Imperial College London

Department of Computing

Challenges in Cooperation between Internet Service Providers and Peer-to-Peer Applications

by

Konstantinos G. Gkerpinis

Submitted in partial fulfilment of the requirements for the MSc Degree in Advanced Computing of Imperial College London

September 2010

Abstract

Most modern Peer-to-Peer applications select their neighbors at random, in a way which is totally network oblivious. This practice leads to a construction of an overlay topology which negates the underlay routing conducted by Internet Service Providers. The latter lose their ability to traffic engineer their networks, while the applications are forced to experience degraded performance. One of the most promising proposed solutions to this problem is to have the ISPs guide P2P clients in the selection of their peers via servers called oracles. An application may send a list of candidate peers to an oracle. The oracle can rank the peers according to criteria such as locality or congestion and return the ordered list to the client.

The project explores the challenges lying in this cooperation between ISPs and P2P applications. It begins with a study of a large repertoire of ranking criteria which can be employed by the oracles. Through extensive experiments, we demonstrated the merits and demerits of each of these strategies and explored how they can be combined for the formulation of more elaborate ranking policies. As a next step, the project provides an investigation of the weaknesses of the 'oracle' approach. Emphasis is placed on the problem of malicious oracles, which was shown to have a detrimental effect on the performance of the P2P clients, as well as on the well-being of the whole network.

We propose two novel detection algorithms which can be utilized by the applications for the identification of hostile oracles. The adoption of our detection schemes was found to improve the download performance of the P2P protocol by 15% and reduce the utilization of the bottleneck links by up to 17%. Finally, numerous variations and enhancements of these algorithms are discussed and evaluated. Among those, we suggest a detection mechanism which can be used for the discovery of the optimum oracle in a multi-oracle environment.

Acknowledgements

First and foremost, I would like to thank my project supervisor, Dr Peter Pietzuch, for his invaluable guidance throughout the project. I am deeply indebted to him for his advice and his constant encouragement.

I would also like to thank my mother Eleni, my father Grigoris and my sister Evangelia for their unlimited support during my studies at Imperial College.

Contents

Chapter 1

Introduction

The purpose of this introductory chapter is to provide a first brief overview of the area of the project and outline its nature and its scope. The first section includes an introduction to the Application Layer Traffic Optimization (ALTO) problem, which is the problem that the project addresses. A detailed presentation of ALTO and the existing approaches for its resolution will be provided in Chapter 2. Section 1.2 presents the aims and the contributions of the project and Section 1.3 describes the organization of the material within this report.

1.1 An introduction to the ALTO problem

During the last decade, the Peer-to-Peer (P2P) paradigm has become increasingly popular. The definition of the term "peer-to-peer" has been highly debatable. Yet, P2P applications yield a number of common characteristics. More specifically, they consist of autonomous entities which can communicate and exchange content without the use of a dedicated, centralized coordinator [1]. This constitutes a departure from the traditional client/server architecture and allows each Internet node to assimilate both roles by simultaneously sending and retrieving content from other nodes called peers.

Modern P2P applications have their roots in older distributed systems used for file sharing such as Usenet [2]. The field was revolutionized in the late 90s, when Napster gained extreme popularity overnight [3]. Although legal actions resulted in the demise of Napster's domination, many new P2P applications emerged as potential successors, such as Kazaa [4], eMule [5] etc. Nowadays, P2P still consumes the largest portion of the Internet resources. As of June 2010, Cisco calculates that P2P file-sharing networks carry 3.5 exabytes per month, accounting for 55.6% of the total usage [6]. Furthermore, it is estimated that P2P traffic will grow at a compound annual rate of 16% until 2014. This is in accordance with related studies (such as that of Karagiannis et al. [7]) which have demonstrated that, although P2P has managed to hide or masquerade as other types of traffic, in reality it is still growing.

On the one hand, this boom in P2P usage can be seen as a positive development for Internet Service Providers (ISPs). P2P is regarded as one of the main reasons why users choose to pay for faster access links [8]. As a result, it can be considered to be a source of revenue for the ISPs. On the other hand, P2P has introduced fundamental changes to the structure and the functionality of the Internet. For instance, by multiplying the amount of traffic load of the links, P2P applications force the ISPs to upgrade the speed and the quality of their backbone links. Furthermore, the P2P paradigm has led to a reconsideration of the asymmetrical bandwidth assumption [7]. As individual hosts can function

both as clients and as servers, the demand for larger upload speeds emerged. The traditional ADSL links may not be able to sustain this large bidirectional flow of traffic anymore.

More importantly, P2P influenced the fundamentals of the Internet traffic control. Traditionally, ISPs traffic-engineered their networks in the context of a given demand pattern [9]. With P2P, this demand pattern is much more dynamic. P2P applications enjoy the privilege of being able to select the remote hosts with which they communicate. The content can be retrieved from multiple candidate peers and thus a potential requester can choose among them. This flexibility in the selection of the destinations demands a reconsideration of the old conventional traffic engineering policies of the ISPs. What is worse is that most of the times the selection of the remote peer is made at random and is totally network oblivious. In other cases, it may be based on inaccurate and outdated information or may be guided by inappropriate criteria.

Ultimately, the choices of the user may lead to the construction of an overlay topology, which is in no accordance with the underlay Layer-3 topology. This can practically nullify the Network Layer routing that is conducted by the ISPs. No matter how the providers decide to forward their traffic, they are restricted by the uninformed and continuously changing peering decisions of the application layer. This disharmony between the overlay and the underlay topology causes problems both to the P2P users and to the ISPs themselves. The former experience degraded performance, while the latter lose the ability to manage the traffic traversing their network. This problem was defined by IETF as the Application-Layer Traffic Optimization (ALTO) problem [10, 11, 12]. It does not only apply to P2P applications; it can be encountered in other use cases as well, as it will be discussed in Chapter 2.

So far, a variety of solutions have been proposed by multiple research groups, but they can all be classified into three main categories:

1. The approaches of the first category (such as that of Madhyastha et al. [13]) do not involve any interaction between the applications and the ISPs. They are mainly based on measurements conducted by the application clients (or trusted entities acting on behalf of them) and aim at reverse-engineering the network.

2. The second group of solutions (such as that of Xie et al. [14]) encourages the cooperation between the application users and the ISPs. They are all based on the same principle: the ISP runs a server which can provide guidance in the selection of the remote party. When a client wishes to discover the optimal remote host to connect to, it can query this server and ask it to rate or rank the candidate hosts, according to a suitable criterion. The server's recommendation can aid the client in making the optimal decision, which will be of benefit both to the application itself and to the ISP. A variety of different strategies can be employed by the server for the rating/ranking of the candidate hosts. Others may be based on locality, while others may involve performance or charging criteria. A server of this type will be subsequently referred to as an *Oracle*, following the terminology of Aggarwal et al. [15]. Similarly, this set of solutions will be referred to as *the Oracle approach*.

3. Finally, the innovative solutions of the third group (such as that of Jacobson et al. [16]) propose a clean-slate redesign of the Internet and a reconsideration of the way routing is currently performed on the network layer.

1.2 Contributions of the Project

The project begins with a detailed survey of the existing work on the ALTO problem. The first challenge was to explore all the different proposals, identify their common features and group them

according to the principles on which they rely. This allowed for a careful comparison between them. From this study, the Oracle approach involving the collaboration between application users and ISPs emerged as the most interesting one to investigate. This was due to three reasons. Firstly, it currently draws most of the attention of the research community. Secondly, so far the experimental results have proved to be very promising. Thirdly, there are still many unresolved issues and unexplored pathways in this area. As a result, our work focused on the further study of this approach.

Nevertheless, the project does not intend to be a 'yet another' study confirming the benefits of an Oracle service and the merits of locality. On the contrary, it intends to build up on the existing work, fill some of the prominent gaps and address new challenges. The ultimate goal is to suggest solutions that further improve the performance of the Oracle approach. More precisely, the main contributions of the project are outlined below:

- Existing studies have mainly experimented only on a small subset of the potential ranking criteria which can be adopted by an oracle. The literature lacks a more detailed comparison of the proposed ranking techniques. This work aims at exploring a larger repertoire of these strategies. This will allow for a careful evaluation of how they influence the performance of the two parties (application users and ISPs). The project discusses the merits and the demerits of each criterion and demonstrates how different policies can be combined in order to ensure more sophisticated and accurate ALTO guidance.
- Our work sheds light on some of the weaknesses of the Oracle approach which have not been fully exposed. The main emphasis is placed on the problem of '*malicious*' Oracles, i.e. non-cooperating oracles which provide misleading service. Experiments will demonstrate that the existence of these oracles may make the overall system collapse and may have detrimental effects on the performance of the P2P applications, as well as on the greater good of the Internet.
- Novel solutions to the problem of malicious oracles are proposed. More precisely, two Detection algorithms which can be used for the identification of a misleading ALTO service are developed. These two detection strategies are discussed in detail, they are compared and they are tested on numerous different types of oracles. Experimental results illustrate that the adoption of these detection schemes can considerably improve the download performance of a P2P file-sharing application while at the same time guarantee a better distribution of the traffic load among the links of the network. Finally, different alternative versions of these two detection strategies are discussed, along with a proposal involving the utilization of detection for the discovery of the optimum oracle.

1.3 Structure of the report

The report is organized as follows:

- Chapter 2 provides a background study of the ALTO problem and its consequences in more detail and enumerates the different contexts within which it can be encountered. It also explores existing approaches for solving the problem and discusses their benefits and drawbacks. Emphasis is placed on the proposals which involve cooperation between the application users and the ISPs with the help of an Oracle-like entity.
- Chapter 3 describes the architecture of a distributed system which can benefit from the incorporation of an Oracle solution. This system will be used as a basis for the subsequent evaluation of the proposed techniques and algorithms. BitTorrent was selected as an appropriate P2P protocol for our study. The chapter begins by explaining how the entities of a BitTorrent swarm interact

with each other and outlines the rules and the elements of the protocol which were considered in the design. It then proceeds to analyze how this architecture can be enhanced with the introduction of an Oracle solution and presents the features of the protocol used for the interaction between an Oracle and a BitTorrent tracker. Finally, the chapter provides a complete list of all the ranking strategies which can be employed by an oracle of our system, both sincere and malicious ones.

- Chapter 4 includes a presentation of the network topology which was used for the evaluation of the system. Firstly, the requirements that such a topology must fulfill are outlined. Then, a detailed description of the selected network map is provided. Its construction was based on careful research and consideration of data collected from numerous resources.
- Chapter 5 is concerned with the implementation of the system within the SSFNet simulation environment. A short presentation of SSFNet is given and it is accompanied by an explanation of the reasons why it was selected as a suitable modeling framework. After that, more details of the implementation are presented. These include the structure of the designed code, the software engineering principles which were adopted and the programming challenges which were encountered.
- Chapter 6 provides the first evaluation of the designed system. It begins by confirming that the Oracle solution can significantly improve the download performance of the P2P system, as well as satisfy the desires of the ISPs. Then, it continues by supplying a detailed evaluation of the different implemented ranking strategies and compares and contrasts their outcomes. It also shows how the combination of these strategies into more advanced ranking criteria can further enhance performance. Finally, and most importantly, it demonstrates some of the weaknesses of the approach, placing particular emphasis on the problem of malicious oracles.
- Chapter 7 contains the heart of the contribution of this project. It describes the algorithms which we propose for the detection of malicious non-cooperating oracles. Two distinct detection techniques are explored, each of which is based on a different principle. The strong and weak points of both strategies are analyzed and a number of variants and extensions are proposed. Moreover, the Chapter surveys how detection can be used for the discovery of the optimal oracle in scenarios with more than one oracles per AS.
- Chapter 8 proceeds with an evaluation of the two detection strategies and their alternative versions. The proposed algorithms are shown to improve the overall performance of the P2P application, as well as contribute to the well-being of the whole network. The strategies are compared with each other and the results of this comparison are explained in detail.
- Finally, Chapter 9 concludes by summarizing the results and the contributions of this study and gives an outlook on future work.

Chapter 2

Background Research

This chapter presents the existing literature on the ALTO problem and the potential solutions which have been proposed. Section 2.1 analyzes the problem and its detrimental consequences and enumerates the different contexts within which it can be encountered. The following sections discuss the three categories of suggested solutions. Section 2.2 presents techniques which attempt to reverse-engineer the network without the cooperation of the ISPs. These approaches mainly rely on measurements and exhibit innate weaknesses. Section 2.3 lists the benefits of the collaboration between applications and ISPs. Section 3.3 presents a totally different suggestion which involves a clean-slate redesign of the current network architecture.

2.1 Analyzing the ALTO problem

One of the primary objectives of the Internet Service Providers is to manage the traffic which crosses their network. To this direction, they employ techniques which aim at finding optimal routes from a specific source to a specific destination [10]. These solutions are applied on the Network Layer (Layer 3) or sometimes on the Link Layer (Layer 2) and attempt to optimize routing, based on criteria such as latency, link capacity, available bandwidth or even financial costs of links. This is at the best interest of both the ISPs themselves and their customers.

Nowadays, distributed applications (both peer-to-peer and client/server) consume a large fraction of the network resources in order to exchange data. Many of these applications exhibit a common feature: they have the freedom to select the nodes with which they exchange content. The same service is often offered by multiple principals and thus a potential requester of the service can select among them. The ALTO problem stems from the fact that this selection is performed in a way which is network oblivious. The applications have little or no knowledge of the underlying network architecture and make their choices either at random or based on inaccurate or deficient information which may lead to suboptimal solutions. As a result, this selection of destinations imposes an overlay topology which negates the underlay routing performed on the Network Layer.

The heart of the problem is exactly this clash between the underlay (mainly Layer-3) routing conducted by the ISPs and the dynamic Layer-5 topology determined by the applications. The underlay routing is doomed, as it is restricted by the unpredictable destination selections performed on the application layer, while any attempts for sophisticated overlay 'routing' are bound to fail, as the applications lack knowledge of the characteristics of the network. All these cause problems both to the ISPs and to their clients.

As it has been discussed in the introduction, a typical example is that of the bandwidth-greedy P2P file-sharing applications. In P2P, a client seeks to retrieve content which is available in replicas in multiple peers. The client chooses a subset of these peers and sets up connections to them. A random network-oblivious peering can result in various undesirable situations [9]. Firstly, a client may select to peer with a distant node, even if the same content can be retrieved from other peers which are closer. For example, a node in London may end up peering with a node in Sydney, even if the same content is available on a node in Edinburgh. Such a scenario could obviously lead to inefficiency (the benefits of localized peering have been demonstrated by Karagiannis et al. in [17]). Secondly, even locality is not always the answer to the problem. A peering with a local node over a congested link or over an expensive inter-domain link may probably be a bad choice.

The problem, as described, harms both the ISPs and the users [18]. ISPs lose control over the traffic traversing their internal network, which they cannot manage and engineer. This effect is boosted by the oscillation in the behavior of the user applications, which constantly react to any traffic management attempt and adapt the overlay topology [9]. In addition, as ISPs cannot limit the transit traffic which crosses inter-domain links, they are burdened with the high costs of the utilization of these links. Even for a Tier-1 ISP (provider ISP), heavy traffic over inter-domain links may be obnoxious because it might lead to potential violations of peering agreements with other Tier-1 ISPs.

Meanwhile, user applications experience degraded performance (e.g. low download times). This stems from the inability of these applications to determine the optimal paths by themselves, as well as from the inability of the ISPs to control phenomena such as link overloading and congestion under the given circumstances. Finally, potential attempts of the applications to reverse-engineer the network by themselves via measurements imposes an extra load on each one of them individually (and of course an extra traffic overhead on the network).

There are numerous manifestations of the ALTO problem in a variety of distributed systems, all of which involve the selection from a list of candidate destinations. An enumeration of these use cases is provided below [10]:

1. **Standard P2P file-sharing**: Up to this point, research has focused mostly on this case, due to the large amount of traffic which it generates and the increasing popularity of protocols such as BitTorrent [19]. The scenario is simple: a user exchanges files (or segments of files) with other users (peers). A file (or file segment) can be retrieved from a number of peers and thus a subset of them must be selected, either from the user himself or from a central coordinator such as a tracker.

2. **Mirror selection**: Media or software distributors often maintain copies of their content on caches or servers at different geographical locations, in order to ensure robustness and load balance. Again, the user has to select a server among the candidate ones. Unfortunately, the choice is either totally random or is based on inaccurate geographical criteria.

3. **Media Streaming**: In this scenario, media content is provided by distributors to multiple clients, either real-time or near-real-time. The users in their turn actively participate in the distribution of the content by sharing it with other users-peers. This paradigm is usually referred to as *Hybrid P2P-Content Distribution Networking Architecture* (or more simply *Peer-assisted CDN*).

   Such an architecture which integrates CDN- and P2P-based media distribution is thoroughly presented by Xu at al. [20]. In this approach, when a new media file is released, the CDN server begins distributing it to requesting clients. New clients who join the system can retrieve parts of the content either directly from the CDN server or from other clients-peers. The CDN server maintains a list of active supplying clients and functions as an 'index server' or a 'tracker' by

connecting requesting peers with supplying peers. Once the P2P streaming capacity is sufficient, the CDN server stops supplying content and can proceed to remove it from its storage (this transition is called a *handoff*). It then continues to act only as an 'index server' and any new requesting client will have to retrieve the content from other peers. *Limited contribution* policies can be applied by the server in order to control phenomena such as free-riding. Of course, the handoff mechanism proposed by Xu et al. is optional and the CDN servers can continue to 'seed'.

Other analogous hybrid architectures have been discussed by Pakkala et al. [21] and by Karagiannis et al. [17]. In [22], Huang et al. studied the hybrid P2P-CDN paradigm by conducting experiments on Akamai and Limelight, with users downloading both from CDN servers and from peers. The future of this technology seems promising (VeriSign, CacheLogic, Grid Networks, Internap and Joost have already announced Hybrid P2P-CDN services [22]), however the problem of selecting optimum peers is encountered here as well.

4. **Real-Time Communications**: Communications applications allow for real-time exchange of text messages as well as media such as audio or video. Nevertheless, this traffic is often obscured due to NATs, firewalls or proxies. As a remedy, these applications forward their traffic towards relay nodes. Again, the issue of choosing from a list of candidate relays arises and current solutions are not always satisfactory. For example, Skype's relay detection mechanism was proven to be unable to select the optimum relay nodes in terms of round-trip time [23].

ISPs have attempted to take action in order to moderate the consequences of the ALTO problem on them. However, most such measures have been against the interests of the users. Ultimately, they have not succeeded in solving the problem. Some of the actions taken by the ISPs are summarized below [24]:

- Many content exchanging applications used to run over fixed ports. ISPs tried to shape, restrict or block all traffic that was forwarded towards these particular ports. In defense, modern applications are designed to run over non-standard randomly selected ports.
- ISPs have tried to use deep packet inspection in order to limit traffic associated with certain protocols such as BitTorrent. In response, user applications (e.g. uTorrent) resort to encryption techniques. In addition, many customers have chosen to abandon an ISP which imposed such strict traffic shaping regulations and switch to another.
- ISPs have sometimes tried to apply debatable techniques of flow throttling. This has led regulatory authorities to intervene and investigate [25].

2.2 ISP-Independent Approaches

A first approach towards a solution to the ALTO problem is to have the application (or a trusted party acting on behalf of a set of applications) reverse-engineer the network. This can be achieved by conducting active measurements and using the results to infer the characteristics of the network. Relevant work is presented below.

2.2.1 IDMaps

IDMaps (Internet Distance Map Service) [26] is one of the earliest approaches that tried to face the problem of neighbor selection. It can be viewed as a measurement infrastructure which estimates the distance between Internet hosts by conducting low-head network probing. Distance is mainly measured in terms of latency (round-trip delay), a metric which can be collected very easily. Francis et al., the

designers of IDMaps, marked the importance of bandwidth as a performance metric, but they stated that its collection is very costly. Measurements are conducted from multiple intermediate points on the Internet.

In the IDMaps architecture, IP addresses are clustered into *Address Prefixes* (APs). An AP is a range of IP addresses and all hosts which fall into this range are considered to be equally distant from any other node on the Internet. APs are assigned to *Tracers*, entities which are distributed around the global network. The first step is the calculation of the distances between the APs and their corresponding Tracers, as well as between the Tracers themselves. These raw distances are called *Virtual Links*. Ultimately, the distance between two APs is the sum of three values: the distance of the first AP from its tracer, the distance of the second AP from its tracer and the distance between the two tracers. IDMaps provides its clients with the Virtual Links so that they can create virtual distance maps. Francis et al. demonstrated that IDMaps facilitated the selection of the nearest neighbor without imposing large overhead. However, the accuracy of the system needed to improve.

2.2.2 Network Coordinate Systems

A Network Coordinate System (NCS) maps the IP address of an Internet host to a low-dimensional coordinate space [27]. The vector distance between two points in this coordinate space illustrates the network distance between the corresponding hosts in the Internet. In most NCSs, the distance is defined in terms of round-trip propagation and transmission delay.

A variety of different NCSs have been proposed. In their NCS, Ng et al. [28] implemented a novel mechanism called Global Network Positioning (GNP). In GNP, a small set of hosts called *Landmarks* initially compute their own coordinates. The coordinates of the Landmarks are then distributed to the rest of the nodes, which can in turn calculate their own coordinates relative to those of the Landmarks. The number of dimensions of the NCS used in GNP is a crucial parameter of the approach. GNP was evaluated by Ng et al. and was found to perform better than IDMaps.

On the other hand, Vivaldi [29] is a NCS which does not require landmarks or any other type of distinguished hosts. It is also vary scalable, as it demands the collection of latency measurements only from a small number of hosts and thus minimizes the required communication. With Vivaldi, network probing can be loaded on top of the traffic generated by the applications and thus this technology can easily be deployed into them. Another coordination mechanism is PIC [30]. The main advantage of PIC is that it can compute accurate coordinates even when some nodes are malicious.

2.2.3 Meridian

Wong et al. [31] introduced Meridian, a new framework which offers node selection services based on network location. The designers stated that *network embedding*, i.e. the practice of constructing Coordinate Systems, is not the best approach, as it is inaccurate and incomplete. Thus, they departed from it and suggested a new loosely-structured architecture which makes use of direct (latency) measurements and does not need to map them onto an absolute globally-consistent coordinate space.

