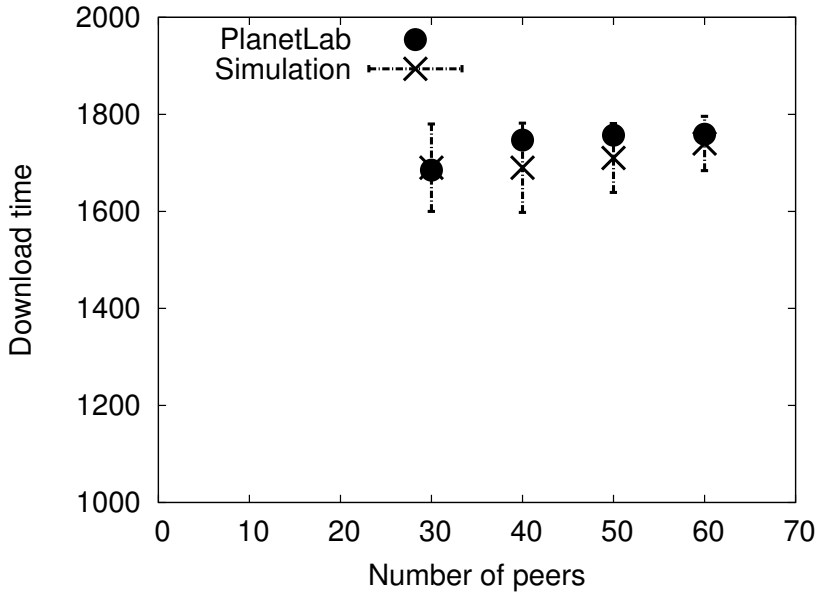


Figure 6.1: Donwloading times comparison between PlanetLab ToroVerde's clients and simulated ones

the overlay size was up to 60 and the file dimension 32 Megabyte. The validation was conducted in the second scenario described in the previous section.

Figure 6.1 shows the comparison between the simulated downloading times and real ones. We plot the mean and the confidence interval of 95% for simulation results and compare these with the PlanetLab's results. For network of 20, 30, 40 and 60 peers we can see how the simulated downloading times' measures fit with reals, and these latter are included in the confidence intervals.

6.3 System architecture

We developed a prototype in C++, composed by a tracker and several peers. The tracker is very simple, since it has only to keep track of connected peers and to give lists of addresses to the joining peers. The choice of which address include in the list is done at random. Peers are organized in several modules that carry out the various tasks.

6.3.1 Outgoing queue

This is an intermediate layer over UDP transport that organizes messages in 2 queues with different priority. This allows to send small control messages before the bigger ones, thus limiting their delay and to limit upstream rate. Indeed the task to takes messages from it and put them in the UDP socket is performed by thread that works at a fixed rate.

6.3.2 Event list

All timeouts and planned actions are considered as events. They are scheduled in an event list, and automatically executed by a dedicated thread. Events can also be delayed and deleted. They are used to manage several aspects like keep alive and retry policies and the refresh of IN bitmasks towards neighbors. It is performed every time a piece or when the timeout expires.

6.3.3 Message receiver

There is a thread that listens to the UDP sockets, parses messages and executes the correspondent actions.

6.3.4 File manager

It performs all I/O disk operations, manages temporary files and updates IN and OUT own bitmasks. It maintains all received blocks and it is responsible for the decoding of pieces. It also keeps up to date the popularity classes used by the rarest first algorithm.

6.3.5 Neighborhood manager

This module maintains the set of known neighbors and it is responsible for the choice of how many and which peers to enable.

6.4 Experimental results

In this section we present preliminary results we obtained from comparing ToroVerde and BitTorrent. We considered several scenarios characterized by different arrival and

Parameter	Value
Block size	1210 Bytes
Blocks per piece (k)	1000
c, δ	0.01
Max connection	40
Min connection	2
Connection timeout	25 s
Keep alive timeout	8 s
Buffer map refresh timeout	2 s
Initial N_{EN}	8
Min N_{EN}	4
Max N_{EN}	20

Table 6.2: System parameters for ToroVerde.

departure patterns and peer churning. We conducted experimental results on PlanetaLab for small sized overlay networks and resorted to simulations for medium-sized systems. Simulation results have been validated against measured performance on PlanetLab; at the same time, the simulator allowed us to optimize some of the system parameters for ToroVerde.

6.4.1 System parameters and scenarios

The experiments we carried out on PlanetLab required the definition of several system parameters for ToroVerde. The simulator we developed supported the optimization of several of them and led us to consider the values reported in Table 6.2 . The default client settings have been used for BitTorrent.

We considered 32MByte and 100MByte files to be distributed to N peers by one seed. It follows that ToroVerde segments the file in 28 and 87 pieces while BitTorrent uses 128 and 400 pieces (BitTorrent default settings for piece size is 256KByte) which are enough to exploit the benefits of the rarest first algorithm and piece distribution entropy. The number of blocks per piece in ToroVerde has been set to $k = 1000$ to obtain low overhead of LT codes. Furthermore, the block size (1210 Bytes) has been set to allow a block to be transmitted in a single UDP datagram.

Each peer starts to download the file using an upload bandwidth that is limited to 20KB/s in order to perform a fair comparison with the BitTorrent client that shares this limitation. This limitation is also useful to reduce congestion of PlanetLab nodes that in turn helps to minimize measurement problems.

In each scenario we consider we start with an overlay network containing a tracker and the initial seed, then we assume that up to N peers join the overlay network to download the file: the performance index we consider is the average time to complete the file transfer and the communication overhead computed over the total number of received bytes.

We consider three scenarios:

1. peers join the overlay network and cooperate until all peers completed their download. The arrival pattern is as follows: peers immediately join the overlay network at the startup representing a flash crowd arrival pattern. We denote this scenario as S1.

2. the same as in S1 but on the startup every peer waits for a random initial delay that is distributed according to a negative exponential probability with mean of 200 seconds (S2).

3. peers join and leave the network repeatedly alternating periods where they are connected (ON) and periods when they are not (OFF). Both ON and OFF periods are random; they are distributed according to a negative exponential probability with mean of 100 seconds. Peers start with an OFF period, i.e., they join the overlay network after an initial delay, and the initial seed remains for its whole lifetime in the overlay network. The arrival and departure pattern we consider is one with a high peer churn (S3).

6.4.2 Results from PlanetLab

The first set of results we present are obtained using a small-sized overlay network with $N = 40, 60$ peers. Table 6.3 shows the results for both ToroVerde and BitTorrent protocols in all scenarios we defined. In Table 6.3 average completion times and the estimated 95% confidence intervals are shown for BitTorrent and ToroVerde, along with the gain (in percentage) yielded by the latter protocol. We observe that ToroVerde is

always able to reduce the average completion time of peers. The best performance are obtained in the case of stable network (scenarios S1 and S2): in this case the reduction ranged from by 16% to 22% for a 32MB file while it ranged from 8% to 11% for the 100MB case. In the scenario S3 the advantages of ToroVerde are less evident; in fact, ToroVerde best performance is for the 32MB case with $N = 40$ where the reduction is equal to 6%.

This advantage is due to the ability of peers to spread also pieces in the downloading state that helps especially in the early stages of the file distribution; on the contrary, BitTorrent has to wait for completed pieces in order to forward data to neighbors. Nevertheless, ToroVerde incurs in a slightly higher than BitTorrent communication overhead; in particular, in the scenarios S1 and S2 we report a total overhead of about 4% in the 32MB case and about 3% in the 100MB case. BitTorrent is slightly more efficient: its overhead is 1.2% for the 32MB case and 0.8% in the 100MB case. The higher communication overhead in ToroVerde is because each UDP datagaram carrying a coded block must also contain some header bits for:

- TFRC signalling and data;
- seed of the random number generator used to select the packets combined in the LT coded block;
- Bloom filter.

The communication overhead in ToroVerde where the TFRC extra bits are filtered out from the total number of received bytes is equal to about 1.7%. It is worth pointing out that for BitTorrent TFRC functionality is not required because rate is shaped by means of TCP, that in turns requires higher protocol overhead than UDP. Furthermore, ToroVerde overhead can be reduced by using a shorter Bloom filter for each coded block or increasing the block size so as to reduce the overhead incurred by the header bits.

6.4.3 Simulation results

Using simulations, we conducted experiments in the same scenarios described for PlanetLab, scaling up to 300 peers. We believe that this size for the overlay is realistic; indeed, previous research showed that a very large number of BitTorrent swarms are very small-sized with average measured size of a few hundreds peer.

Table 6.3: Average completion times (with 95% confidence interval) measured on PlanetLab.

	40 peers			60 peers		
scenario	BitTorrent	ToroVerde	Diff	BitTorrent	ToroVerde	Diff
	32MB file size					
S1	2380 ± 5.37	1889 ± 8.4	-20%	2437 ± 3.1	1922 ± 7.9	-21%
S2	2082 ± 58.9	1750 ± 69.1	-16%	2324 ± 37.2	1821 ± 66.6	-22%
S3	1829 ± 61.1	1747 ± 58.1	-4%	1867 ± 77.0	1759 ± 59.2	-6%
	100MB file size					
S1	6137 ± 41.48	5661 ± 87.31	-8%	6510 ± 15.5	5769 ± 11.9	-11.4%
S2	5912 ± 221.08	5357 ± 380.99	-10%	6135 ± 186.6	5606 ± 16.6	-8.6%
S3	5452 ± 143.6	5179 ± 72.23	-5%	5721 ± 134.22	5492 ± 201.23	-4%

Table 6.4: Average completion times (with 95% confidence interval) for simulated protocols.

scenario	100 peers			300 peers		
	BitTorrent	ToroVerde	Diff	BitTorrent	ToroVerde	Diff
	32MB file size					
S1	2566 ± 44	1931 ± 6.58	-24%	2670 ± 32.3	2202 ± 22.7	-17%
S2	2317 ± 36.5	1816 ± 7.7	-22%	2412 ± 31.4	2005 ± 19.5	-17%
S3	1874 ± 22.46	1712 ± 8.8	-9%	2036 ± 15.8	1766 ± 4.5	-13%

The results for both BitTorrent and Toroverde are reported in Table 6.4 for the 32MB case. The simulation results confirm what we experimentally observed on PlanetLab. The reduction on the average download time ranges from 17% to 22% in scenarios S1 and S2. Simulation also reveals that in larger overlays ToroVerde yields larger gains also in the high churn scenario S3.

6.5 Conclusions

In this chapter we have presented a push-based P2P content distribution application exploiting the digital fountain concept through rateless codes. We defined a complete

specification of the ToroVerde protocol and provided a detailed description of all the employed strategies. We developed both a detailed simulator and a complete prototype that we used to perform a fair comparison between ToroVerde and BitTorrent. Early results suggest that average download delay can be reduced by ToroVerde in several realistic scenarios for small-to-medium sized files in overlays composed of a few hundred peers.

Chapter 7

P2P protocols and Internet Service Providers: a game theoretical approach

In this chapter we propose a game theoretic framework to help the design of techniques promoting the ISP cooperation in P2P streaming platforms and aiming at the minimization of inter-ISP traffic.

We first introduce the ISP game as two-ISP game and for this simple scenario we discuss the Nash equilibrium, the Pareto optimality, and a fairness criterion to refine the equilibrium points. Then we introduce the Evolutionary Game Theory model of the ISP game that allows us to investigate large scale systems. Finally we present a simulation based investigation that complements the theoretical studies we present.

7.1 Internet Service Provider awarness

In P2P applications, the participating nodes (or peers) form an overlay network which is largely agnostic on the underlying IP network. The overlay/underlay topology mismatch affects the performance of existing P2P platforms that can generate large volumes of unnecessary inter-ISP network traffic.
This increases the cost associated with P2P traffic for individual Internet Service Providers (ISPs) which face the problem to manage a vast amount of unnecessary inter-domain traffic (98). Such traffic level can cause congestion on valuable inter-ISP

links and more generally an overall performance degradation in an ISP network.

In this chapter we focus the attention on the role of the ISPs. In particular, we do not consider them as simple providers of network connectivity but as the main players for supporting P2P streaming platforms.

We can say that the P2P streaming is traditionally a game among two different kinds of players: the content provider and the users/peers. The existing proposals basically share the idea of helping one of these players, or in some case both of them, by providing information related with the underlying IP network. We move a bit further by introducing a third type of players, i.e, **the ISPs**. In our study the ISPs become active players of the P2P streaming by shifting their role from passive network providers to active game players that are able to modulate the network resources they require/offer to the P2P streaming platform. In particular, we propose a game theory framework to represent the ISP behavior and then, by using this framework, we investigate the characteristics of the interactions among the ISPs.

Closest to our proposal are the works (106) and (100), that study the need to localize P2P file sharing traffic within the ISP boundaries. The paper (106) proposes an overlay structure composed by mTrackers (similar in the spirit to the one described in Section 7.4) aiming at minimizing transit ISP costs. The paper (100) describes an ISP-friendly variant of BitTorrent to reduce the cross-ISP traffic.

Another paper that is related with the technique proposed in this chapter is (99). We share with this paper several model assumptions. In particular, we use the same fluid assumption whose consequences are that the peers (and also the streaming server), are able to share their upload among all the participant peers. That is, the stream is seen as a fluid that can be split in a continuous manner among all the requesting peers.

7.2 P2P Streaming: Peers and ISPs

In this section we describe the basic features of P2P streaming systems we address. Table 7.1 summarizes the notation used in this chapter.

The video originates from a server node; denote by c_s (in chunk per second (cps))[1] the streaming server upload capacity. Let r denote the rate of the video that has to

[1] In the following we express all the measures in terms of chunk per second (cps).

N	Set of peers in the system (with $\|N\| = n$)
r	Video streaming rate (in cps)
c_s	Streaming server upload capacity (in cps)
c_j	j-th peer upload capacity (in cps)
M_i	i-th partition of peers, with $i = 1, \ldots, k$, and $\|M_i\| = m_i$
I_i	Aggregate rate at which the stream flows into M_i
O_i	Aggregate bandwidth resource that partition M_i offers to the system
α_i	Locality control parameter for the i-th partition M_i
$u_i(\cdot)$	Payoff function for the i-th ISP

Table 7.1: Notation

be streamed to all participating peers. We assume a churnless system. Let N denote the set of peers in the system (with $|N| = n$), and with c_j the upload capacity of j-th peer, for $j = 1, \ldots, n$. The aggregate bit rate out of the server cannot exceed c_s, while the aggregate bit rate out of the j-th peer cannot exceed c_j, for $j = 1, \ldots, n$.

We use the definition introduced in (99), i.e., when all participating peers receive the video at rate r, we say that the system provides *universal streaming*. The system can perform universal streaming if it is possible to copy and route the bits so that all peers receive fresh bits at rate r. On the other hands, when the system is not providing universal streaming, we say that it works in degraded service mode. The following result presented in (99) defines the conditions that allow the universal streaming.

The maximum achievable streaming rate, $r_{\max}$, is given by

$$r_{\max} = \min \left\{ c_s, \frac{c_s + \sum_{j=1}^{n} c_j}{n} \right\}. \tag{7.1}$$

If the parameters are such that $r_{\max} \geq r$ the system is providing universal streaming.

Remarks. It is important to point out that the previous result is based on the following model simplifications: *i) fluid assumption*; *ii) complete connectivity*; *iii) churnless system*. The first assumption states that the stream is seen as a fluid and hence it can be divided among an arbitrary number of peers (in a continuous manner). Obviously this is a simplification because the stream is divided into a discrete number of packets

(or chunks). The second assumption states that the P2P overlay topology is a completely connected graph. The third assumption concerns the peers activity, i.e., the peers do not alternate between active and in-active states but they are always active.

Since we use Theorem 7.1 in many derivations we obviously inherit the used model simplifications. In Section 7.4 we discuss these simplifications, and their effects.

Let $M_1, M_2, \ldots, M_k$ be a partition of the set N, i.e., $\bigcup_{i=1}^{k} M_i = N$, and for any i and j (with $1 \leq i, j \leq k$), $M_i \cap M_j = \emptyset$. With m_i we denote the size of the i-th partition. A partition can be, for instance, an ISP, or an Autonomous Systems. Let $M_i \in N$ be a partition that does not include the streaming server. If the partition is an ISP, the connection point between the ISP and the remaining part of the set of peers, can be the link (or the links) connecting the ISP networks' to the rest of the Internet.

The aggregate rate that flows into the partition, denoted by I_i, is defined as the sum of all the contributions copied from peers that do not belong to M_i towards peers of M_i. To simplify the derivations we assume that the upload capacity of all the peers in M_i is equal to c_i.

If we denote by r_l the average rate at which a generic peer $p_l \in M_i$ receives the streaming, we can write

$$r_l = \frac{I_i}{m_i} + w_i,$$

where $\frac{I_i}{m_i}$ is the average contribution that p_l receives from peers outside of M_i, and w_i is the average contribution that p_l receives from peers of M_i.
If the system is providing universal streaming then $r_l \geq r$ and hence $I_i \geq m(r - w_i)$. To simplify the derivation we assume that the streaming system is so efficient that $r_l = r$, hence we have that

$$I_i = m_i(r - w_i). \tag{7.2}$$

To derive w_i we focus on a generic peer of the partition (that we call *tagged peer*), and we compute its average upload rate directed towards other peers of the same partition. Let focuses the attention on the possible destination peers that can be "served" by the tagged peer. According to the original assumptions of the Theorem 7.1, i.e., completely connected overlay topology, this set corresponds to to the entire set of peers (without the tagged peer to avoid self loops). In our derivations we can relax this assumption and this set can be a proper subset of N. Let $N' \subseteq N$ (with $|N'| = n'$) be the set of

all the neighbors of tagged peer and denote by $M'_i = N' \cap M_i$, the peers of N' that belong to the partition M_i (with $|M'_i| = m'_i$). If we assume that the destination peers are selected with a uniform probability distribution among the whole set of peers N' we can write

$$w_i = c_i \frac{m'_i}{n'}.$$

In the previous equation it can happen that $w_i > r$, and in this case Equation (7.2) assumes negative values. To avoid this we assume that

$$w_i = \min\left\{ c_i \frac{m'_i}{n'}, r \right\}, \tag{7.3}$$

that is, the average contribution that a peer may receive from the other peers belonging to the same partition cannot exceed the streaming rate r. The definition of I_i implies that if the sum of all the contributions from peers outside of M_i is smaller than I_i, then there are peers of M_i that receive the streaming rate at rate smaller than r!

Lets O_i the aggregate amount of bandwidth that partition M_i offers to the system. This is defined as the sum of all the possible contributions that the peers of M_i can offer to peers outside of M_i, i.e.,

$$O_i = m_i(c_i - w_i). \tag{7.4}$$

It may happen that the actual aggregate rate at which the stream flows out of partition M_i is smaller that O_i. This occurs when the amount of resources offered by M_i is greater than the that required by the system.

Furthermore, if the streaming server belongs to partition M_i this must be accounted by adding the server contribution to r_l. If we use the same assumptions of (99) (i.e., fluid assumption and completely connected overlay topology), the contribution of the streaming server per peer is equal to c_s/n, and from this it follows that $r_l = (I_i/m_i) + w_i + (c_s/n)$. Equations (7.2) and (7.4) turn out to be $I_i = m_i(r - w_i - (c_s/n))$ and $O_i = m_i(c_i - w_i) + (n - m_i)(c_s/n)$.

In Figure 7.1 we present some numerical results that illustrate the impact on the required bandwidth of several different parameters (e.g., the partition size m_i and the upload capacity c_i). For these results we assume that $w_i = c_i(m_i - 1)/(n - 1)$. In particular, the left plot of this figure shows three curves that represent the values of the ratio I_i as a function of m_i computed for different values of the upload capacity of

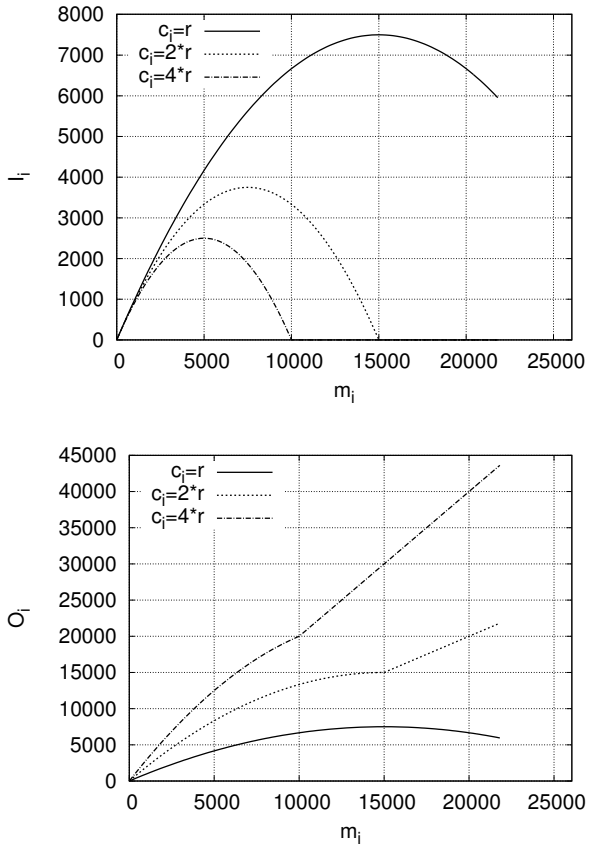


Figure 7.1: I_i (left plot) and O_i (right plot) as function of m_i for different values of upload capacity of the peers belonging to the partition, with $r = 1.0$ and $n = 30,000$ (note the effects of the min operator on the curves of the right plot)

the peers inside the partition. The rightmost figure shows the corresponding plot for the values of O_i. For these derivations we assumed that the streaming server does not belong to M_i.