In Meridian, each node organizes its peers into concentric rings of exponentially increasing radii. A query related to a node selection problem is forwarded towards those nodes that are best fit to provide a solution (in the sense that they possess the required information to answer the query). Meridian also makes use of a gossip protocol which caters for the dissemination of information on potential peers. The system can be used to address issues such as the discovery of a central leader or the identification of the peers which are located in a region defined by latency constraints. It is a scalable and robust

architecture and generates less errors than the traditional coordinate systems. Its disadvantage is that it is less general when compared to them.

2.2.4 iPlane

Madhyasta et al. [13] proposed a new network service called iPlane (Information Plane), which provides a sophisticated up-to-date link-level map of the Internet. iPlane tries to infer end-to-end path properties using a small number of vantage points and without having to introduce new infrastructure. iPlane's power stems from the fact that it does not only rely on latency but considers a broader set of metrics such as bandwidth, capacity and loss rate. It can also view the network both from a global perspective and with respect to local isolated anomalies. It provides information to clients which query it but it also incorporates measurements collected by these clients individually.

iPlane divides paths into segments. The properties of each segment are predicted separately and then the results are combined so that the performance of the composite path is determined. In order to facilitate the discovery of path intersections, the system clusters network interfaces that belong to the same Autonomous System. The performance of each link is measured by the closest vantage point. iPlane has been tested against a variety of use cases. It was shown to reduce loss-rate and jitter in VoIP applications and it improved download times in BitTorrent file-sharing.

2.2.5 General Comments on ISP-Independent Approaches

Most of the reverse-engineering techniques which were described above are bound to fail to provide an adequate solution to the ALTO problem. This is due to a variety of reasons [32]:

1. Most of these systems determine the distance between two nodes based on latency. However, latency may not be the best metric to evaluate performance, especially when it comes to content distribution applications. Other parameters such as capacity, available bandwidth and loss rate may play an equal or more important role.

2. Many of these reverse-engineering algorithms have a slow convergence or may ultimately get trapped to a local sub-optimal solution. This is a feature which is also commonly encountered in the bandwidth reciprocity mechanisms of various BitTorrent clients [32].

3. The measurements which are conducted by the various interacting parties of these systems add a considerable amount of traffic load to the network.

4. Certain characteristics of the network are hard to reverse-engineer by using the aforementioned techniques, or they may even be impossible to infer. ISP policies driven by financial costs or by inter-ISP agreements fall into this category. In addition, some of the techniques used to determine the structure of the network fail to capture certain topological elements. For example, topologies with multipath routing or overlay paths shorter than the direct path might be hard to pinpoint.

2.2.6 The case of Ono

Choffnes et al. [24] presented a new approach, which tries to address the ALTO problem without conducting any path monitoring or probing, but again without the cooperation of the ISPs. This approach tries to achieve biased localized neighbor selection, by recycling network views gathered at low cost from Content Distribution Networks. Choffnes et al. designed a plugin to the Vuze (former

Azureus [33]) BitTorrent client, which implemented their idea. Their implementation was written in Java and was named Ono.

In general, CDNs comprise multiple geographically dispersed replicated servers. When a user requests content, he is dynamically forwarded to the optimum server, via DNS redirection or URL rewriting. The optimum server is most frequently the one across the path with the lowest latency. The CDN determines the redirection on the basis of information such as geographical data, connectivity or network measurements. CDNs maintain informed databases which reflect the status of the network and allow them to perform these redirections.

The Ono approach tries to exploit this information which is already collected by the CDNs. Instead of asking from the ISP to help them determine their peers, the Ono clients use the CDN for hints. When two clients query the CDN and are redirected towards the same replica server, it can be inferred that they are located close to the server and, by transition, close to each other. Thus, they constitute a good candidate pair of peers. Most probably, these peers will be located within the boundaries of the same ISP and a peering relationship should not involve inter-domain traffic.

Choffnes et al. defined the ratio map of an Ono client as a set of tuples *(server, frequency with which the client is redirected to the server)*. The similarity in the redirection patterns between two clients is measured by computing the *cosine similarity* of the ratio maps, which is a newly defined metric, analogous to the normalized dot product of two vectors. More information on this metric can be found in [24]. Ono prefers to peer with clients with which it has high cosine similarity values. The ratio maps of other clients can be obtained either via direct interaction with the corresponding clients or from distributed storage or trackers.

Using results collected from 120,000 Ono clients worldwide, Choffnes et al. showed that, with their new approach,

- over 33% of the time, the selected peers are within the same AS.
- the paths between the peers have two orders of magnitude lower latency and 30% lower loss rates.
- download rates can increase by up to 207% on average.

The main advantage of the Ono approach is that it does not require any additional infrastructure (e.g. servers, caches etc). Moreover, although it does not yield the benefits of the ISP-Cooperation approaches, it does not have to rely on a trust relationship between the ISPs and the users. This trust is a prerequisite for the success of these cooperation approaches which will be discussed later on. On the other hand, Ono's main weakness is that it totally relies on the CDN. Should the CDN choose to change its behavior, Ono would have to be reviewed. In addition, it is not known in the first place whether the redirections performed by the CDN are the optimal ones. Essentially, Ono does not solve the problem but asks another party to solve it instead.

2.3 Approaches involving the collaboration of the ISPs

As it has been explained, by conducting measurements, applications try to re-invent the wheel and most of the times this proves to be unsuccessful. A better idea would be to allow an application to exploit the information that is already in the hands of the ISPs. ISPs have full knowledge of the networks which they administer and it is at their best interest to help their clients construct their overlay topology. Thus, ISP-P2P collaboration is advisable.

The benefits of this cooperation for the ISPs are the following [34]:

- They regain their ability to traffic-engineer their networks. By better managing the traffic flow, they can improve the service which they provide to their customers and ensure fairness among the various types of applications (P2P, WWW traffic, E-mail services etc).
- They can achieve localization of the traffic within their networks. As a result, they can reduce the utilization of the transit links which connect them to their providers and thus minimize the cost which is incurred on them.
- The end users have no reason to conduct individual measurements in order to reverse-engineer the network. Thus, the latter is relieved from the burden of the traffic introduced by such measurements. This is obviously at the interest of the ISPs.

Similarly, end users also benefit from this cooperation:

- They are relieved from the tedious task of probing the Internet so as to reverse-engineer it.
- By taking advantage of the knowledge of the ISP, they can boost performance and achieve improved download times and lower latency.

2.3.1 Older studies

The study of Karagiannis et al. [17]. was one of the older ones which praised the idea of the collaboration between applications and ISPs. However, it did not propose a concrete way of exercising this cooperation. It suggested that the ISPs introduce *proxy-trackers* at the edge of their networks. These proxy-trackers would intercept requests for peers originating from the local clients and would then redirect them towards other clients within the same AS. Nevertheless, this solution was not investigated in detail, as the authors argued against the deployment of new infrastructure.

A similar pathway was followed by Bindal et al. in [35]. This study suggested that the ISPs utilize P2P traffic shaping devices installed on top of the edge routers in order to impose a biased neighbor selection. These devices can help an ISP keep track of the peers of a distributed application. Whenever a client sends a request for a new list of peers to a coordinating entity such as a BitTorrent tracker, the ISP can intercept the corresponding response and substitute the IDs of peers outside the local AS with the IDs of internal peers. Furthermore, as an extra method of guaranteeing locality, the ISP may choose to reset a subset of the TCP connections between external and internal peers, thus forcing the latter to pursue local neighbors. It is apparent that these proposed policies are more of an enforcement than a collaboration.

However, Bindal et al. discussed one more technique for implementing biased neighbor selection, which is closer to the modern *Oracle-like* approaches that will be described below. They suggested that ISPs could publish their IP address ranges to the distributed applications, encouraging them to select local destinations. Alternatively, they can communicate locality information by appending suitable HTTP headers to the messages exchanged by the users. For example, they proposed the introduction of a new header called 'X-Topology-Locality', which could contain a locality tag. The application trackers could make use of these tags to optimize their peer recommendations. This technique can be actually seen as a precursor of the *p-distances* of the P4P approach, which will be presented in section 2.3.3.

2.3.2 The Oracle Approach

Aggarwal et al. [15, 18, 27, 34] presented a simple but effective solution to the ALTO problem. They proposed the introduction of an *oracle*, a service provided by the ISP, whose purpose would be to guide the end users of a P2P system in the selection of their peers. Instead of selecting his peers at

random, a user can forward a list of candidate peers to the oracle. The oracle ranks the peers and returns the ranked list. The end user can then proceed to choose his peers on the basis of the oracle's recommendation.

Aggarwal et al. emphasize on the fact that, as the oracle is provided by the ISP, it has access to information on the structure and utilization of the network that can be used during the ranking process. The oracle can rank potential neighbors according to coarse-grained distance metrics such as:

- whether the potential neighbor is located inside or outside the AS
- the number of AS hops, as it can be derived by the BGP protocol
- the distance to the edge of the AS, as it can be derived from the applied Interior Gateway Protocol (e.g. OSPF)

Neighbors within the oracle's AS can further be ranked according to the following fine-grained criteria:

- geographical data
- performance-related metrics such as the available bandwidth or the expected latency
- the utilization of the intra-AS links

The end user has no obligation to adhere to the oracle's recommendation, but it is at his best interest to do so. Instead of ranking, the oracle can provide a simplified classification of the potential peers (e.g. in three categories: good, medium, bad). An AS may comprise only one Oracle server, or may have many replicated versions of the same oracle.

Akonjang et al. [36] designed a protocol which specifies the format of the messages that are exchanged between the Oracle server and a client. The *Oracle Protocol* was created by appropriately adapting DNS. UDP was used as the transport protocol, due to its speed and its stateless nature. The two main messages of the new protocol are the 'Query Message' (sent by the client to the server) which contains unsorted IP addresses and the 'Response Message' (sent by the server to the client) which contains sorted IPs. The Oracle ranks the IPs based on information retrieved by the Oracle Database. The Oracle Database contains static data (such as the network topology) as well as dynamic data (such as link congestion), which must be updated at specified time intervals.

In a sense, the oracle service provides an abstraction of a coordinate system. Its advantage against the other traditional coordinate systems is that it does not only rely on latency but also considers features such as the geographical location, the link capacity etc. Aggarwal et al. suggest that by enabling interaction between different oracles, the construction of a global coordinate system is possible. An oracle can provide an accurate ranking only for those nodes which are located within its AS. For nodes that are outside the AS, the oracle should first segregate them according to their parent ASs. Nodes belonging to ASs in the close neighborhood will be ranked higher than nodes belonging to ASs further away. Moreover, nodes in the immediate neighborhood can be further classified to nodes belonging to customer ISPs, peer ISPs or provider ISPs. Most probably, the oracle would assign a higher rank to the nodes belonging to customer ISPs, followed by those belonging to peer ISPs. Nodes belonging to provider ISPs would be ranked lower. If the oracle desires an accurate ranking for the nodes within a particular AS, it can send a list of these nodes to the oracle server of the corresponding AS and ask for the ranking. Nevertheless, Aggarwal's proposal for interaction between oracles was not investigated any further.

It could be argued that a weakness of the oracle service is that it reveals confidential information about the network topology and performance. Aggarwal et al. comment that the ranking criteria are not available to the users and an ordered list by itself cannot reveal much. Another debatable issue is whether the oracle logs could be used against the users in the context of piracy hunting. Aggarwal et

al. claim that, as the service can be used is a variety of use cases (from legal content distribution to real-time communications), it cannot be treated as a piracy logger. In other words, querying the oracle does not imply participation in the illegal exchange of protected content. In addition, to further protect their identity, the users can attempt to permute the last bytes of the IP addresses of the potential peers. Nevertheless, this is bound to affect the efficiency of the ranking.

The evaluation of the oracle approach in a P2P system produced the following results:

- With the help of the oracle, download times for P2P users decrease significantly. P2P users are still able to locate the available content with the same probability.
- The use of the oracle increases locality (sometimes even by a factor of 7).
- Localized P2P graphs maintain the nice graph properties which are typical of random overlays (small node degree, small graph diameter, small mean path length, connectedness), even under heavy churn.
- The approach maintains its benefits across different user-behavior patterns. Even the presence of a large number of free-riders does not adversely affect localization. Furthermore, the approach maintains its benefits across different network topologies.

2.3.3 The P4P Approach

Xie et al. [14, 9] proposed a new architecture called *P4P*. P4P stands for Proactive network Provider Participation for P2P, or simpler Provider Portal for P2P. The P4P framework is based on the same principles as the Oracle approach which was described above. It allows for cooperation between P2P users and ISPs towards a solution to the ALTO problem. Although the designers primarily focus on P2P applications, P4P can be used in a variety of use cases such as server selection, high bandwidth remote backup and online gaming.

The P4P Working Group [37], which is part of the Distributed Computing Industry Association (DCIA), leads the research in this particular field. It aims at promoting the adoption of the new technology by the involved parties, as well as at supporting researchers who work on this area. Its participants are prominent ISPs, P2P software distributors and technology researchers.

The P4P framework consists of three planes: the data plane, the management plane and the control plane. The *Data Plane* is optional and is concerned with differentiating and prioritizing application traffic. The *Management Plane* is responsible for monitoring the behavior of the control plane. The core of P4P's pioneering functionality lies within the *Control Plane*. The main entity of this plane is the *iTtracker*. The iTracker is the equivalent to the oracle which was described in Aggarwal et al.'s approach. An ISP can have one or more iTrackers for scalability and robustness. The IP address of the iTracker can be resolved by a simple DNS query (using DNS SRV with *p4p* as the symbolic name). P4P can be applied both on tracker-based P2P systems and in trackerless ones. In the case of a tracker-based system, the communication scenario is as follows:

- The end user registers with the application tracker.
- The application tracker queries the iTracker and determines the peers for the end user, taking into account the iTracker's recommendation as well as the application requirements.
- The application tracker notifies the end user of the peers that it has selected for him

In a trackerless system, the end user would have to interact directly with the iTracker.

The iTracker provides the following 3 interfaces:

- The *Policy* interface. Via this interface, the applications can get knowledge of the usage policies and the guidelines of the network. A policy is a strategy that determines how the ISP would like its network resources to be utilized. Examples of such strategies could be: coarse-grain time-of-day link usage policies, traffic ratio balance policies, heavy-usage thresholds etc.

- The *Capability* interface. This interface allows peers or content distributors to request 'capabilities' that are offered by the ISPs. For example, an ISP may provide different classes of services, caches or on-demand servers. An application tracker could ask for such a service to facilitate content distribution. It is possible that these capabilities could also function as a revenue for the ISPs.

- The *p4p-distance* interface. This is the heart of the P4P framework. Through this interface, the applications can ask for an evaluation of the intra- and inter-domain links of the network. The p4p-distances (also called p-distances) are actually cost values assigned to the network links and reflect the ISP's assessment of these links. The higher the p-distance, the less the ISP favors traffic forwarding over the corresponding link. p-distances can be determined on the basis of a variety of parameters, such as OSPF weights, BGP preferences, link utilization, financial cost of the links, congestion metrics etc. The following analysis will focus on the p4p-distance interface.

The iTracker represents the network as a graph $G = (V, E)$, where V is a set of nodes and E a set of links. Each node in the graph is referred to as a PID (opaque ID) and corresponds to a cluster of end users. In most cases, a PID represents a set of users that belong to the same Point of Presence. However, users may also be clustered into PIDs according to other criteria (such as exhibiting common network properties). Each time a new user enters the network, his IP address is mapped to a tuple (PID, AS). If two PIDs i and j are connected, the iTracker assigns a p-distance p_{ij} to the corresponding link. The iTracker can be queried by an application tracker and can provide the p-distances through the p4p-distance interface. The application tracker can then make its peering decisions based on these distances.

The application tracker can utilize the p-distances in a variety of ways. For example, it can treat the distances as simple ranks, i.e. the higher the distance, the lower the rank. In addition, it may select to apply robustness constraints, demanding that a client in a given PID has to connect to a minimum number of clients in other PIDs. Finally, it may want to maximize the matching of peer download and upload, as described by Qiu et al. in [38].

An ISP may enforce access control to some of the iTracker's services, in order to ensure security and privacy. It may also choose to supply more accurate information to preferred or trusted applications. However, this would negatively affect the neutrality of the network.

Xie et al. have developed a mathematical model to calculate p-distances using optimization decomposition. They begin with the ultimate ISP traffic engineering objective, which is to minimize the maximum link utilization. In order to find a distributed solution to the optimization problem, they introduce dual (shadow price) variables. This allows them to decompose the problem into independent sub-problems for each peering session and to decouple the iTracker from the applications. After the end of a cycle of interaction between the iTracker and its clients, the dual variables are updated using the projected super gradient method. More information on the model can be found in [9].

The recent news from field testing are very encouraging. Comcast, a large cable broadband ISP in the USA, released the first real-world data on P4P technology [39]. The P4P experiments were conducted in cooperation with Pando, Yale and three other P4PWG member ISPs, from July 2 to July 17, 2008. The results showed that P4P technology reduced Comcast's incoming Internet traffic by 80%. Moreover, the end users were seen to experience increased download times of up to 85%. Verizon, a leading Telecommunications, company conducted its own tests using Pando and observed download

performance improvements of approximately 200% [40]. The increase in performance was observed to climb up to 600% in some cases.

The Oracle approach can be viewed as a special case of the p4p-distance interface, in which a PID represents exactly one IP address and p4p-distances are ranks. The one-to-one PID-IP correspondence of the Oracle approach enables better fine-grained control. On the other hand, clustering IPs, as performed by P4P, has the advantage of exposing less confidential information about the network. At the same time, by avoiding per-user queries, it allows for better scalability. A simple ranking of the peers, as in the Oracle approach, offers simplicity and robustness. However, the richer p-distance semantics of P4P allow for more precise traffic control. It is obvious that both of the approaches have advantages and disadvantages, but in any case the basic principles behind them are the same.

2.3.4 ISP-Driven Informed Path Selection Service (IDIPS)

During the SHIM6 session of the 71^{st} IETF meeting, Saucez et al. [41] presented the ISP-Driven Informed Path Selection Service, which can also be used to address the ALTO problem. IDIPS is a distributed request-response service which is responsible for ranking paths on the basis of specified ranking criteria. It involves two parties: a client and the IDIPS server. The client has a list of m candidate source addresses, as well as a list of n candidate destination addresses. The occurrence of multiple source addresses is possible in a multihoming environment. The client wants to determine the optimum path, among all the possible $m \cdot n$ paths that can be formed. The interaction between the two parties is conducted as follows:

- The client sends a request to the IDIPS server containing the lists of the sources and the destinations. The client may also specify parameters or criteria which may be used by the server in the ranking process. For example, the request may contain the client's DSCP [42].
- The server receives the request and determines all the valid paths between the sources and the destinations. It then ranks the paths on the basis of the client's criteria as well as of other factors like

  1. routing information derived from BGP, OSPF, IS-IS etc.
  2. active or passive measurements such as RTT, bandwidth, TCP flow analysis etc.
  3. network policies

  The server creates a response message containing the ranked list of paths and returns it to the client. In the response, the server may also include a value indicating the time for which this list should be considered to be valid.
- The client receives the response and makes a selection based on the ranked list of paths.

The details of the IDIPS protocol can be found in [42]. In order to reduce the request load and encourage scalability, the server may select to rank paths between prefixes and not between complete addresses. The IDIPS service is typically intra-domain-oriented, but it may also be extended to allow for cooperation between different ISPs.

The IDIPS approach can be considered to be a more general version of the Oracle approach of Aggarwal et al. Its extra feature is that it caters for cases with multiple sources and thus can provide a back end for protocols like SHIM6 and LISP [32]. This may prove to be a valuable extension in multi-homing environments. In [43], Sausez et al. evaluated the processing time of the IDIPS server. The results showed that, from the client's point of view, the cost associated with IDIPS is negligible, i.e. no more than that of a DNS request.

2.3.5 The IETF ALTO Working Group

All the previous studies led IETF to realize the importance of the cooperation between the Network and the Application Layer. In order to proceed with standardization in this area, IETF formed the ALTO Working Group (ALTO WG) [44]. As it is stated in the charter of the group [45], its goal is to develop Internet standards for a service that will provide applications with the information they require in order to perform "better-than-random peer selections". Once again, the work of the group mainly focuses on P2P applications such as BitTorrent and Distributed Hash Tables, but also considers other use cases such as mirror selection. So far, its three main objectives can be summarized in the following list:

1. To define the problem with the appropriate formalism and to introduce terminology which can be used by all researching teams.
2. To explicitly state the requirements that an ALTO service must fulfill.
3. To design a concrete Request/Response protocol for the communication between the *ALTO Clients* and the *ALTO Servers*.

The WG states that the ALTO Client can be either a *Resource Consumer* or a *Resource Directory*. The term *Resource Consumer* is used to refer to hosts which wish to access a resource, such as a shared file [10]. The term *Resource Directory* is used to describe a party which is logically separate from the Resource Consumer and aids the latter in the discovery of potential *Resource Providers*. For example, in a BitTorrent system, the *Resource Directory* is the Tracker, while the *Resource Consumers* are the numerous BitTorrent clients. The WG also extends the idea of the Application-ISP collaboration. They state that the ALTO Service can be provided not only by the Network Operators, but also by third parties acting on behalf of the ISPs.

Under the umbrella of the ALTO WG, several proposals for a concrete ALTO Protocol have emerged. Some of them are totally new approaches, while others are previous studies (such as the Oracle and the P4P ones) which have been refined so that they comply with the IETF requirements. The following contributions are the most notable ones:

- The first major contribution came from Yang (a member of the P4P team) and Penno (who had previously worked on an architecture called ALTO Information Export Service [46]). This proposal combines features from these two studies. It mostly relied on the principles of P4P [47]. Its basic characteristic is that the server sends information about network regions to the client in advance. The client caches this information and uses it to rank candidate peers whenever it is required. Therefore, the rating/ranking of the peers is conducted within the client itself, on the basis of information provided by the server.
- **PROXIDOR**: The *PROXIDOR* [48] protocol is an attempt to combine different features from older approaches. Its design is the result of the collaboration of three parties: the group behind the Oracle protocol (Aggarwal and Feldmann), the developer of IDIPS (Sausez) and the team which suggested the Proximity Services (Davie and Previdi). *Proximity Services* [49] was a proposal for the extension of the existing routing databases (ISIS, OSPF, BGP) so that they facilitated localization services such as ALTO. As its name suggests, PROXIDOR incorporates characteristics from all these approaches (PROXimity, IDips and ORacle). It is practically an extension of the Oracle Protocol of Aggarwal et al. which supports multiple source addresses (in an IDIPS fashion). In PROXIDOR, the client sends a list of candidate destinations to the server and the latter ranks them and returns the evaluated list.
- **H12**: As it is obvious, the previous two contributions constitute two different approaches concerning the communicated data between an ALTO server and an ALTO client. In the first

proposal (called H1 [50]), the client does not have to reveal its destinations and thus retains its privacy. On the other hand, the server has to share sensitive (and sometimes confidential) information about its network. In the second proposal (referred to as H2), the client is obliged to reveal its candidate peers, while the oracle does not have to expose details of its network. H12 attempts two bridge the gap between these two approaches. The client may choose to send the oracle specific IP addresses or descriptions of larger Internet regions, depending on how much information it is willing to reveal. Similarly, the oracle may select to reply by ranking particular addresses or broader domains, according to its own policy. H12 uses HTTP over TCP. The encoding of the message body is done with XML.