The derivations of I_i and O_i are based on the assumption that the peers do not use any *proximity* information. A possible criterion that enforces locality among the peers belonging to M_i could be accounted by increasing the average contribution that a generic peer $p_l \in M_i$ receives from peers of the same partition. This can be obtained by defining

$$w_i(\alpha_i) = w_i + (c_i - w_i) \cdot \alpha_i, \tag{7.5}$$

where α_i is a parameter that controls the locality. That is, with $\alpha_i = 0$ we represent the random choice, i.e., there is no locality at all, and $w_i(\alpha_i) = w_i$. With $\alpha_i = 1$ represents the extreme case where all the chunks leaving the peers of M_i remain inside the partition. We can derive expressions for I_i and O_i that account for the locality, i.e.,

$$I_i(\alpha_i) = m_i(r - w_i(\alpha_i)). \tag{7.6}$$

and

$$O_i(\alpha_i) = m_i(c_i - w_i(\alpha_i)). \tag{7.7}$$

If the streaming server does not belong to the partition M_i, $I_i(\alpha_i)$ cannot be arbitrarily small (e.g., zero) because to ensure that all the peers of M_i receive the streaming rate (e.g., fresh bits) at rate r we must assume that $I_i(\alpha_i) \geq r$. If this does not hold, i.e., aggregate rate at which the stream flows into M_i is smaller than the streaming rate r, we have that the arrival rate of "fresh bits" to M_i is not sufficient to ensure that the peers in the partition receive the stream at rate r. This can be also formally verified by using a version of Theorem 7.1 restricted to the partition M_i. In particular, we can write that the maximum streaming rate, $r_{\max,i}$ that can achieved by the peers belonging to partition M_i is given by

$$r_{\max,i} = \min\left\{I_i, \frac{I_i + m_i \cdot c_i}{m_i}\right\}$$

Note that in the previous equation the connection point plays the same role played in Equation (7.1) by the streaming server. We can easily note that if, in the previous equation, $I_i < r$, this implies that $r_{\max,i} < r$. From this we must add a constraint on the possible values for α_i, i.e.,

$$\alpha_i \leq \frac{(m_i - 1)(r(n-1) - c_i\ m_i)}{c_i\ m_i(n - m_i)}.$$

In Figure 7.2 $I_i(\alpha_i)$ and $O_i(\alpha_i)$ are shown. We can see that the curves presented in Figure 7.2 allow one to point out the effects of the locality parameter α_i on both $I_i(\alpha_i)$ and $O_i(\alpha_i)$. In both cases we observe that enforcing the locality decreases $I_i(\alpha_i)$ and $O_i(\alpha_i)$.

From the simple derivations we made we can note that:

- Enforcing the locality is useful to reduce the required bandwidth. From the technical side there are several possible solutions to enforce the locality (see for instance (105) or (108)).

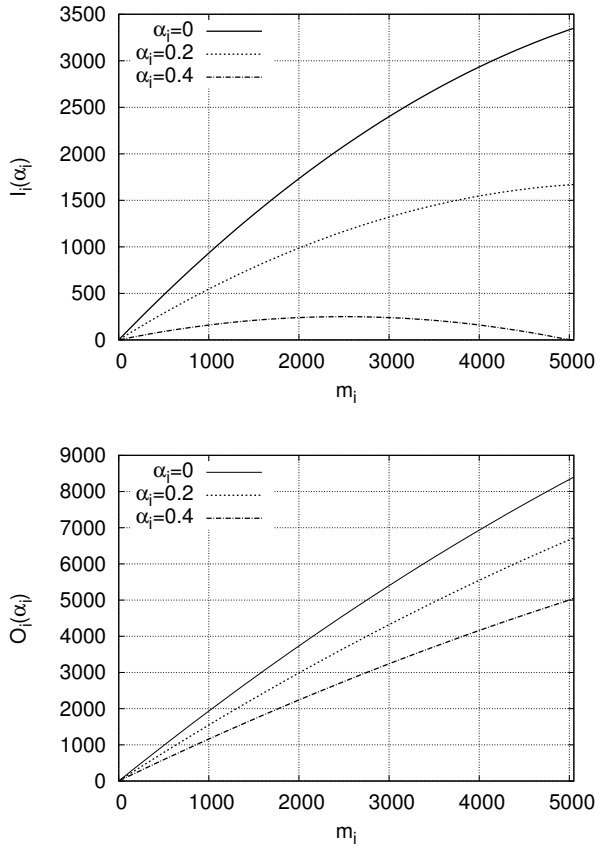


Figure 7.2: $I_i(\alpha_i)$ (left plot) and $O_i(\alpha_i)$ (right plot) as function of m_i with $r = 1$, $c_i = 2{*}r$, and $n = 30,000$ for different values of α_i

- From the ISP side the optimal solution occurs when $I_i(\cdot) = r$ and $O_i(\cdot) = 0$. This can be obtained by increasing the ISP redirection capabilities and dropping all the communications from peers that belong to the ISP towards peers that are outside (e.g., bandwidth throttling). It is clear that this *extreme* ISP free-riding behavior is not "socially" acceptable.

  Note that this ISP view of the P2P streaming system originates questions related with the role of free-riders, the cooperation among the participant ISPs, and so on. These questions are similar to those originated in P2P systems (file sharing and streaming) at peers' level; in the next section we will address these similarities and/or differences.

The ISPs point of view. According to the definition, I_i (for $i = 1, \ldots, k$) is the least value for the aggregate rate that M_i must receive from outside such that its peers receive a streaming rate at least equal to r. When the system is providing universal streaming, the real aggregate rate may be greater than I_i because the streaming rate received by each peer can be greater than r (this can be due to the overhead of the streaming platform, to duplicate chunks, and so on). On the other hand, if the current aggregate rate is smaller than I_i there are peers of the partition that do not receive a streaming rate at least equal to r. Obviously, with a significant churn the current aggregate rates are not constant but vary according to the overlay dynamics. If we assume that the streaming server belongs to the ISP-1 we can state that the system can provide universal streaming iff

$$\begin{aligned} m_1(c_1 - w_1) + (n - m_1)(c_s/n) + \sum_{i=2}^{k} O_i &\geq m_1(r - w_1 - (c_s/n)) + \sum_{i=2}^{k} I_i \\ (n - m_1)\frac{c_s}{n} + \sum_{i=1}^{k} O_i &\geq \sum_{i=1}^{k} I_i - m_1\frac{c_s}{n} \\ c_s + \sum_{i=1}^{k} O_i &\geq \sum_{i=1}^{k} I_i. \end{aligned} \tag{7.8}$$

It is interesting to observe that the previous inequality states that the universal streaming is possible if the amount of available resources (resources offered by the ISPs and offered by the streaming server) is greater than or equal to the amount of requested resources (by the ISPs)!

7.3 Game Theory Formulation

In this section we present several game models that involve peers and ISPs. In Section 7.3.1 we first introduce some simple models where the players are the peers of the P2P streaming platform. These models allow us to highlight peculiarities that will be also helpful to understand the ISP game models of Section 7.3.2. In Section 7.3.3 we introduce an analysis of the ISP game based on the Evolutionary Game Theory.

7.3.1 Peer Game Model

In this section we investigate the interactions among peers in a P2P streaming platform. We start with a simple scenario with two peers and the streaming server. We denote by

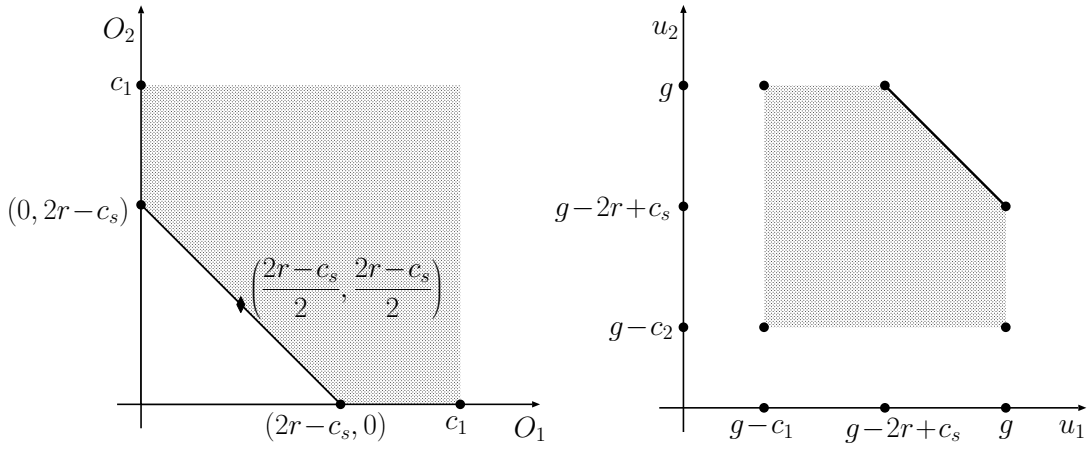


Figure 7.3: Set of values of O_1 and O_2 that allow universal streaming (left plot); set of positive payoff values (right plot)

c_1, c_2 the upload capacity of the peers while c_s is the upload capacity of the streaming server. We assume that $r < c_s < 2r$, i.e., the upload capacity of the streaming server is such that the two peers cannot be served only by the server but the cooperation between the peers is necessary.

In the following we denote by O_1 (resp. O_2) the maximum rate of bits from peer 1 towards peer 2 (resp. from peer 2 towards peer 1). Obviously we have that $0 \leq O_i \leq c_i$, for $i = 1, 2$. We can say that the peer i (for $i = 1, 2$) uses O_i to modulate its cooperation level. With these assumptions Theorem 7.1 states that both the peers receive the streaming at rate r (i.e., the system is providing universal streaming) iff

$$\frac{O_1 + O_2 + c_s}{2} \geq r.$$

We can use this condition to define the payoff function of the two-player game. Player i ($i = 1, 2$) may receive chunks both from the streaming server and from the other peer. We define the cost of peer i equal to O_i. If the condition of Theorem 7.1 are such that the system is providing a universal streaming (i.e., $r_{\max} \geq r$) then i receives a gain[1] of g. The term g quantifies the user satisfaction/un-satisfaction. That is, the peer receives a gain of g (resp. $-g$) if $\frac{O_1+O_2+c_s}{2} \geq r$ (resp. $\frac{O_1+O_2+c_s}{2} < r$). If (O_1, O_2)

[1]Without loss of generality we can assume that g is expressed in cps (chunk per second).

is the action profile, we can then define the payoff as

$$\begin{aligned} u_1(O_1, O_2) &= -O_1 + g \cdot 1\left(\frac{O_1 + O_2 + c_s}{2} \geq r\right), \\ u_2(O_1, O_2) &= -O_2 + g \cdot 1\left(\frac{O_1 + O_2 + c_s}{2} \geq r\right) \end{aligned} \tag{7.9}$$

where $0 \leq O_1 \leq c_1$ and $0 \leq O_2 \leq c_2$ and

$$1\left(\frac{O_1 + O_2 + c_s}{2} \geq r\right) = \begin{cases} 1 & \text{if } \frac{O_1+O_2+c_s}{2} \geq r \\ -1 & \text{otherwise} \end{cases}$$

The above payoff function consists of two terms: the first term denotes the cost (expressed in terms of bandwidth) that user i pays, while the second terms denotes the gain of user i. Note that the gain depends on the quantity of available bandwidth resources i offers, on the quantity offered by the other peer, and on the quantity of resources offered by the streaming server. The form of this dependency is defined by Theorem 7.1.

It is reasonable to assume that $O_1, O_2 < g$, since users will only cooperate with each other if cooperation can benefit both users and gives them positive payoffs. Let $u(O_1, O_2) = (u_1(O_1, O_2), u_2(O_1, O_2))$ be the payoff profile. Figure 7.3 (left plot) shows the set of values for O_1 and O_2 that allow the universal streaming (gray region), while the right plot shows the region with positive payoff values (gray region of the right plot).

We can see that all the values of O_1, O_2 that are on the segment between $(0, 2r - c_s)$ and $(2r - c_s, 0)$ in Figure 7.3 are *Nash equilibrium points (NE)*. In particular, if we assume that (x, y) is a point on this line segment and $u(x, y)$ is the corresponding payoff profile we have that for any $h > 0$

$$\begin{aligned} u_1(x, y) &= - x + g \\ u_1(x + h, y) &= - x - h + g \text{ and } u_1(x, y) > u_1(x + h, y) \\ u_1(x - h, y) &= - x + h - g \text{ and } u_1(x, y) > u_1(x - h, y). \end{aligned}$$

Since the same type of derivation can be done for $u_2(x, y)$, we can then conclude that any value (x, y) on the line segment between $(0, 2r - c_s)$ and $(2r - c_s, 0)$ is a NE point. On the other hand, we must point out that not all the payoff profiles on this line segment are "socially" acceptable. Note that there are points such as $O_1 = 0$ and

$O_2 = 2r - c_s$ (and vice versa). In other words the NE condition does not guarantee a fair contribution of the two peers. As first effort we try to refine these equilibrium points by using the Pareto optimality (103), and (104). In particular, we know that a payoff profile $u(O_1, O_2)$ is Pareto optimal if there is no other payoff profile $u(O_1', O_2')$ such that $u_1(O_1', O_2') \geq u_1(O_1, O_2)$ and $u_2(O_1', O_2') \geq u_2(O_1, O_2)$. Pareto optimality means that no one can increase his/her payoff without degrade other's. It is clear from Figure 7.3 that all the values on the line segment between $(0, 2r - c_s)$ and $(2r - c_s, 0)$ are Pareto optimal. Hence, not even the Pareto optimality allows us to discriminate among the values on the segment $(0, 2r - c_s)$ and $(2r - c_s, 0)$.
We postpone the introduction of the discrimination criterion that enforces a fair sharing among peer 1 and peer 2 after the discussion of the case when the bandwidth requests of the two peers are different.

It is worth to point out the difference between the peer game model we presented here and the one proposed in (101). In particular, the model presented in this reference assumes that the peers may have only two different cooperation levels, i.e., they can choose a non-cooperative or a cooperative behavior. Our model instead allows us with a "continuous range" of cooperation level (i.e., O_i may vary from 0 up to c_i). Thank to this continuous modulation of the cooperation level, the peer level model proposed here allow interesting studies that cannot be done with the models proposed in (101).

We now investigate a version of the game that is slightly different from the previous one. The only difference in this case is that the two peers have different rate requests, i.e., the peer 1 requires a video rate equal to r_1 and the peer 2 a rate equal to r_2, with $r_1 \neq r_2$ (e.g., $r_1 > r_2$). This version of the game could represent a P2P platform that uses a *Multiple Description Coding Technique* or *Scalable Video Coding* (96) and then the peers may have different rate requests. In this game c_1, c_2, and c_s are the upload capacity of the peer 1, of the peer 2, and of the streaming server respectively, while O_1 and O_2 are the maximum rate of bits from peer 1 towards peer 2, and from peer 2 towards peer 1. We assume that $r_1 < c_s < r_1 + r_2$. That is, the upload capacity of the streaming server is not sufficient to serve the two peers in a client-server manner (right most inequality), and on the other hand, if the left most inequality does not hold, there is no hope of having universal streaming (in this case the condition holds iff peer 1 receives the streaming at least at rate r_1 and peer 2 at least at rate r_2). With these

assumptions we have that the P2P streaming platform can offer a universal streaming iff

$$O_1 + O_2 + c_s \geq r_1 + r_2.$$

We can define the payoff as

$$\begin{aligned} u_1(O_1, O_2) &= -O_1 + g \cdot 1\left(O_1 + O_2 + c_s \geq r_1 + r_2\right), \\ u_2(O_1, O_2) &= -O_2 + g \cdot 1\left(O_1 + O_2 + c_s \geq r_1 + r_2\right) \end{aligned}$$

It is easy to see that we can repeat all the reasoning we made for the case with $r_1 = r_2$. In particular, we have that all the values on the line segment between $(r_1 + r_2 - c_s, 0)$ and $(0, r_1 + r_2 - c_s)$ are NE and also Pareto optimal. Also in this case we need a criterion that allows us to discriminate among the values on the line segment between $(r_1+r_2-c_s, 0)$ and $(0, r_1+r_2-c_s)$. In particular, we are interested in a fairness criterion that allocates the O_i-s proportionally to the amount of required resources. A similar fairness criterion has been proposed in (97) where it has been called *effort fairness*. A version of this criterion for our game provides us the following constraint

$$\frac{O_1}{r_1} = \frac{O_2}{r_2}.$$

With some simple algebraic manipulation we can derive that the values of O_1 and O_2 satisfying the previous condition are

$$\begin{aligned} O_1^* &= \frac{r_1(r_1 + r_2 - c_s)}{r_1 + r_2} \quad \text{and} \\ O_2^* &= \frac{r_2(r_1 + r_2 - c_s)}{r_1 + r_2}. \end{aligned}$$

Furthermore, it is easy to see that when $r_1 = r_2 = r$ we have that $O_1^* = O_2^* = \frac{2r-c_s}{2}$. Note that when $r_1 = r_2$ the effort fairness criterion we use corresponds to the proportional fairness used in (101).

7.3.2 ISP Game Model

In the following we consider an ISP version of the game where the players are the k ISPs. We start with the simple scenario with two ISPs, later on we generalize to the case with an arbitrary number of interacting ISPs. Without loss of generality we can assume that streaming server is in the ISP n. 1, i.e., partition M_1.

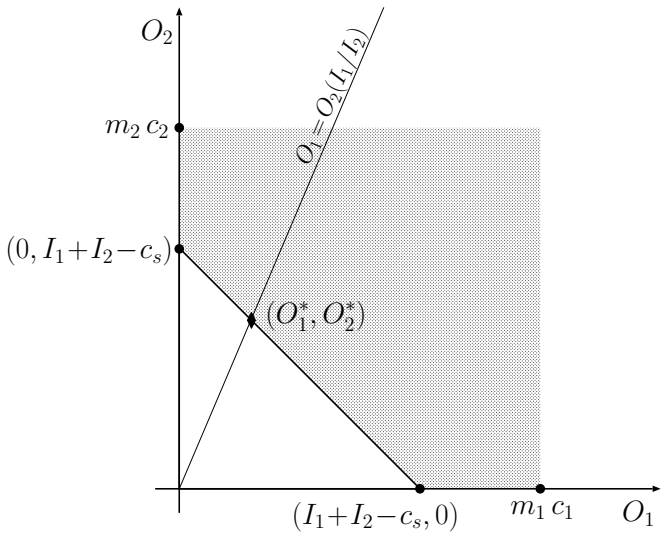


Figure 7.4: Set of values of O_1 and O_2 that allow universal streaming for the two ISPs system

By using Inequality (7.8) we can define the payoff function of the two ISPs game as follows:

$$\begin{aligned} u_1(O_1, O_2) &= -O_1 + g \cdot 1\,(c_s + O_1 + O_2 \geq I_1 + I_2) \\ u_2(O_1, O_2) &= -O_2 + g \cdot 1\,(c_s + O_1 + O_2 \geq I_1 + I_2)\,, \end{aligned} \tag{7.10}$$

If m_1 and m_2 and c_1, and c_2 are the number of participant peers and the upload capacity in the ISP-1 and ISP-2 respectively, we have that $O_1 \leq c_1 \cdot m_1$ and $O_2 \leq c_1 \cdot m_2$.

It is worth noticing that c_s does not represent a variable in Equation (7.10). That is, each ISP may vary the aggregate bandwidth that the peers of the ISP send to external peers, but the upload bandwidth of the streaming server c_s is fixed (or in other words it not part of the ISP game we are considering).

Figure 7.4 shows the set of values for O_1 and O_2 that allow the universal streaming for the two ISPs (gray region). By using the same arguments used for the peer level games we can see all the values on the segment line between the values $(I_1 + I_2 - c_s, 0)$ and $(0, I_1 + I_2 - c_s)$ are NE points and Pareto optimal. We can refine these equilibrium points by means of the effort fairness criterion

$$\frac{O_1}{I_1} = \frac{O_2}{I_2}$$

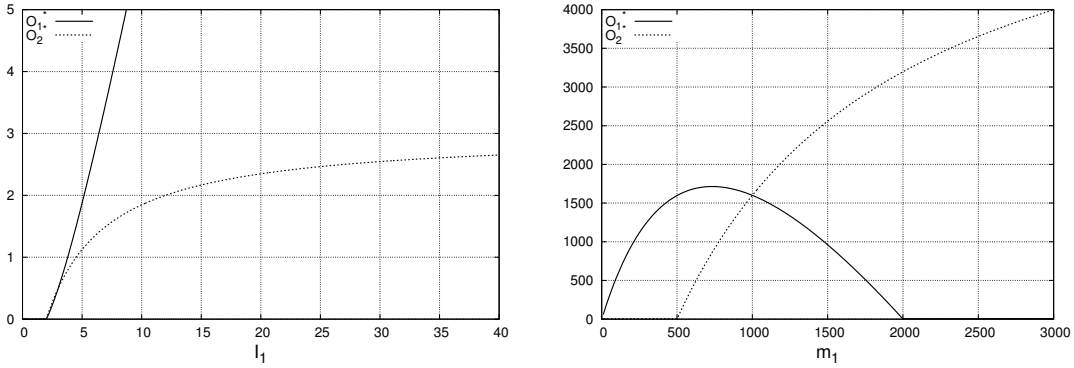


Figure 7.5: Set of values of O_1^* and O_2^* as function of I_1 (left plot), and as function of m_1 (right plot)

and then we have

$$O_1^* = \frac{I_1(I_1 + I_2 - c_s)}{I_1 + I_2} \text{ and } O_2^* = \frac{I_2(I_1 + I_2 - c_s)}{I_1 + I_2}. \tag{7.11}$$

The previous equations establish the relations among the several different quantities in the effort fair point. In particular, we can see what are the effects on the values O_1^* and O_2^* if the bandwidth request of ISP-1 grows. To this aim it is interesting to observe that $\lim_{I_1 \to \infty} O_1^* = \infty$ while $\lim_{I_1 \to \infty} O_2^* = I_2$. Figure 7.5 shows these behaviors when $I_2 = 3$ and $c_s = 5$. Note that the ISP bandwidth request I_i may change as a consequence of a variation of its redirection level, or of the upload capacity, or of the number of peers of the ISP. We must point out that Figure 7.5 (left plot) only shows the effects of the variation of I_1 on O_1^* and O_2^* but, if we consider that the I_i may change for several different reasons the interleaving of these effects may have unexpected (or non-intuitive) effects on O_1^* and O_2^*.

Figure 7.5 (right plot) shows one of these interesting effects. In particular, this figure shows the behavior of O_1^* and O_2^* as function of the size of the ISP-1. To this end we use Equation (7.2) to derive I_i as function of m_i. Hence we compute the values O_1^* and O_2^* as function m_1. For O_2^* we can observe a trend similar to the one observed in Figure 7.5 (left plot) but for O_1^* we observe a very different behavior. Note that this is not only due to the relation between I_i and m_i induced by Equation (7.2), but it is a more general trend that holds when I_i decreases as m_i increases. The situation depicted in this figure can be originated, for instance, by a strong variation in the size of one of the ISPs (for example due to a ISP market drift).