The WG is trying to combine elements from all the above studies in order to finalize the definition of the ALTO Protocol. The current design is based on HTTP and uses a REST-like interface [11]. It includes both H1 and H2 approaches. The work is still in progress and it is constantly being reviewed.

Other issues that are currently being examined by the ALTO WG are [44]:

- **Security**: An important security concern is the way ALTO servers can be shielded against attacks generated by hostile ALTO clients (such as Denial of Service attacks, or attacks exploiting the vulnerabilities of the software). Another open issue is that of 'malicious' oracles which offer misguiding ALTO information. This area has not been explored in depth yet.
- **Collaboration with caching**: ALTO can be used for redirecting clients towards local caches which store the content they wish to acquire. The mechanism which will enable such a service is still a subject of research.
- **ALTO Discovery**: Finally, a lot of work is currently focused on the development of efficient algorithms that will enable applications to discover the ALTO servers. If the ALTO client is the same entity as the Resource Consumer, the discovery could be achieved using DHCP or PPP. However, if the ALTO client is a Resource Directory like a tracker, more advanced mechanisms have to be employed. These mechanisms are still under investigation.

Many of the members of the ALTO Working Group are supported by NAPA-WINE. NAPA-WINE (Network-Aware P2P Applications over WIse Networks) is a Specific Targeted Research Project (STREP) funded by the European Commission within the Seventh Framework Programme (FP7). The NAPA-WINE project is currently investigating how an ALTO service can be used in the case of Live P2P Television [51]. One of its main aims is to promote the cooperation between ISPs and P2P High Quality TV systems for the efficient construction of an overlay topology. NAPA-WINE has tested how the ALTO solution works on Live Streaming applications by conducting experiments within an event-based simulation environment [52]. The results showed that the peers which were supported by ALTO received chunks of data two times faster than the peers which made their selection based on individual measurements.

2.4 Redefining the Network Architecture

A more renovating approach towards a solution to the ALTO problem is to reconsider the current architecture of the Internet and develop a novel clean-slate network pattern which facilitates content distribution. At the moment, the location-oriented routing which is performed on the network layer is negated by the fact that applications select their peers on the basis of content exchange. If the network were built to make forwarding decisions based on the content in the first place rather than reference specific hosts/locations, the problem of bi-level routing would not be encountered. A number of studies towards this direction have been published. Some of these approaches are presented below, with more emphasis placed on Content Centric Networking.

2.4.1 Content Centric Networking

Jacobson et al. [16] introduced Content-Centric Networking (CCN), which is one of the most recent approaches based on the aforementioned idea. According to Jacobson et al., CCN is a communications architecture built on *named data*, rather than on *named hosts*. Packet 'addresses' do not name locations; they name content. CCN interestingly exhibits all of the characteristics of IP that contributed to its success. Firstly, it is simple and secondly it does not pose strong demands on the underlying layer-2 (stateless, unreliable, unordered, best-effort delivery suffice). CCN can either replace IP or be layered on top of it.

CCN has two types of packets: the *Interest* and the *Data* packets. An Interest packet signifies a host's interest in specific content. When a host wants to retrieve this content, it broadcasts Interest packets over all available connectivity. Data packets contain the actual content. They are sent as responses to Interest packets by hosts which have a copy of the requested content. A host can express interest in content that does not exist yet. A field called *ContentName* is used to match corresponding Interest and Data packets. *CCN names* are the CCN equivalents of IP addresses. They are opaque, binary objects and they consist of a number of components. Just like IP addresses, they are hierarchically composed and can easily be resolved via a similar mechanism. A database of CCN names can be represented as a *Name Tree*. A Name Tree is ordered and therefore the name of a segment of data X that follows a segment of data Y can be easily found by traversing the Name Tree.

When a packet arrives on a *face* of a CCN node, the node performs a longest-match lookup, in the same way an IP node would perform a traditional IP lookup. A CCN node has three data structures:

- The Forwarding Information Base (FIB). It is the equivalent to the IP FIB. It is used for forwarding Interest packets to sources that can potentially satisfy the interest. The IP FIB forwards packets only to one face, while the CCN FIB can forward replicas of an Interest packet to a number of outgoing faces.
- The Content Store (CS). It is the equivalent to the buffer memory of an IP router. Its difference is that it keeps the packets in memory as long as possible and thus applies a LRU or LFU replacement policy.
- The Pending Interest Table (PIT). PIT records express interest in data. As an Interest packet traverses CCN nodes, relevant records are added to the PIT tables of these nodes. In this way, the Interest packet leaves a trail, which can be followed by the Data packet in order for the latter to reach the original requester.

A detailed description of the operation of the CCN node can be found in [16]. Jacobson et al. analyze how the three structures are prioritized during the lookup process and how they are updated. What is important is that one can easily notice the structural and functional similarities between a CCN and an IP node. CCN's forwarding model is a superset of IP's forwarding model, but with fewer restrictions (e.g. it allows for forwarding towards multiple outgoing faces). This reinforces the belief that, should CCN be employed, the software running in traditional IP routers would not have to undergo radical changes. All the routing policies that are used for IP could easily be incorporated to CCN.

Security in CCN is addressed in a novel way. CCN does not secure the connections over which the content travels but the content itself. Thus, security is content-based, just like the rest of the network architecture. In traditional IP, the content has to be retrieved directly from the original sender (or a trusted intermediary) in order to be considered trusted. In CCN, as the content itself is secured, the locations which it traverses do not necessarily have to be regarded as trusted. CCN authenticates the binding between the name and the content and this signed binding constitutes a certificate for the content. As a result, keys can be distributed just like any other data. The authenticated binding

between a name and a key can be used to certify the key. Finally, it must be noted that CCN's design facilitates shielding against flooding attacks.

Other important aspects of the CCN architecture are summarized below:

- Data packets cannot loop in a CCN.
- CCN data senders are stateless. Thus, unlike in TCP, the original data requesters have the responsibility of re-asking for undelivered data, by resubmitting unsatisfied Interest packets.
- In CCN, flow balance operates in a hop-by-hop fashion, i.e. all intermediate nodes are involved in the flow control. This eliminates the need to apply policies to control congestion in the middle of a path.
- CCN can simultaneously use multiple layer-2 technologies (Ethernet, 802.11 etc). A *Strategy layer* is responsible for managing the concurrent use of these technologies and for making the optimal connectivity choices.

Jacobson et al. implemented a prototype of their approach and used it to conduct experiments. As far as Web traffic is concerned, CCN's performance was shown to surpass that of HTTPS. CCN also matched the performance of unsecured HTTP and this demonstrates its superiority, as it can guarantee both secure delivery and good download times. When it comes to content distribution, CCN's completion time was not observed to rise with the increase of the number of peers, as opposed to TCP's completion time which increased linearly. In [53], Jacobson et al. developed a Voice-over-CCN application in order to compare it with traditional VoIP. The results were satisfactory, but the delay in the delivery of some of the packets was noticeable.

2.4.2 Other Related Studies

TRIAD [54]: TRIAD (Translating Relaying Internet Architecture integrating Active Directories) is an approach which shares some common elements with CCN. TRIAD defines a Content Layer, within which clients can request content using URLs as the format for content identification. The requester and the provider of the content interact solely in the context of these URL-based location-independent names and IP addresses are only used as transient routing tags. The content layer contains *content routers* which are responsible for forwarding requests to the *content servers* and *content caches* which store the content closer to potential requesters. Instead of the content, the client may finally retrieve a 'network pointer', i.e. a HTTP/TCP connection through which it can access the content. TRIAD has its own routing protocol called Name-Based Routing Protocol (NBRP). It is a flexible, extensible and scalable approach. One of its drawbacks is that it demands the establishment of a trust relationship between the content routers.

ROFL [55]: Although it does not focus on content distribution, ROFL (Routing Over Flat Labels) is a good example of an attempt to depart from the location-centric architecture of the Internet. ROFL suggests that location information is totally abolished from the network layer and routing is conducted solely on names. It proceeds to use flat labels with no semantic content as names. Correct routing is ensured with the use of a circular namespace.

RTFM [56]: RTFM (Rendezvous, Topology, Forwarding and physical Media architecture) is a publish/subscribe network architecture which transfers the control over the distribution of the content from the sender to the receiver. A data source publishes content and nodes-clients subscribe to the publication. Two identifiers, a public one and a private one, are assigned to each segment of data, with the private one being available only to the publisher. A directory service, analogous to DNS, can be queried to map the name of a segment of data to the corresponding identifier. These identifiers

are used as labels in the data packets and can be used by the routers to make forwarding decisions. Therefore, they are essentially an equivalent to the CCN names. Moreover, just like in CCN, the same packet can be forwarded to multiple destinations. RTFM's main weakness is that the identifiers that it uses are unstructured. On top of that, RTFM does not offer a cryptographic binding between the identifier and the content and thus does not yield many of CCN's security related benefits.

DONA [57]: In DONA, DNS names are replaced with flat names and a new name-based anycast primitive is introduced upon the IP layer. Each segment of data is said to be associated with a *principal* (mainly the publisher of the content). The names of the segments of data are of the form $P : L$, where P is the cryptographic hash of the principal's public key and L is a label chosen by the principal. DONA names are globally-unique, application-independent, location-independent, self-certifying and persistent. Resolution Handlers (the equivalent of DNS servers) resolve DONA names to locations. The DONA designers present a variety of extensions to the system which can be found in [57]: augmenting the resolution handlers to support caching, implementing a subscription system etc. Jacobson et al. [16] criticize DONA, mainly for the fact that it does not bind name and content for security and that it does not support requests for content which has not yet been generated.

2.4.3 General Remarks

Despite their differences, the basic principle behind all these proposals is the same. They all suggest a reconsideration of the fundamentals of the Internet. Locality is not seen as an appropriate criterion for making routing decisions anymore. If the new network model is designed so that it facilitates content exchange, then the ALTO problem could be resolved to a very large extent. Nevertheless, no matter how promising a novel clean-slate content-based architecture of the Network Layer might seem, it would be very difficult to persuade the Internet community to adopt it. It is undeniable that rebuilding the Internet is a very challenging and ambitious aim. It unavoidably demands the reconsideration of numerous protocols and services. Software will have to be revised and even hardware may have to be rebuilt. Although with CCN these changes would be minimized, they would still be significant. It is argued that there is still a long way to go until such an approach is employed.

2.5 Conclusion

This chapter provided a detailed presentation of the ALTO problem and all its potential manifestations. It also discussed the three categories of proposed solutions in detail and highlighted the benefits and drawbacks of all of them. It is apparent that the independent approaches which rely on measurements cannot guarantee optimal results. In addition, the redesign of the Network Layer seems to be very far-fetched at present. Thus, the approaches which suggest the cooperation between ISPs and P2P applications emerge as the most promising ones. This justifies the fact that they have greatly drawn the attention of the research community. Consequently, our work will also focus on the challenges of the Oracle approach and will try to investigate some of its unexplored pathways.

Chapter 3

Designing a P2P System with Oracle support

The scope of the project is to evaluate and enhance the performance of the Oracle solution which was described in the previous chapter. A distributed system using the BitTorrent protocol for file-sharing constitutes an excellent case base for this study. The ultimate goal is to create a model of this P2P system using a scalable packet level simulator and subsequently augment the model by adding Oracle support. The first section of this chapter describes the architecture of the distributed system, the entities it consists of and the features of the BitTorrent protocol which will be considered. The second section presents the design of the system after the introduction of the oracle(s) and discusses the alternative ranking criteria which can be adopted by such an oracle.

3.1 A P2P system running BitTorrent

The design of the system was based on the original BitTorrent specification which was published by Bram Cohen [58], as well as on BitTorrent Protocol Specification v1.0 [59]. The system consists of the BitTorrent clients and the tracker, whose responsibility is to help the clients peer with each other and exchange chunks of data. Following the terminology and the conventions outlined in [59], in the following paragraphs the term *Peer* will be used to refer to any host which runs the BitTorrent protocol and participates in the swarm. On the other hand, the term *Client* will be used to refer to the particular instance of the BitTorrent protocol which is in the center of attention, runs on the local machine and is connected to multiple other Peers. In other words, the readers may think of themselves as the Client which has to establish connections with other Peers in order to download data.

Firstly, the torrent initiator needs to set up a machine (the original seed) containing the complete file which is to be shared. The file is divided into a number of pieces. All pieces are of equal length, except for the last one. Obviously, the number of pieces is equal to $\lceil \frac{file\ size}{piece\ size} \rceil$. Additionally, the initiator is required to generate the metainfo torrent file and upload it on a web server. This auxiliary file contains information such as the name of the file which will be shared, the size of the file in bytes, the size of a piece in bytes, the SHA1 hash values of all the pieces of the file and the announce URL, i.e. the URL which will be used by the Client to send requests to the Tracker. The Client needs to download the torrent file in order to acquire this information. For the scope of our project, the generation of the torrent file, its contents and its retrieval by the Peers are not of importance. Thus, we can assume that the torrent file is distributed to the Peers in an out-of-band channel, following the approach of Katsaros et al. in [60]. The mechanism of this distribution will not need to be discussed in more detail.

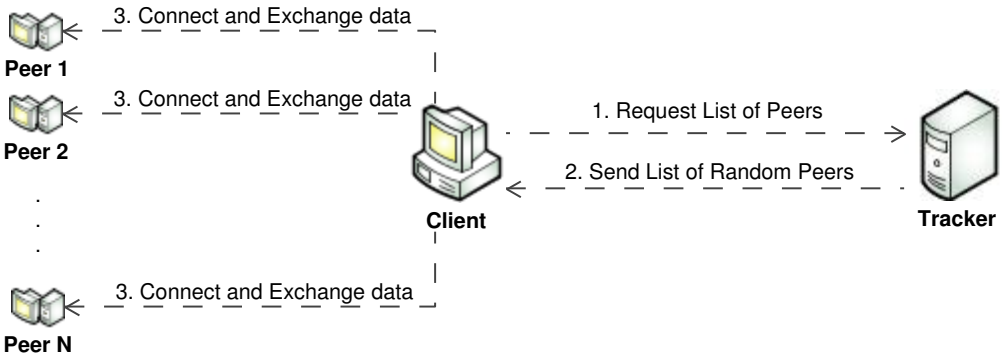


Figure 3.1: Architecture of the Distributed P2P system.

Figure 3.1 presents an overview of the interaction between the different entities of the system. This can be summarized as follows:

1. A Client requests a list of peers from the Tracker.
2. The Tracker responds by sending the list of peers back to the Client.
3. The Client initiates connections with Peers from the list and starts exchanging content with them.

The following subsections focus on the design of the two parts of the BitTorrent Protocol: The Tracker Protocol which defines the interaction between the Tracker and the Client and the Peer Wire Protocol which incorporates the heart of BitTorrent' s functionality and is concerned with the exchange of data and commands among the Peers.

3.1.1 Tracker HTTP/HTTPS Protocol

The Tracker Protocol is layered on top of HTTP/HTTPS and is concerned with the exchange of messages between the client and the tracker. When a client joins the swarm, it sends an HTTP request to the tracker using the GET method. The URL used for sending the request is the one that has been retrieved from the contents of the metainfo torrent file. Via the *Tracker Request* message, the client announces its entrance to the swarm and asks the tracker for a list of peers. It also lets the tracker know whether it is a seed (i.e. owns all the pieces of the file) or not.

A similar *Tracker Request* message is sent to the tracker in the following occasions:

- When the client completes the download of the file and becomes a seed.
- When the client exits the swarm gracefully i.e. by normally 'stopping' a torrent with the use of the corresponding command and not due to a sudden network failure or an unexpected shutdown of the client machine.
- Periodically, at regular intervals $T_{announce}$.

The value of the *event* parameter in the *Tracker Request* is used to notify the tracker of the type of the event that led to the transmission of the message. This allows the tracker to remove the client from the list of peers participating in the swarm if the event that generated the request was the departure of the client. The *Tracker Request* message also includes a large list of parameters, some of which are mandatory and some of which are optional. For instance, the client uses the request to provide the tracker with metrics which can be used in conjunction with the metrics sent by other clients to keep

track of the performance of the swarm. At this point, these parameters are not of importance to the system which is being designed and they do not need to be discussed further.

The tracker responds by sending a 'text/plain' *Tracker Response* message, which contains a list of D peers, where D is a default value. If the number of peers in the swarm is smaller than D, then the list is unavoidably shorter. The peers contained in the list are selected at random from the pool of all the participants in the swarm. However, the tracker is sophisticated enough to avoid naive choices such as recommending a seed to another seed. It is assumed that the *Tracker Response* messages are *compact*. This means that the list of peers has the form of a String consisting of multiples of 6 bytes. Each peer is represented by 6 bytes, 4 used for its IP address and 2 used for its port number.

3.1.2 Peer Wire Protocol

The term *Peer Wire Protocol* refers to the set of rules governing the exchange of chunks of data between the peers over TCP. The designed system will comprise all the key features of the Peer Wire Protocol. These are presented in the following paragraphs:

Initial Handshake

The interaction between two peers begins with an initial exchange of *Handshake* messages. A client initiating a connection sends a *Handshake* message to the remote peer. This is a required message and it is the first message sent by the client. Its length is $49 + pstrlen$ bytes, where $pstrlen$ is the length of the String identifier of the protocol. For the first version of BitTorrent, $pstrlen = 19$ and therefore the length of the *Handshake* is 68 bytes. The peer receiving the Handshake message replies to the initiator with another Handshake message. As soon as the initiator receives the response, the connection is considered established and the two peers can proceed to exchange messages of other types.

A client maintains a counter $numConnections$ of its currently established connections with other peers. Two thresholds are used to control the mechanism of the initiation and the acceptance of new connections: the minimum number of connections $minNumConnections$ and the maximum number of connections $maxNumConnections$. If $numConnections$ is smaller than $minNumConnections$, the client will attempt to actively open a new connection to a remote peer, provided that it is not already connected to all the peers contained in the list which was sent by the tracker. If $numConnections$ is equal to $maxNumConnections$, the client will reject connections initiated by remote peers.

Interested/NotInterested messages

For each one of its connections, a BitTorrent client maintains state information on whether it is interested in a piece that the remote peer owns, as well as on whether the remote peer is interested in a piece the local client has. It is of prime importance that this information is kept up-to-date, as it plays a crucial role in the Choking/Unchoking algorithm, which will be described below. Thus, as it is discussed in [59], the client has two state variables:

- *amInterested*: it is true iff the client is interested in a piece owned by the remote peer.
- *isInterested*: it is true iff the remote peer is interested in a piece owned by the client.

When the connection is initially established, both of these variables are set to false. Their values are kept up-to-date with the exchange of *Interested/NotInterested* messages, which are 1 byte long. The

client sends an *Interested* message to a peer whenever *amInterested* changes from false to true. This can happen on two occasions:

- when the client receives a *Bitfield* message from the remote peer, immediately after the successful completion of the Initial Handshake and the establishment of the connection
- when the client receives a *Have* message from the remote peer.

The client sends a *NotInterested* message to a peer whenever *amInterested* changes from true to false. This can happen when the client finishes downloading a piece and this piece was the only reason why the client was interested in that peer.

KeepAlive Messages

A client is allowed to close a connection if it has not received any messages of any type for a specified time interval T_{close}. This is due to the fact that the client interprets this silence as a problem in the connection or considers the remote peer to be unreachable. In order to keep the connection alive, the peer can send a *KeepAlive* message, if no other message of any type has been sent for a period of time $T_{keepalive}$ (with $T_{keepalive} < T_{close}$).

A connection is considered to be inactive if it has not received any non-KeepAlive messages for a certain period of time $T_{inactive}$ (with $T_{inactive} > T_{close}$). A client drops a connection as soon as it becomes inactive. This means that even *KeepAlive* messages cannot keep an idle connection alive for time longer than $T_{inactive}$. This feature is not in the original BitTorrent specification, but it is part of most modern BitTorrent clients such as Vuze [33] and uTorrent [61].

Bitfield Messages

Bitfield messages are exchanged between two connected parties only once, immediately after the completion of the Initial Handshake. They are the first messages to be exchanged after the establishment of the connection. Their purpose is to inform the remote peer about which pieces of the file the local client owns. The bitfield is practically an array of bits, each of which corresponds to a piece of the file being shared. The value of a bit is set to 1 iff the client sending the *Bitfield* message owns the corresponding piece, otherwise it is set to 0. A *Bitfield* message is $bflen + 1$ bytes long, where $bflen$ is the length of the bitfield in bytes and depends on the total number of pieces the file is divided into. A client does not send a *Bitfield* message if it owns no pieces of the file. Hence, if a peer does not receive a *Bitfield* message after the Initial Handshake, it may assume that the remote party possesses no pieces of the file.

Have Messages

As soon as a client finishes downloading a specific piece, it sends a *Have* message to each of the peers which it is connected to. The purpose of this message is to notify the peers that the client now owns this particular piece. In this way, the peers can update the stored bitfield of the corresponding client and reconsider their interest in it. A *Have* message is 5 bytes long and it contains an integer which represents the index of the piece.

There has been heated debate over whether *Have* messages should be sent to peers which already own a copy of the announced piece. It is argued that under normal circumstances, a peer will not re-download a piece that it already has and therefore there is no incentive in sending a *Have* message for

this piece to that peer. This design choice is often referred to as *Have Suppression* and it is estimated to reduce the number of exchanged *Have* messages by at least 50%, thus reducing the protocol overhead by 25-35% [59]. However, it is advocated that the exchange of *Have* messages even in such cases is useful, as it allows seeds to keep track of the piece availability among their peers and permits the deployment of rare-piece policies [62]. In order to simplify the design, as well as to allow for future extensions involving rare-piece policies exercised by the seeds, the Have suppression technique will not be included in the system.