At this point we introduce the version of the game with k ISPs. Let $I_1, \ldots, I_k$ be the incoming aggregate rates of the k ISPs.
In this case the Nash equilibriums can be found on the hyperplane satisfying the following relation:

$$\sum_{i=1}^{k} O_i = \sum_{i=1}^{k} I_i - c_s.$$

Furthermore, the effort fair points $(O_1^*, \ldots, O_k^*)$ are the solution of the following system of equations

$$\begin{cases} \sum_{i=1}^{k} O_i^* = \sum_{i=1}^{k} I_i - c_s \\ \frac{O_i^*}{I_i} = h \qquad\qquad \text{for } 1 \leq i \leq k, \end{cases}$$

where h is a constant. From the second equations we can derive that

$$O_j^* = I_j \frac{O_i^*}{I_i} \quad \forall\, 1 \leq i, j \leq k$$

by substituting this in the first equation we obtain that

$$O_i^* + \sum_{j=1, j\neq i}^{k} I_j \frac{O_i^*}{I_i} = \sum_{j=1}^{k} I_j - c_s \quad \forall\, i.$$

From this equation with simple algebraic manipulation we can derive that

$$O_i^* = I_i \left(1 - \frac{c_s}{\sum_{j=1, j\neq i}^{n} I_j} \right) \quad i = 1, \ldots, k. \tag{7.12}$$

7.3.3 Evolutionary Analysis of Streaming Games

In this section we present a procedure that ISPs can use to compute the effort-fair values O_i^*, for each $i = 1, \ldots, k$. Note that these values can be computed by using Equation (7.12) but this requires the knowledge of all the values I_j for $j = 1, \ldots, k$, with $i \neq j$, i.e., besides its bandwidth request I_i each ISP has to know the I_j values of all the other ISPs. Unfortunately, an ISP does not disclose (to the other ISPs) information related with the services it offers to the subscribers (i.e., the I_i).

To avoid dissemination of its private information to all other ISPs we use a two tier structure of trackers composed by a *super-tracker* and a collection of *ISP-trackers*, one for each ISP. This kind of architecture is inspired to the tracker architectures proposed in (105) and (102).

In our proposal the ISP-i's tracker sends to the super-tracker the values I_i and the current value for O_i. Then the super-tracker is able to compute the indicator function $1(\cdot)$ and it can send the output of this function to the ISPs. In this manner, ISPs do not disseminate their I_i-s and O_i-s but they only send these values to a (trusted) super-tracker. Obviously the implementation of this type of tracker architecture must ensure that the ISP-trackers and the super-tracker mutually trust each other. Furthermore, this architecture could also be enhanced with measurement tools that help to avoid cheating behaviors. The two tier tracker architecture can be viewed a sort of "oracle" that ISPs can use to know if system is providing universal streaming or not.

By exploiting the two tier tracker infrastructure and by using ideas from Evolutionary Game Theory (EGT) (107) we device a distributed strategy to allow ISPs to compute the effort fairness point.

In EGT each player has a local information of the game. In particular, in case of the ISP game the local information are bandwidth request I_i, the current value for O_i, and a super-tracker to know if the system is providing universal streaming or not. The player learns which moves are better by trying them and looking how his utility function changes. The evolution of the system of players is ruled by equations called *replicator dynamic* (107). The replicator dynamic considers the evolution of each player behavior and shows the temporal evolution of players moves. In this way, if the game is repeated the replicator could reach an equilibrium, that is the equilibrium of the game.

The common EGT approach is to consider games with a finite number of strategies. On the other hand the game we propose has infinite strategies because O_i can take values over a continuous interval. For this reason we cannot use classical equation of replicator dynamic, but we create a new dynamic with the same kind of proprieties.

In the following we illustrate the method for the two ISPs game. In this case we look for a replicator dynamic that has an equilibrium in (O_1^*, O_2^*). The equation of ISP-i should depend only on O_i and I_i and from the knowledge of being in the universal streaming region or not (test performed by using the measured quantity $\overline{I_i}$). In particular, if the ISP is in the universal streaming region it should decrease its O_i

to increase its utility function. If we denote by O_i^s the value of the O_i at s-th iteration step and O_i^0 the initial values of the O_i (for $i = 1, \ldots, k$), we use a replicator dynamic described by the following equations

$$O_i^{s+1} = \begin{cases} O_i^s(1-\beta) & \text{if universal streaming} \\ O_i^s + \beta I_i & \text{otherwise} \end{cases} \tag{7.13}$$

where $0 < \beta < 1$ is a parameter that tunes the convergency speed. We can assume that the ISP-i decreases O_i proportionally to same value computed in the previous iteration. If the ISP has a high O_i it is probably far from the bound of the region, so it could decrease O_i a lot. If, instead, the ISP has a low O_i it is probably near by the bound, so it should pay attention on decreasing O_i. On the other hand, if the ISP is not in the universal streaming region it would try to increase his O_i faster to reach that region. In this case, it is reasonable that the ISP increases O_i depending on its request, i.e., I_i. We can see the evolution of the dynamic system with the help of Figure 7.3.3.

If we start the iterative method from a point (O_1^0, O_2^0) that is in the universal streaming region, then the system will move out the region in a radial way, i.e., following the line to the point $(0, 0)$. On the other hand, if we start from a point that is out of the universal streaming region then we will move along the vector $(\beta I_1, \beta I_2)$. This means that in this case we always move in the same direction, i.e., all the movements are parallel to the line $t^* : O_1 I_2 = O_2 I_1$. In this manner there is no need to know the bandwidth request of the other ISPs to choose its move (i.e., the I_j-s (for $j \neq i$)).

We must point out that with a similar iteration strategy the process will oscillate around the effort-fair points and the size of this fluctuation interval depends on β. That is, a small value of β yields smaller fluctuation interval but with lower convergency speed, and a large value of β yields higher convergency speed but larger fluctuation interval. To overcome this problem we can adopt a variable β, i.e., a value of β that is function of s and of the stability of convergency process. We empirically derive a suitable updating strategy for the values of β i.e., the value of B fluctuates less when the value of O_i is close to the effort-fair point.

In the following we prove that the iterative method converges to the effort-fair points. Each ISP converges to its own O_i^*

Proof. For the sake of readability we only present the proof only for the case $n = 2$ (the proof for $n > 2$ follows the same ideas but with the use of a cumbersome notation).

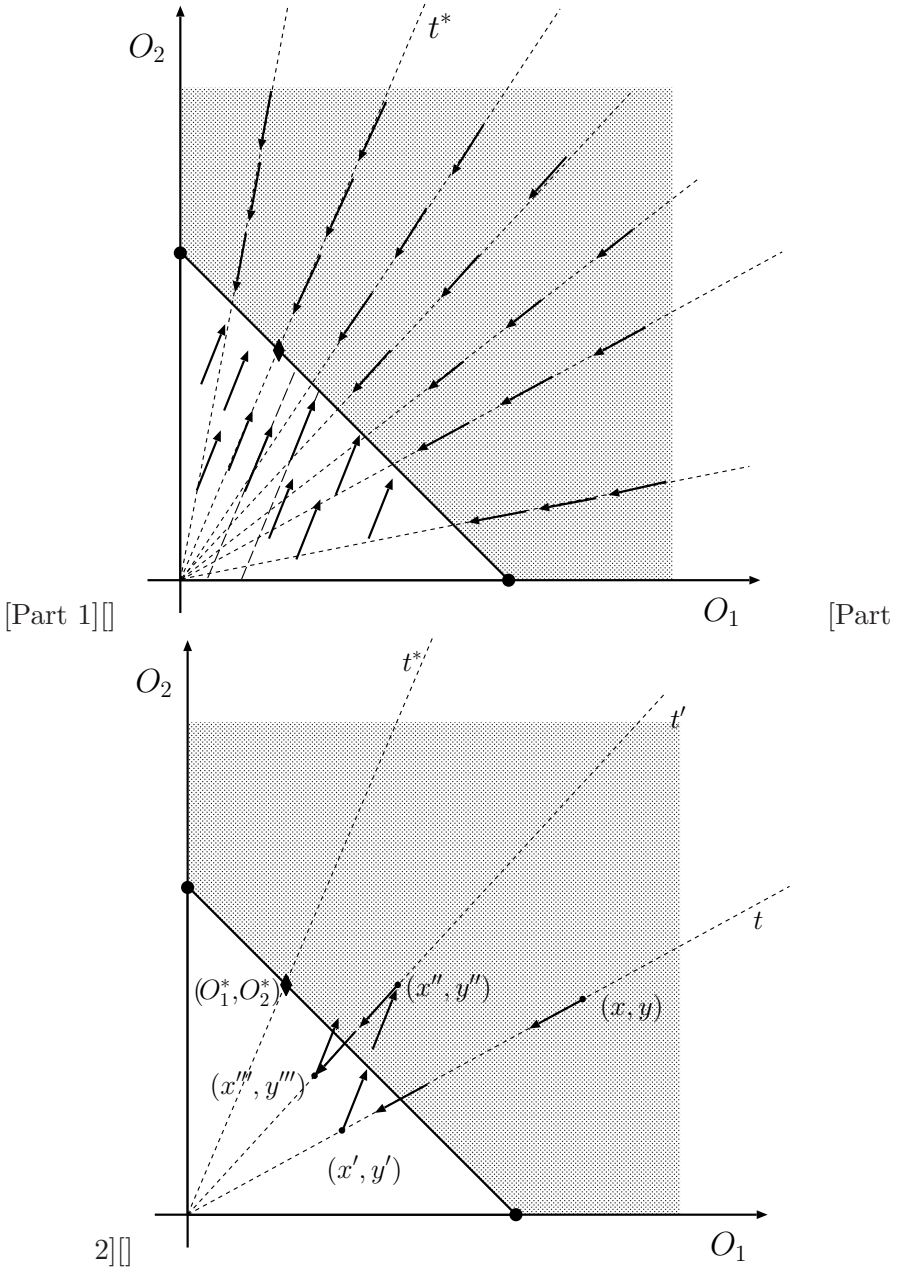


Figure 7.6: (a) Vector field of the dynamic system (the symbol ♦ denotes the point (O_1^*, O_2^*)); (b) Evolution of the replicator dynamic

First of all, we show that the line $t^* : O_1I_2 = O_2I_1$ *is a fixed line of the replicator dynamic, i.e., if the staring point of the iteration, i.e.,* $p^{(0)} = (O_1^0, O_2^0)$ *is a point on the line* t^* *then for any other iteration step* s *we have that* $p^{(s)} = (O_1^s, O_2^s) \in t^*$. *If* $p^{(0)} \in t^*$ *then*

$$p^{(0)} = \left(O_1^0, \frac{I_2}{I_1}O_1^0\right).$$

now there are two cases:

- $p^{(0)}$ *is in the universal streaming region and then*

$$p^{(1)} = p^{(0)} - \beta p^{(0)} = \left(O_1^0(1-\beta), \frac{I_2}{I_1}O_1^0(1-\beta)\right),$$

  so $p^{(1)} \in t^*$.

- $p^{(0)}$ *is not in the universal streaming region. In this case we have that*

$$p^{(1)} = p^{(0)} + \beta(I_1, I_2) = \left(O_1^0 + \beta I_1, \frac{I_2}{I_1}(O_1^0 + \beta I_1)\right),$$

  and again $p^{(1)} \in t^*$.

We can conclude that t^* *is a fixed line of the replicator dynamic.*
We prove now that the line t^* *is an attractor of the dynamic system. Let* $p = (x, y)$ *be a generic point (see Figure 7.3.3) in the universal streaming region. In this case the system will move in a radial direction along the line* t *until it falls out of the universal streaming region, e.g., in the point* $p' = (x', y')$. *This point is not in the universal streaming region, so the system will move along a new line, parallel to* t^*, *until it returns into the universal streaming region. Let* $p'' = (x'', y'')$ *be the new point reached by the system. The system will move along a new radial line* t', *that is closer to* t^* *with respect to the previous line* t. *Going on we have that the radial line tends to* t^*, *so we can conclude that* t^* *is an attractor of the system.*
Finally, we have that the system oscillates around the effort-fair point (O_1^*, O_2^*). *Indeed, each* O_i^s *will oscillate into the interval*

$$[O_i^* - \beta O_i^*, O_i^* + \beta I_i].$$

This means that using an opportune function for decreasing β *we obtain for each ISP-i the convergence to its effort-fair point* O_i^*.

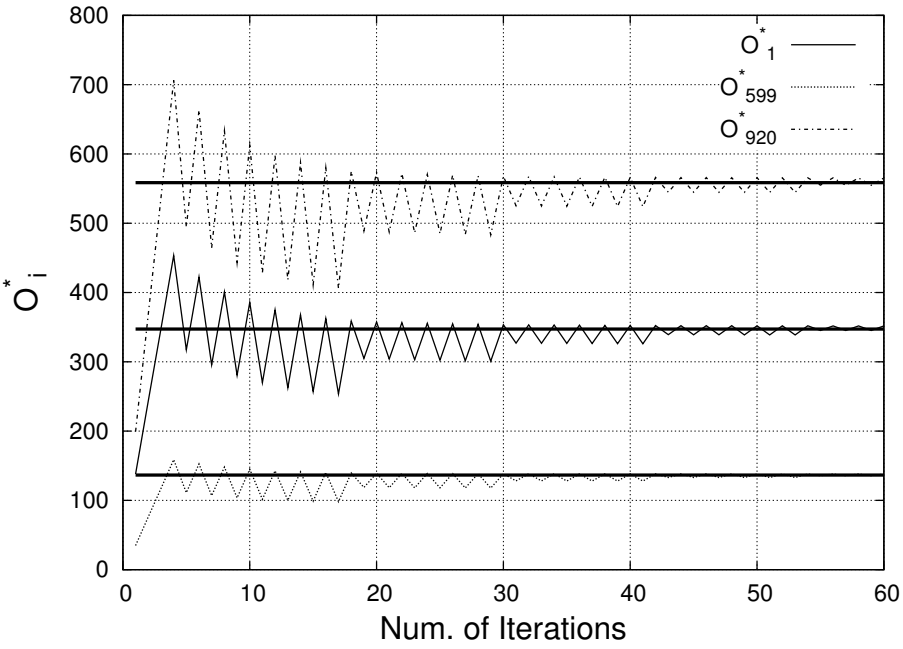


Figure 7.7: Numerical Experiment with $k = 1,000$, $r = 50$ (cps), $c_s = 50 \cdot r$, $m_i \in$ Unif$[100, 1000]$, and $I_i \in$ Unif$[r, m_i \cdot r]$

Numerical Example. To show that the proposed numerical technique can be used for "realistic" system sizes we investigate its performance on the following system. We assume that the streaming rate r is equal to 50 (cps)[1] and the number of ISPs k is equal to $1,000$. To compute the values I_i, for $i = 1, \ldots, k$ we use the following idea. We choose the values m_i, i.e., the number of peers in each ISP, by selecting them from a uniform distribution in the interval $[100, 1000]$. From m_i we derive the values I_i by choosing them from a uniform distribution in the interval $[r, m_i \cdot r]$ (these two values correspond to the minimum and the maximum of I_i). The streaming server capacity is $c_s = 50 \cdot r$.

To emulate the realistic scenario we assume that each ISP-i can use a "black box" function that receives as input the I_i and O_i^s (s is the iteration step) and gives as output the answer "the ISP-i is providing universal streaming" or "the ISP-i is *not* providing universal streaming". This function emulates the test that the ISP can perform by using the measured incoming aggregate streaming rate.

[1] If we assume that a chunk is $\sim$ 1 KB then a streaming rate $r = 50$ cps corresponds to $\sim$ 400 kbps.

Figure 7.7 shows the evolution of three values of O_i^* derived by applying our EGT based approach. For each of the presented O_i^* the figure also shows the corresponding theoretical values computed by using Equation (7.12).

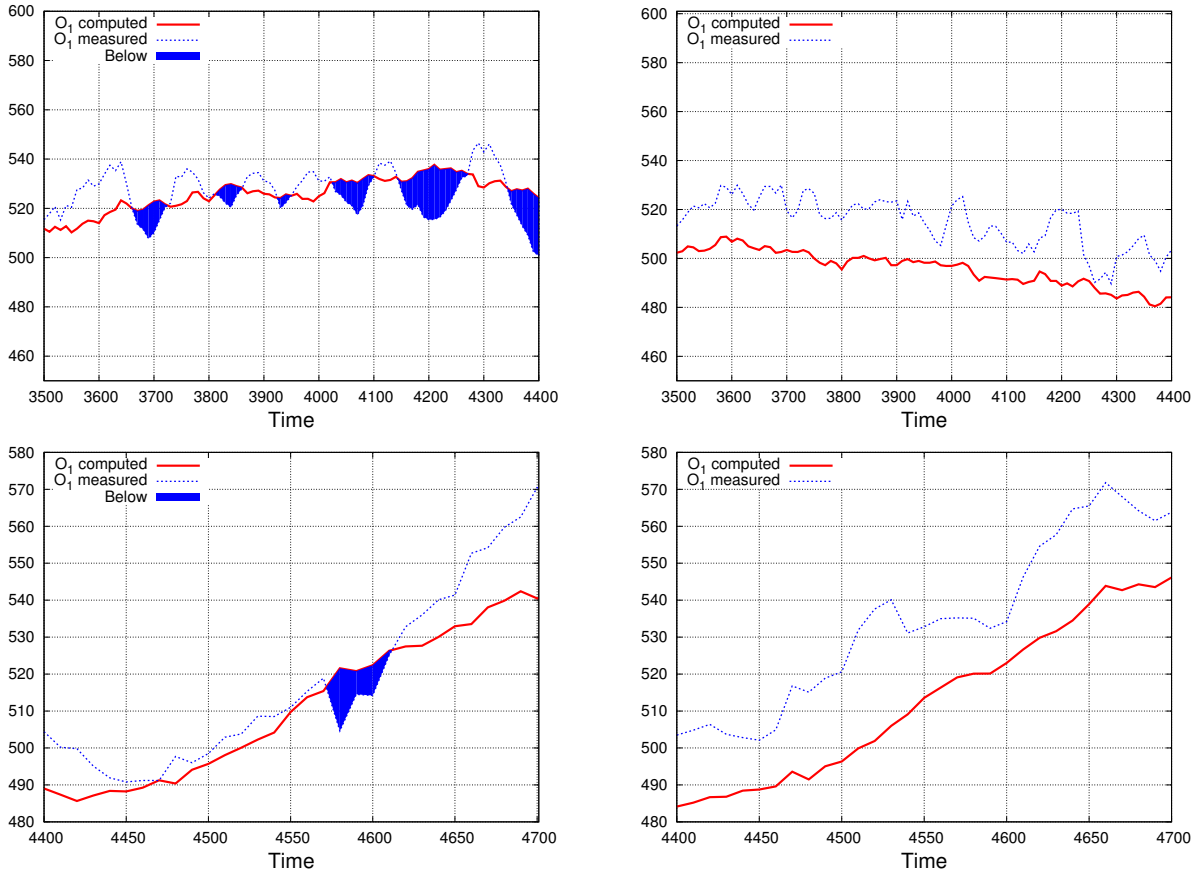


Figure 7.8: Two simulation traces reporting the values of O_1 (computed and measured) with the following parameters: $\lambda_1 = 1.0$, for $t = [0..4500]$, and $\lambda_1 = 1.5$ for $t > 4500$, $\lambda_2 = 1.0$, $\mu_1 = \mu_2 = 0.001$, $c_1 = c_2 = 2 \cdot r$, $\delta = 30$ secs. The left upper and lower plots report the traces in the interval $[3500, 4400]$ and $[4400, 4700]$. The right upper and lower plots report the simulation traces in the same interval with the same set of parameters but when the super-tracker increases the I_i-s of 5%. The filled regions denote time intervals where the system is not able to satisfy the ISP bandwidth demands

7.4 A ISP Streaming Traffic Management Algorithm

In Section 7.3 we introduced a simple game-theoretic framework but in all the models we present the equilibrium points can only be achieved via common knowledge. That is, each ISP/peer needs to know the streaming server upload capacity and the I_i-s of all the other ISPs. Obviously this hypothesis is not realistic! The models we developed only allow us to investigate the existence and the peculiarities of the equilibrium points. Nevertheless, by using the ideas presented in Section 7.3.3 we can define a distributed algorithm that implements the EGT version of the game without the constraints that the ISP must be aware of the I_i-s of all the other ISPs.

The proposed distributed algorithm could be implemented by means of a two tier structure of trackers (similar in the spirit to the tracker architecture proposed in (105) and (102)). At the top level there is the *super-tracker* and at the lower level there is a collection of *ISP-trackers* (one for each ISP). Each ISP-tracker must be able to measure the current values of *i)* the number of active peers in the ISP, *ii)* the number of peers that receive the video at rate r, and *iii)* the values for the incoming and outgoing aggregate rates. Furthermore, the ISP-tracker is able to modify the bandwidth request and the bandwidth offered to the other ISPs. This could be done by increasing/decreasing the locality parameter (e.g., the parameter α_i of Equation (7.6)). The bandwidth offered to the other ISPs could also be modified by using bandwidth throttling.

Each ISP-tracker periodically (e.g., each δ sec) sends the collected measures (i.e., the number of active peers, the number of peers that receive the video at rate r, and the measured incoming and outgoing aggregate rates). With these measures the super-tracker knows if the system is in a universal streaming region or not, and hence it is able to play the EGT-game described in Section 7.3.3. In particular, the super-tracker sends an appropriate message (e.g., `universal-streaming` or `no-universal-streaming`) to the ISP-trackers asking them to change the values of the O_i-s according to Equation (7.13).

In the basic schemata we introduced the super-tracker, that is the only agent accumulating global knowledge. We assume that all ISP-trackers trust each others. Using this architecture an ISP does not need to know the I_i-s of all the other ISPs. Moreover, the super-tracker may cheat on purpose by increasing the values of I_i so as to make the players contribute slightly more than needed to anticipate the effects of peer churn

or mass arrival/departure phenomena. Obviously, we must say that many un-touched important problems still remain. A non exhaustive list contains issues such as incentive mechanisms to enforce the cooperation among the ISPs, control techniques to check the ISP honesty, confidentiality of the communications between the super-tracker and ISP-trackers, and so on.
Note that the two-tier architecture composed by the super-tracker and the collection of ISP-trackers only aims at the ISP traffic management and does not replace the P2P architecture but it only helps the ISPs in the traffic management. In other words, if the P2P streaming architecture uses a tracker similar to the BitTorrent protocol to assist the streaming distribution (e.g., to provide peer addresses) this tracker does not interfere with the ISP traffic management.

A Simulation Study. To verify the effectiveness of the proposed schemata in a realistic scenario , we develop a discrete time simulator that implements the proposed strategy.
With the simulator we try to capture in an abstract manner the peculiarities of a generic P2P streaming platform. In particular, the peers are grouped into k ISPs.

We assume that both the inter-arrival times and those of service (i.e. how much a peer remains connected) follow a negative exponential distributions, with parameter λ_i and μ_i. We also assume that a peer can immediatly join the network, without waiting. Using a notation common in queueing theory we denote the number of peers in the i-th ISP at time t with $m_i(t)$, and the i stochastic processes are independent $M/M/\infty$ $(m_i(t), t \geq 0, \text{with } i = 1, \ldots, k)$.

In the simulator, we assume that the upload capacities can be shared among a discrete number of peers, i.e., we do not use the fluid assumption. Hence each peer (and also the streaming server) can forward the streaming to a limited number of other peers.
The tracker keeps track of all active peers. When a (new) peer p joins the system it contacts the tracker (the P2P tracker) to receive a list of peers that are already in the overlay (possibly including the streaming server). The peer p contacts a subset of peers of this list. For each contacted peer p' if its upload capacity is not already saturated then peer p establishes a link with p'. On the other hand, if the upload capacity of p' is already allocated the connection is refused. When the number of incoming links

connecting peer p with other peers reaches a pre-defined threshold (i.e., b links) this means that p receives the video streaming at rate r.