Request/Piece Messages

As it was already explained, the file is divided into pieces. Each piece is further divided into chunks of data called blocks. The client can request a block from a remote peer by sending a *Request* message. A *Request* message is 13 bytes long. Its payload contains the index of the piece the requested block belongs to, as well as information used to define the particular block being requested. Obviously, a client requests blocks of pieces it does not have from peers that have them.

When a client receives a request for a block, it responds by sending a *Piece* message containing the corresponding block. The name of this type of message is misleading, as the message does not contain the whole piece, but just a portion of it. A client will not satisfy a request if it currently chokes (blocks) the requesting peer. It can either ignore the request or respond with a *Choke* message to ensure that the remote party is aware that it is being choked. The *Piece* message is $9 + blcklen$ bytes length, where $blcklen$ is the length of the block.

Piece/Block Downloading Strategy

Selecting the next piece/block to download among the candidate ones is an important element that can affect the performance of the Peer Wire protocol. A client must adopt the following three downloading strategies:

1. **Strict Priority**: As it is illustrated by Frioli et al. in [63], if a client has began downloading blocks of a specific piece, it should attempt to end up downloading all the blocks of this piece, before starting requesting blocks of other pieces.

2. **Rarest First Strategy**: According to this strategy, when deciding on the next piece to request, the client should select the piece which is the rarest one among the peers the client is connected to. In order to determine the rarity of the pieces, the client is required to maintain a copy of the bitfield of the peers and update it whenever it receives a *Have* message. Since having all the peers simultaneously go for the rarest piece would be inefficient, the strategy is enhanced by allowing a client to select randomly one of the R rarest pieces. The Rarest First Download Strategy facilitates the distribution of data in the swarm and speeds up the completion of downloads.

3. **Random Strategy**: As explained by Katsaros et al. [60], the Rarest First strategy is not efficient at the beginning of the download, when the client joins the swarm for the first time. At this point, it is important that the client acquires its first pieces as quickly as possible. If it chooses to download the rarest pieces first, the download will not be fast (due to the rarity of these pieces) and thus it will take more time for the client to become an uploader. To alleviate this problem, the first n pieces to be downloaded must be selected at random.

Choking - Unchoking

Our design of the Peer Wire protocol also incorporates the Choking/Unchoking algorithm. This algorithm is a very important feature of the protocol, firstly because it caps the number of concurrent uploads and secondly because it allows a client to discover better peers and reciprocate their good upload behavior. Choking a remote peer means not fulfilling any of its requests and not uploading any data to it. All *Request* messages received by a choked peer are ignored. As it is stated in the unofficial BitTorrent Protocol Specification [59], the client should maintain two additional state variables per connection:

- *amChoked*: it is true iff the client is choked by the remote peer.
- *isChoked*: it is true iff the client has choked the remote peer.

With the establishment of a connection, peers begin as choked. As a result, the aforementioned variables are initially set to false and they are kept up-to-date via the exchange of *Choke* and *Unchoke* messages. The choking/unchoking mechanism abides by the following rules:

- The choked/unchoked state of a peer is reconsidered periodically every T_{choke} seconds. T_{choke} must be large enough to prevent fibrillation i.e. too frequent choking and unchoking of the same peer. Every T_{choke} seconds the peers are sorted according to the rate with which they upload data to the client. The U peers which have the highest rate and are interested become unchoked and the rest of the peers become choked. By unchoking the peers which uploaded the most data, the client reciprocates and behaves in a tit-for-tat fashion. If the client is a seed, then the peers are instead sorted according to the rate with which they downloaded data from the client. This again guarantees that the seeds contribute the most to the data exchange in the swarm. The integer U is often referred to as the number of upload slots, because it gives the maximum number of peers a client can upload data to at the same time. In the original specification, U was defined to be equal to 4, however modern BitTorrent clients determine the number of upload slots based on the upload bandwidth of the host.
- Unchoking a previously choked peer means sending it an *Unchoke* message. Choking a previously unchoked peer means sending it a *Choke* message. Both of these messages are 1 byte long and carry no payload. A peer receiving an *Unchoke* message can start sending requests for blocks to the client. A peer receiving a *Choke* message must consider that all pending requests which were sent to the client will not be answered. Moreover, it must refrain from sending any new requests to the client (until it is unchoked).
- Every $3T_{choke}$ seconds, a random peer is unchoked, regardless of its upload rate. This mechanism is referred to as optimistic unchoking. Peers whose connections have been established more recently are preferred by the optimistic unchoking strategy. The benefit from this feature of the Peer Wire protocol is twofold. Firstly, it allows the peers which have just joined the swarm for the first time to acquire their first pieces. Secondly, it aids the client to discover potentially better peers.

Request Pipelining

A BitTorrent client is designed to keep a number of Q unfulfilled requests queued up on each connection. This practice is called Request Queuing or Request Pipelining. When a client is unchoked by one of its peers, it sends Q block requests. When a *Piece* message arrives and fulfils one of the requests, a new request is generated, so that the total number of unfulfilled requests remains equal to Q (providing that the client is still interested in this peer). The purpose of this mechanism is to boost up TCP

performance. If Request Pipelining were not used, then the client would not be receiving any data from a peer during the time period between the beginning of the transmission of a new *Request* message and the beginning of the reception of the corresponding *Piece* message. This time period is equal to one Round Trip Time and it is obvious that wasting it would result in serious performance degradation. As a side note, Cohen [58] underlined the fact that requests which can't be written out to the TCP buffer immediately must be kept in memory, so that they can all be discarded should the client be choked by the remote peer.

Features Not Included

Two of the elements of the Peer Wire protocol were decided not to be included in the design:

- Super Seeding: Super Seeding is an algorithm employed primarily by the initial seeds and aims at helping them reduce the amount of data they need to upload in order to give birth to new seeds in the swarm. Super Seeding mode was not part of the original specification but was introduced much later. Moreover, it is only recommended for the torrent initiating seeds, i.e. a very small fraction of the swarm. As it does not constitute a key element of the protocol, it is safe to ignore it.
- End Game Mode: End Game mode is a strategy employed by a client when its download is close to completion. In some cases, the last one or two requested blocks are downloaded at low download rates. As a remedy, the client sends requests for these blocks to all its peers. When a copy of a block is finally downloaded successfully, *Cancel* messages are sent to the peers to abort the reception of multiple replicas of it. The inclusion of the End Game Mode to the designed protocol would introduce significant complexity, as it would greatly multiply the number of exchanged messages (*Request, Piece, Cancel* ones). At the same time, the effect of the presence or the absence of this feature in the evaluation of the Oracle solution was estimated to be negligible. So, it was finally omitted.

3.2 Introducing Oracle Support

The system described in the previous section is bound to suffer from the ALTO problem. Hence, it is augmented by applying the Oracle solution described in Chapter 2. Again, the term *Oracle* will be used to refer to the entity providing the ALTO service.

Figure 3.2 illustrates the architecture of the enhanced system, which is inspired by the works of Aggarwal et al. [15] and Xie et al. [9], as well as the Internet Draft published by the ALTO Working Group [11]. The use case proceeds as follows:

1. A Client requests a list of peers from the Tracker.
2. The Tracker contacts the Oracle which serves the AS of the Client and asks for a ranking of the peers participating in the swarm. The ranking must be performed on the basis of the IP address of the requesting Client.
3. The Oracle ranks the peers and sends the ranked list back to the Tracker. The ranking may be performed based on data which can be retrieved from a semi-static database.
4. The Tracker selects the D best (higher-ranking) peers and sends them to the Client.
5. The Client initiates connections to Peers from the list sent by the Tracker and starts exchanging content with them.

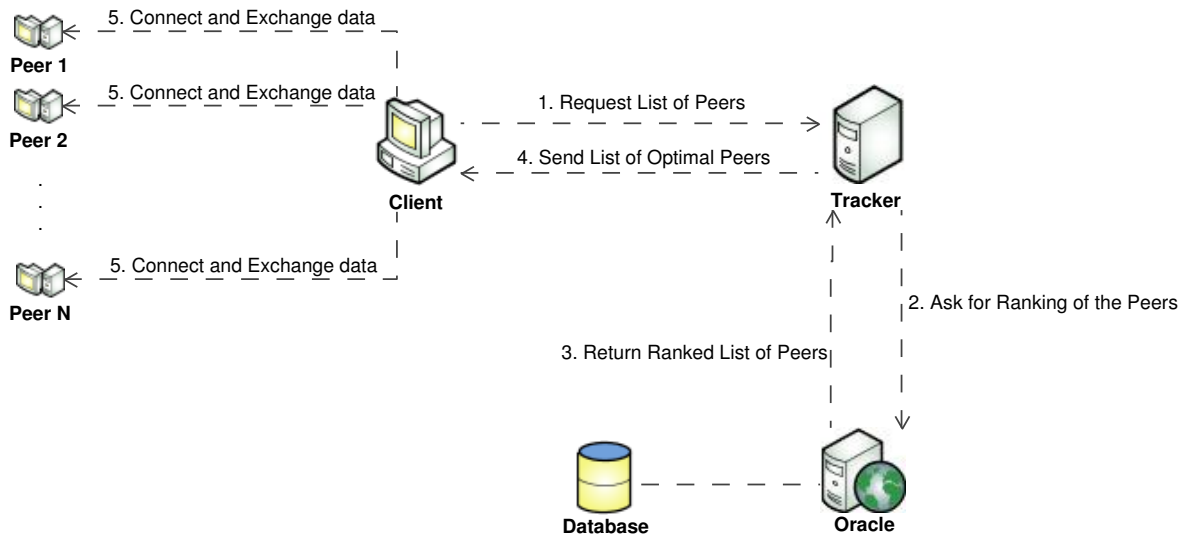


Figure 3.2: Architecture of the P2P system with Oracle Support.

The rules governing the exchange of messages between the Tracker and the Oracle constitute a separate protocol. As it was described in Chapter 2, alternative versions of this protocol have been proposed. Two examples are the *Oracle Protocol* defined by Akonjang et al. [36] and the *ALTO Protocol* proposed by Alimi and the rest of the ALTO Working Group [11]. This project mainly follows the design of Aggarwal et al.; however, as the basic principles behind both protocols are the same, and since only the key features of Akonjang's specification are retained, the design described in this section can cover both approaches.

The format of the messages exchanged between the Tracker and the Oracle is a simplified version of the format described in the Oracle Protocol specification of Akonjang et al. [36]. The messages are of two types: the *Query* messages and the *Response* messages. A *Query* message is sent from the Tracker to the Oracle and contains the list of peers to be ranked. The *Response* message is sent from the Oracle to the Tracker as a reply to a *Query* message. It contains an ordered list of peers, which results from the appropriate ranking of the peers contained in the corresponding *Query* message.

Messages of both types consist of a header section and a payload section. The header section has a fixed length of 16 bytes. 12 of these bytes are used for the various fields enumerated in the specification. Examples of these fields are:

- The Message Length field, indicating the total size of the message in bytes.
- An Identification field used to correlate queries and responses.
- A set of Flags used to communicate different kinds of information.
- The number of IP addresses included in the message.

Examining these fields in more detail is out of the scope of the project. However, one thing worth noting is that the scenario under study demands the incorporation of an extra field which was not part of the specification of Akonjang. This is a field used to indicate the IP address of the Client. The Tracker needs to communicate the IP address of the Client to the Oracle, since the ranking performed by the latter must be tailored to the case of this specific Client. The original specification did not include a field of this type, as it was designed for the particular case in which the Oracle (ALTO) Client (i.e. the party which queries the Oracle) is also the P2P Client. In that case, the Oracle would

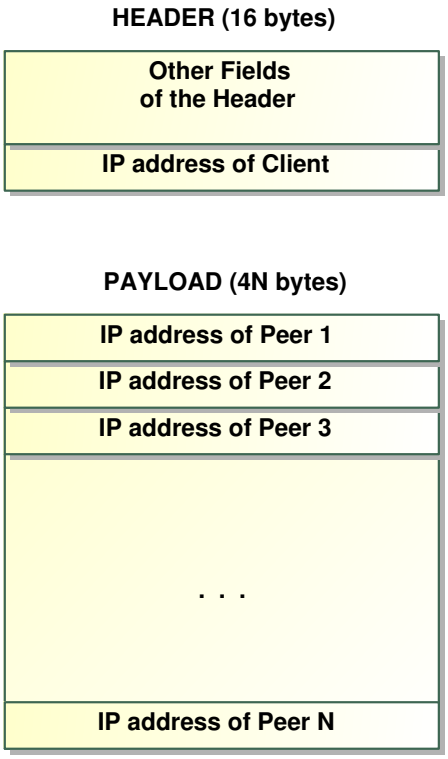


Figure 3.3: Format of the Messages to/from the Oracle.

just have to rank on the basis of the source IP of the received requests. However, in our case, the Oracle (ALTO) Client (the Tracker) is different from the application Client, and the ranking has to be conducted with respect to the IP of the latter. Introducing this extra field to the header of the Oracle messages means an increase of 4 bytes in the size of the header (assuming that the designed system uses IPv4 addresses).

The payload of the *Query* message consists only of IP addresses. These are the IPv4 addresses of the peers in a random order. Obviously, the length of the payload of the message is a multiple of 4 bytes. The payload of the corresponding *Response* message contains the same list of peers, but the list is now ordered according to the ranking strategy employed by the Oracle. The specification introduces two additional fields per IP Address: the Option Field and the Value field. The designers stated that these fields could be optionally used in the *Response* message for the potential transmission of additional information, especially in a future extension of the protocol which would support the interaction between oracles. The use of these fields is out of the scope of this project, so we we can assume that they are not included in the payload of the *Response* message.

In order to apply its ranking strategy, the Oracle may consult a *Database* containing static as well as dynamic data. For example, it may store information on the topology (whose nature is more static) as well as measurements of the utilization of the links of the AS (more dynamic). The dynamic records of the Database need to be kept up-to-date, so that they always reflect the current state of the network.

A design choice that had to be made was the actions of the tracker in the case in which the number of candidate peers N is smaller than the number D of peers included in the *Tracker Response* message. The issue that arises is whether the tracker should consult the oracle or not in this case. Two alternative approaches exist:

1. The tracker should not consult the oracle. No matter if the tracker has the list of peers sorted or

not, all N peers will ultimately be included in the *Tracker Response*. Skipping this contact will relieve both the tracker and the oracle from the corresponding communication overhead and it will also allow the client to receive the response sooner.

2. The tracker should still consult the oracle. Having the peers ranked in order of preference is always beneficial for the clients. This is due to the fact that a client attempts to connect to the peers sequentially in the order in which they are included in the *Tracker Response*. Consulting the oracle ensures that better peers appear first in the response and the Client connects to them as quickly as possible.

As both of these alternatives present merits and demerits, the Tracker is designed to support both of them.

A second crucial design choice was concerned with the Transport Layer Protocol which would be utilized in the communication between the Tracker and the Oracle. Again, at it was also illustrated in the previous chapter, different approaches can be followed:

1. Akonjang et al. decided to use UDP instead of TCP. Their choice was driven by a number of reasons. According to them, UDP is faster and imposes less overhead on the operating system. Thanks to its stateless nature, it is suitable for servers like the Oracle which accept requests of small size from a large number of clients. It is also appropriate for applications that do not demand guaranteed delivery of the messages and prefer to have packets discarded rather than delayed. Akonjang et al. perceived the Oracle as an application of this type and therefore preferred to deploy it on top of UDP.

2. Kiesel et al. [12] in their ALTO Requirements Internet Draft explicitly stated that ˜the ALTO client protocol MUST use TCP based transport″. In addition, Alimi et al. [11] also selected TCP as the Transport Layer Protocol. This was a one-way choice, as their ALTO Protocol is built on top of HTTP. HTTP was regarded as a natural design choice for a number of reasons. Firstly, it permits the use of existing infrastructure such as HTTP caches. Secondly, the communication can benefit from the existing authentication and encryption mechanisms of HTTP. Finally, the applications requesting the Oracle service usually have an embedded HTTP client which can be reused.

The system was designed to support delivery both over TCP and over UDP. This gives it flexibility and will allow for the evaluation of the two alternatives.

3.2.1 Ranking Strategies

As it was discussed in Chapter 2, the Oracle can employ numerous different ranking criteria. Each one of the Oracle-like approaches that have been published has come with a list of proposed ranking strategies. However, the experiments which have been conducted so far have focused their attention on only a couple of these strategies. In this work, the Oracles are made to employ a large repertoire of ranking criteria, both efficient and unfruitful ones. This will ultimately allow for a more careful evaluation and comparison between them.

Oracles are designed to support the following ranking criteria:

- **'Simple' Strategy**: This strategy only considers whether the candidate peer is located within the local AS or not. Peers within the oracle's (and obviously the client's) AS are preferred over peers outside the AS. This is the simplest and less sophisticated ranking strategy which can be adopted.

- **'AS Distance' Strategy**: This ranking criterion makes use of the AS-hop distance between the client and the candidate peer. The larger the distance, the lower the peer is ranked. This strategy is an extension to the previous one. Instead of treating all the peers outside the AS in the same manner, preference is granted to those situated within the most local ASs. The 'AS-hop distance' metric can be obtained from the BGP protocol running on the AS Border routers.
- **'Charging-related' Strategy**: This criterion is typically determined by the relationship between the oracle's AS and the peer's AS. Peers in customer ASs are preferred over peers in peer ASs and peers in peer ASs are preferred over peers in provider ASs. This allows the ISP to maximize its revenue and minimize the cost incurred by the traffic directed towards its providers. However, the application of such policies must be handled with great care. Their abuse may render an oracle malicious and may ultimately harm the performance of the swarm as well as the well-being of the whole network.

For further discriminating among the peers located within its AS, an oracle is designed to employ the following ranking strategies:

- **'Num of IP Hops' Strategy**: The ranking decisions are determined by the number of IP hops in the routes connecting the client with the candidate peers. The larger the number of IP hops in a route, the lower the corresponding peer is ranked. This is a pure locality-based criterion. ISPs have knowledge of the topology of their networks and can easily determine the number of IP hops. For instance, the information can be retrieved from the OSPF protocol running on the routers, provided that the cost of all links is equal to 1.
- **'Path Cost' Strategy**: This criterion considers the cost of the route connecting the client with the candidate peer, with the cost of each link being a function of its capacity. More precisely, the cost of a link is calculated using the formula $Cost = \lceil \frac{A}{link\ capacity} \rceil$, where A is a constant. Then, the total cost of the route is computed by summing the individual costs of the links. The larger the total cost, the less the corresponding peer is preferred. Modern Cisco routers which run OSPF protocol use the aforementioned methodology to determine the cost of the paths [64], with A being equal to 100,000,000 bps and the link capacity measured in bps. Therefore, we may assume that the data for the application of this criterion can be retrieved directly from the routers, provided that they exercise the OSPF protocol following the rule described above.
- **'Access Link - Centric' Strategies**: This group of ranking policies are based on the capacity of the access link of the candidate peer. Two alternative approaches can be followed. The first and more general one is to set high preference to peers with faster access links. The higher the capacity of the link, the higher a peer will be ranked. We will refer to this approach as the 'Fastest Access Links' ranking strategy. The second approach is formulated to specifically fit the requirements of the BitTorrent protocol. This approach prefers peers whose access links are of the same capacity as the link of the requesting Client. This will ultimately allow the tracker to match clients with similar capabilities. Fast clients will peer with other fast clients, while low capacity clients will finally connect with other low capacity clients. In a sense, this latter approach attempts to mimic the tit-for-tat algorithm of BitTorrent. It is tailored to this protocol's characteristics and should not be exercised globally. We will refer to this approach as the 'Matching Access Links' ranking strategy.
- **'Traffic Engineering' Strategy**: This technique relies on the values of the utilization of the links in order to make ranking decisions. It allows the Oracle to traffic engineer its network, manage the distribution of the load and minimize congestion. Decisions should be based on measurements of the utilization, which must be conducted periodically. These measurements can be collected from the routers of the network and can be stored in the Oracle Database. The time interval between two successive updates of the utilization records of the Database is

a very crucial parameter of the strategy. Failing to update frequently enough may result in uninformed ranking decisions based on outdated information. On the other hand, too frequent updates may not be a good practice either. Firstly, they will place a significant burden on the system. Secondly, it must be ensured that a change in the values is not due to occasional random fluctuations but it reflects an actual alteration in the network conditions.

An Oracle in the designed system can exploit this information to evaluate a candidate peer. Firstly, the Oracle retrieves the utilization values of all the links in the path connecting the Client to the candidate peer. These values are then used to assess the quality of the path. This can be done either on the basis of the average utilization of all the links or more appropriately by considering the maximum utilization. The larger the maximum utilization value, the less preferred this path is and the lower the corresponding peer is ranked.

Using the Oracle solution to avoid congestion is a practice that has not been praised by all the research teams who have worked on this area. Aggarwal et al. [15] and Xie et al. [9] are in favor of it. On the other hand, Kiesel and the rest of the IETF ALTO Working Group [12] state that using ALTO for congestion control is inappropriate and criticize the ranking based on metrics related to the instanteous congestion status of the links. Furthermore, they point out that congestion control should best be exercised via more traditional techniques such as TCP based transport.

Of course, some of the above criteria can be combined in order to form more complex ranking algorithms. This may result in the development of more efficient composite strategies which retain the merits of their components and suppress their weaknesses. For instance, 'Num of IP Hops' is a strong locality-based strategy but it does not consider any performance-related metrics such as the bandwidth of the links. On the other hand, the 'Access Link - Centric' strategies are good performance-related criteria but they are completely oblivious of the network topology. The performance of the system could be enhanced if these two approaches are blended to form a composite one, which prioritizes peers that are both local and with good access links.

Algorithm 1 provides a simple but effective way in which these two strategies can be combined. We assume that the first general variation of the 'Access Link - Centric' strategy is applied i.e. preference is given to peers with the fastest access links. The algorithm compares two peers on the basis of this new composite criterion and returns the one that is better (higher ranking). Initially, the number of IP hops and the capacity of the access link is determined for both peers. If one of the peers outmatches the other according to both sub-criteria (i.e. it has a faster access link and it is over a path with fewer hops), then this peer is the winner. If $peer1$ beats $peer2$ on the basis of the first sub-criterion but $peer2$ beats $peer1$ according to the second sub-criterion, then the decision depends on the extent to which the one peer surpassed the other. More precisely, let us consider that according to the first criterion $peer1$ was better that $peer2$ by a% and according to the second criterion $peer2$ was better than $peer1$ by b%. $peer1$ outmatches $peer2$ if $a \geq b$; otherwise, $peer2$ outmatches $peer1$.

Concluding, one should consider that the Oracle may employ a bad unfruitful ranking strategy. Thus, the Oracle solution may result in not being a win-win strategy and may damage the performance of the swarm and the general health of the network. Our system is designed so that an Oracle may employ one of the following malicious ranking strategies:

- **Malicious Strategy 1**: Always prioritize peers belonging to customer ASs. As it was already discussed, an Oracle can abuse the 'Charging-related' strategy to promote the financial incentives of the corresponding ISP. In the worst case, the Oracle may attempt to push as much traffic as it can towards the customer ASs. A totally obnoxious ranking strategy is one that prefers peers in the customer ASs even over peers within the Oracle's local AS. So, ultimately the list of peers

Algorithm 1 Ranking Strategy combining 'Num of IP Hops' and 'Fastest Access Links' criteria

```
Retrieve the number of IP hops for peer1: hops1
Retrieve the number of IP hops for peer2: hops2
Retrieve the bandwidth of the access link for peer1: bandwidth1
Retrieve the bandwidth of the access link for peer1: bandwidth2

if hops1 ≤ hops2 and bandwidth1 ≥ bandwidth2 then
  return peer1
end if
if hops1 ≥ hops2 and bandwidth1 ≤ bandwidth2 then
  return peer2
end if

if hops1 < hops2 and bandwidth1 < bandwidth2 then
  difHops ← (hops2 − hops1)/hops2
  difBW ← (bandwidth2 − bandwidth1)/bandwidth2
  if difHops ≥ difBW then
    return peer1
  else
    return peer2
  end if
end if

if hops1 > hops2 and bandwidth1 > bandwidth2 then
  difHops ← (hops1 − hops2)/hops1
  difBW ← (bandwidth1 − bandwidth2)/bandwidth1
  if difHops < difBW then
    return peer1
  else
    return peer2
  end if
end if
```

will be ordered according to the following pattern:

$$[peers\ in\ customer\ ASs,\ peers\ in\ the\ local\ AS,\ peers\ in\ peer\ ASs,\ peers\ in\ provider\ ASs]$$

with the higher-ranking peers appearing first. Of course, the malicious ISP may benefit from the adoption of such a strategy. However, as it will be demonstrated, this scheme can have adverse effects both on the performance of the P2P application and on the well-being of the entire network. This phenomenon has been discussed by Kaya et al. [65] and Piatek et al. [66]. The former built a mathematical model to quantify the risks stemming from the adoption of a '*No valley and prefer customer policy*'. The latter illustrated the same problem by extracting evidence from a large scale trace measurement of BitTorrent usage. However, the current bibliography lacks a more detailed study based on experiments conducted on a system which comprises both the P2P application and the oracle(s).

- **Malicious Strategy 2**: Always prioritize certain peers in order to generate an attack on them. The Oracle may select to reply to any query of the Tracker by always placing the same set of N peers on top of the rank [67]. This obnoxious strategy will lead a large number of BitTorrent

clients to try to connect to these N particular peers and this will finally cause a Denial of Service (DoS) attack against them. In addition, this behavior will result in the degradation of the download performance of the clients that get the unfruitful malicious recommendations.

- **Malicious Strategy 3**: Deliberately prioritize bad peers. The Oracle may deliberately provide bad recommendations (for example by assigning a high rank to peers in distant ASs). An Oracle may choose to follow this behavior when it wants to penalize a Tracker or discriminate against it and in favor of other competitor Trackers. In other words, the Oracle service can prove to be a tool in the hands of the ISPs for expressing preferences and favoring some Trackers over others.

Chapter 4

Topology

The performance of the distributed system which was described in the previous chapter will be evaluated by conducting simulations on a suitable network topology. The selection of an appropriate topology was a task of prime importance, as it could fundamentally influence the quality of the results. This chapter is dedicated to the detailed presentation of this topology and aims at clarifying the design choices that were made. It begins by enumerating the requirements that the simulated network had to fulfill and proceeds to illustrate the network map at inter-AS as well as at intra-AS level.

4.1 Requirements

As Floyd et al. discussed in [68], simulating the Internet is a very challenging task due to its scale and heterogeneity. A simulated topology should be a subset of the Internet which manages to capture as much of its diversity and distinct characteristics as possible. In a sense, it should be seen as a 'miniature' of the whole Internet. More precisely, for the scope of the project, the network topology had to encompass the following features:

- It had to consist of more than one ASs. The number of ASs had to be large enough to allow for the evaluation of the performance of the system when different oracles in different ASs employ different ranking strategies. However, this number had to be capped by the scalability capabilities of the simulator and the available computational resources.
- The ASs should be organized in a multi-tier hierarchy. This hierarchy had to exhibit multiple instances of the two possible AS relationships: customer-provider and peer-to-peer.
- The topology within each AS should be elaborate enough to allow for the comparison and the evaluation of the different ranking strategies which were enumerated in the previous chapter. This means that the topology should consist of at least six routers which should be connected with numerous links of different capacities. The design should be such that the links exhibit a variety of different utilization values. This means that there should be congestion hotspots, as well as links of lower utilization. Again, the intra-AS topology should not be too complex to permit the execution of the simulation runs within reasonable time frames. Aggarwal et al. conducted their simulations on ASs with a much simpler star intra-AS topology [18]. Although this would simplify the modeling, it would not provide a clear representation of the image of the Internet and it would not be able to illustrate the differences between the ranking strategies under test.

4.2 Inter-AS topology

After considerable research, the Greek Internet was found to constitute an appropriate test case, as it presents a variety of interesting features. Ultimately, a subset of the AS topology of Greece was selected to be modeled. This subset manages to fulfill all the aforementioned requirements.

In order to construct a complete and accurate map of the Greek AS topology, one has to collect information from a number of different sources. This section has been based on data retrieved from:

- the Cooperative Association for Internet Data Analysis (CAIDA) [69]. The web site of CAIDA provided information on the relationships between ASs, as well as the correspondence between the names and the Autonomous System Numbers (ASNs) of the ASs.
- Robtex, the Swiss Army Knife Internet Tool [70]. Robtex complemented CAIDA's data on the relationships between ASs.
- ADSLgr[71], a Greek independent online community whose web site contains news, articles, interviews, data and results of experiments related to the Greek broadband.
- The web sites of Greek ISPs [72, 73, 74].
- Ping/Traceroute experiments conducted on hosts served by multiple Greek ISPs.

Subsection 4.2.1 provides a presentation of the characteristics of the Greek Internet that were collected from all the previous sources. Subsection 4.2.2 describes the subset of the Greek AS topology which was finally chosen to be simulated.

4.2.1 The Greek Internet

In their majority, the large Greek ISPs are Tier-2 ASs. They purchase IP transit from international Tier-1 ISPs, such as Level 3, GBLX and Tiscali, as well as from large foreign Tier-2 ISPs, such as Seabone.

A large number of peering relationships between these Greek ISPs has been made possible with the help of GR-IX [75]. GR-IX is an Internet Exchange Point (IXP), whose purpose is to facilitate the interconnection between the Greek providers. It was established in 2009 by the Greek Research and Technology Network (GR-Net), in cooperation with the ISPs and the National Hellenic Research Foundation (NHRF), and it is located in Athens. It is managed by GR-Net, which acts as a trusted third party. Thanks to GR-IX, the Greek ISPs can establish peerings and achieve direct exchange of IP traffic, without the intervention of foreign Tier-1 and Tier-2 ISPs. Currently 12 ISPs participate in the GR-IX project. However, not all of them peer with each other via GR-IX. The web page of GR-IX [75] provides a peering matrix which presents the peering relationships between the participant ISPs. For example, as of September 2010, OteNet peers with 9 of the 11 other ISPs via GR-IX, while Forthnet peers with only 2 of them.

Certain pairs of ISPs that do not peer via GR-IX have agreed on private peerings, subject to undisclosed terms and conditions. Some of these peerings have not been officially announced by the corresponding parties yet, but traceroute experiments can prove their existence (for example, the private peering between Forthnet and Hellas On Line).

Finally, some of the Greek ISPs do not peer at all and exchange traffic via their providers i.e. foreign Tier-2 and Tier-1 ISPs (for example Forthnet and CYTA). This often proves to be problematic, as the communication between two Greek hosts is conducted via routes that cross the borders of the country and thus suffers from severe latency.

In their turn, the tier-2 ISPs act as providers for a large number of tier-3 Greek ASs. This ASs are generally run by smaller providers and organizations such as universities, banks, hospitals, television channels, large enterprises etc.

4.2.2 Final Inter-AS Topology Map

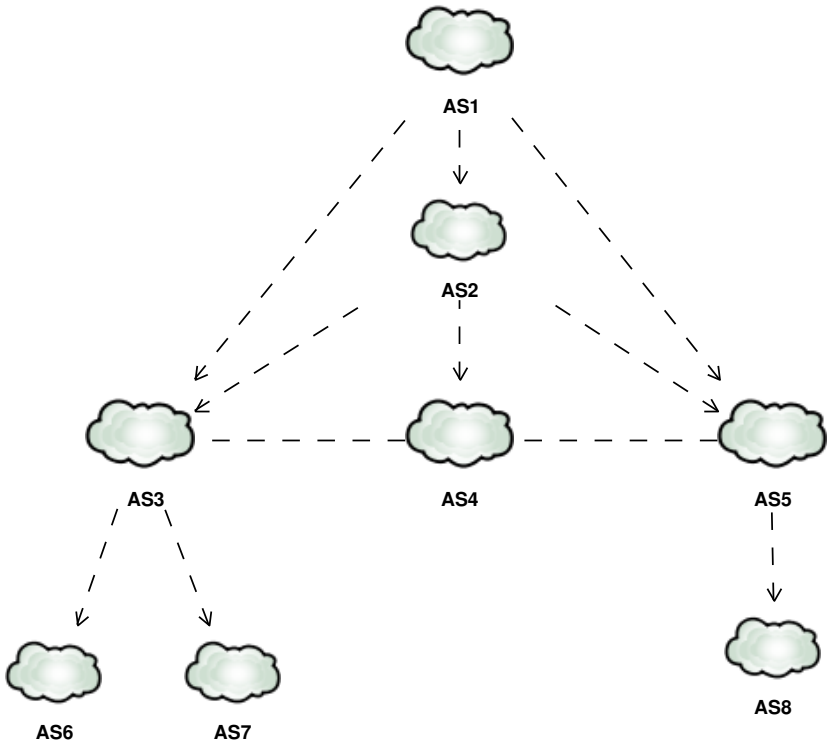


Figure 4.1: Inter-AS Topology

Figure 4.1 depicts the inter-AS topology which was finally selected to be modeled. This is a subset of the Greek Internet map which was described in the previous section. The drawing has the form of a mixed graph i.e. a graph in which some edges may be directed and some may be undirected. A directed edge is used to visualize a provider-customer relationship, with the arrow of the edge pointing from a provider AS to a customer AS. Undirected edges are used to represent peering relationships between the ASs.

As shown in the figure, AS1 is an international Tier-1 ISP. AS2 is a large non-Greek Tier-2 ISP, which buys IP transit from AS1. AS3, AS4 and AS5 are large Greek Tier-2 ISPs. AS3 and AS5 are costumers of AS1 and AS2. AS4 buys IP transit from AS2 only. In addition, AS4 peers with AS3 and AS5. AS6, AS7 and AS8 are smaller Greek Tier-3 ASs. The following example is an instantiation of this topology, in the context of the Greek Internet.

Example

- AS1 is Level 3 Communications (with ASN 1239) [76]. Level 3 operates one of the largest IP transit networks in North America and Europe.
- AS2 is Telecom Italia Sparkle Seabone (with ASN 6762) [77]. Seabone is a large Italian ISP. It buys IP transit from Level 3 (as well as from other tier-1 ISPs such as Sprint and GBLX)

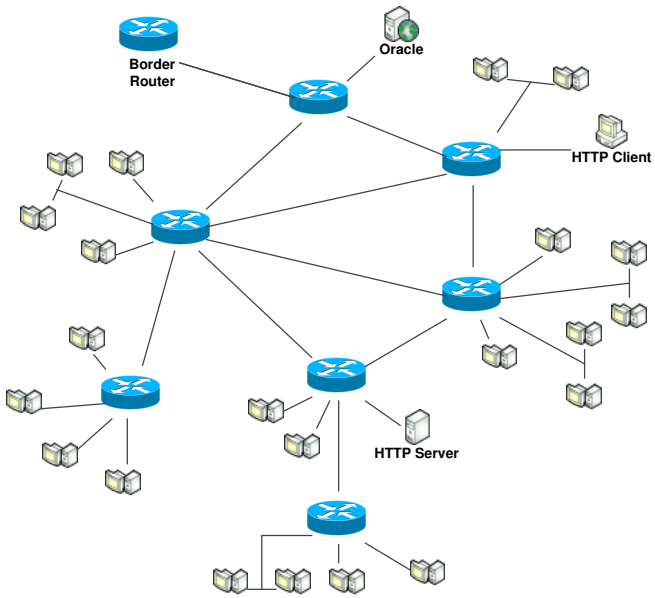


Figure 4.2: Intra-AS Topology 1

- AS3 is Forthnet (with ASN 1241) [73]. Forthnet is one of the largest Greek Tier-2 ISPs. It buys IP transit both from Seabone and directly from Level 3.
- AS4 is On Telecoms (with ASN 41920) [72]. On Telecoms is another large Greek Tier-2 ISP. It is a customer of Seabone and it peers with Forthnet. This peering is not established via GR-IX [75], but it is a private one. Traceroute experiments were conducted to prove its existence.
- AS5 is CYTA (with ASN 6866) [74]. CYTA is a Greek Tier-2 ISP, but it is managed by the Cyprus Telecommunications Authority. It is a customer of Level 3 and Seabone. It peers with On Telecoms via GR-IX [75]. It does not peer with Forthnet. Hosts in the CYTA network communicate with hosts in the Forthnet network either via Seabone or via Level 3. Traceroute experiments were carried out and they indicated that the communication is primarily conducted via Seabone and the route via Level 3 is used as a back-up.
- AS6 is islet number 3 of Syzefxis (with ASN 8951) [78]. Syzefxis provides Internet access to the organizations of the Greek Public Sector (such as ministries, hospitals, town halls, military camps etc). The network of Syzefxis is divided into 6 islets and each islet may be a customer of a different Greek Tier-2 ISP.
- AS7 is Space Hellas (with ASN 8499) [79]. Space is a service provider in the areas of telecommunications, IT and Security for the enterprise, government and military sectors.
- AS8 is Lemontel (with ASN 35660) [80]. Lemontel is a Greek-Cypriot company which provides telephony and Internet services.

4.3 Intra-AS topology

Figures 4.2 and 4.3 depict the two topologies that were used for the interior of the ASs. Topology 1 was used for AS1, AS2, AS6, AS7 and AS8 and topology 2 was used for AS3, AS4 and AS5. Both topologies consist of 8 routers forming a partially connected mesh. Each of the routers runs OSPF

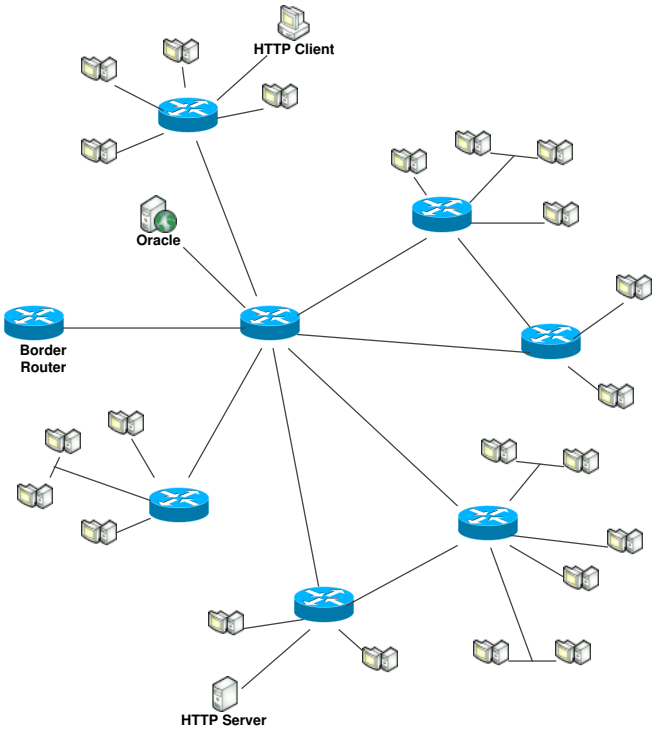


Figure 4.3: Intra-AS Topology 2

as the Interior Gateway Protocol (IGP) and the border router also runs BGP for the interconnection with the other ASs.

The figures show a small subset of the BitTorrent clients for demonstration purposes. The whole network of the 8 ASs consists of 600 BitTorrent clients. The distribution of the clients within the ASs is inspired by the patterns followed in [18] and is in accordance with the number of subscribers in each AS (again the sites of the ISPs provided information on this).

Apart from the BitTorrent clients, the topology also contains the oracle(s) of the AS, as well as a number of HTTP clients and servers, used for the simulation of the background traffic. In addition, the trackers managing the swarms can be arbitrarily placed in any of the ASs. For the experiments involving only one swarm, the tracker was selected to be situated within AS3. In total, the simulated network consists of over 700 hosts running the IP protocol (including routers, BitTorrent clients, oracles, trackers and HTTP clients and servers).

The second topology, used for the three large Greek Tier-2 ISPs, is based on the network maps of these ISPs, as it can be retrieved from their web pages. Therefore, it is in accordance with the geography of the country and can fit the Greek map. Four of the eight routers are located in Attica, i.e. in the capital, Athens, and its surrounding area, where approximately 36% of the population resides [81]. The rest of the routers are placed in Points of Presence in other large Greek cities (such as Patra and Thessaloniki).

The access links of the BitTorrent clients are Asymmetric DSL links, with download bandwidth ranging from 1Mbps to 20Mbps. Once again, the web pages of the ISPs were used as a guide to determining the distribution of the capacity of the access links. As a result, the international top-tier ASs were made to have a larger portion of higher bandwidth clients than the lower-tier ASs.

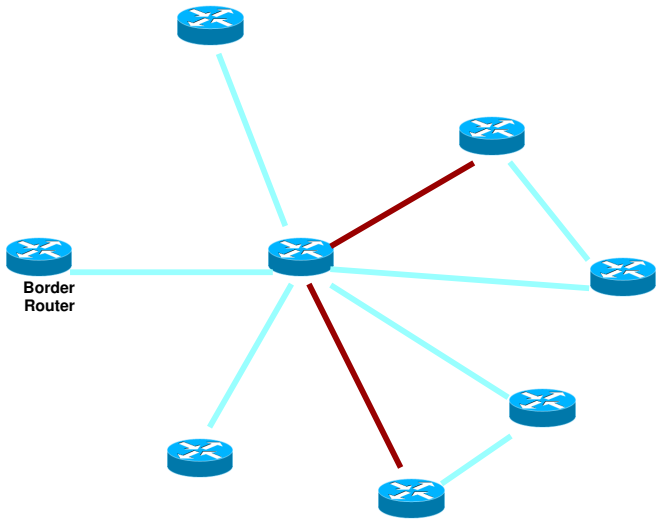


Figure 4.4: Links that tend to be highly congested in Greek tier-2 ASs. The red (dark) colour is used to mark these links.

4.4 General Remarks

1. The value of the capacity of the links was chosen on the basis of the information retrieved from the sites of the ISPs and from the portal of ADSLgr. However, only a small portion of the background traffic was simulated. Besides, in the cases of the larger ASs, even the number of BitTorrent clients had to be scaled down. This means that the number of hosts operating in the network is a small subset of the whole set of Internet users and as a result, if the capacities of the links were given their true value, the links would ultimately appear to be abnormally under-utilized. Hence, the capacity of all the links (apart from the last-hop ADSL ones) was appropriately scaled down so that it is in correspondence with the number of simulated hosts. The values of utilization were thus kept at reasonable levels, in accordance with the results of related studies. More precisely, a large portion of the links was made to exhibit utilization values of less than 50%. This complies with the work of Akella et al. [82], as well as with the observations derived from the traffic measurements conducted on the Sprint IP Backbone by Fraleigh et al. [83]. In addition, a small number of links was made to be congestion hot-spots carrying 'high load', with 'high load' following the definition of Jiang et al. [84], i.e. having utilization of over 80%. As demonstrated by Akella et al., these links can be both intra-AS and inter-AS ones. For illustration purposes, Figure 4.4 shows where bottlenecks tend to appear in the networks of Greek tier-2 ISPs.

2. The propagation delay over a link was defined to be equal to $t = \frac{s}{v}$ where s is the distance between the two ends of the link and v is the propagation speed. The propagation speed was selected to match the quality of the links of the corresponding ASs.

Chapter 5

Implementation

This chapter is concerned with the implementation of the designed system in SSFNet. The first section outlines the features of SSFNet framework and explains the reasons why it was selected as a suitable simulation environment. The second section presents the existing models of SSFNet which were used and the necessary adaptations that had to be made for the requirements of this project. Section 5.3 proceeds to provide more details regarding the implementation. It illustrates the most interesting aspects of the designed artefact and sheds light on some of the choices and assumptions that were made. Finally, the last part of this chapter provides comments on the definition of the network topology with the help of the Domain Modeling Language.

5.1 The Simulation Environment

For the deployment of the designed system, a variety of different modeling environments were considered. Among them were the OPNET Modeler [85], the OMNET++ simulator [86], the Network Simulator NS2 [87] and the Scalable Simulation Framework SSFNet [88]. The simulators were assessed on the basis of the system under design and in accordance with the evaluation criteria specified in the work of Naicken et al. [89] (scalability, usability, hardware limitations etc). Early enough, SSFNet [88] emerged as one of the most suitable choices.