When a peer p leaves the system all the peers with incoming arcs from p must re-wire the broken link towards another active peer having enough available upload capacity. During this search for new partners the peer p does not receive the video streaming at rate r.

On the top of this simple simulator we implement the distributed algorithm with a super-tracker and k ISP-trackers (one for each ISP). Each δ secs each ISP-tracker sends the measured metrics to the super-tracker (i.e., the number of active peers, the number of peers that receive the video at rate r, and the measured incoming and outgoing aggregate rates). The super-tracker plays the EGT-game described in Section 7.3.3 and sends to the ISP-trackers the message `universal-streaming` or `no-universal-streaming` to ask them to change the values of the O_i-s according to Equation (7.13). Note that since the simulation does not use the fluid assumption the simulator implements a discrete approximation of Equation (7.13).

The main goal of our simulation is to demonstrate that the game's logic is still valid in more realistic conditions. In particular, we show that the distributed algorithm is able to play the EGT-game described in Section 7.3.3 even if the peer behaviors dynamically change the available/requested bandwidth resources. Moreover, we note that it is indeed possible to achieve the same results of those described in Section 7.3.3 without the model simplifications such as the fluid assumption, complete connectivity, and in presence of discrete values of the upload capacities, the parameter β of Equation (7.13), and so on.

In the following we present some qualitative results that allows us to illustrate the some of the peculiarities of the proposed algorithm.

In particular, we perform experiments with two ISPs. The two stochastic processes representing the arrival and the departure of the peers are two independent $M/M/\infty$ with arrival rate $\lambda = 1$, and service rate $\mu = 0.001$. The upload capacity of all nodes is assumed equal to $2r$, where r is the video streaming rate. We assume that the chunk size is such that a peer may correctly reproduce the streaming video when it has at least 10 incoming arcs, and each peer may have no more than 20 outgoing arcs towards other peers. The super-tracker and the ISP-trackers exchange information each 30 secs.

Figure 7.8 (upper left plot) shows a portion of a simulation trace that reports the values O_1 **computed** by using Equation (7.13) and the one **measured** by counting the outgoing arcs from peers belonging to ISP-1 towars peers of the other ISP. In this plot we emphasize the periods where the measured O_1 is smaller than the computed value. In these time intervals the amount of resources that ISP-1 is smaller than the requests and the peers of the other ISP may not receive the streaming video at the required rate. The right upper plot allows us to illustrate a simple and naive possible solution to mitigate this problem. The basic idea is that the super-tracker receives the values of the I_i but plays the EGT-game with increased values, say $\hat{I}_i$ (with $\hat{I}_i = I_i + 0.05 * I_i$). In other words, the super-tracker cheats the ISP-trackers, and hence it injects in the system spare capacity to smooth the effect of peer dynamics.

To evaluate the robustness of the algorithm to quick variations of the bandwidth requests, e.g. because of mass arrivals, during the experiments, at time $t = 4500$, we increase the parameter λ_1 to 1.5. Figure 7.8 (lower left plot) reports the values of O_1 (computed and measured) during the time interval $[4400 - 4700]$, i.e., during the redoubling time interval. We do not report the behavior for $t > 4700$ because the system stabilizes (even if $\lambda_1 = 2\lambda_2$). Figure 7.8 (lower right plot) shows the behavior of the algorithm when the super-tracker uses increased values for I_i (i.e., it works with $\hat{I}_i = I_i + 0.05 * I_i$).

We must point out that the presented preliminary simulation study only provides some idea on the qualitative behavior of the proposed schemata. A deeper study must be done to improve the knowledge of the phenomena under study. A non exhaustive list of the issues that must be investigated includes the definition of metrics to measure the time intervals where system is not able to satisfy the ISP requests, the evaluation of the impact of peer dynamics (e.g., impact of statistical characteristics of the arrival and departure processes), the impact of the parameter β, the relations among the amount of spare capacity injected by the super-tracker (by cheating the ISP-trackers) and the peer dynamics, and so on.

7.5 Conclusions

In this chapter we have presented a game theory framework that can be used for the design of ISP strategies aiming at providing efficient and reliable support for P2P

streaming application. Our framework allows us to illustrate the existence of equilibrium points, the role of possible strategies to refine these points, and how to use ideas from the evolutionary game theory to derive techniques, that a player of the game can use, to computed the (operational) points assuming only limited knowledge on the state of the other players.

The presented simulation results show that the algorithm that implements the EGT-game is able to face the impact of peer dynamics, and it can help the ISP traffic management, also considering the modeling simplifying assumptions.

7. P2P PROTOCOLS AND INTERNET SERVICE PROVIDERS: A GAME THEORETICAL APPROACH

Chapter 8

The impact of P2P protocols on ASes network topology

In this chapter we investigate how the resources' diffusion generated by a P2P file-sharing overlay influence costs and rewards of Autonomous Systems (ASes) organized in a network topology. First we briefly overview the Internet AS-level and BGP-4, the external routing protocol used to exchange reachability information among ASes. Then we introduce our model for both AS-level and P2P overlay. Finally we present some experiments showing how our model is able to compute costs and rewards for complex network topologies, taking into account several parameters like peering and transit agreements, inter-AS routing and different AS size.

8.1 The Internet AS-level

A rigorous definition of AS is given in RFC 1930 (115), Section 3, that describes it as a group of IP prefixes sharing a unique and clearly defined routing policy. Here "prefix" is referred to a CIDR block, a group of one or more classful networks (A, B or C networks). Usually routers that belong to the same AS are under the same technical administration, share common metrics to route packets within the AS, and use an inter-AS protocol to forward network messages to other. An AS appears to its neighbors having a single coherent internal routing plan, and announces to them routes that are reachable through it.

In Internet each AS is uniquely identified by an Autonomous System Number

(ASN), a 32 bit integer as specified in RFC 4893 (126). ASN are assigned by the Internet Assigned Numbers Authority (IANA) to Regional Internet Registry (RIRS). RIRSs are responsible for specific geographic areas and in turn assign ASNs to organizations that make request.

The protocol used nowadays to exchange routing and reachability informations is the Border Gateway Protocol (BGP) (114, 122, 123, 125). The current version is BGP-4. BGP-4 is an exterior gateway protocol used for intra-AS routing, in contrast to the interior gateway protocols, like Routing Information Protocol (RIP) (116) or Intermediate System To Intermediate System (IS-IS) (121), used within the same AS. Each AS runs some BGP-4 gateways that discover routes exchanging reachability informations with the other gateway. Each one announces the destinations (i.e, other ASes), that are reachable through it.

BGP is a *Path Vector Protocol*, so gateways exchange full AS-paths, according to their routing policies. Policies determine what are the best paths to reach a specific destination. A common parameter is the length of AS-path (124), preferring shorter over longer ones, but the AS administration can specify more complex policies.

8.1.1 Traffic Agreements: peering and transit

According to the BGP Topological Model (cf. RFC 1655 (124), Section 2), a direct connection between two AS is both a physical connection (i.e., a shared network formed at least from one border gateway for each AS) and a BGP session running on the border routers (gateways). A connection is demanding in terms of economic resources (hardware, maintenance, technical administration), so commercial agreements are settled between directly connected ASes.

The two established methods to exchange Internet traffic between directly connected networks are **peering** and **transit**. In peering two networks do not charge any fees to each other, while a transit accord occurs when an AS pays to another some fees to reach other parts of Internet. In this case the traffic travel across the seller AS and is forwarded to the next hop to destination.

Peering and transit agreements can influence the way messages are routed on the ASes topology because BGP policies used from a network are based on economic and commercial considerations (123). Indeed is convenient to an AS announces routes for other networks it peers only to its customers. Usually traffic between two peering

partners is not forwarded. In the same way it is not allowed to route traffic from peering partners to seller ASes and vice-versa. The reasons behind these policies are simple to understand: if an AS announces these kind of routes, it would be providing free transit over its network for its peers or buy transit from another network and giving it away freely to a peer. For sake of clarity we illustrate some allowed paths in Figure 8.1 (page 116):

- AS_8 can see all the networks because networks AS_5, AS_6, AS_7 buy transit from it.
- AS_1 can see AS_2 and its customers directly, but not AS_3 through network AS_2.
- Traffic from AS_4 to AS_7 is routed by AS_6 , but not through AS_5.
- AS_4 can see Network B through its peer AS_5, but not via its transit customer AS_2.

Peering and transit accords also induce a hierarchy on ASes topology. At the top there are **Tier 1** networks (AS_8 in the example in Figure 8.1), that sell transit traffic to all its partners and can reach every other AS without pay any settlements or buy transit traffic. A **Tier 2** network (the other ASes in Figure 8.1) buys transit traffic but has also some peering agreements. Finally sometimes is used the term **Tier 3** to refer to networks that only purchase transit traffic.

8.2 Modelling Autonomous Systems

We are interested in studying the costs that must be sustained, and the gains that can be achieved from the agreements set among the ASes. In particular, starting from a formal description of the network topology, we want to be able to estimate gains and costs for each AS induced by a particular traffic scenario.

We consider a network composed by N ASes. We use a *traffic matrix* $\mathbf{X}$ to represent the data flows between the ASes:

$$\mathbf{X} = \begin{bmatrix} x_{11} & \dots & x_{1N} \\ \vdots & x_{ij} & \vdots \\ x_{N1} & \dots & x_{NN} \end{bmatrix}. \tag{8.1}$$

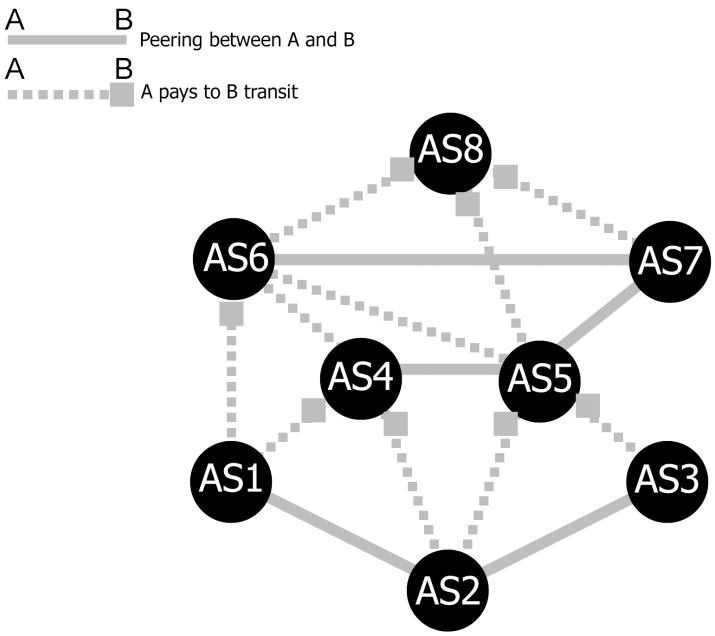


Figure 8.1: ASes topology used for experiments

Each element x_{ij} is the amount of traffic exchanged on the logical link between the AS i and the AS j. A logical link between i and j is a BGP session over a physical connection between two gateways. The physical connection might spread over several physical links. The network topology is described by a graph G:

$$G = (V, E_p, E_t, \sigma : (\mathbb{N} \times \mathbb{N}) \rightarrow \mathbb{R}). \tag{8.2}$$

Element $V = \{AS_1, \ldots, AS_N\}$ contains the vertexes of the graph that corresponds to the N ASes i.e., the list of ASes ids. Two type of agreements, *pure peering* and *transit*, can be settled on a link. Two ASes connected with a peering link do not charge any fee each other to communicate over that link. Instead in a transit agreement the customer node i pays to the seller node j. To take this issue into account, edges of the graph are defined as ordered pairs and grouped in two disjoint sets E_p and E_t. For each peering aggrement between ASes i and j, both the pairs (i, j) and (j, i) are inserted in E_p, while a transit accord is represented from the pair (i, j) in E_t, where i is the AS id of the customer and j the seller's one.

Despite agreements are confidential, it is a reasonable assumption that cost is proportional to the traffic carried on links. These costs are computed using the relations σ. It associates to an ordered pair (i, j) the fee payed from AS_i for a traffic unit carried on the link from AS_i to AS_j. The relation returns this cost as a real number if there is a transit aggrement between AS_i and AS_j with the first as customer, 0 otherwise.

Using the G representation of the ASes network we build a *Cost Matrix* $\mathbf{C}^k$ and a *Reward Matrix* $\mathbf{R}^k$ for each AS. A value $c_{ij}^k \neq 0$ means that the AS_k is involved in the communication between i and j, and c_{ij}^k is the cost it paids for a traffic unit that travel from AS_i to AS_j. In the same way the *Reward Matrix* $\mathbf{R}^k$ defines how much AS_k gains when two other ASes communicate.

The Algorithm 8.2.1 describes the function *BuilCostRewardsMatrices()* that computes $\mathbf{C}^k$ and $\mathbf{R}^k$ from the network description G. In Algorithm 8.2.1 line (i) is used the function *DijkstraModified()* (Algorithm 8.2.2) to find the shortest path between any pair of ASes. This function is a modified version of the Dijkstra algorithm that solve the all sources shortest path problem considering transit and peering aggrements to eliminate not allowed routes (see Section).

The purpose of Algorithm 8.2.2 is to build the predecessor matrix P. The element P(i,j) of this matrix stores the predecessor of the AS_j on the shortest route to it from AS_i. The funcion repeat the classical single source Classical Dijkstra algorithm for all vertices using the *AllowedNeighbors()* (Algorithm 8.2.2 line (i)) to restrict the neighborhood to the nodes that are on allowed paths. Algorithm 8.2.3 take into account that an AS does not annouce routes between peering patners, between peering ASes and transit seller or between two seller nodes (Algorithm 8.2.2) line (ii)). Instead it will announce routes for all its neighbors to its customers nodes (Algorithm 8.2.2) line (i)).

Once computed all the shortest allowed routes (Algorithm 8.2.1 line (i)), the matrices $\mathbf{C}^k$ and $\mathbf{R}^k$ are built using the predecessor matrix P to retrieve the path among each pair of ASes. If between two nodes that are on the route from AS_i to AS_j there is a transit aggrement, the fees computed using the σ relation are stored in the C_{ij}^k element of the customer AS_k and in the R_{ij}^s reward matrix of the seller AS_s (Algorithm 8.2.1

lines (ii)(iii)).

Algorithm 8.2.1: BUILDCOSTREWARDMATRICES(V, E_p, E_t, σ)

$P \leftarrow DijkstraModified(V, E_p, E_t)$ (i)
for each $(i, j) \in (V \times V), i \neq j$
do
- $k \leftarrow j$
- **while** $P(i, k) \neq NULL$
  - $p \leftarrow P(i, k)$
  - **if** $(p, k) \in E_t$
    - **do** $\{R_{ij}^k = \sigma(p, k);\ C_{ij}^p = \sigma(p, k)$ (ii)
  - **if** $(k, p) \in E_t$
    - **do** $\{C_{ij}^k = \sigma(k, p);\ R_{ij}^p = \sigma(k, p)$ (iii)
  - $k \leftarrow p$

Algorithm 8.2.2: DIJKSTRAMODIFIED(V, E_p, E_t, σ)

$D(i, i) \leftarrow 0, \forall i \in V$
$D(i, j) \leftarrow \infty, \forall (i, j) \in V \times V, i \neq j$
$P(i, j) \leftarrow NULL, \forall (i, j) \in V \times V$
for each $i \in V$
do
- $Q \leftarrow V$
- $u \leftarrow \arg\min\{D(i, j) : j \in Q\}$
- **while** $(Q \neq \emptyset) \wedge (D(i, u) < \infty)$
  - **do**
    - $Q \leftarrow Q \setminus \{u\}$
    - $S \leftarrow Q \cap AllowedNeighbors(E_p, E_t, P(i, u), u)$ (i)
    - $D(i, j) \leftarrow \min\{D(i, j), D(i, u) + W(u, j)\}, j \in S$
    - $P(i, j) \leftarrow \arg\min\{D(i, p) + W(p, j)\} : p \in \{P(i, j), u\}, j \in S$

return (P)

Algorithm 8.2.3: ALLOWEDNEIGHBORS(E_p, E_t, p, u)

if $(p \equiv NULL) \vee ((p, u) \in E_t)$
  do $neighbors \leftarrow \{j : (u, j) \in (E_p \cup E_t)\} \cup \{j : (j, u) \in (E_p \cup E_t)\}$ (i)
  else
  do $neighbors \leftarrow \{j : (j, u) \in E_t)\}$ (ii)
return $(neighbors)$

For sake of clarity, here we report an example releated to the network showed in Figure 8.2. The topology is made of six ASes connected with six links. There are some routes that are not allowed e.g, $AS_1 \overset{L_1}{\rightarrow} AS_2 \overset{L_4}{\rightarrow} AS_3$ becouse AS_2 does not announce routes between its peering patners. As results of these forbideen paths AS_3 is connected

only to AS_1, AS_2 and AS_4, but not with AS_5 and AS_6. AS_2 case is particular interesting in this network becouse it is directly connected to all ASes and involved in all communications that originate from $i \in \{AS_1, AS_2, AS_3\}$ and ends in $j \in \{AS_5, AS_6\}$. All routes that involve AS_2 and a transit aggrement influence its cost and reward matrices. In the following we enumerate all the route from AS_1 to other that involve AS_2:

$$
\begin{aligned}
Route_{12} &: AS_1 \xrightarrow{L_1} AS_2 \\
Route_{13} &: AS_1 \xrightarrow{L_2} AS_4 \xrightarrow{L_3} AS_2 \xrightarrow{L_4} AS_3 \\
Route_{15} &: AS_1 \xrightarrow{L_2} AS_4 \xrightarrow{L_3} AS_2 \xrightarrow{L_5} AS_5 \\
Route_{16} &: AS_1 \xrightarrow{L_2} AS_4 \xrightarrow{L_3} AS_2 \xrightarrow{L_6} AS_6
\end{aligned}
$$

E.g., when AS_1 send a packet to AS_6, it follows the $Route_{12}$, so AS_2 gain a quantity $\sigma(4,1)$ from AS_4 for the usage of the link L_3 but pays $\sigma(2,6)$ to AS_6 for the traffic that travel on L_6. In Figure 8.2 we report the C^2 and R^2 matrices computed for the AS_2.

Once computed the C^k and R^k matrices for each AS, it is simple retrieve the total cost C_k sustainded and the total reward R_k for the AS_k simply using:

$$
C_k = \vec{1} \cdot (X \circ C^k) \cdot \vec{1}' \cdot \text{ and } R_k = \vec{1} \cdot (X \circ R^k) \cdot \vec{1}' \cdot
$$

where $\circ$ define the entry wise product between matrices and the $\cdot$ is the standard row-by-column matrix product.

8.3 P2P Resource Diffusion

In this section we describe the probabilistic model of the P2P traffic travelling among the considered ASes. We first define the peer-to-peer scenario (Section 8.3.1), then we provide the analytical representation of peers (Section 8.3.2). In Section 8.3.3 we study the evolution of the system in order to characterize the resource diffusion.

8.3.1 Network Scenario

We consider that peers are distributed across N different ASs. We assume that the total number of peers in the system is a discrete random variable that follows a given

Table 8.1: Model Notations

Notation	Description
N	Number of Autonomous Systems (AS)
s_i	Probability that a peer belongs to the i-th AS
$G(z)$	Generating function of the distribution of the number of peers in the network
α_i	Probability that a peer in the i-th AS holds the resource
β_i	Probability that a peer in the i-th AS wants the resource
γ_i	Probability that a peer in the i-th AS holds the resource but do not share it
ξ_i	Probability that a peer in the i-th AS does not share the resource after getting it
n_i	Number of peers holding the resource in the i-th AS
p_i	Number of peers requiring the resource in the i-th AS
q_i	Number of peers holding but not sharing in the i-th AS
u_i	Generating function of the number of peers holding the resource in the i-th AS
v_i	Generating function of the number of peers requiring the resource in the i-th AS
w_i	Generating function of the number of peers holding but not sharing in the i-th AS

probability distribution $p(m)$. We express this distribution with its generating function $G(z)$:

$$G(z) = \sum_{m=0}^{\infty} p(m) z^m \tag{8.3}$$

We only take into account peers that can participate in the diffusion, i.e., peers that either hold or request the resource. We denote by s_i the probability that a peer is in the i-th AS. In each AS peers are divided into three different classes: a) peers holding the resource and available for sharing it, b) peers requiring the resource, and c) peers holding the resource, but not sharing it i.e., freeloaders. The class of each peer is determined randomly, according to a given initial probability: α_i for class a), β_i for class b) and γ_i for class c) (with $\alpha_i + \beta_i + \gamma_i = 1$). The number of peers of the three classes are respectively denoted by n_i, p_i and q_i. We denote by ξ_i the probability that a peer that gets the resource decides to not share it. Peers requiring the resource can either get it from peers lying in the same AS or from peers belonging to others ASs. All model notations are summarized in Table 8.1.

8.3.2 The Model

We represent the P2P system by introducing the distribution of the number of peers and its corresponding generating function.

We call $\Pi(n_1...n_N, p_1...p_N, q_1...q_N)$ the joint distribution of the number of peers in each class, for each of the N ASs. The generating function $g(\cdot)$ of this distribution can be computed from parameters $G(z)$, s_i, α_i, β_i and γ_i as:

$$g(u_1...u_N, v_1...v_N, w_1...w_N) = G\Big(\sum_{i=1}^{N} s_i\Big[\alpha_i u_i + \beta_i v_i + \gamma_i w_i\Big]\Big). \tag{8.4}$$

An intuitive interpretation of Equation (8.4), is that each peer randomly chooses both its AS and its class with probability $s_i\alpha_i$, $s_i\beta_i$ or $s_i\gamma_i$, which corresponds to $z = \sum_{i=1}^{N} s_i\Big[\alpha_i u_i + \beta_i v_i + \gamma_i w_i\Big]$.

The marginal distribution corresponding to the i-th AS is defined as:

$$\Pi_i(n_i, p_i, q_i) = \sum_{\substack{n_1...n_{i-1}, n_{i+1}...n_N,\\ p_1...p_{i-1}, p_{i+1}...p_N,\\ q_1...q_{i-1}, q_{i+1}...q_N}} \Pi(n_1...n_N, p_1...p_N, q_1...q_N) \tag{8.5}$$

We call $g_i(u_i, v_i, w_i)$ the generating function of $\Pi_i(n_i, p_i, q_i)$. Using the properties of the generating function and (8.4), we have that:

$$\begin{aligned} g_i(u_i, v_i, w_i) &= g(1...1, u_i, 1...1, 1...1, v_i, 1...1, 1...1, w_i, 1...1) = \\ &= G\Big(s_i\Big[\alpha_i u_i + \beta_i v_i + \gamma_i w_i\Big] + 1 - s_i\Big) \end{aligned} \tag{8.6}$$

where we set to 1 all the transformed variables except the ones corresponding to the i-th AS.