SSFNet is a a public-domain standard used for the discrete-event simulation of large-scale networks. It is a packet-level simulator which allows for modeling at and above Layer 3. Its latest release in written in Java. It comprises two derivative frameworks, each of which is organized within a different Java packet. The first one, *SSF.Net*, is responsible for the modeling of those elements that are closely related to the network itself such as the links (both shared and point-to-point ones), the network interfaces, the packet queues etc. The second framework, *SSF.OS*, is concerned with the modeling of the protocols and the processes of the operating systems which are installed within the hosts. It includes a large repertoire of protocols, ranging from IP to HTTP. Each designed protocol extends the *ProtocolSession* abstract class and all the protocols running on a host (i.e. all the ProtocolSessions) are incorporated within a single 'container', a *ProtocolGraph*. In other words, each node of the network has a unique ProtocolGraph containing a number of ProtocolSessions, with each one of them corresponding to one of the protocols running on the node. The design of the SSF.OS package is based on *x-kernel* [90], an object-oriented framework used for the development of network protocols.

SSFNet also relies on the Domain Modeling Language (DML), a standardized syntax that allows the definition and the configuration of network models. DML is used to define the topologies of the simulated networks in a hierarchical fashion, with the top network being a composition of lower level

building blocks. Moreover, DML serves an additional purpose; it is used for the configuration of the network properties and the protocols installed within the hosts, with the help of an 'inline attribute substitution' mechanism. It is easy to use and it is appropriately documented. Finally, DML code is reusable, portable, machine-independent and can be easily stored in a relational database.

The collection of data in SSFNet is possible with the help of a scalable network monitoring infrastructure implemented within the classes of the *SSF.Util.Streams* package. In SSFNet, data is exported in the form of record streams. Each record is a sequence of bytes and is encoded according to Java's standard primitive data type serialization rules. A *Recorder* has to be used for the storage of a record stream. The Recorder first *connects* to a file (or *data sink* according to the terminology of SSFNet) and then *emits* records . At the end of the simulation, all Recorders are disconnected from the data sinks. The exported records can only be read by a suitably defined *Player*. Firstly, the Player *connects* to the file (or *data source*) and reads the stored records. Then, it decodes them, demuxes them and can process them further. To help the users, the authors have designed a pseudo-protocol called *ProbeSession* which can be installed within any host and can provide a Recorder for the collection of the records. Then, it is up to the users to create appropriate *Player* classes to decode and post-process the measurements.

The latest SSFNet version comes with an extra tool called *Raceway Views*. Raceway allows for the visualization of the defined topologies; it receives the DML configuration files as an input and generates hierarchical graphical representations of the network. Raceway can also be used to create animations illustrating the flow of data in the network and to dynamically plot the collected measurements.

SSFNet was finally selected for the following reasons:

1. It supports the execution of parallel simulations on a variety of platforms such as Windows, Linux and Solaris. The SSFNet simulation kernel maintains multiple event queues and can execute them on different processing units, guaranteeing the necessary synchronization. The user is given the option to partition the hosts of the modeled network into groups called *alignments*. Each alignment can then be executed on a different processor. The SSFNet documentation states that even if a machine has only one processor, having multiple threads running in parallel can boost performance. For example, a simulation conducted on a typical Dual Core machine with Windows was shown to be 1.7 times faster when it uses parallelism.

2. It is distributed at no cost.

3. It satisfies one of the most important requirements: scalability. As the name of the simulator implies, its creators placed particular emphasis on ensuring the scalability of the framework, both in terms of modeling and in terms of performance. The authors have demonstrated that SSFNet can run efficient simulations of models consisting of thousands of nodes (even up to 33,000).

4. It comes with existing models for a variety of protocols such as IP, TCP, UDP, OSPF, BGP and HTTP, all of which are required for the study of the designed P2P system. Moreover, the source code of these models is open and this allows for the incorporation of additional elements if this happens to be necessary.

5. The models of the protocols are written in Java, a programming language with vast support and documentation.

6. It has already been used successfully in previous studies of the performance of P2P Systems. Examples are the Epichord project of Leong et al. [91] and the Oracle proposal of Aggarwal et al. [15].

7. It simulates a variety of networking phenomena such as packet loss, propagation delay etc.

5.2 The use of existing Protocol Models

The newly designed protocols (i.e. the P2P BitTorrent Protocol and the Oracle Protocol) lie on the Application Layer. For the rest of the simulated layers of the protocol stack, as well as for the modeling of the background traffic, the existing SSFNet models were used. Some of them had to be adapted, extended or simplified, so that they operated appropriately and in accordance with the requirements of the designed system. The following list provides a brief overview of these models. Their basic characteristics are outlined and any modifications that were introduced to them are described.

- **TCP**. SSFNet provides a complete model of the TCP protocol. Two different variants of TCP are implemented. The first one, the Tahoe variant, includes Slow Start, Congestion Avoidance and Fast Retransmission. The second one, the Reno variant, extends Tahoe by adding a Fast Recovery mechanism. The model was created by Hongbo Liu and Andy Ogielski and has been revised many times. Its documentation, which can be found in SSFNet's web portal [88], provides a detailed description of the implemented features. Nevertheless, the model had to undergo important changes, so that it behaved correctly in the context of the designed application. The following list presents some of the changes that were introduced:

  1. In the original model, when a new connection was initiated, the TCP socket was bound to a random local port. This feature was fixed by demanding that the socket is bound to the port specified by the application.

  2. The TCP Protocol Specification [92] states that if an unsolicited FIN message is received, TCP has to notify the application that the remote party has initiated the termination of the connection. This mechanism was not appropriately implemented in the model and it would sometimes cause the algorithm to reach a dead end. The problem was fixed by enhancing the notification mechanism according to the specification.

  3. When the protocol sends the final ACK of the connection termination process, it has to wait for a period of time equal to the double of the maximum segment life (MSL) time, to ensure that the ACK it sent was received. The existing *2MSL* timer of the model was problematic, as it was using a wrong time unit. The timer was appropriately fixed.

  4. Several other minor bugs were fixed, such as missing *break* commands in *switch-case* statements and null pointers.

- **sOSPF**. SSFNet's *sOSPF* package was selected for the simulation of the Open Shortest Path First version 2 protocol, which was used for routing decisions within the ASs. The more advanced *OSPFv2* model was not used, as it is still under development and lacks important features. Although OSPFv2 is more dynamic and allows for links with costs not equal to 1, it is not able to process external route information originating from inter-AS routing protocols such as BGP. On the other hand, despite its static nature, sOSPF is more stable and it has been proven to work correctly in a multi-AS environment. Thus, sOSPF was finally used, but it was appropriately augmented so that it implemented extra functionality. In a sense, the newer but incomplete OSPFv2 was used as a guide to the extension of sOSPF. Firstly, the key features of OSPFv2 that were absent from sOSPF were identified and isolated. Then, these features were added to sOSPF. The most important of them was the support for different treatment of the links. sOSPF was made to calculate the cost of the links on the basis of their capacity, exactly as it was described in section 3.2.1. sOSPF was originally authored by Philip Kwok with the help of Andy Ogielski. OSPFv2 was originally authored by Hagen Bohm with advice from Dirk W. Jacob and Anja Feldmann.

- **UDP**. SSFNet includes a simple model of UDP, which offers a BSD-like socket interface. It was

created by Andy Ogielski. Its usage is demonstrated via two demo examples, one of a UDP client and one of a UDP server. These examples assume that UDP datagrams are never lost, but they always reach their destination. In the experiments of this project, packet drops will obviously be considered. Therefore, whenever the trackers use UDP to communicate with the oracles, suitable timers have to be introduced so that packet loss is handled at the application layer.

- **BGP4**. SSFNet's model of BGP is based on Internet Engineering Task Force's Request for Comments number 1771 (RFC 1771 [93]). BJ Premore is its main author. It is a very sophisticated and detailed model and includes the majority of the features of the protocol. As it is stated in the documentation, all the types of messages of the protocol and all the standard path attributes have been included. Moreover, among others, the implementation offers support for route reflection, update message handling, route flap damping, policy configuration and route filtering. Furthermore, it includes all the timers of BGP, some of which can be optionally made to be jittered. The BGP model of SSFNet was used without modifications.
- **WWW**. The WWW model of SSFNet was used as a basis for the simulation of the background traffic. As it was already discussed, each AS incorporates a number of HTTP Clients and HTTP Servers, which exchange WWW content. Monitoring the performance of this exchange will be used to evaluate how the proposed ALTO solutions interact with the operation of the rest of the Internet applications. The WWW packet of SSFNet contains classes which model the empirically observed HTTP traffic. As it is stated in the Readme files included within SSFNet, the WWW packet was designed by Andy Ogielski and was insipred by the empirical model of Feldmann et al. [94]. For the scope of the project, the model was simplified, so that it only included the core functionality of the HTTP protocol; clients send HTTP Requests to the servers and the servers reply with HTTP Response messages which include the requested content. Some of the more sophisticated features of the model, such as the generation of requests for multiple objects of variable size, were excluded, as they added unnecessary overhead to the simulations. On the other hand, new features were added. Most notably, the *Stream Package* was used to probe the time needed for the completion of the Request-Response conversations between the two parties.

5.3 Details of the implementation

Following the programming guidelines proposed by the authors of SSFNet, we proceeded with the implementation of the described system. The Java code is organized in four packages.

The *SSF.OS.BitTorrent* package includes the classes that are responsible for modeling the functionality of the BitTorrent protocol and are used by the BitTorrent clients as well as by the Tracker. The messages exchanged in the context of this protocol are included within the *SSF.OS.BitTorrent.messages* auxiliary package. The third package, *SSF.OS.OracleProtocol* contains the classes which model the interaction between the Tracker and the Oracle, i.e. the 'client' and the 'server' of the Oracle Protocol. Finally, a smaller *SSF.OS.ALTOplayers* package comprises the Record Players which will be used for the post-processing of the collected record streams.

In many cases, code which is written within a simulation environment does not respect some of the fundamental Software Engineering principles. One of the most important reasons is that the architecture of the simulator itself usually encourages (or even demands) the adoption of programming anti-patterns. For example, the protocol implementation framework of SSFNet can easily lure the user into writing code with multiple Circular Dependencies. By examining the existing SSF.OS.BGP4 packet, it can be calculated that it involves 12 circular dependencies. Similarly, the SSF.OS.TCP packet suffers from 10 cycles. Nevertheless, a lot of attention was placed so that the designed code would be of good quality.

Eventually, all four packets of the newly designed system were made to exhibit no circular dependencies. Various techniques were used to achieve this desirable result. Most notably, the *Dependency Injection* [95] technique was applied and it allowed the decoupling of highly dependant components. Interfaces were extracted and static factories were utilized for the creation of the objects whose classes implemented these interfaces. The factories were introduced in such a way that in the future the system can be easily enhanced to adopt the *Abstract Factory* pattern [96].

In addition, the *Abstractness* of the designed code (defined as the ratio of the number of abstract classes and interfaces to the total number of classes [97]) is 22%, with the average Abstractness of the rest of the packages of SSFNet being 9% (for instance, the SSF.OS.TCP and SSF.OS.OSPF packages have an abstractness of 0%). This is a good indication that the designed code is polymorphic and abides by the principles of Object Oriented programming. Finally, it must be pointed out that the *McCabe Cyclomatic Complexity* [98] of our code is 20% lower than the average McCabe Cyclomatic Complexity of the whole SSFNet framework. The only weakness of the *SSF.OS.BitTorrent* and *SSF.OS.OracleProtocol* packages is their high *Efferent Couplings* (i.e. the number of packages that they depend on). Unfortunately, there is no remedy for this problem, as the application interacts with a large number of the existing protocols and unavoidably depends on their corresponding implementations.

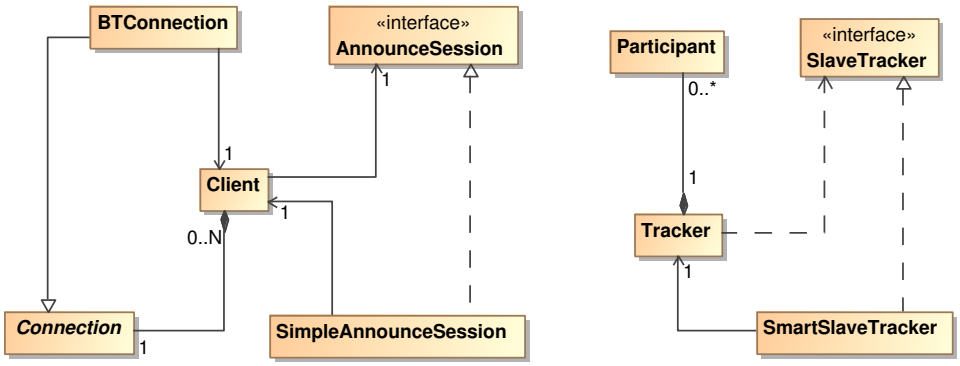


Figure 5.1: Key Classes of SSF.OS.BitTorrent package

Figure 5.1 shows the UML diagram of the most important classes of the SSF.OS.BitTorrent package. As it is obvious, the classes can be grouped into two different categories: the ones which are related to the Tracker and the ones which are related to the BitTorrent Client. If it is required, the two groups of classes can be split into two completely independent Java packages without having to alter the existing code.

The *Tracker* is a protocol, in the sense that it extends SSFNet's ProtocolSession class and can be installed within a ProtocolGraph. Its role is to manage all the functionality of the tracker. Whenever a Tracker Request message arrives from a remote client over a TCP connection, the Tracker class must create a new session whose purpose will be to serve this particular request. A class representing such a session must implement the *SlaveTracker* interface. In our implementation of this interface (called *SmartSlaveTracker*), all the necessary steps are performed so that a suitable Tracker Response message is constructed. The *SmartSlaveTracker* can be configured so that it gives random peer recommendations or utilizes the services of the oracles. In the latter scenario, the control is passed to the Oracle Protocol, whose development will be described below. The tracker maintains a list of all the clients participating in the swarm (instances of the class *Participant*). Clients who have not re-announced for a certain time interval are considered to have departed and are therefore removed from the list.

Similarly, the *Client* class runs within the ProtocolGraph of a BitTorrent client and is responsible for managing the individual connections and assembling the file being downloaded. A client can have up to

a maximum number of connections running at the same time. Our implementation of the *Connection* interface incorporates all the features of the BitTorrent protocol which were discussed in Chapter 3. The Client class is also responsible for initiating the announcements to the Tracker, whenever this is required by the specification. An instance of a class implementing the *AnnounceSession* interface must be created for this purpose. As soon as the communication with the tracker is completed (with or without success), this session object serves no other purpose and can be discarded. A new session will be created for the next announcement.

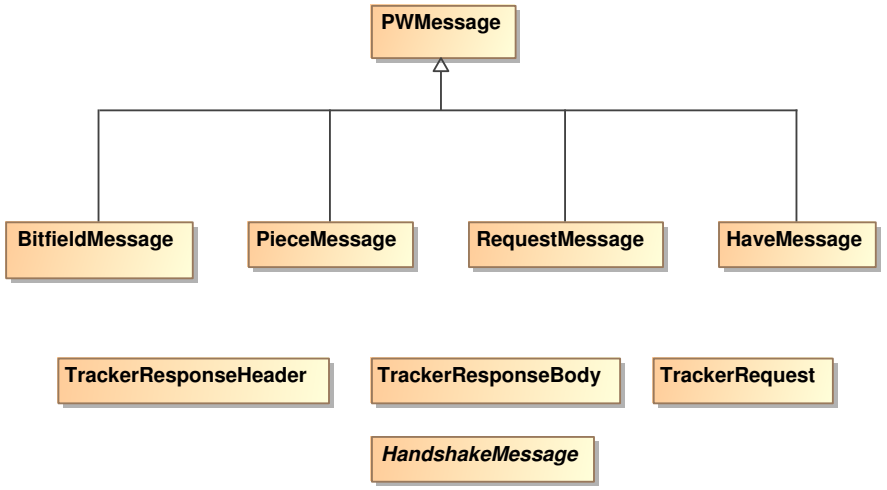


Figure 5.2: Key Classes of SSF.OS.BitTorrent.messages package

Figure 5.2 depicts the UML class diagram of the SSF.OS.BitTorrent.messages package. Three classes are used to represent the messages exchanged between the Tracker and a Client. The first one corresponds to the *Tracker Request* message, while the other two are used to model the *Tracker Response* message. The latter may have a variable size, depending on the number of peers contained in the response. SSFNet demands that such messages are split into two parts: a 'header' indicating the size of the message and a 'body' containing the communicated data. This is the reason why two classes are needed for the representation of the Tracker Response. The messages of the Peer Wire Protocol are objects of the class *PWMessage*. Certain types of these messages carry extra information (e.g. the *Bitfield* message contains the sender's bitfield) and for these types, additional classes are created and are made to extend the PWMessage class.

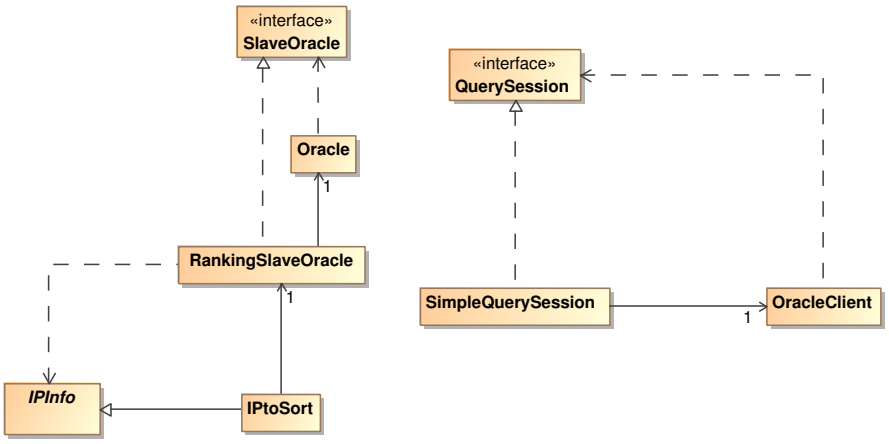


Figure 5.3: Key Classes of SSF.OS.OracleProtocol package

Figure 5.3 shows the most important classes contained within the SSF.OS.OracleProtocol package. The *Oracle* class is run by the oracles and its purpose is to accept incoming queries from potential clients. It is capable of accepting requests both over TCP and over UDP. For each new query, the Oracle generates an individual session *SlaveOracle* which will be responsible for serving this particular ALTO client. In our implementation of the *SlaveOracle* interface, the *RankingSlaveOracle*, the list of IP addresses contained within the query message is ordered according to a configurable ranking criterion and is sent back to the ALTO client, layered on top of the same Transport Protocol as the one which carried the corresponding query.

RankingSlaveOracle uses the received IPs to construct an ArrayList of *IPInfo* objects. The *IPToSort* class implements the *IPInfo* interface, as well as Java's Comparable interface. As a result, it has a *compareTo()* method which conducts a comparison between two candidate IPs and determines which of them is preferred with respect to the employed ranking strategy. The *Collections.sort()* method can obviously be used to sort the ArrayList of *IPToSort* objects with maximum efficiency, on the basis of this comparison scheme.

The *OracleClient* class defines the client-side part of the Oracle Protocol and is embedded within the ProtocolGraph of the potential ALTO client (in our case the Tracker). Whenever the ALTO client wishes to consult an oracle, the *OracleClient* class generates a new session *QuerySession* which is responsible for initiating the communication with this particular oracle, sending the query and waiting for the reception of the corresponding response. In the context of our system, a tracker will have one instance of the *OracleClient* class and this instance will create multiple *QuerySession* sessions, one for each of the requests sent to an oracle.

The final package, SFF.OS.ALTOplayers, contains four classes used for the decoding, the demuxing, the grouping and the manipulation of the collected measurements. Their implementation was based on the paradigm of the SSF.Util.Streams.BasicPlayer class. The first one is used for the post-processing of the completion times and the second one for the calculation of the utilization of the links of the network. The third one post-processes the measurements concerned with the performance of the HTTP communication which is conducted in the background. Finally, the last one is concerned with the average download performance of the swarm.

During the implementation phase, the following additional assumptions had to be made:

- Although the messages which are exchanged between the parties were made to have their appropriate size, the fields that are not of interest for the scope of the project were not included in them. Examples of such fields are the Strings specifying the versions of the protocols or the SHA1 values used for security reasons. This decision reduced the number of objects in the Java Heap and speeded up the simulations, without interfering with the quality of the results.
- The model was made to support the coexistence of multiple swarms. It is assumed that each of the swarms is managed by a different tracker.
- Regarding the distribution of the pieces of the file at the beginning of the simulation, the model proposed by Qiu et al. in [38] was adopted. According to this study, it is reasonable to assume that the number of pieces in each leecher is uniformly distributed within the interval $[0, N-1]$, where N is the total number of pieces of the shared file. As the BitTorrent clients employ a Rarest First piece selection strategy, it can also be assumed that these pieces are chosen randomly from the set of all the pieces of the file.
- The number of upload slots and the minimum and maximum number of connections per torrent are determined from the upload bandwidth of each client, with the use of a strategy similar to the one exercised by the uTorrent client [61]. Again, the values are appropriately scaled down so that they correspond to the size of the simulated network.

- When the study of a longer fraction of the lifetime of the system is demanded, additional downloaders have to be added to the swarm as the simulation proceeds. This issue was addressed in a way inspired by the work of Le Blond et al. [99]. After a client completes the download of the file, it remains in the swarm as a seed for an amount of time T and then gracefully departs. This client is then substituted by a new one, which will be running within the same SSFNet node.

5.4 Definition of the Topology

The final step of the implementation was the definition of the topology of the network with the use of the DML language. Following the principles of DML, the top network was hierarchically composed from smaller pre-configured sub-Nets. In this case, a sub-net was one of the eight ASs. In their turn, the ASs were also composed from a number of smaller building blocks. Thanks to this multi-stage structure and the reusing of the same components, code duplication was diminished.

In addition, DML was used to specify the protocols which were to be installed in each of the hosts of the network, or similarly the classes extending the *ProtocolSession* class which were to be used by each host. As it was required, the protocols were declared in the order of the layer which they belonged to, beginning with the protocols of the Application Layer and ending with IP. Furthermore, DML was used for the configuration of the protocols, both the newly implemented ones and the existing ones. For instance, the DML syntax was used to specify parameters such as the number of pieces of the file which would be exchanged via BitTorrent or the window size of TCP. Finally, suitable probes were set up for the collection of the measured data. Figure 5.4 shows the AS topology as it appears in Raceway Views. Figure 5.5 illustrates a subset of the internal network of AS1, again as it is depicted by Raceway Views.