We can compute the probability that a peer in the i-th AS holds the resource, given that the AS is not empty (by empty we mean that there are no peers that can participate in the resource diffusion as mentioned in Section 8.3.1) as:

$$\bar{\alpha_i} = \sum_{n_i+p_i+q_i \neq 0} \frac{n_i}{n_i + p_i + q_i} \Pi_i(n_i, p_i, q_i). \tag{8.7}$$

It can be shown that $\bar{\alpha_i}$ can be computed in the following way:

$$\bar{\alpha_i} = \int_0^1 \Big[\frac{\partial g_i(u_i, v_i, w_i)}{\partial u_i}\Big]_{u_i=y, v_i=y, w_i=y} dy. \tag{8.8}$$

Indeed we can show that we can obtain (8.7) from (8.8) by solving step by step the right term of the equation (see Appendix A, Proof 2).

If we calculate (8.8) with the definition (8.6) we get:

$$\bar{\alpha_i} = \alpha_i(1 - g_i(0,0,0)) \tag{8.9}$$

Hence we can write:

$$\alpha_i = \frac{\bar{\alpha_i}}{(1 - g_i(0,0,0))} = \frac{\bar{\alpha_i}}{(1 - G(1 - s_i))} \tag{8.10}$$

The term $1-G(1-s_i)$ corresponds to the probability that the i-th AS is not empty. The same computation can be made for β and γ:

$$\beta_i = \frac{\bar{\beta_i}}{(1 - g_i(0,0,0))},\ \bar{\beta_i} = \int_0^1 \Big[\frac{\partial g_i(u_i, v_i, w_i)}{\partial v_i}\Big]_{\substack{u_i = y \\ v_i = y \\ w_i = y}} dy \tag{8.11}$$

$$\gamma_i = \frac{\bar{\gamma_i}}{(1 - g_i(0,0,0))},\ \bar{\gamma_i} = \int_0^1 \Big[\frac{\partial g_i(u_i, v_i, w_i)}{\partial w_i}\Big]_{\substack{u_i = y \\ v_i = y \\ w_i = y}} dy \tag{8.12}$$

8.3.3 System dynamics

We now study the evolution of parameters α_i, β_i and γ_i, and show how they characterize the resource diffusion in the system. In particular, we model the resource diffusion among the ASs with an *embedded time* process: time is not considered explicitly, instead is modeled by a discrete variable m that increases of one unit whenever a resource transfer is completed. With this assumption we compute α_i^m, β_i^m and γ_i^m, that correspond to the values of parameters α_i, β_i and γ_i at time m.

Parameters α and β vary only due to a resource transfer. For sake of simplicity, in this paper we neglect peers that give up requesting the resource and peers that quit sharing it. However these assumptions could be easily removed by adding new parameters and difference equations on α and β. We expect that as time tends to the infinity, every requests will be satisfied, that is (for any i-th AS):

$$\lim_{m\to\infty} \alpha_i^m = \alpha_i^0 + (1 - \xi_i)\beta_i^0 \tag{8.13}$$

$$\lim_{m\to\infty} \beta_i^m = 0 \tag{8.14}$$

$$\lim_{m\to\infty} \gamma_i^m = \gamma_i^0 + \xi_i\beta_i^0 \tag{8.15}$$

where α_i^0, β_i^0 and γ_i^0 represent the initial system parameters.We are interested in studying the evolution of α_i^m, β_i^m and γ_i^m until all transfers are completed. Note that changes in these parameters affect the joint distribution of the number of peers per class, i.e. $\Pi^m(n_1..n_N, p_1..p_N, q_1..q_N)$.

We can define $\bar{\alpha}_i^{m+1}$ as function of the system parameters at time m:

$$\begin{aligned}\bar{\alpha}_i^{m+1} = & \sum_{\substack{(\sum_k p_k = 0 \vee \\ \sum_k n_k = 0) \\ n_i+p_i+q_i \neq 0}} \frac{n_i}{n_i+p_i+q_i}\Pi^m(n_1..n_N, p_1..p_N, q_1..q_N) + \\ & + \sum_{\substack{(\sum_k p_k \neq 0 \wedge \\ \sum_k n_k \neq 0) \\ n_i+p_i+q_i \neq 0}} \left[\frac{n_i+1}{n_i+p_i+q_i}\frac{p_i}{\sum_k p_k}(1-\xi_i) + \right. \\ & \left. + \frac{n_i}{n_i+p_i+q_i}\left(1-\frac{p_i}{\sum_k p_k}(1-\xi_i)\right)\right] \cdot \\ & \cdot \Pi^m(n_1..n_N, p_1..p_N, q_1..q_N)\end{aligned} \tag{8.16}$$

The first addendum on the r.h.s. accounts for the case in which no transfer occurs since either all requests in the system have been satisfied and there are no more peers requiring the resource ($\sum_k p_k = 0$), or there are no resources in the system ($\sum_k n_k = 0$). The second addendum on the r.h.s. considers the case where a resource is actually transferred. If the destination of the transfer is the i-th AS and the considered peer is not a freeloader (with probability $\frac{p_i}{\sum_k p_k}(1-\xi_i)$), n_i is increased by one (first term in square brackets), otherwise n_i remains constant (second term in square brackets).
By developing (8.16) we obtain:

$$\begin{aligned}\bar{\alpha}_i^{m+1} = & \sum_{n_i+p_i+q_i \neq 0} \frac{n_i}{n_i+p_i+q_i}\Pi^m(n_1..n_N, p_1..p_N, q_1..q_N) + \\ & + \sum_{\substack{(\sum_k p_k \neq 0 \wedge \\ \sum_k n_k \neq 0) \\ n_i+p_i+q_i \neq 0}} \frac{1}{n_i+p_i+q_i}\frac{p_i}{\sum_k p_k}.\Pi^m(n_1..n_N, p_1..p_N, q_1..q_N)(1-\xi_i)\end{aligned}$$

From (8.7) we can write:

$$\bar{\alpha}_i^{m+1} = \bar{\alpha}_i^m + \Delta_i^m(1-\xi_i) \tag{8.17}$$

where we define

$$\Delta_i^m = \sum_{\substack{(\sum_k p_k \neq 0 \wedge \\ \sum_k n_k \neq 0) \\ n_i + p_i + q_i \neq 0}} \frac{1}{n_i + p_i + q_i} \frac{p_i}{\sum_k p_k} \Pi^m(n_1..n_N, p_1..p_N, q_1..q_N) =$$

$$= \sum_{\substack{\sum_k p_k \neq 0 \\ n_i + p_i + q_i \neq 0}} \frac{1}{n_i + p_i + q_i} \frac{p_i}{\sum_k p_k} \Pi^m(n_1..n_N, p_1..p_N, q_1..q_N) -$$

$$- \sum_{\substack{\sum_k p_k \neq 0 \\ n_i + p_i + q_i \neq 0}} \frac{1}{p_i + q_i} \frac{p_i}{\sum_k p_k} \Pi^m(0..0, p_1..p_N, q_1..q_N). \tag{8.18}$$

$\Delta_i^m(1-\xi_i)$ represents the variation of $\bar{\alpha}_i$ at time m. In the same way the evolution of $\bar{\beta}_i$ and $\bar{\gamma}_i$ can be calculated as:

$$\bar{\beta}_i^{m+1} = \bar{\beta}_i^m - \Delta_i^m \tag{8.19}$$

$$\bar{\gamma}_i^{m+1} = \bar{\gamma}_i^m + \Delta_i^m \xi_i \tag{8.20}$$

Δ_i^m can be computed exploiting the generating function representation of the number of peers in the system. We define $\bar{p}_i = \sum_{k \neq i} p_k$ and we denote with $\bar{v}_i$ its corresponding transformed variable. It can be shown that (See Appendix A, Proof 3):

$$\Delta_i^m = \int_0^1 \left[\int_0^1 \left[\frac{\partial}{\partial v_i} \hat{g}_i(u_i, v_i, w_i, \bar{v}_i) \right]_{\substack{v_i = xy \\ \bar{v}_i = x}} dx \right]_{\substack{u_i = y \\ w_i = y}} dy. \tag{8.21}$$

where

$$\begin{aligned} \hat{g}_i(u_i, v_i, w_i, \bar{v}_i) = g(1...1, u_i, 1...1, \bar{v}_i...\bar{v}_i, v_i, \bar{v}_i...\bar{v}_i, 1...1, w_i, 1...1) - \\ g(0...0, \bar{v}_i...\bar{v}_i, v_i, \bar{v}_i...\bar{v}_i, 1...1, w_i, 1...1) = \\ = G\Big(1 + s_i\Big[\alpha_i(u_i - 1) + \beta_i(v_i - \bar{v}_i) + \gamma_i(w_i - 1)\Big] + B(\bar{v}_i - 1)\Big) - \\ G\Big(1 - A + s_i\Big[\beta_i(v_i - \bar{v}_i) + \gamma_i(w_i - 1)\Big] + B(\bar{v}_i - 1)\Big) \end{aligned} \tag{8.22}$$

and $A = \sum_k s_k \alpha_k$ and $B = \sum_k s_k \beta_k$.

By calculating (8.21) with (8.22) we obtain:

$$\Delta_i^m = \int_0^1 \frac{s_i\beta_i}{s_i\beta_i(y-1)+B}\Big[G\left(1+s_i(y-1)\right) - \\ -G\left(1-B+s_i(1-\beta_i)(y-1)\right) - \\ -G\left(1-A+s_i(1-\alpha_i)(y-1)\right) + \\ +G\left(1-A-B+s_i\gamma_i(y-1)\right)\Big]dy. \tag{8.23}$$

Finally, from (8.17), (8.19), (8.20) and (8.23) we are able to compute $\alpha_i^m, \beta_i^m, \gamma_i^m$ for any step m and for each AS i. Given the initial parameters α_i^0, β_i^0 and γ_i^0 we have, by applying (8.10), (8.11) and (8.12), that:

$$\alpha_i^m = \frac{\bar{\alpha}_i^m}{(1-G(1-s_i))} \tag{8.24}$$

$$\beta_i^m = \frac{\bar{\beta}_i^m}{(1-G(1-s_i))} \tag{8.25}$$

$$\gamma_i^m = 1-(\bar{\alpha}_i^m+\bar{\beta}_i^m) \tag{8.26}$$

8.4 Traffic among Autonomous Systems

The goal is to define the traffic related to the resource across the N AS's, hence we define X_{ji} as the probability that there is a resource transfer from the AS_j to the AS_i:

$$X_{ji} = \sum_{\substack{n_1\ldots n_N,\, p_1\ldots p_N,\, q_1\ldots q_N\,:\\ \sum_{k=1}^N n_k\neq 0,\, \sum_{k=1}^N p_k\neq 0}} \frac{n_j}{\sum_{k=1}^N n_k}\frac{p_i}{\sum_{k=1}^N p_k}\Pi(n_1\ldots n_N, p_1\ldots p_N, q_1\ldots q_N). \tag{8.27}$$

Note that conditions that allows a resource transfer from AS_j to AS_i are $n_j \neq 0$ and $p_i \neq 0$. Since in Equation (8.27) n_j and p_i are both to the numerator, it is equivalent to use $\sum_{k=1}^N n_k \neq 0$ and $\sum_{k=1}^N p_k \neq 0$.
It can be shown, using techniques similar to the proof of Equation(8.8), that:

$$X_{ji} = \int_0^1\int_0^1 \left[\frac{\partial}{\partial u_j}\frac{\partial}{\partial v_i}g(u_1\ldots u_N, v_1\ldots v_N, w_1\ldots w_N)\right]_{\substack{u_1\ldots u_n=x\\ v_1\ldots v_n=y\\ w_1\ldots w_N=1}} dx\,dy. \tag{8.28}$$

The complete derivation of Equation (8.28) is given in the Appendix A, Proof 1. By calculating Equation (8.28) with Equation (8.4) we obtain:

$$X_{ji} = \frac{\alpha_j s_j}{A}\frac{\beta_i s_i}{B}\left[1-G(1-A)-G(1-B)+G(1-A-B)\right] \tag{8.29}$$

Note that $\sum_{i,j} X_{ji}$ is equal to the probability that there is at least one peer holding the resource and one peer requiring it in the whole system.

8.4.1 Different Search Policies: internal search first

Let supposes the AS searches resources first among its peers and if it is not present it will search in the others AS's. Then the definition of X can be split in the following two cases:

$$X_{ii} = \sum_{\substack{n_1...n_N,\, p_1...p_N,\, q_1...q_N\,:\\ n_i \neq 0,\, \sum_{k=1}^{N} p_k \neq 0}} \frac{p_i}{\sum_{k=1}^{N} p_k} \Pi(n_1...n_N, p_1...p_N, q_1...q_N). \tag{8.30}$$

to consider the internal search first, and

$$X_{ji} = \sum_{\substack{n_1...n_N,\, p_1...p_N,\, q_1...q_N\,:\\ n_i = 0,\\ \sum_{k=1}^{N} p_k \neq 0,\, \sum_{h=1}^{N} n_h \neq 0}} \frac{n_j}{\sum_{k=1,k\neq i}^{N} n_k} \frac{p_i}{\sum_{k=1}^{N} p_k} \Pi(n_1...n_N, p_1...p_N, q_1...q_N), i \neq j. \tag{8.31}$$

to describe the external search. By using the generating function we can write:

$$X_{ii} = \int_0^1 \left[\frac{\partial}{\partial v_i} \Big(g(u_1...u_N, v_1...v_N, w_1...w_N) + \right.$$
$$\left. - g(u_1..u_{j-1}, 0, u_{j+1}..u_N, v_1...v_N, w_1...w_N) \Big) \right]_{\substack{u_1...u_n = x \\ v_1...v_n = y \\ w_1...w_N = 1}} dx\, dy. \tag{8.32}$$

$$X_{ji} = \int_0^1 \int_0^1 \left[\frac{\partial}{\partial u_j} \frac{\partial}{\partial v_i} g(u_1..u_{j-1}, 0, u_{j+1}..u_N, v_1...v_N, w_1...w_N) \right]_{\substack{u_1...u_n = x \\ v_1...v_n = y \\ w_1...w_N = 1}} dx\, dy. \tag{8.33}$$

The derivations of the previous equations can be easily obtained using technique similar to the one used in the appendix to prove Equation (8.28), and have been omitted for brevity. By calculating Equation (8.32) and (8.33) with Equation (8.4) we obtain:

$$X_{ji} = \begin{cases} \frac{\alpha_i s_i}{A - \alpha_j s_j} \frac{\beta_j s_j}{B} \left[G(1-\alpha_j s_j) - G(1-A) - G(1-B-\alpha_j s_j) + G(1-A-B) \right] & j \neq i \\ \frac{\beta_j s_j}{B} \left[1 - G(1-\alpha_j s_j) - G(1-B) - G(1-B-\alpha_j s_j) \right] & j = i \end{cases} \tag{8.34}$$

8.4.2 Different Search Policies: weighted search

Now instead suppose the AS associate a weight with each other AS and searches are done with probability proportional to the assigned weight. This strategies can be useful for example for preferring AS with peering aggrements over more expensive transit links.

$$X_{ji} = \sum_{\substack{n_1...n_N, p_1...p_N, q_1...q_N : \\ \sum_{k=1}^{N} n_k \neq 0, \sum_{k=1}^{N} p_k \neq 0}} \frac{\sigma_{ij} n_j}{\sum_{k=1}^{N} \sigma_{ik} n_k} \frac{p_i}{\sum_{k=1}^{N} p_k} \Pi(n_1...n_N, p_1...p_N, q_1...q_N). \quad (8.35)$$

It is possible to prove that:

$$X_{ji} = \int_0^1 \int_0^1 \left[\frac{\partial}{\partial \hat{u}_j} \frac{\partial}{\partial v_i} \left[g(u_1...u_N, v_1...v_N, w_1...w_N) \right]_{u_k = \hat{u}^{\sigma_{ik}}} \right]_{\substack{u_1...u_N = x \\ v_1...v_N = y \\ w_1...w_N = 1}} dx\, dy. \quad (8.36)$$

Equation (8.36) can be derived with technique very similar to used in the Appendix A, Proof 1, and as for Equation (8.32) and (8.33), the derivation has been omitted for brevity. As in the previous cases, by applying Equation (8.36) to Equation (8.4) we obtain:

$$X_{ji} = \frac{\beta_i s_i}{B} \int_0^1 \alpha_j s_j \sigma_{ij} x^{\sigma_{ij}-1} \left[G'(1 - A + \sum_{k=1}^{N} \alpha_k s_k x^{\sigma_{ik}}) - G'(1 - A - B + \sum_{k=1}^{N} \alpha_k s_k x^{\sigma_{ik}}) \right] dx \quad (8.37)$$

8.5 Considering more resources

To create a more realistic scenario we suppose that each peer holds more than one resource. We denote with N_{max} the maximum number of resources available in the whole system. Each one is characterized by a popularity level that describes the probability that a peer has such a resource. We assume that the popularity of the resources follows a Zipf Mandelbrot (119) distribution. The resource distribution is characterized by three parameters: q_Z, s_Z and $\bar{n}_Z(i)$. The first two parameters define the Zipf Mandelbrot distribution and they are constant for the whole system, whereas $\bar{n}_Z(i)$ represents the mean number of resources per peer, and we allow it to depend on the autonomous

system i. In particular, let us define $f_i(k)$ with $1 \leq k \leq N_{Max}$ as the probability that a peer in the $i-th$ AS holds resource k, computed as:

$$f_i(k) = \frac{\bar{n}_Z(i) \cdot H}{(k+q_Z)^{s_Z}} \tag{8.38}$$

with

$$H = \frac{1}{\sum_{l=1}^{N_{Max}} \frac{1}{(l+q_Z)^{s_Z}}}$$

It is easy to show that with (8.38) and the above definition we have $\sum_{l=1}^{N_{Max}} f_i(l) = \bar{n}_Z(i)$. Note that since $f_i(k)$ is a probability distribution the following constraints must hold:

$$\bar{n}_Z(i) \leq \frac{(1+q_Z)^{s_Z}}{H}.$$

8.6 Model validation

We validate our model using a discrete event simulation. In our simulated environment the P2P overlay is distributed among different autonomous systems. Peers seek and exchange resources they need one another. We compare simulation's results with model's outcome, showing how the first confirm the model.

We use PeerSim (117) to implement our simulation. PeerSim is a discrete event simulator providing a collection of features that help the implementation and analysis of network protocols or multi-agents simulations.

In our simulated set up the whole peers population is grouped into ASes of different sizes. The number of resources types in the system is fixed to N. Resource's popularity is modeled according to the Zipf-Mandelbrot (119) discrete distribution.

Once established how many kinds of resources there are in the overlay, resources and requested are assigned to peers according to 8.5. A peer owns a resource with probability $0 \leq \beta \leq 1$ or it need for it with probability $1-\beta$. A peer can supply to others items it owns and requires those that it needs.

Simulation's length is subdivided into rounds, and each round into time slots. A round has exactly a number of time slots equal to the peers' number active in the overlay. In this way all peers are scheduled exactly once for one simulation round. When scheduled, a peer makes a request to another for a specific resource.

Table 8.2: Simulation parameters

Description	Value
n_{tot}	5000
n_1, n_2, n_3	2500, 1800, 700
N	10
s	1
q_1, q_2, q_3	0.5, 0.2, 0.7

The resource's search is made according to the policy chosen. In our simulation each peer can use both the uniform and the weighted strategy explained in Section 8.4.1.

We validate the model using the weighted strategy (Section 8.4.2). Indeed the uniform strategy can be considered a special case where all weights are equals. Here we present a comparison between the model and the simulation for the weighted case with a P2P overlay distributed over three ASes. The connection between AS_1, AS_2 and AS_3 is shown in Figure 8.3. A fraction n_i of the total peer population n_{tot} is assigned to each AS. A certain number of resources type N is distributed in the overlay.

The simulation's most relevant parameters are reported in Table 8.2. α and q_1, q_2 and q_3 are parameters of the Zipf-Mandelbrot distribution, respectively the skewness factor and the plateau factors for the three ASes. This latter controls the plateau shape near the lowest ranked objects i.e., the flatness of the distribution's tail.

We assign the following weight matrix W:

$$W = \begin{bmatrix} 4 & 2 & 1 \\ 1 & 2 & 1 \\ 1 & 2 & 4 \end{bmatrix}$$

In W the element w_{ij} corresponds to the weight that the AS_i assign to AS_j. According to W, each peer seeks the needed resources with higher probability in its same AS, and prefers shorter paths to longer ones when communicating with other ASes.

In Figures 8.4, 8.5 and 8.6 we show the comparison between the evolution over time of both the model and simulator. The simulator's outcomes are the mean of 50 runs and the 95% confidence interval is shown. We compared all the transfers among all ASes: the model outcomes are always included in the 95% confidence interval of

simulation results. Further confirmation comes from Figure 8.7 where we confronted the final steady state for both simulation and model, confirming the previous results.

8.7 Experiments

We conduct several experiments using both the P2P resource diffusion model (Sections 8.3 and 8.4) and the AS-level network model (Section 8.2). In the following we present the results changing model parameters. We compute costs and rewards for ASes in a complex network topology, taking into account peering and transit accords, non trivial routes and different ASes sizes.

In Figure 8.1 (page 116) the topology used for experiments is depicted. The network is composed from eight different ASes, and nodes are connected with links with peering or transit agreements. In the system there are $N = 10$ different resources types. To each link is assigned a specific cost c for link usage.

8.7.1 Changing ASes size

We first conduct several experiments changing the size of ASes. Different sizes are computed according to Zipf-Mandelbrot distribution with parameters $N = [1 \dots 8]$, $s = [0 \dots 1.4]$ with a step of 0.2, and $q = 2$. Figure 8.8 shows how networks' sizes changes. When $s = 0$ all the ASes have the same dimension, while moving on higher value of the s parameter the distribution becomes more skewed. Here we use the uniform strategies for resources finding. We set $c = 1$ for all links. We compute rewards (Figure 8.9) and the costs (Figure 8.10) for each AS. In Figure 8.11 we compute the net profit, that is the difference between rewards and costs. Our results show how the profit changes in function of the different AS size. We observe that in this particular topology ASes AS_1, AS_2, and AS_7 only pay for transit access or have peering agreements, so their net profit is negative. AS_8 always increase its profit, taking advantage from its wider population when the distribution is more skewed. Note that for AS_5 there is a trend inversion when $s = 0.6$. We can conclude that the more skewed distribution has a positive effect on AS from 1 to 3 and 8, negative for 4 to 7.

8.7.2 Changing links' weights

We use the weighted strategy to increase the weights on peering links. We set $c = 1$ for all links. The total profit for each AS is shown in Figure 8.12. The absolute value of fees paid in the overall network globally decreases. AS_1, AS_2, AS_3 and AS_6 pay less than using the uniform strategy and AS_6 and AS_8 have a lower profit.

Figure 8.13 shows the results if only the AS_7 increases weights on its peering links. The most important thing to note is that the changed policy of AS_7 has effect on the overall network. We report in the figure the results changing the weight from $w = 0$ to $w = 7$. Indeed for increasing value of w the total rewards of AS_7 increase, while the gains of AS_5, AS_6 and AS_8 diminish. AS_1, AS_2, AS_3 and AS_4 benefit from the new policy of AS_7, that uses less their transit links preferring its peering partners.

8.7.3 Changing links' costs

In this experiment we change the cost c for the links connected to AS_8, from $c = 1.1$ to $c = 1.4$. The results are shown in Figure 8.14. Higher costs have affect only to the direct connected ASes. Their profit diminishing but other networks are not affected.

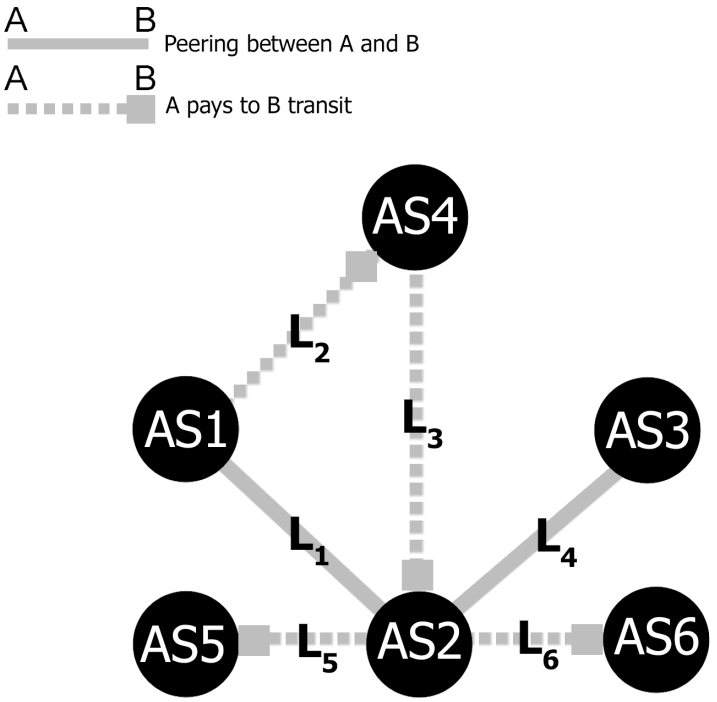
8.7.4 Changing AS agreements

Finally we made some change to the agreements among networks. The results are reported in Figure 8.13. With denote with M_1 the old topology and with M_2 the new. We change the agrements in the following way: the links between AS_1 and AS_4, and from AS_2 to AS_4 are now peering; from AS_5 to AS_7 is settled a transit agreement where AS_5 is the customer and the same for the connection between AS_7 and AS_6, with AS_7 as customer. Also here if we increase the weights to peering ASes we can see that there is a gain for all networks. Also in this topology AS_1, AS_2, and AS_3 only pay. But AS_4 and AS_5 have now a positive profit because they do not route requests of AS_2. It is interesting note that here give up a transit agreements, which still generate profit, is more convenient.

8.8 Conclusions

In this chapter we have presented a model able to compute costs and rewards for Autonomous Systems (ASes) organized in a complex network topology. We take into

accounts peering and transit agreements as well as inter-AS routing policy. We merge an AS-level model with a new P2P resources diffusion model we have developed. Peers are considered being spread among the ASes, each one having its own parameters in terms of resource availability and demand. We were able to describe how a P2P protocol impact on the underlying AS-layer. The main goal we achieved is providing the system administrator a tool to minimize its cost related to the traffic produced by the users of its network. Our model can easily compute costs and rewards distribution in the ASes network that depending on different parameters (peering and transit agreements, routing policy, different AS size etc.).



$$\mathbf{C^2} = \begin{bmatrix} 0 & 0 & 0 & 0 & \sigma(2,5) & \sigma(2,6) \\ 0 & 0 & 0 & 0 & \sigma(2,5) & \sigma(2,6) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma(2,5) & \sigma(2,6) \\ \sigma(2,5) & \sigma(2,5) & 0 & \sigma(2,5) & 0 & 0 \\ \sigma(2,6) & \sigma(2,6) & 0 & \sigma(2,6) & 0 & 0 \end{bmatrix}$$

$$\mathbf{R^2} = \begin{bmatrix} 0 & 0 & \sigma(4,2) & 0 & \sigma(4,2) & \sigma(4,2) \\ 0 & 0 & 0 & \sigma(4,2) & 0 & 0 \\ \sigma(4,2) & 0 & 0 & \sigma(4,2) & 0 & 0 \\ 0 & \sigma(4,2) & \sigma(4,2) & 0 & \sigma(4,2) & \sigma(4,2) \\ \sigma(4,2) & 0 & 0 & \sigma(4,2) & 0 & 0 \\ \sigma(4,2) & 0 & 0 & \sigma(4,2) & 0 & 0 \end{bmatrix}.$$

Figure 8.2: Example topology of six ASes with the cost and reward matrices for AS_2

Figure 8.3: ASes topology used to validate the model

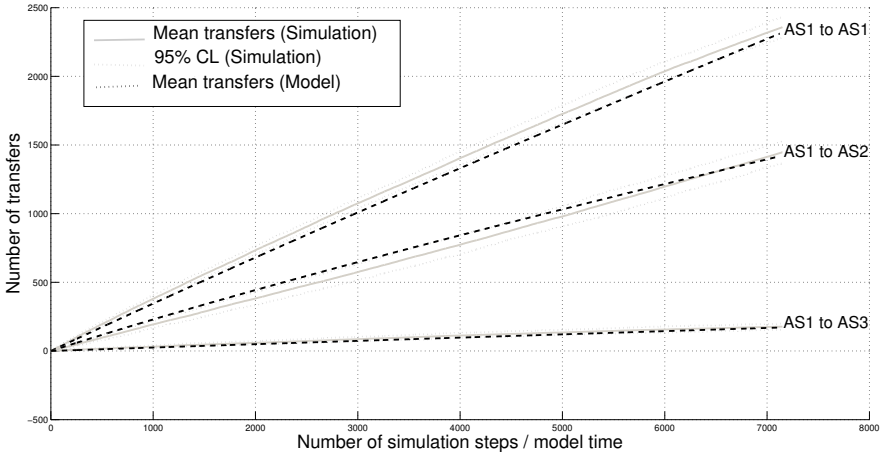


Figure 8.4: Comparison between the model and the simulation for transferts from AS_1 to others

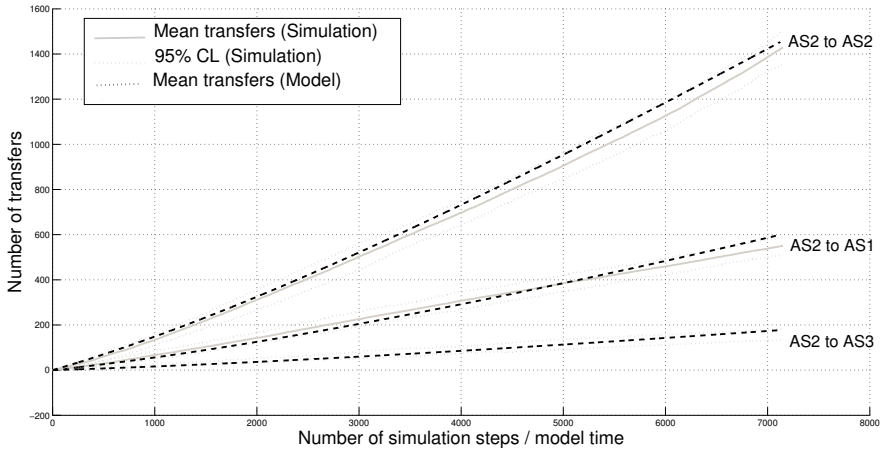


Figure 8.5: Comparison between the model and the simulation for transferts from AS_2 to others

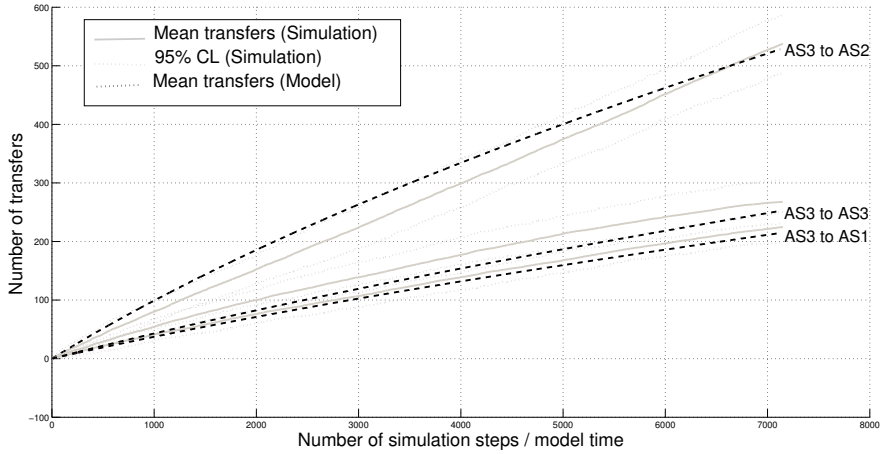


Figure 8.6: Comparison between the model and the simulation for transferts from AS_3 to others

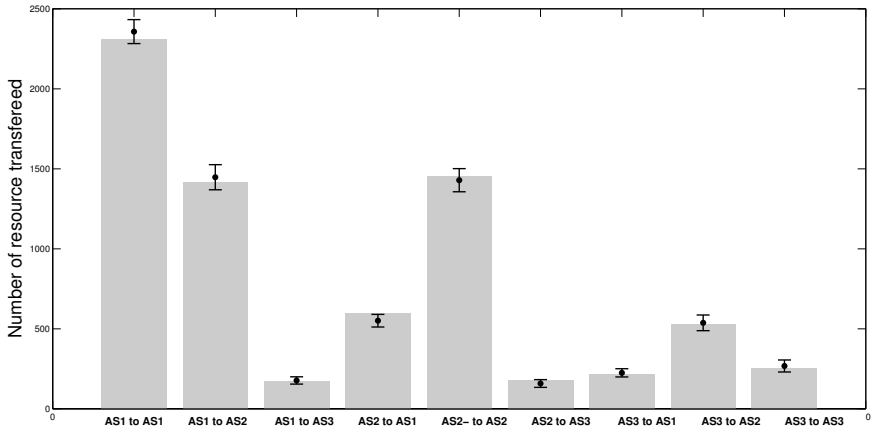


Figure 8.7: comparison between the model and the simulation in the steady state, the bars are the model's outcome compared with the mean of 50 simulation run and 95% confidence interval (dots with bars)

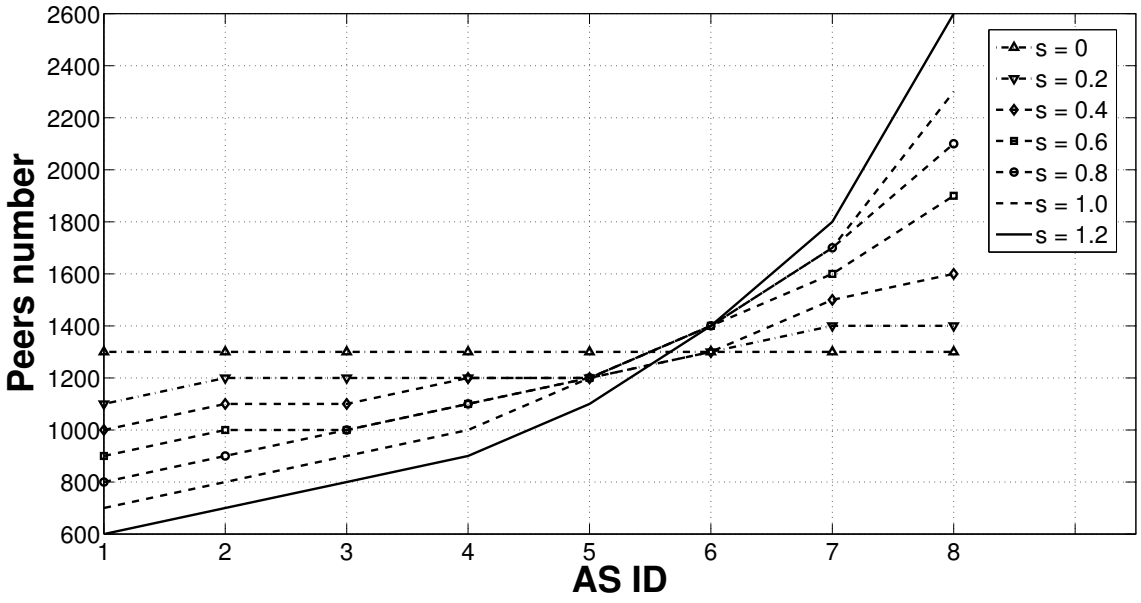


Figure 8.8: Different configurations of ASes sizes

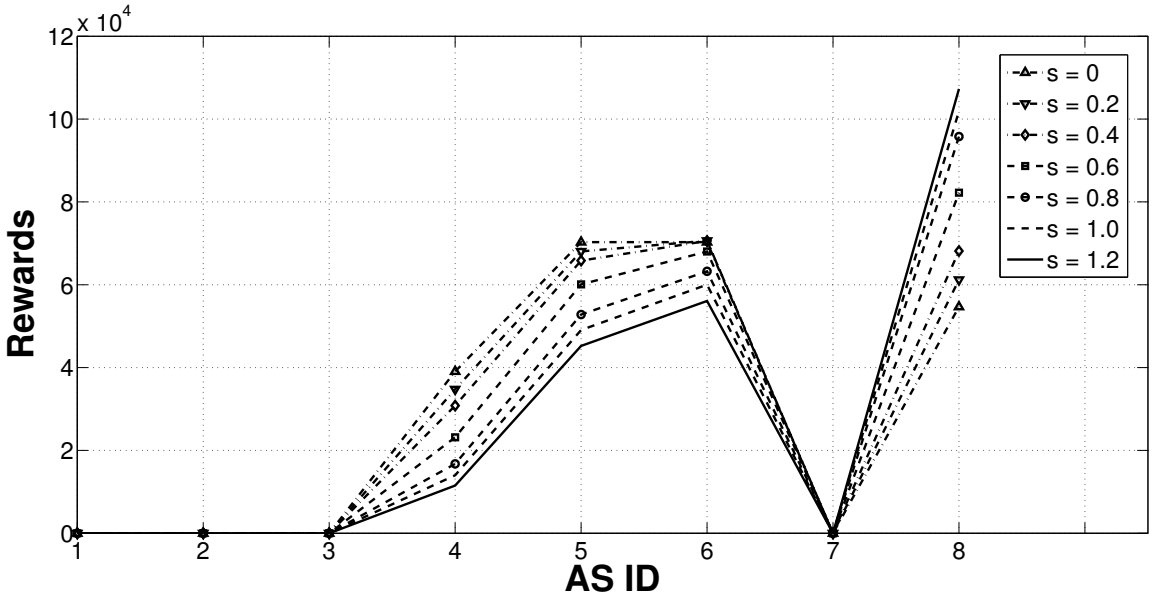


Figure 8.9: Rewards distribution considering several configurations of ASes size

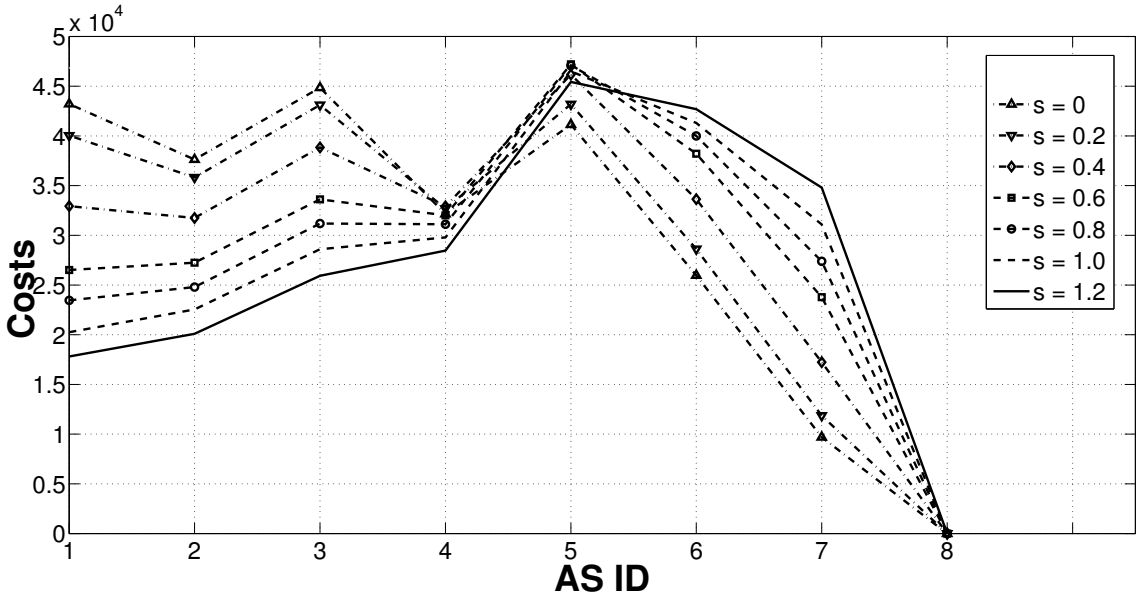


Figure 8.10: Costs distribution considering several configurations of ASes size

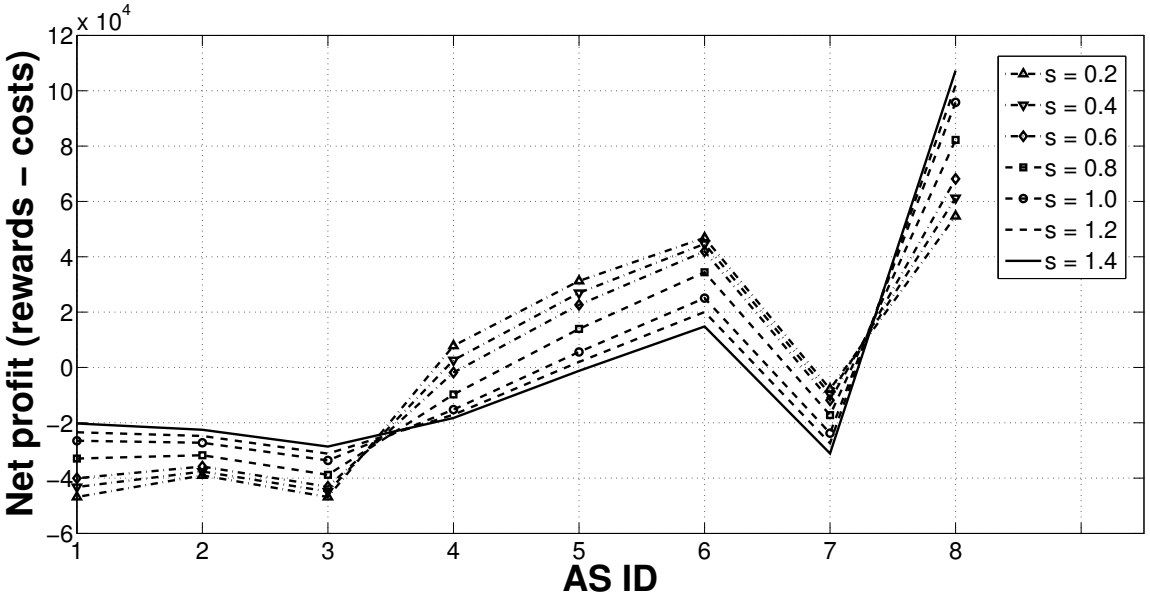


Figure 8.11: Distribution of the difference reward-cost (net profit) considering several configurations of ASes size

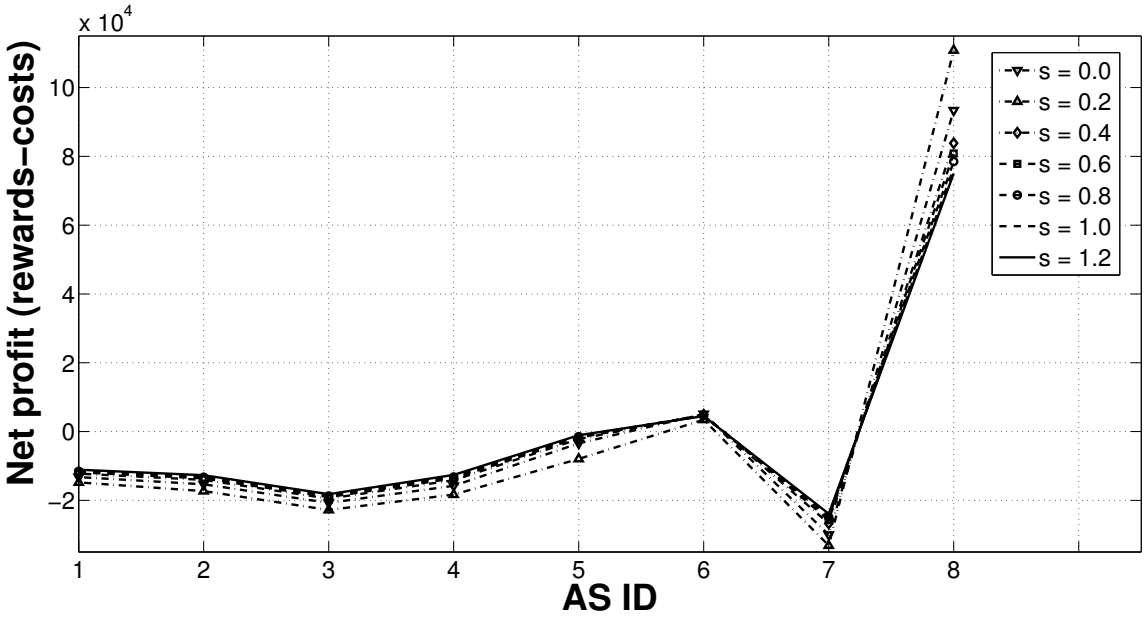


Figure 8.12: Distribution of the difference reward-cost (net profit) when all ASes increase weights to their peering links