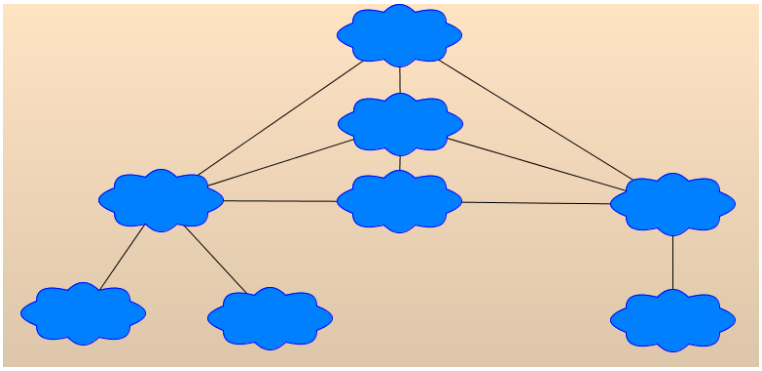


Figure 5.4: AS Topology in Raceway Views

One of the challenges in the definition of the topology was the correct configuration of the BGP routers. This was due to two reasons. Firstly, in a network topology presenting multiple instances of the types of the different possible AS relationships, one has to be very careful when determining which routes must be advertised by a router and which must not. Secondly, the definition of route filters is a feature of SSFNet's BGP4 model that is still under development and has not been fully exercised yet. The existing documentation is not rich enough and the authors themselves advise the user to use the policy mechanism of BGP4 with extra care. Therefore, the final configuration had to be tested thoroughly so that it was ensured that it behaved appropriately.

On the whole, the policies were defined so that they complied with the following rules:

1. An AS should filter the traffic which comes from one of its providers and is directed towards one of its peers.

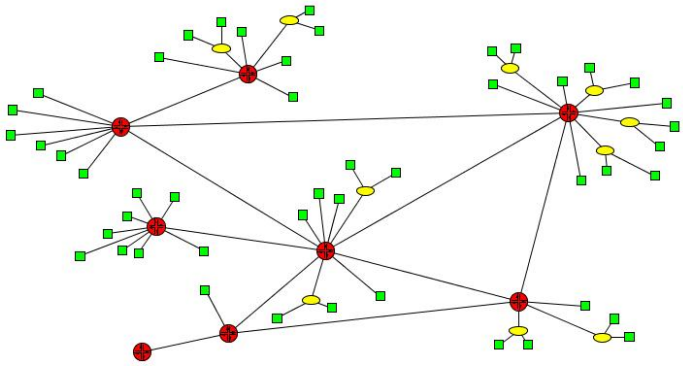


Figure 5.5: Topology of AS1 in Raceway Views

2. An AS should filter the traffic which comes from one of its providers and is directed towards another one of its providers.

3. An AS should filter the traffic which comes from one of its peers and is directed towards one of its providers.

4. An AS should filter the traffic which comes from one of its peers and is directed towards another one of its peers.

The rules were retrieved by the web portal of the Wireless Network of Tripoli (TRWN [100]).

In addition, as it was already described in Chapter 4, traceroute experiments were conducted so that the *Local Preference* of the routers was determined. The results of the experiments were used in order to configure the *local_pref* parameter for each route in DML: the higher the value of the parameter, the more the corresponding route is preferred by the AS. Listing 1 provides an example of the definition of the route filters in DML. This fragment of code is a part of the configuration of the Border Router of AS3. It aims at defining the input and output filters for this router with regard to its neighboring BGP speaker of AS2. More precisely:

- Line 2 defines the identity of the neighbor. It specifies the AS where it belongs (in this case AS2), as well as the address of the interface through which it accepts BGP messages. Moreover, it declares the id of the interface on the local border router of AS3 which the neighbor will need to use for the BGP communication.

- Line 3 defines three parameters of the BGP sessions which will be established with the neighbor. The first one of them is *Hold Timer Interval*. Its value specifies the maximum number of seconds which are allowed to elapse between the receipt of two successive messages from the neighbor. If this interval is exceeded, the connection is considered to be lost. The second parameter is the *Keep-Alive Timer Interval*. It corresponds to the maximum number of seconds which are allowed to elapse between two successive transmissions of messages. When this timer expires, the local router should send a Keep-Alive message in order to preserve the connection. The third parameter,the *Minimum Route Advertisement Interval* (MRAI), defines the minimum allowed time (in seconds) between two successive route announcements to the neighbor.

- Lines 4-20 define the filter for incoming route advertisements which originate from the border router of AS2. An input filter consists of several sub-filters called clauses and when combined, all of these clauses form the complete inbound filtering policy of AS3 with respect to AS2. A clause consists of two main parts: a predicate and an action. The predicate specifies the routes which this sub-filter applies to. This is achieved with the help of a canonical expression called matcher. This canonical expression is checked against a route and if they are found to match,

Listing 1 Input and Output Filters of the Border router of AS3 for its neighbor AS2

```
1     neighbor [
2          as 2  address 0(1)  use_return_address 0(1)
3          hold_time 90  keep_alive_time 30  mrai 30
4          infilter  [
5              clause [
6                  precedence 1
7                  predicate [atom [ attribute nlri_nhi matcher "^5" ] ]
8                  action [ primary permit atom [ attribute local_pref  type set  value 120 ] ]
9              ]
10             clause [
11                 precedence 2
12                 predicate [atom [ attribute nlri_nhi matcher "^8" ] ]
13                 action [ primary permit atom [ attribute local_pref  type set  value 120 ] ]
14             ]
15             clause [
16                 precedence 3
17                 predicate []
18                 action [ primary permit ]
19             ]
20         ]
21
22         outfilter [
23             clause [
24                 precedence 1
25                 predicate [atom [ attribute nlri_nhi matcher "^3" ] ]
26                 action [ primary permit ]
27             ]
28             clause [
29                 precedence 2
30                 predicate [atom [ attribute nlri_nhi matcher "^6" ] ]
31                 action [ primary permit ]
32             ]
33             clause [
34                 precedence 3
35                 predicate [atom [ attribute nlri_nhi matcher "^7" ] ]
36                 action [ primary permit ]
37             ]
38         ]
39    ]
```

then the action of this filter is applied to that particular route. For example, the first clause of the input filter (Lines 5-9) has a predicate whose matcher is "^5" (Line 7). This means that this clause applies to all the advertised routes to AS5. The action which will be applied on these routes is 'permit' (Line 8), i.e. they will not be denied. Moreover, line 7 performs an additional configuration; it sets the local_pref attribute for these routes to 120.

Each clause of a filter also has another important attribute, the precedence. The precedence defines the order in which the clauses of a filter will be checked as potential matches for an incoming route. In other words, each newly advertised route will be first compared to the predicate of the clause with precedence 1, then to the predicate of the clause with precedence 2 etc. If a match is found, the action of the matching clause is applied to the route and all further clauses are ignored. If all clauses are checked and no matches are found, the route is denied.

As it has already been discussed, the first clause permits all routes to AS5 and sets their local preference to 120. Similarly, the second clause permits all routes to AS8 and sets their local preference to 120 (Lines 10-14). As the default value for the local preference is 100, these definitions mean that AS3 prefers to reach AS5 and AS8 via AS2 (and not via its other provider AS1). The third clause (Lines 15-19) simply permits all the remaining routes, without changing their local preferences.

- Lines 22-38 define the filter for outgoing route advertisements to AS2. Again, the syntax of the clauses follows the same rules. Clause 1 (Lines 23-27) permits the advertisement of all routes to AS3 (i.e. the local AS). Similarly, clauses 2 and 3 permit the advertisement of all routes to AS5 and AS6 respectively. As there is no other clause in the outbound filter, all other routes are blocked. Hence, the only routes that AS3 advertises are those towards hosts in its own network and the networks of its customers, following the rules which were discussed above.

Chapter 6

Evaluating the Oracle approach: Merits and Pitfalls

This chapter presents the results of the evaluation of the system which was described in Chapter 3. It begins with a statement of the goals which we attempted to achieve through the experiments that were executed. It then continues with some general remarks on the methodology and the metrics used for the evaluation. After that, it proceeds with analyzing the results of the experiments in detail. The chapter ends with a summary and a brief discussion of our findings.

6.1 Aims

Extensive experiments were conducted within the simulation framework with the help of the newly designed models which were described in Chapter 5. A variety of different issues were addressed:

1. Through the first series of simulations, we aimed at confirming that the Oracle approach can aid in the resolution of the ALTO problem. More specifically, we tested whether it constitutes a Win-Win solution for the two cooperating parties (P2P applications and ISPs) and we attempted to quantify the extent to which these two parties benefit from the introduction of such a service.

2. As a second step, we experimented on the different ranking criteria which can be employed by the oracles. Although a variety of different ranking policies have been proposed in the bibliography, very few of them have been tested. Our goal was to compare them and determine which of them perform better in the context of a P2P file sharing system. We also aimed at identifying which of these strategies are of benefit to both collaborating parties and which ones constitute a blessing for only one of them. Finally, we attempted to exercise the 'combination' algorithm which was presented in Section 3.2.1 so as to evaluate how the merits of different criteria can be married.

3. In addition, we conducted experiments with view to exposing the potential pitfalls of the Oracle approach. The first goal was to test the robustness of the system during the initial phase of the lifetime of a swarm. Furthermore, we aimed at measuring the effect of the presence of malicious oracles on the performance of the P2P application, as well as on the overall state of the network. We tried to shed some more light on this unexplored issue and check whether the hostile intentions of the oracles can make the ALTO service a threat on the balance of the system.

6.2 General Remarks

The simulations were conducted on the topology which was described in Chapter 4. It was assumed that the file being shared was a 350 MB 43-minute .avi video. This practically corresponds to a one-hour television show without the commercials. This choice was determined by the fact that television content has been proven to be one of the most popular types of data being distributed over BitTorrent [101]. The file was assumed to be divided into 700 pieces, with each piece comprising 32 blocks of 16kB each.

The effectiveness of the different variations of the oracle approach will be evaluated on the basis of the extent to which they satisfy the two cooperating parties: the BitTorrent users and the ISPs. The satisfaction of BitTorrent users is determined by the download performance that they experience. The download rate is an appropriate metric that can be used to illustrate their happiness.

On the other hand, ISPs wish to achieve the following three objectives:

1. Minimize the costly traffic traversing the inter-AS links which connect them to their customers (the ASs from which they buy IP transit). This is equivalent to minimizing the utilization of these links. Therefore, the utilization value of the inter-AS links can be used to measure the satisfaction of the ISPs. The higher this utilization is, the lower the satisfaction. It is assumed that this metric is calculated in 95 percentiles, which is the most popular charging model in the Internet [102]. The calculation follows the algorithm provided in [103].

2. Minimize the congestion on the links of their AS. Therefore, the utilization of the intra-AS links can also be used as an evaluation metric. Again, the 95 percentile model is used for this calculation,

3. Maximize the satisfaction of their customers. These include the users of the P2P applications, as well as all the background users exchanging other types of traffic (such as WWW or E-mail). As it has already been discussed, the exchange of HTTP traffic has been modeled in our system. Hence, by measuring the performance of this background communication, we can have an indication of how successful the ISP is in catering for the needs of its customers. We calculate the duration of the HTTP sessions, defined as the time required for the transmission of an HTTP Request and the reception of the corresponding fixed-size HTTP Response. The larger the duration of these HTTP sessions, the less the Quality of Experience for the HTTP clients and the less the ISP fulfills its goal.

6.3 Overview of the performance of the Oracle Solution

We begin the evaluation by demonstrating the general benefits of the application of the Oracle approach. This section is an introductory one and it is meant to illustrate that the adoption of the Oracle service can alleviate the consequences of the ALTO problem and can prove to be a Win-Win solution that satisfies the interests of both the BitTorrent users and the ISPs.

It is assumed that two swarms are operating simultaneously on the network. Moreover, ASs are assumed to comprise Oracles which exercise the simplest and less sophisticated ranking criterion; they segregate candidate peers according to whether they belong to the local AS or not. Internal peers are ranked higher than external peers. We run simulations on the traditional standard architecture which does not use the Oracle service. We then introduce the Oracle support and repeat the experiments.

Figure 6.1 plots the cumulative distribution function of the download rate of the BitTorrent clients for both the standard and the Oracle-enhanced scenario. It is obvious that the incorporation of the Oracle

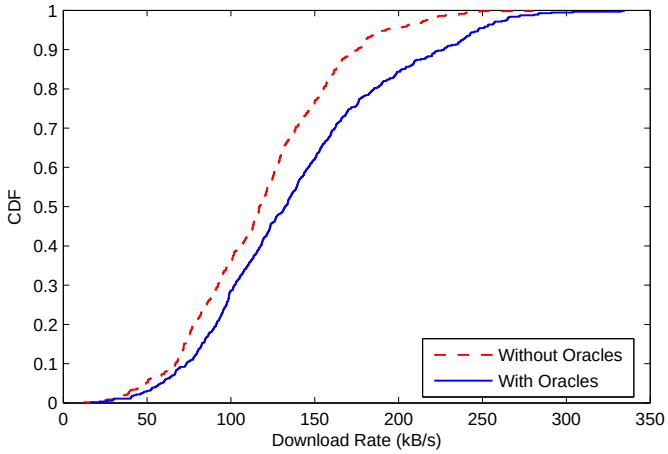


Figure 6.1: CDF of Download Rate before and after the introduction of simple Oracles.

solution boosted the download performance of BitTorrent significantly. More precisely, the average download rate improved by 19% and the median increased by 13%. At the same time, the coefficient of variation of the rates (defined as the ratio of the standard deviation to the mean) did not change and this proves that the fairness of the protocol was not negatively affected.

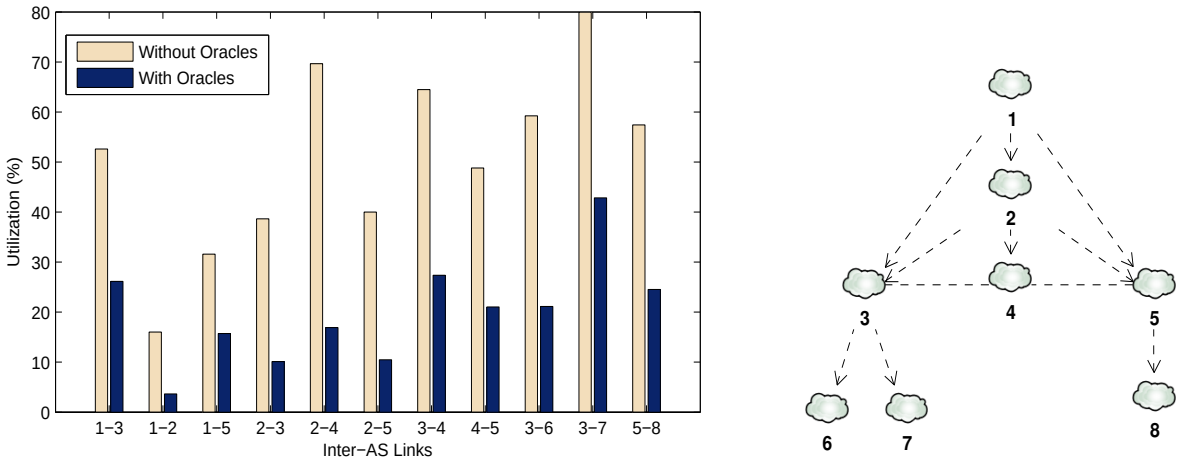


Figure 6.2: Utilization of inter-AS links before and after the introduction of simple Oracles. The pairs X-Y on the horizontal axis indicate the indices of the ASs connected by the corresponding link.

Figure 6.2 depicts the utilization of the inter-AS links in 95 percentiles. As it is obvious, ASs managed to localize traffic, as the utilization decreased by 63% on average. For instance, the traffic crossing the link which connects AS1 and AS2 was reduced by 77%. This implies that the communication between nodes of these two ASs will become faster, as well as that AS2 will be exposed to fewer charges.

Figure 6.3 plots the utilization of the intra-AS links of a typical Tier-3 AS. It can be seen that utilization decreased for some links and increased for others. The links of the first group (e.g. 1,2,3) are the ones which form part of the path that leads to the border/exit of the AS and they were obviously relieved from a large portion of their load thanks to the reduction of inter-AS traffic. The links of the second group (e.g. 4) are the ones which are only used for the exchange of content between the PoP of the AS. Their utilization increased as a side-effect of the localization of the traffic.

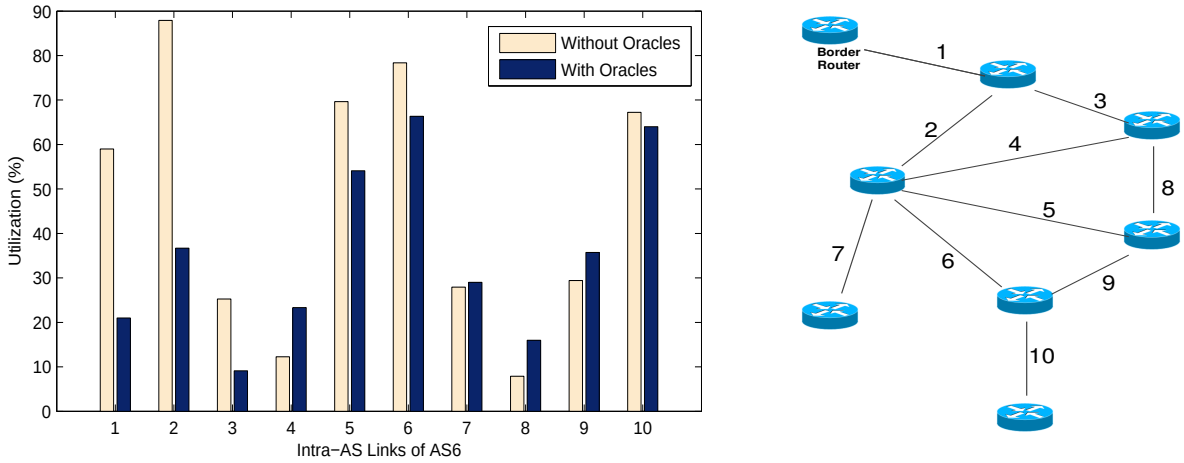


Figure 6.3: Utilization of interior links of a typical Tier-3 AS before and after the introduction of simple Oracles.

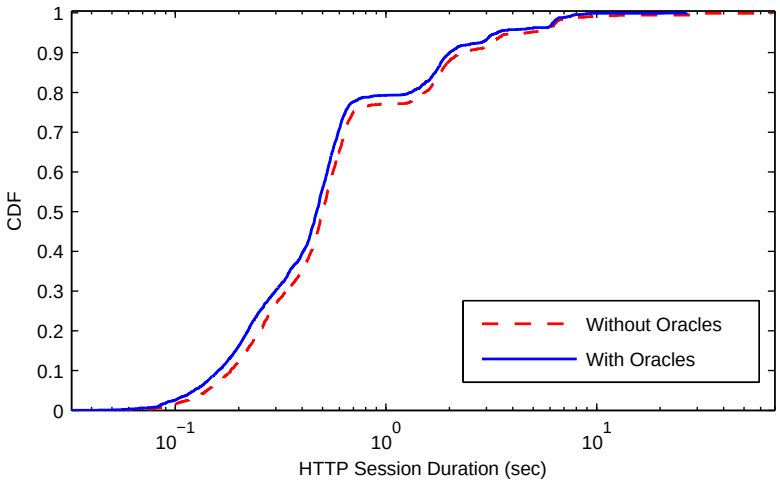


Figure 6.4: CDF of Session Duration of Background HTTP traffic before and after the introduction of simple Oracles.

Figure 6.4 depicts the CDF of the HTTP session duration. Again, it can be seen that the Oracle solution improved the QoE of the non-P2P network users, as the average time required for the HTTP communication decreased by 22%. In addition, the CDF of the session duration for the scenario without oracles can be seen to have a much longer tail. This is in accordance with the fact that in the oracle-guided case the coefficient of variation of the session duration decreased by 38%. Indeed, thanks to the introduction of the oracles, cases of very problematic interactions between HTTP clients and HTTP servers were diminished. This is due to the fact that the number of over-congested links was reduced.

Finally, Figure 6.5 presents the average duration of HTTP sessions between parties in the same AS, as well as between parties in different ASs. It can be seen that the improvement of the total average session duration is the combined effect of the improvement of the communication with both local and external HTTP servers. This is an important conclusion, because it illustrates the fact that the Oracle approach has promoted the well-being of the network both at the intra-AS and at the inter-AS level.

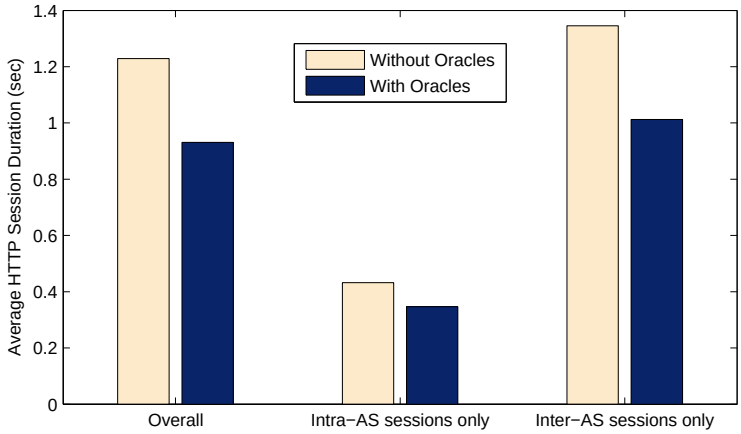


Figure 6.5: Average HTTP Session Duration before and after the introduction of simple Oracles.

TCP VS UDP

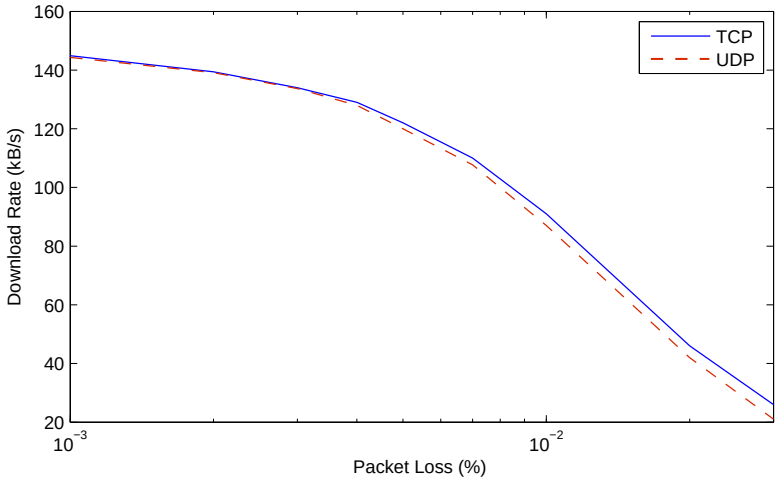


Figure 6.6: Download Performance for different values of the Packet Loss rate at each network interface.