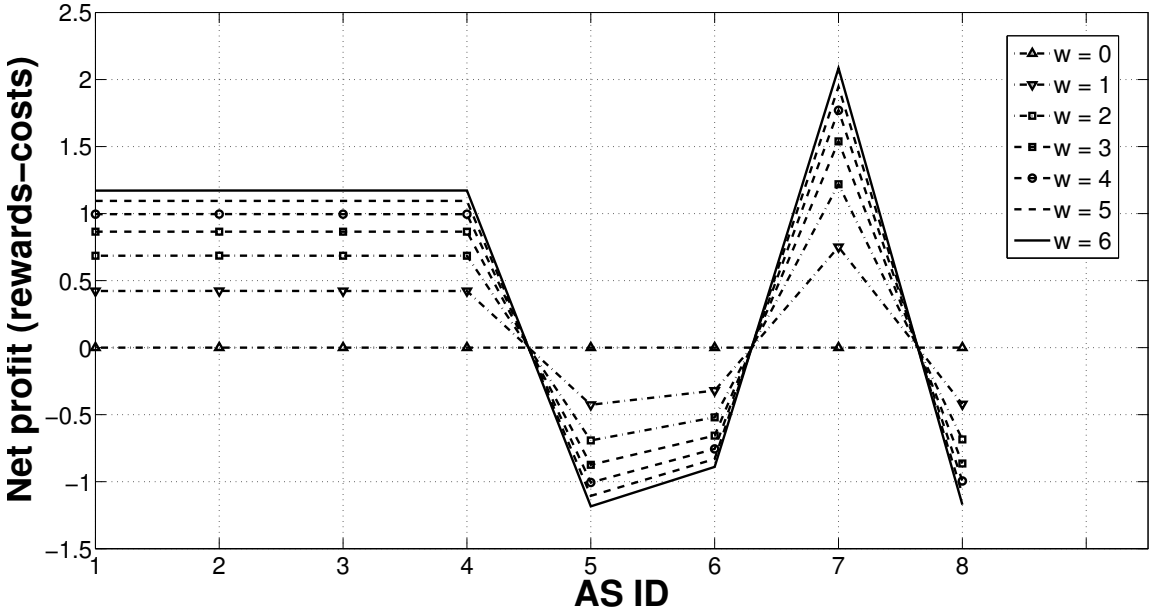


Figure 8.13: Distribution of the difference of the net profit respect to $w = 0$ when only AS_7 increase weights to its peering links

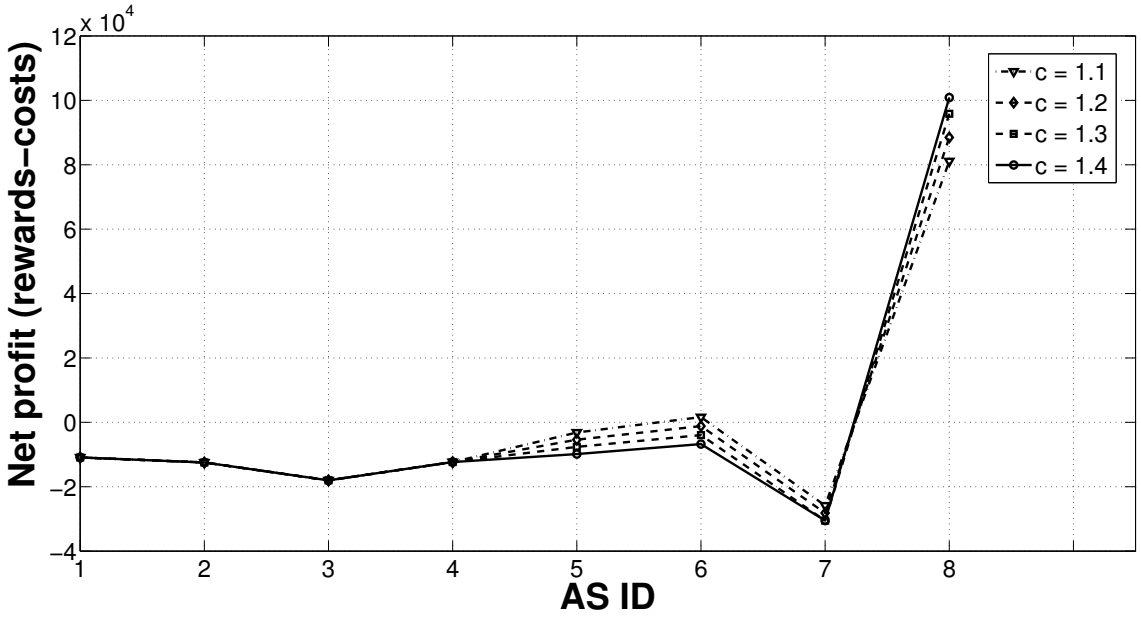


Figure 8.14: Net profit when all only AS_8 increase its trasnsit costs

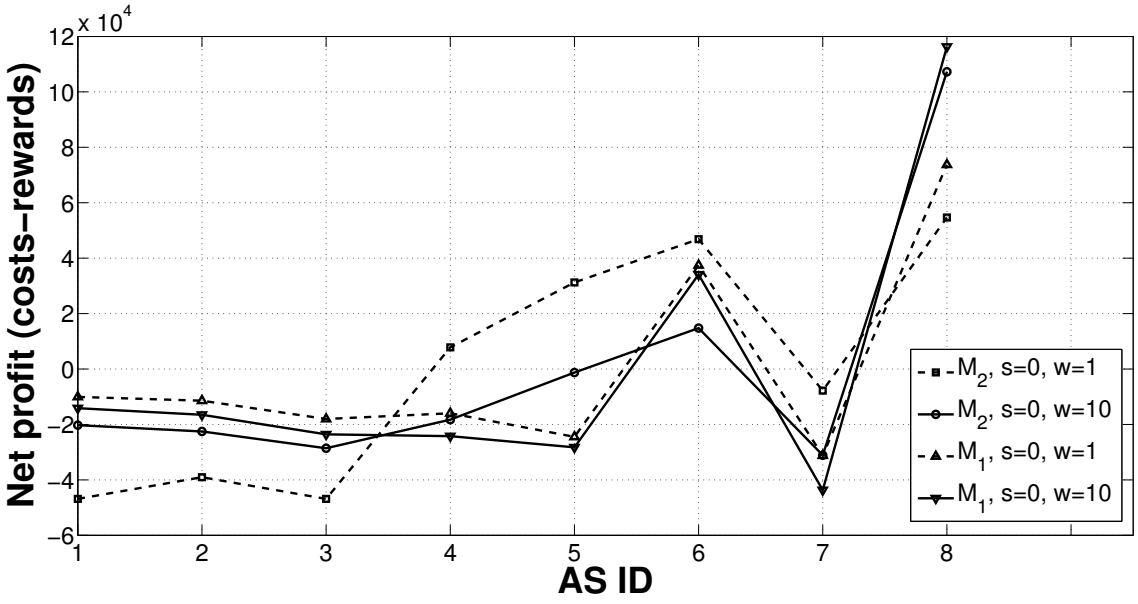


Figure 8.15: Comparison between two different topologies

Chapter 9

Conclusions and future work

We state the conclusions of this thesis and discuss opportunities for further investigation.

9.1 Summary

The primary objective of this dissertation is the optimization of P2P protocols. The issue has been considered from two points of view: protocol's performance improvement and network awareness. A particular attention has been given to file-sharing protocols. Our main contribution with respect to performance improvements has been to develop ad-hoc strategies that support LT codes in P2P file-sharing protocols. We develop both a BitTorrent modification (Chapter 4) and a complete new protocol, ToroVerde (Chapter 6), built around the digital fountain concept. Regarding network awareness we propose both a game-theoretical approach to design IPS-aware policies in P2P streaming networks and an analytical model for resource diffusion in P2P file-sharing overlay distributed over Autonomous Systems topology. Other contributions of this thesis are a new P2P simulator (Chapter 5), that achieves significant improvements respect with GPS and a novel measurement strategy for P2P closed source network protocols.

We started out by studying a popular P2P-TV streaming system: PPLive. Our aim was to study a widely-used P2P application to obtain useful and meaningful informations about P2P protocols' behaviour. We move around the closed source nature of the PPLive protocol using a passive/active measurement technique. To this end, we use a

distributed crawler to capture snapshots of PPLive overlay networks. We derive several characteristics of the overlay network, such as topological informations, users behavior and sessions length. Moreover we also point out the relations between bandwidth (upload and download) and the crawling accuracy. Our work contribute also to show limits and completeness of active measurements acquired crawling the PPLive overlays, concluding that a simple crawling strategies is not able to get accurate snapshots. The lesson that we can draw is that the derivation of accurate topological information for PPLive (and possibly for similar P2P-IPTV applications) requires the setting of more sophisticated measure methodologies.

In Chapter 4 we investigate how the introduction of LT codes in a well-known protocol, BitTorrent, improve its performance. In particular we show by simulations that in some network conditions i.e., when peers expose high dynamical behaviour (high churn rates and flash crowds), and considering files of small size, better performance can be achieved. In such conditions our proposed protocol yields a gain about of 10%, in terms of downloading times. We also show how the standard protocol strategies must be properly modified for rateless codes. These results demonstrate that the use of some kind of network coding, in our case LT codes, can lead to improvements, and these are mainly related to network conditions and peers' behaviour.

In Chapter 6 we implemente a completely new P2P protocol, ToroVerde, a push-based P2P content distribution application exploiting the digital fountain concept through rateless codes. We provide a complete protocol specification and developed both a detailed simulator and a complete prototype that we used to perform a fair comparison between ToroVerde and BitTorrent. ToroVerde achieves less download delays in several realistic scenarios for small-to-medium sized file and overlays composed of a few hundred peers.

We presente several possible techniques for network awareness. We develope a game theory framework (Chapter 7) that can be used for the design of ISP strategies aiming at providing efficient and reliable support for P2P streaming application. Our framework allows us to illustrate the existence of equilibrium points in the ISPs game and the role of possible strategies to renew these points. We have also shown how to use ideas from the evolutionary game theory to derive techniques that a player can use to compute the (operational) equilibrium points assuming only limited knowledge on the state of the other players.

The analytical model we have presented in Chapter 8 is able to compute costs and rewards for Autonomous Systems (ASes) organized in a complex network topology. Peering and transit agreements as well as inter-AS routing policy are taken into considerations to compute utility functions. Also in this case we have given particular attention to file-sharing protocols. We merge an AS-level model with a new P2P resources diffusion model we develop. The main advantage of the proposed probabilistic model is that it exploits properties of the Z-Transform to consider very large models with low computational cost. Peers are considered being spread among the ASes, each one having its own parameters in terms of resource availability and demands. Using both models we describe how a P2P protocol impacts on the underlying AS-layer. Indeed the main goal is providing the system administrator some tools to minimize its cost related to the traffic produced by the users of its network. In particular we show how this can be achieved by either trying to settle different peering agreements with other ASes or by applying intelligent routing schemes that can provide a cheaper use of its resources.

In summary our work focuses on P2P file-sharing protocols. We have shown how using network coding techniques lead to improvements. We have modified BitTorrent and have implemented a new protocol from scratch. At the same time we have developed several techniques to optimize the interaction between P2P protocols and ISPs, using both a game theoretical framework and a P2P diffusion model.

9.2 Topics for further investigation

There are several areas that are doubtless interesting and where we would perform further researches to improve the contributions made in this dissertation. How network coding can improve P2P protocols performance is a question still open and network awareness promises to be a productive research topic in the near future, for the its economical and commercial consequences. In the following we summarize several possible improvements to the work presented in this thesis.

9.2.1 Network coding and P2P overlay

We have demonstrated that using coding techniques lead to improvement in P2P file-sharing protocols. As a possible future development it would be interesting investigating

the performance of Toroverde in different conditions from that already presented i.e., for larger files as well as for larger overlays. A challenging goal is also devise techniques to reduce the communication overhead of the protocol: a possibility could be allowing coded block to be forwarded more than once hence avoiding the need for Bloom filters bits in each sent coded blocks. Another interesting topic is evaluate possibility for the recombination of already coded blocks. The objective is improve blocks diversity in the overlay network, reducing the possibility that the same block is received twice, and consequently reducing the signaling overhead required for content reconciliation. It would be interesting to explore the use of different coding techniques.

9.2.2 Applying network coding to other P2P protocols

While this thesis focuses main on file-sharing protocols, it would be interesting investigate how other types of protocols like live-streaming or Video-On-Demand can benefit from coding techniques.

9.2.3 Extend the game theoretical framework

The future developments of the research presented in Chapter 7 is following several directions. One is to deepen the simulation based analysis investigating relation between peers dynamics and their effects on the fluctuation utility functions. Another way we are currently investigating is developing more sophisticated payoff functions that include information about the size of the ISP and the subscription fees it apply to its subscribers. These model enhancements could explore other different interactions/competitions among the ISPs e.g., competition for attracting users, and competition on subscription fees. Furthermore, we can also investigate other fairness criteria, based for instance, on function that accounts for the ISP rewards.

9.2.4 Improving the cooperation between Autonomous Systems and P2P protocols

Our objective for further development is extend and generalize our model for different kind of P2P protocols. Another interesting direction is study techniques that improve the interaction between P2P application level protocol and external routing protocols, like BGP. One solution, for example, could be include geographical information

for packet routing, making protocols aware of optimal (e.g. shortest, less congested, cheaper) paths.

Appendix A

Here we report some proofs cited in Chapter 8

A.1 Proof 1

In order to simplify the notation, we denote

$g(\cdot) = g(u_1...u_N, v_1...v_N, w_1...w_N)$ and $\Pi(\cdot) = \Pi^m(n_1..n_N, p_1..p_N, q_1..q_N)$

$$\frac{\partial}{\partial v_i} g(\cdot) = \sum_{n_1...n_N,\, p_1...p_N,\, q_1...q_N} p_i u_1^{n_1}..u_N^{n_N} v_1^{p_1}..v_i^{p_i-1}..v_N^{p_N} w_1^{q_1}..w_N^{q_N} \Pi(\cdot) \tag{A.1}$$

$$\frac{\partial}{\partial u_j}\frac{\partial}{\partial v_i} g(\cdot) = \sum_{n_1...n_N,\, p_1...p_N,\, q_1...q_N} n_i p_i u_1^{n_1}..u_i^{n_i-1}..u_N^{n_N} v_1^{p_1}..v_i^{p_i-1}..v_N^{p_N} w_1^{q_1}..w_N^{q_N} \Pi(\cdot) \tag{A.2}$$

$$\left[\frac{\partial}{\partial u_j}\frac{\partial}{\partial v_i} g(\cdot)\right]_{\substack{u_1...u_n = x \\ v_1...v_n = y \\ w_1...w_N = 1}} = \sum_{n_1...n_N,\, p_1...p_N,\, q_1...q_N} n_i p_i x^{\sum_{k=1}^N n_k - 1} y^{\sum_{k=1}^N p_k - 1} \Pi(\cdot) \tag{A.3}$$

$$\int_0^1 \left[\frac{\partial}{\partial u_j}\frac{\partial}{\partial v_i} g(\cdot)\right]_{\substack{u_1...u_n = x \\ v_1...v_n = y \\ w_1...w_N = 1}} dx = \sum_{\substack{n_1...n_N,\, p_1...p_N,\, q_1...q_N : \\ \sum_{k=1}^N n_k \neq 0}} p_i \frac{n_i}{\sum_{k=1}^N n_k} y^{\sum_{k=1}^N p_k - 1} \Pi(\cdot) \tag{A.4}$$

$$\int_0^1 \int_0^1 \left[\frac{\partial}{\partial u_j}\frac{\partial}{\partial v_i} g(\cdot)\right]_{\substack{u_1...u_n = x \\ v_1...v_n = y \\ w_1...w_N = 1}} dx\, dy = \sum_{\substack{n_1...n_N,\, p_1...p_N,\, q_1...q_N : \\ \sum_{k=1}^N n_k \neq 0,\, \sum_{k=1}^N p_k \neq 0}} \frac{n_i}{\sum_{k=1}^N n_k} \frac{p_i}{\sum_{k=1}^N p_k} \Pi(\cdot) \tag{A.5}$$

A.2 Proof 2

♢ We can show that we can obtain (8.7) from (8.8) by solving step by step the right term of the equation. Starting from explicit form of 8.5 we have:

$$g_i(u_i, v_i, w_i) = \sum_{n_i,p_i,q_i} u_i^{n_i} v_i^{p_i} w_i^{q_i} \Pi_i(n_i, p_i, q_i)$$

$$\frac{\partial g_i(u_i, v_i, w_i)}{\partial u_i} = \sum_{n_i,p_i,q_i} n_i u_i^{n_i-1} v_i^{p_i} w_i^{q_i} \Pi_i(n_i, p_i, q_i)$$

$$\Big[\frac{\partial g_i(u_i, v_i, w_i)}{\partial u_i}\Big]_{u_i=y,v_i=y,w_i=y} = \sum_{n_i,p_i,q_i} n_i y^{n_i+p_i+q_i-1} \Pi_i(n_i, p_i, q_i)$$

$$\int \Big[\frac{\partial g_i(u_i, v_i, w_i)}{\partial u_i}\Big]_{u_i=y,v_i=y,w_i=y} dy = \sum_{n_i,p_i,q_i} \frac{n_i}{n_i + p_i + q_i} y^{n_i+p_i+q_i} \Pi_i(n_i, p_i, q_i) + c$$

Equation (8.8) is proven by computing the integral between 0 and 1.♢

A.3 Proof 3

♢ By definition of generating function we have:

$$\hat{g}_i(u_i, v_i, w_i, \bar{v}_i) = \sum_{n_i,p_i,q_i,\bar{p}_i} u_i^{n_i} v_i^{p_i} w_i^{q_i} \bar{v}_i^{\bar{p}_i} \hat{\Pi}_i(n_i, p_i, q_i, \bar{p}_i) \tag{A.6}$$

Elaborating (8.22) we obtain:

$$\begin{aligned} &g(1...1, u_i, 1...1, \bar{v}_i...\bar{v}_i, v_i, \bar{v}_i...\bar{v}_i, 1...1, w_i, 1...1) - \\ &\qquad g(0...0, \bar{v}_i...\bar{v}_i, v_i, \bar{v}_i...\bar{v}_i, 1...1, w_i, 1...1) = \\ &\sum_{n_i,p_i,q_i,\bar{p}_i} \sum_{\substack{n_1...n_{i-1},\, n_{i+1}...n_N,\\ p_1...p_{i-1},\, p_{i+1}...p_N \wedge \sum_k P_k = \bar{p}_i,\\ q_1...q_{i-1},\, q_{i+1}...q_N}} \Pi^m(n_1..n_N, p_1..p_N, q_1..q_N) u_i^{n_i} v_i^{p_i} w_i^{q_i} \bar{v}_i^{\bar{p}_i} - \\ &\sum_{p_i,q_i,\bar{p}_i} \sum_{\substack{p_1...p_{i-1},\, p_{i+1}...p_N \wedge \sum_k P_k = \bar{p}_i,\\ q_1...q_{i-1},\, q_{i+1}...q_N}} \Pi^m(0..0, p_1..p_N, q_1..q_N) v_i^{p_i} w_i^{q_i} \bar{v}_i^{\bar{p}_i} \end{aligned} \tag{A.7}$$

Comparing (A.7) with (A.6) we have:

$$\hat{\Pi}_i = \sum_{\substack{n_1 \dots n_{i-1},\, n_{i+1} \dots n_N,\\ p_1 \dots p_{i-1},\, p_{i+1} \dots p_N \wedge \sum_k P_k = \bar{p}_i,\\ q_1 \dots q_{i-1},\, q_{i+1} \dots q_N}} \Pi^m(n_1..n_N, p_1..p_N, q_1..q_N) - \tag{A.8}$$
$$\sum_{\substack{p_1 \dots p_{i-1},\, p_{i+1} \dots p_N \wedge \sum_k P_k = \bar{p}_i,\\ q_1 \dots q_{i-1},\, q_{i+1} \dots q_N}} \Pi^m(0..0, p_1..p_N, q_1..q_N)$$

By using a technique similar to the one used in the proof of (8.8) we can compute from (8.21):

$$\frac{\partial \hat{g}_i}{\partial v_i} = \sum_{n_i, p_i, q_i, \bar{p}_i} p_i u_i^{n_i} v_i^{p_i - 1} w_i^{q_i} \bar{v}_i^{\bar{p}_i} \hat{\Pi}_i(n_i, p_i, q_i, \bar{p}_i)$$

$$\left[\frac{\partial \hat{g}_i(u_i, v_i, w_i, \bar{v}_i)}{\partial v_i}\right]_{\substack{v_i = xy \\ \bar{v}_i = x}} =$$

$$\sum_{n_i, p_i, q_i, \bar{p}_i} p_i u_i^{n_i} (xy)^{p_i - 1} w_i^{q_i} x^{\bar{p}_i} \hat{\Pi}_i(n_i, p_i, q_i, \bar{p}_i) =$$

$$\sum_{n_i, p_i, q_i, \bar{p}_i} p_i u_i^{n_i} x^{p_i + \bar{p}_i - 1} w_i^{q_i} y^{p_i - 1} \hat{\Pi}_i(n_i, p_i, q_i, \bar{p}_i)$$

$$\int \left[\frac{\partial \hat{g}_i(u_i, v_i, w_i, \bar{v}_i)}{\partial v_i}\right]_{\substack{v_i = xy \\ \bar{v}_i = x}} dx =$$

$$\sum_{n_i, p_i, q_i, \bar{p}_i} \frac{p_i}{p_i + \bar{p}_i} x^{p_i + \bar{p}_i} u_i^{n_i} w_i^{q_i} y^{p_i - 1} \hat{\Pi}_i(n_i, p_i, q_i, \bar{p}_i) + c$$

$$\int_0^1 \left[\frac{\partial \hat{g}_i(u_i, v_i, w_i, \bar{v}_i)}{\partial v_i}\right]_{\substack{v_i = xy \\ \bar{v}_i = x}} dx =$$

$$\sum_{n_i, q_i, p_i + \bar{p}_i \neq 0} \frac{p_i}{p_i + \bar{p}_i} u_i^{n_i} w_i^{q_i} y^{p_i - 1} \hat{\Pi}_i(n_i, p_i, q_i, \bar{p}_i)$$

$$\left[\int_0^1 \left[\frac{\partial \hat{g}_i(u_i, v_i, w_i, \bar{v}_i)}{\partial v_i}\right]_{\substack{v_i = xy \\ \bar{v}_i = x}} dx\right]_{\substack{u_i = y \\ w_i = y}} =$$

$$\sum_{n_i, q_i, p_i + \bar{p}_i \neq 0} \frac{p_i}{p_i + \bar{p}_i} y^{n_i + p_i + q_i - 1} \hat{\Pi}_i(n_i, p_i, q_i, \bar{p}_i)$$

$$\int \left[\int_0^1 \left[\frac{\partial \hat{g}_i(u_i, v_i, w_i, \bar{v}_i)}{\partial v_i}\right]_{\substack{v_i = xy \\ \bar{v}_i = x}} dx\right]_{\substack{u_i = y \\ w_i = y}} dy =$$

$$\sum_{n_i, q_i, p_i + \bar{p}_i \neq 0} \frac{p_i}{p_i + \bar{p}_i} \frac{1}{n_i + p_i + q_i} y^{n_i + p_i + q_i} \hat{\Pi}_i(n_i, p_i, q_i, \bar{p}_i) + c$$

$$\int_0^1 \left[\int_0^1 \left[\frac{\partial \hat{g}_i(u_i, v_i, w_i, \bar{v}_i)}{\partial v_i} \right]_{\substack{v_i = xy \\ \bar{v}_i = x}} dx \right]_{\substack{u_i = y \\ w_i = y}} dy = \tag{A.9}$$

$$\sum_{n_i + p_i + q_i \neq 0, p_i + \bar{p}_i \neq 0} \frac{p_i}{p_i + \bar{p}_i} \frac{1}{n_i + p_i + q_i} \hat{\Pi}_i(n_i, p_i, q_i, \bar{p}_i)$$

If we insert (A.8) into (Last) we obtain (8.21).♢

References

[1] "PPStream, http://www.PPStream.com; PPLive, http://www.pplive.com; SOPCast, http://www.sopcast.com; TVAnts, http://www.tvants.com; Joost, http://www.joost.com; Babelgum, http://www.babelgum.com; Zattoo, http://www.zattoo.com; Tvunetworks, http://www.tvunetworks.com." 15

[2] "PPLive internet site," Registered in 2004. http://www.pplive.com/en/. 15, 25

[3] D. Stutzbach and R. R., "Capturing Accurate Snapshots of the Gnutella Networks," in *Proc. of Global Internet Symposyum*, 2005. 26, 27

[4] D. Ciullo, M. Mellia, M. Meo, and E. Leonardi, "Understanding P2P-TV Systems Through Real Measurements," in *Proc. GLOBECOM 2008*, (New Orleans, FL, USA), Nov 2008. 25

[5] C. Wu, B. Li, and S. Zhao, "Characterizing peer-to-peer streaming flows," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, 2007.

[6] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into PPLive: A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Transactions on Multimedia*, vol. 9, no. 8, 2007. 25

[7] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt, "Measurement and Modeling a Large-scale Overlay for Multimedia Streaming," in *Proc. of the Int. Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine 2007)*, (Vancouver, BC, Canada), Nov 2007. Details available at http://cairo.cs.uiuc.edu/~longvu2/pplive.html. 25, 26, 28, 34, 35

[8] D. Stutzbach, R. Rejaie, and S. Sen, "Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems," in *Proc. of the ACM SIGCOMM Internet Measurement Conference*, (Berkeley, CA, USA), 2005. 26

[9] "NetLimiter Bandwidth Shaper," Shareware edition. http://www.netlimiter.com/. 31

[10] T. Karagiannis, A. Broido, M. Faloutsos, and K. C. Claffy, *Transport layer identification of p2p traffic*, in Proc. ACM IMC04, Taormina, Sicily, Italy, October 2004.

[11] http://www.bittorrent.com/ 2, 12

[12] Bitorrent protocol specification v1.0 http://wiki.theory.org/BitTorrentSpecification, June 2005 2, 5, 12, 13, 60

[13] Zheng Weimin , Ben Y. Zhao, Zheng Dongdong Hongliang Yu, *Understanding User Behavior in Large Demand Systems* Eurosys Conference 2006 3, 48

[14] Kunwoo Park; Dukhyun Chang; Junghoon Kim; Wonjun Yoon; Kwon, T., *An Analysis of User Dynamics in P2P Live Streaming Services* in 2010 IEEE International Conference on Communications (ICC) 3, 48

[15] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. *Analysing and improving bittorrent performance.* In Proc. IEEE Infocom'2006, Barcelona, Spain, April 2006. 2

[16] E. W. Biersack, P. Rodriguez, and P. Felber. *Performance analysis of peer-to-peer networks for le distribution.* In Proc. Fifth International Workshop on Quality of Future Internet Services (QofIS'04), Barcelona, Spain, September 2004. 2

[17] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. *Measurements, analysis, and modeling of bittorrent-like systems.* In Proc. ACM IMC'2005, Berkeley, CA, USA, October 2005. 2

[18] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. A. Hamra, and L. Garces-Erice. *Dissecting bittorrent: Five months in a torrent's lifetime.* In Proc. PAM'04, Antibes Juan-les-Pins, France, April 2004. 2

[19] S. Jun and M. Ahamad. *Incentives in bittorrent induce free riding.* In Proc. SIGCOMM'05 Workshops, Philadelphia, PA, USA, August 2005. 2

[20] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. *The bittorrent p2p le-sharing system: Measurements and analysis.* In Proc. 4th International Workshop on Peer-to-Peer Systems (IPTPS'05), Ithaca, New York, USA, February 2005. 2

[21] D. Qiu and R. Srikant. *Modeling and performance analysis of bittorrent-like peer-to-peer networks.* In Proc. ACM SIGCOMM'04, Portland, Oregon, USA, Aug. 30 - Sept. 3 2004. 2

[22] Yang, X., de Veciana, G. *Service Capacity of Peer to Peer Networks.* In: Proceedings of INFOCOM 2004, Hong Kong (March 2004) 22

[23] Qiu, D., Srikant, R. *Modeling and Performance Analysis of Bit Torrent-Like Peer-to-Peer Networks.* In: Proceedings of ACM SIGCOMM 2004, Portland, USA (2004) 22

[24] Hossfeld, T., Leibnitz, K., Pries, R., Tutschku, K., Tran-Gia, P., Pawlikowski, K. *Information diffusion in eDonkey-like P2P networks.* In: Proceedings of Australian Telecommun. Networks and Applications Conference (ATNAC), Bondi Beach, Australia (2004) 22

[25] Leibnitz, K., Hofeld, T., Wakamiya, N., Murata, M. *Modeling of Epidemic Diffusion in Peer-to-Peer File-Sharing Networks.* In: Proceedings of Proceedings of the Second International Workshop on Biologically Inspired Approaches to Advanced Information Technology Bio-ADIT 2006, Osaka, Japan (2006) 22

[26] Massouli, L., Vojnovic, V. *Coupon Replication Systems.* In: Proceedings of the ACM SIGMETRICS 2005, Banff, Alberta, Canada (2005)

REFERENCES

[27] Stutzbach, D., Zhao, S., Rejaie, R. *Characterizing Files in the Modern Gnutella Network.* In: Proceedings of Multimedia Systems Journal (March 2007)

[28] Caroglio, G., Gaeta, R., Garetto, M., Giaccone, P., Leonardi, E., Sereno, M. *A Fluid-Diffusive Approach for Modelling P2P Systems.* In: Proceedings of the 14-th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems - MASCOTS 2006, Monterey, CA, USA (2006) 22

[29] C. Dana, D. Li, D. Harrison, C. Chuah, *BASS: BitTorrent Assisted Streaming System for Video-on-Demand*, Multimedia Signal Processing, 2005 IEEE 7th Workshop on (2005) 16

[30] B. Liu, Y. Cui, B. Chang, B. Gotow, Y. Xue, *BitTube: case study of a web-based peer-assisted video on demand system,* , Multimedia, 2008. ISM 2008. Tenth IEEE International Symposium on In Multimedia, 2008 16

[31] A. Vlavianos, M. Iliiofotou, M. Faloutsos, *BiToS: Enhancing BitTorrent for Supporting Streaming Applications*, In 9th IEEE Global Internet Symposium 2006 (April 2006) 16

[32] D. Erman, K. De Vogeleer, A. Popescu, *On piece selection for streaming bittorrent*, Fifth Swedish National Computer Networking Worshop (SNCNW2008) 16

[33] J. J. D. Mol, J. A. Pouwelse; M. Meulpolder; D. H. J. Epema; H. J. Sips, *Give-to-Get: Free-riding-resilient Video-on-Demand in P2P Systems*, Multimedia Computing and Networking 2008 16

[34] P. Shah, J.F. Paris, *Peer-to-Peer Multimedia Streaming Using BitTorrent*, Performance, Computing, and Communications Conference, 2007. IPCCC 2007 16

[35] J. J. D. Mol, A. Bakker, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips, *The Design and Deployment of a BitTorrent Live Video Streaming Solution*, The IEEE Int'l Symposium on Multimedia 2009, December 2009. 16

[36] M. Luby, *LT codes*, Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on (2002). 5, 19, 42, 44, 45, 66

[37] E.Hyytia, T. Tirronen, J. Virtamo *Optimizing the Degree Distribution of LT Codes*, in RESIM 2006, 6th International Workshop on Rare Event Simulation, 2006, Bamberg, Germany 45

[38] W. Yang, N. Abu-Ghazaleh, *GPS: A General Peer-to-Peer Simulator and its Use for Modeling BitTorrent*, Proceedings of 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '05) , Sept 27-29, 2005, Atlanta. 46, 55

[56] K. Calvert, M. Doar, E. W. Zegura. *Modeling Internet Topology*, IEEE Communications Magazine, June 1997. 49, 59

[40] P. Erdős, and A. Rényi, *On Random Graphs* in Publ. Math. Debrecen 6, p. 290297 (1959). 56

[41] M. Mathis, J. Semke, and J. Mahdavi. *The macroscopic behavior of the tcp congestion avoidance algorithm*, SIGCOMM Comput. Commun. Rev., 27(3):6782, 1997. 55

[42] C. Gkantsidis, P. Rodriguez *Network Coding for Large Scale Content Distribution*, INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, Vol. 4 (2005), pp. 2235-2245 vol. 4.

[53] D. Bickson, R. Borer *The BitCod Client: A BitTorrent Clone using Network Coding* In P2P '07: Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing (2007), pp. 231-232 19, 20

[61] H. Luan and D. H.K. Tsang *A Simulation Study of Block Management in BitTorrent* In Proceedings of the 1st international conference on Scalable information systems (InfoScale 2006) (01 June 2006) 19, 20

[45] B. Dodson, D. Petkanics, Z. Ives, S. Khanna, *Combining coding and block schemes for p2p transimissions* 19, 20

[46] B. Li, Y. Keung, S. Xie, F. Liu, Y. Sun, and H. Yin, *An Empirical Study of Flash Crowd Dynamics in a P2P-based Live Video* Streaming System, in Proc. of IEEE Globecom, Nov. 2008. 48

[47] C. Wu, B. Li, Senior Member *rStream: Resilient and Optimal Peer-to-Peer Streaming with Rateless Codes* IEEE Transactions on Parallel and Distributed Systems (2008). 19

[48] M. Grangetto, R. Gaeta, and M. Sereno, *Rateless codes network coding for simple and efficient p2p video streaming* in Proc. of IEEE International Conference on Multimedia and Expo, July 2009. 19

[49] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications* in Proceedings of the ACM SIGCOMM '01 Conference, August 2001 12

[50] Petar Maymounkov, David Mazires *Kademlia: A Peer-to-peer Information System Based on the XOR Metric* 12

[51] R. Ahlswede, N. Cai, S. R. Li, and R. Y. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46:1204–1216, 2000. 3, 17

[52] A. Al-Hamra, A. Legout, and C. Barakat. Understanding the Properties of the BitTorrent Overlay. *Computing Research Repository (CoRR)*, abs/0707.1820, 2007. 2, 14

[53] D. Bickson and R. Borer. The BitCod Client: A BitTorrent Clone using Network Coding. In *Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing (P2P 2007)*, 2007. 19, 20

[54] V. Bioglio, R. Gaeta, M. Grangetto, and M. Sereno. On the fly gaussian elimination for LT codes. *IEEE Communication Letters*, 13(2):953–955, December 2009. 68

[55] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2005. 70

[56] K. Calvert, M. B. Doar, A. Nexion, and E. W. Zegura. Modeling Internet Topology. *IEEE Communications Magazine*, 35:160–163, 1997. 49, 59

[57] S. Floyd, M. Handley, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification, Sep 2008. URL http://www.aciri.org/RFC/rfc5348.txt. 67, 69, 70

[58] C. Gkantsidis, J. Miller, and P. Rodriguez. Anatomy of a P2P Content Distribution System with Network Coding. In *Proceedings of the 6th International workshop on Peer-To-Peer Systems*, 2006. 3, 18

[59] C. Gkantsidis, J. Miller, and P. Rodriguez. Comprehensive view of a live network coding P2P system. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement 2006*, 2006. 3, 18

[60] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM*, 2005. 3, 18

[61] L. Hao and D. H. K. Tsang. A simulation study of block management in BitTorrent. In *Proceedings of the 1st international conference on Scalable information systems (InfoScale '06:)*, 2006. 19, 20

[62] A. Legout, G. Urvoy-Keller, and P. Michiardi. Understanding BitTorrent: An experimental perspective. Technical report, INRIA Sophia Antipolis / Institut Eurecom, November 2005. 2, 14

[63] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest First and Choke Algorithms Are Enough. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006. 2, 14

[64] S. R. Li, R. Y. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49:371–381, 2003. 3, 17

[65] M. Luby. LT codes. In *IEEE FOCS*, pages 271–280, November 2002.

[66] G. Ma, Y. Xu, M. Lin, and Y. Xuan. A Content Distribution System based on Sparse Linear Network Coding. In *Proceedings of the Third Workshop of Network Coding, NetCod*, 2007. 3, 18

[67] D. Ming Chiu, R. W. Yeung, J. Huang, and B. Fan. Can network coding help in p2p networks. In *Proceedings of the Second Workshop of Network Coding, NetCod*, 2006. 3, 18

[68] S. Spoto, R. Gaeta, M. Grangetto, and M. Sereno. BitTorrent and fountain codes: friends or foes ? In *Proceedings of the Seventh International Workshop on Hot Topics in Peer-to-Peer Systems (HotP2P 2010), Satellite Workshop of the IEEE International Parallel & Distributed Processing Symposium (IPDPS 2010)*, 2010.

[69] M. Wang and B. Li. How Practical is Network Coding? In *Proceedings of the Fourteenth IEEE International Workshop on Quality of Service (IWQoS 2006)*, pages 274–278, 2006. 3, 18

[70] http://www.gnu.org/philosophy/gnutella.html 2, 12

[71] http://music.napster.com 2

[72] http://www.emule-project.net/ 2

[73] www.kazaa.com 2

[74] http://www.skype.com 2

[75] Ho, Tracey and Koetter, Ralf and Mdard, Muriel and Karger, David R. and Effros, Michelle (2003) The benefits of coding over routing in a randomized setting. In *IEEE International Symposium on Information Theory (ISIT '04)*, Chicago, IL, 27 June-2 July, 2004. IEEE , Piscataway, NJ, p. 442. ISBN 0-7803-7728-1 17

[76] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPS and P2P users cooperate for improved performance? Sigcomm Comput. Commun. Rev., 37(3), 2007. 21

[77] C. Buragohain, D. Agrawal, and S. Suri. A Game Theoretic Framework for Incentives in P2P Systems. In Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P '03), 2003. 21

[78] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In Proceedings of the 5th ACM conference on Electronic commerce (EC '04), 2004.

[79] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to- peer systems. In Proceedings of the ACM Sigcomm workshop on Practice and theory of incentives in networked systems (PINS '04), 2004. 21

[80] S. Guobin, W. Ye, X. Yongqiang, Y. Z. Ben, and Z. Zhi-Li. HPTP: Relieving the Tension between ISPs and P2P. In Proc. of IPTPS, 2007. 21

[81] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. A measurement study of a large-scale P2P IPTV system. IEEE Transactions on Multimedia, 9, December 2007. 21

[82] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should Internet service providers fear peer- assisted content distribution? In IMC '05: Proceedings of the 5th ACM Sigcomm conference on Internet Measurement, 2005. 1, 3, 4, 12, 20

[83] J. Kiss. ISPs fear iPlayer overload, Monday August 13 2007. http://www.guardian.co.uk/media. 1, 12, 20

[84] M. H. Lin, J. C. S. Lui, and Dah-Ming Chiu. An ISP-friendly File Distribution Protocol: Analysis, Design and Implementation. IEEE Transcations on Parallel and Distributed Systems. to appear. 20

[85] W.S. Lin, H.V. Zhao, and K.J.R. Liu. Incentive Cooperation Strategies for Peer-to-Peer Live Multi- media Streaming Social Networks. In IEEE Transactions on Multimedia, 11(3):396412, April 2009. 21

[86] Y. Liu, L. Guo, F. Li, and S. Chen. A Case Study of Trac Locality in Internet P2P Live Streaming Systems. In Proceedings IEEE International Conference on Distributed Computing Systems (ICDCS '09), 2009. 21

[87] F. Picconi and L. Massoulie. ISP Friend or Foe? Making P2P Live Streaming ISP-Aware. In Proceedings IEEE International Conference on Distributed Computing Systems (ICDCS '09), 2009. 20

[88] V. Reddy, Y. Kim, S. Shakkottai, and A. L. Narasimha Reddy. Designing ISP-friendly Peer-to-Peer Networks Using Game-based Control. Technical report, The Computing Research Repository (CoRR), February 2009. http://arxiv.org/abs/0912.3856.

[89] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz. P4p: Provider Portal for Applications. In Proceedings of Sigcomm, 2008. 21

[90] Osama Abboud, Aleksandra Kovacevic, Kalman Graffi, Konstantin Pussep, and Ralf Steinmetz. Underlay awareness in p2p systems: Techniques and challenges. In IEEE Computer Society, editor, *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, New York, USA, May 2009. IEEE Computer Society Press, IEEE Computer Society Press. 21

[91] György Dn, Tobias Hossfeld, Simon Oeschner, Piotr Cholda, Rafal Stankiewitz, Ioanna Papafili, and George. D. Stamoulis. Interaction patterns between p2p content distribution systems and ISPs. *IEEE Communications Magazine*, September 2011. To Appear.

[92] Jessie Hui, Wang Dah, and Ming Chiu. Modeling the peering and routing tussle between isps and p2p applications. 20

[93] Bo Liu, Yi Cui, Yansheng Lu, and Yuan Xue. Locality-awareness in bittorrent-like p2p applications. *Trans. Multi.*, 11:361–371, April 2009. 22

[94] Habib Rostami and Jafar Habibi. A mathematical foundation for topology awareness of p2p overlay networks. In *In the 4th International Conference on Grid and Cooperative Computing (GCC2005), Springer LNCS*, pages 906–918. Springer, 2005. 20

[95] A General Peer-to-Peer Simulator and its Use for Modeling BitTorrent Weishuai Yang and Nael Abu-Ghazaleh; Proceedings of 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '05) , Sept 27-29, 2005, Atlanta [PDF] 22

[96] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee. On multiple description streaming with content delivery networks. In *In Proceedings of IEEE Infocom*, 2002.

[97] D. A. Eckhardt and P. Steenkiste. Effort-limited fair (ELF) scheduling for wireless networks. In *In Proceedings of IEEE Infocom*, 2000. 96

[98] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should Internet service providers fear peer-assisted content distribution? In *IMC '05: Proceedings of the 5th ACM Sigcomm conference on Internet Measurement*, 2005. 97

[99] R. Kumar, Y. Liu, and K. W. Ross. Stochastic Fluid Theory for P2P Streaming Systems. In *In Proceedings of IEEE Infocom*, 2007. 85

[100] M. H. Lin, J. C. S. Lui, and Dah-Ming Chiu. An ISP-friendly File Distribution Protocol: Analysis, Design and Implementation. *IEEE Transcations on Parallel and Distributed Systems.* to appear. 86, 87, 89

[101] W.S. Lin, H.V. Zhao, and K.J.R. Liu. Incentive Cooperation Strategies for Peer-to-Peer Live Multimedia Streaming Social Networks. *In IEEE Transactions on Multimedia*, 11(3):396–412, April 2009. 86

[102] N. Magharei and R. Rejaie. ISP-Friendly P2P Streaming. *IEEE Multimedia Communications Technical Committee E-Letter*, October 2009. 96, 97

[103] M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* The MIT Press, 1994. 101, 107

[104] G. Owen. *Game Theory.* Academic Press, 1995. 96

[105] F. Picconi and L. Massoulie. ISP Friend or Foe? Making P2P Live Streaming ISP-Aware. In *Proceedings IEEE International Conference on Distributed Computing Systems (ICDCS '09)*, 2009. 96

[106] V. Reddy, Y. Kim, S. Shakkottai, and A. L. Narasimha Reddy. Designing ISP-friendly Peer-to-Peer Networks Using Game-based Control. Technical report, The Computing Research Repository (CoRR), February 2009. http://arxiv.org/abs/0912.3856. 91, 101, 107

[107] J. Weibull. *Evolutionary Game Theory.* The M.I.T. Press, 1995. 86

[108] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz. P4p: Provider Portal for Applications. In *Proceedings of Sigcomm*, 2008. 101
91

[109] Bittorrent protocol specifications. http://www.bittorrent.org/beps/bep_0003.html.

[110] Emule project website. http://www.emule-project.net/.

[111] Gnutella website. http://www.gnutella.com/.

[112] Kazaa website. http://www.kazaar.com/.

[113] Skype website. http://www.skype.con/.

[114] V.G. Cerf. IAB recommended policy on distributing internet identifier assignment and IAB recommended policy change to internet "connected" status. RFC 1174 (Informational), August 1990. 114

[115] J. Hawkinson and T. Bates. Guidelines for creation, selection, and registration of an Autonomous System (AS). RFC 1930 (Best Current Practice), March 1996. 113

[116] C.L. Hedrick. Routing Information Protocol. RFC 1058 (Historic), June 1988. Updated by RFCs 1388, 1723. 114

[117] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris. The Peersim simulator. http://peersim.sf.net. 8, 128

[118] Taoyu Li, Yuncheng Zhu, Ke Xu, and Maoke Chen. Performance model and evaluation on geographic-based routing. *Comput. Commun.*, 32:343–348, February 2009. 21

[119] B. Mandelbrot. Information theory and psycholinguistics: A theory of word frequencies. MIT Press, MA, USA, 1967. 127, 128

[120] Ricardo Oliveira, Mohit Lad, Beichuan Zhang, and Lixia Zhang. Geographically informed inter-domain routing. 21

[121] D. Oran. OSI IS-IS Intra-domain Routing Protocol. RFC 1142 (Informational), February 1990. 114

[122] D. Perkins. Point-to-Point Protocol for the transmission of multi-protocol datagrams over Point-to-Point links. RFC 1171 (Draft Standard), July 1990. Obsoleted by RFC 1331. 114

[123] D. Perkins and R. Hobby. Point-to-Point Protocol (PPP) initial configuration options. RFC 1172 (Proposed Standard), July 1990. Obsoleted by RFCs 1331, 1332. 114

[124] Y. Rekhter and P. Gross. Application of the Border Gateway Protocol in the Internet. RFC 1655 (Proposed Standard), July 1994. Obsoleted by RFC 1772. 114

[125] J. VanBokkelen. Responsibilities of host and network managers: A summary of the "oral tradition" of the Internet. RFC 1173 (Informational), August 1990. 114

[126] Q. Vohra and E. Chen. BGP Support for Four-octet AS Number Space. RFC 4893 (Proposed Standard), May 2007. 114