The performance of the system was tested for the two alternative Transport protocols, TCP and UDP. Simulations were run for different packet loss scenarios. Figure 6.6 plots the download performance of the BitTorrent clients, for different values of the Packet Loss rate at each network interface. For cases of small packet loss (less than 0.3%), UDP was found to give as good performance as TCP and as a result it could be preferred to TCP for the less load it places on the operating systems. However, as packet loss grows, so does the advantage of TCP over UDP and for cases of large packet loss (e.g. more than 1% lost packets on each network interface) TCP may ensure BitTorrent download rates that are 10% larger than those of UDP.

6.4 Ranking Criteria

One of the objectives of the project is to study and compare the alternative ranking criteria which can be employed by the oracles. A first presentation of the strategies which the oracles were designed

to support was provided in Chapter 3. In this section, these ranking criteria will be tested on the selected topology and they will be compared on the basis of whether they manage to further satisfy the incentives of the BitTorrent clients and the ISPs.

'AS Distance' strategy

We begin by studying the effect of the AS-distance based neighbor selection. Instead of treating all the peers outside the AS in the same manner, preference is granted to those situated within the most local ASs. In order to test this criterion, it is assumed that the clients of our network are divided into 8 equally sized swarms. This guarantees that the number of candidate peers which are located within a client's AS is smaller than the minimum number of connections which it has to establish; thus, the client will be obliged to peer with clients in other ASs as well.

Experiments showed that by considering the number of AS hops, the average utilization of the inter-AS links decreased by about 5%. Meanwhile, the download performance of the swarms improved, but the benefit was not significant enough. More precisely, the median download rate increased by 3%, while the average rate increased by only 1% (when compared to the scenario with simple oracles which treated equally all external peers). This is in accordance with the observations of Aggarwal et al. in [18]. There are two reasons why the improvement of the download performance was not as outstanding as one would expect:

1. The inter-AS links have already been relieved from most of their load thanks to the utilization of the simple oracles. Thus, these links should not be considered as bottlenecks or 'congestion hotspots' anymore. Hence, the presence of one more such inter-AS link in the path connecting a client to its peer should not be a significant burden.
2. The download experience of a BitTorrent client is dominated by characteristics of the remote peers such as the bandwidth of their access links or their load. Such characteristics are bound to influence performance to a great extent (as discussed by Dabek et al. in [29]) and may be more important than the AS distance.

Ranking interior peers

The focus will now be placed on the fine-grained strategies used for ranking destinations within the local AS. Multiple simulations were conducted to test the efficiency of these strategies. In these experiments, it was assumed that all BitTorrent clients belonged to the same swarm. The results of the simulations can be summarized in Table 6.1. For each one of the strategies, the Table gives an indication of how it affected the performance of the P2P application, as well as of whether it improved the utilization of the links and satisfied the interests of the ISPs. Following the terminology used in [9], we define the *Bottleneck Link* as the link with the maximum utilization (calculated using the 95 percentile model as usual). The table is also illustrated with the help of Figure 6.7. To facilitate the study of the results, we once again give a brief summary of the tested ranking policies:

- **'Naive' Strategy**: The naive case in which all the internal destinations are treated in the same way, without discrimination.
- **'Num of IP Hops' Strategy**: The smaller the number of IP hops on the path connecting the client to the candidate peer, the more this peer is preferred.
- **'Path Cost' Strategy**: The smaller the cost of the path connecting the client to the candidate peer, the more this peer is preferred.

Ranking Strategy	Avg Download Rate (kB/s)	Avg Utilization (%)	Utilization of Bottleneck Link (%)
1. Naive	135.7	32	93
2. Num of IP Hops	139.7	23	58
3. Path Cost	144.3	28	82
4. Fastest Access Links	143.1	34	97
5. Matching Access Links	147.3	34	97
6. Traffic Engineering	142.4	22	55
7. Combined 2 and 4	144.9	25	59
8. Combined 2 and 5	148.1	30	78

Table 6.1: Performance of the Ranking Strategies

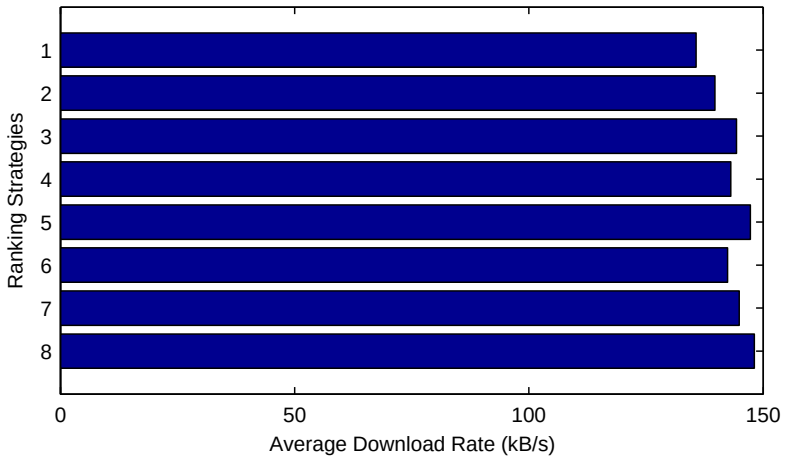
- **'Fastest Access Links' Strategy**: Peers with fast access links are preferred.
- **'Matching Access Links' Strategy**: Peers with access links whose capacity matches that of the requesting client are preferred.
- **'Traffic Engineering' Strategy**: Peers which can be reached via routes that do not contain bottleneck links are preferred.
- **'Combined Num IP Hops-Fastest Access Links' Strategy**: Combination of these two strategies into a composite criterion, as described in Algorithm 1.
- **'Combined Num IP Hops-Matching Access Links' Strategy**: Combination of these two strategies into a composite criterion, in a fashion analogous to that of Algorithm 1.

The results reveal that all the tested ranking criteria were of benefit to at least one of the two cooperating parties. Some of them satisfied both sides, while some others were beneficial to only one of them. In any case, the adoption of any of those fine-grained ranking policies is an improvement when compared to the naive scenario.

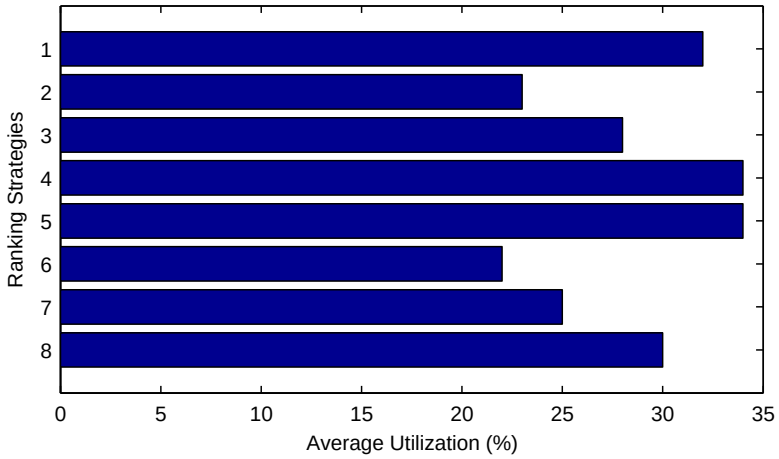
The 'Num of IP Hops' Strategy has been found to be a strong proximity-related criterion, which allows ISPs to manage the traffic within their networks more efficiently. By encouraging clients to connect to peers in their vicinity, ISPs ensured that fewer links were burdened by the traffic corresponding to these BitTorrent connections. That is the reason why this strategy resulted in a 28% decrease in the average traffic traversing the links and it also reduced the utilization of the bottleneck link by more than 37%, when compared to the naive scenario. The utilization of some of the links was found to decrease by up to 50%. However, the improvement of the download performance of the swarm was not found to be equally outstanding. More precisely, the average download rate increased only by 2.9%. This means that a local peer is not always an ideal peer. This can be due to a variety of reasons:

- Although the path to this peer is short (in terms of IP hops), it may comprise a congested bottleneck link.
- The path may also comprise links whose capacity is low.
- The peer itself may be overloaded.

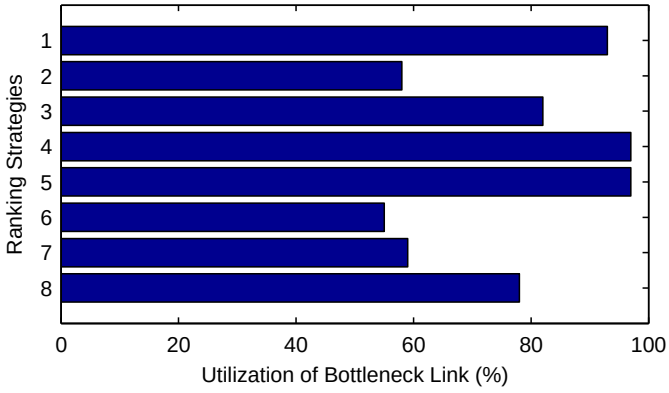
Consequently, it can be deduced that locality alone cannot provide optimum results. We now proceed to employ a performance-related criterion instead. We explore the 'Fastest Access Link' strategy, which prioritizes peers whose access links have the largest capacity. This strategy was praised by Aggarwal et al. in [18], as in their experiments on the Gnutella protocol, it was found to greatly benefit P2P users. Indeed, our experiments also demonstrated that this technique can improve the average download rate



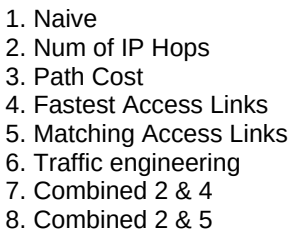
(a) Average Download Rate



(b) Average Utilization



(c) Utilization of the Bottleneck Link



(d) Legend

Figure 6.7: Comparison of the Ranking Strategies

of the swarm by 5.5%. This is a considerable improvement; however, it is not an impressive one. This observation is not in accordance with the results of the work of Aggarwal. This is due to a variety of reasons:

- Contrary to the previously discussed criterion, the 'Fastest Access Links' strategy does not consider locality at all. Although we have discussed that locality is not an answer on its own, it it still a very important parameter. Completely disregarding it is obviously not a good practice. A fast but distant peer may be less preferable than a slow but local one. Aggarwal et al. conducted their experiments on ASs with a simple star topology, where every client was equally distant from all the others. This selection of topology meant that they decided to totally ignore intra-AS proximity as a parameter of the system. That is the reason why the bandwidth of the access link was found to play such a crucial role. Our more complex intra-AS topology manages to capture the importance of locality as an additional factor that influences the performance of the application.
- By always prioritizing the fastest peers, the oracle will eventually impose a great load on them, while the slower ones will result in being underloaded. Connecting to a fast but busy peer may be less preferable than peering with a slow client with very few connections.
- The final reason applies specifically to the case of BitTorrent (and other applications which involve tit-for-tat reciprocity algorithms). Having a slow client peer with a fast one may eventually prove to be futile. The fast client will most probably choke the slow one, in favor of other fast clients. The slow client will then most probably end up exchanging data with other slow clients. Therefore, it might be better if clients are matched according to their capabilities in the first place.

Despite these obstacles, the 'Fastest Access Links' strategy still manages to enhance the download performance of the swarm. However, it is not equally beneficial to the ISPs. On the contrary, it increases the average link utilization by 6% and makes the utilization of the bottleneck link explode, as it is shown in Table 6.1. This is due to the fact that this policy imposes a large amount of traffic on the links which lead to the fastest peers.

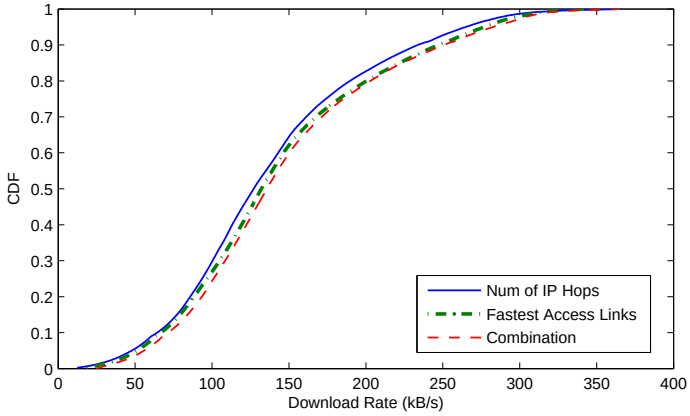


Figure 6.8: CDF of Download Rate for the 'Num of IP Hops' and 'Fastest Access Links' ranking strategies, as well as for their combination.

This is the point where one should consider the combination of strategies. The 'Num of IP Hops' strategy was of benefit to the ISP, while the 'Fastest Access Links' criterion improved the performance of the P2P application more. By combining them appropriately, one can form a composite criterion which satisfies both sides. Indeed, the composite policy managed to increase the average rate of BitTorrent clients by almost 7%. At the same time, the average link utilization decreased by 22%,

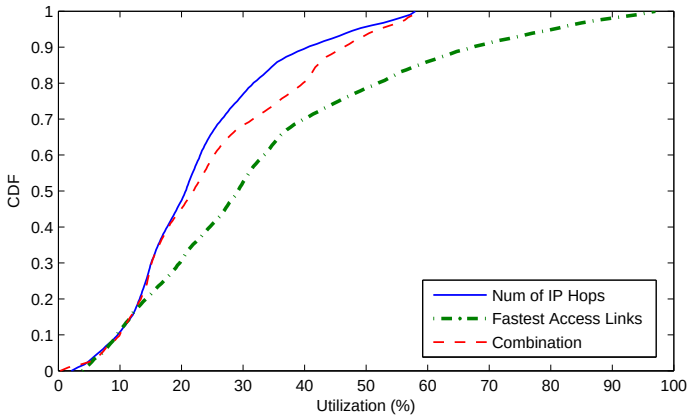


Figure 6.9: CDF of Utilization of intra-AS links for the 'Num of IP Hops' and 'Fastest Access Links' ranking strategies, as well as for their combination.

while the load of the bottleneck link was reduced by more than 36%. Hence, blending the two criteria provided a Win-Win solution for both cooperating parties. In addition, it ultimately guaranteed a download rate that is larger than that which was provided by the component strategies. This is due to the fact that our composite criterion considers both proximity and capacity and thus ends up recommending peers that are optimal in the context of both of these perspectives. Figure 6.8 plots the CDF of the download rates for the three ranking strategies which have been discussed so far. Figure 6.9 also plots the CDF of the utilization of all intra-AS links across the network for these three strategies. The presented benefits of the combined criterion are obvious.

The 'Matching Access Links' strategy prioritizes peers whose last-hop bandwidth matches that of the requesting peers. In this way, fast peers are matched with other fast peers and slow clients peer with other slow clients. As a result, load is distributed, bandwidth is not wasted and the tit-for-tat mechanism of BitTorrent is complemented. Although it does not benefit the ISPs, this ranking criterion was found to considerably improve the performance of BitTorrent. The average download time was found to increase by more than 8.5%. However, as it is illustrated in Figure 6.10, the benefit applies only to the fastest peers. This strategy allows them to blossom and reach the maximum of their potential. The weakest peers receive no benefit at all. On the contrary, some of them may experience worse performance, as they lose every chance they have to enjoy the merits of the connection with a faster peer (e.g. via an optimistic unchoking). This is in accordance with the observation that the median of the download rate slightly decreases. Hence, we can deduce that this criterion introduces unfairness to the system. Furthermore, this strategy has two disadvantages:

- It is specifically targeted to BitTorrent and other protocols which might employ tit-for-tat-ish policies. It does not have a general scope and it will not be of benefit in the other use cases of ALTO.
- It is not efficient in cases in which the swarm comprises a considerable number of fast seeds and a smaller number of slower leechers. In such a scenario, the system would benefit if the leechers tried to connect to the fast seeds, rather than peer with each other and form weak connections.

By combining the 'Num of IP Hops' and the 'Matching Access Links' policies, a new composite strategy can be formed. The combined criterion manages to increase download performance by 9%, while it also reduces the average link utilization by 6% and decreases the traffic on the bottleneck link by more than 16%. Hence, it achieves the best result in terms of the the QoE of the BitTorrent users. However, the benefit for the ISPs is not as large as that of the previous combined strategy that was presented. This is due to two reasons:

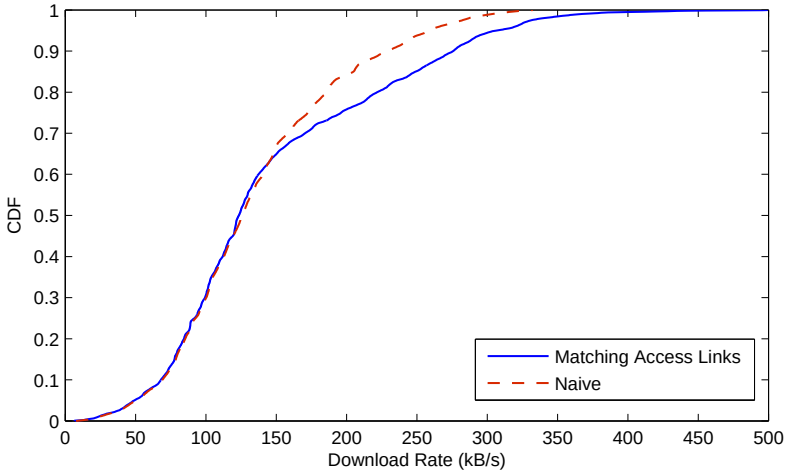


Figure 6.10: CDF of Download Rate for the 'Naive' and 'Matching Access Links' ranking strategies

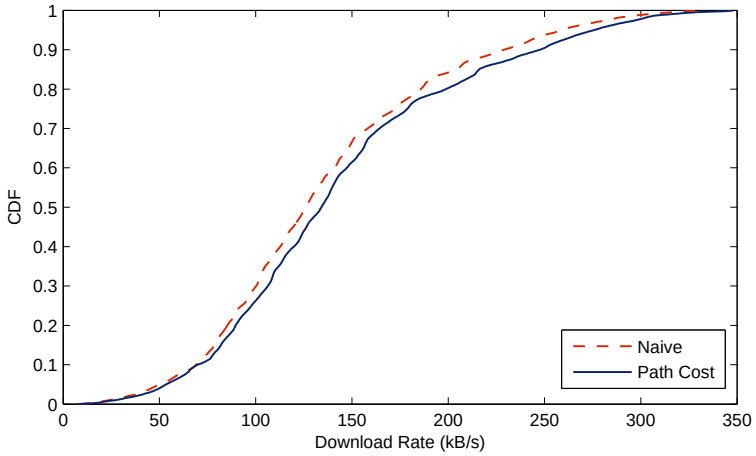


Figure 6.11: CDF of Download Rate for 'Naive' and 'Path Cost' ranking strategies .

- By providing the P2P application with the optimal overlay topology, it allows it to blossom and make the most use out of the available bandwidth.
- Finding a matching peer in the local vicinity may not always be possible. This means that the strategy may often have to sacrifice locality. As a remedy, the ISP may decide not to weigh the two sub-criteria equally and place more emphasis on the 'Number of IP Hops'.

The 'Path Cost' strategy is a strong technique that manages to consider both locality and bandwidth. When this strategy assigns a high rank to a particular peer, this means that the cost of the path leading to this peer is low. This can be either because the path contains a small number of links and/or because these links have a high capacity. Ultimately, the winning peers will be the ones which are located close to the requesting client and can be reached via the fastest possible links at the same time. This strategy manages to increase the average download rate by 6.4% and also reduce the average link utilization by 12.5%. Its advantage is that it does not only consider the quality of the access link, but also that of all the intermediate ones. However, it could not outmatch the combined 'Num of IP Hops - Fastest Access Links' criterion , as the latter allows the oracle to actively control the balance between the two different parameters: locality and bandwidth. Figure 6.11 demonstrates the benefit that the 'Path Cost' strategy introduces to the performance of the BitTorrent clients. One can see

that all the participants of the swarm experience this improvement. Hence, it constitutes a much more fair criterion than the 'Matching Access Links' one.

The 'Traffic Engineering' strategy is a good example of how the ALTO service can function as a means of efficiently controlling the traffic flow. It was the strategy that provided the best results in terms of improvement of link utilization, as it managed to reduce the traffic on the bottleneck link by more than 40%. However, it was also one of the most unstable criteria, as it provided very diverse results in distinct simulation runs (nearly doubling the variance of the measurements, when compared to the rest of the strategies). This is due to the fact that the utilization of the links is a dynamic parameter of the network that constantly changes. Thus, relying on measurements that may not be valid a moment after they were captured has the risk of providing wrong conclusions. Our experimental evidence practically justifies the fears of Alimi et al., which are against the application of such a strategy [11]. However, one should not totally disregard it, as we showed that it can achieve some excellent results for the ISPs. We argue that the strategy may be applied, but with great care and in conjunction with other ranking policies.

In addition,the 'Traffic Engineering' criterion manages to improve the download performance of the swarm by up to 5%. One would think that the P2P application would have benefitted more from the decrease in the traffic. Nevertheless, this is not the case, as the ISP usually has to recommend distant or slow peers if these can be reached via the least congested routes.

Finally, there is one more thing that must be noted. The calculated increase in the average download rate for all of these strategies is restricted by the size and the complexity of our simulated intra-AS topologies. Although these topologies were selected to be elaborate enough, they cannot obviously match the scale and the heterogeneity of real intra-AS domains. Consequently, the improvement which the experiments demonstrated would most probably be more significant in the real Internet.

Summarizing the results

We summarize the key conclusions that can be drawn from this study of the ranking strategies:

1. A ranking criterion may satisfy only one of the two cooperating parties (ISPs, P2P applications).
2. A ranking strategy may work best in a particular ALTO use case or cater for the needs of a specific protocol.
3. The best ranking criteria are the ones that manage to consider multiple parameters of the problem (e.g. locality, link capacity etc.).
4. The appropriate combination of strategies with the use of an algorithm such as the one which we proposed can optimize the ALTO guidance.

6.5 Weaknesses of the Oracle solution

This section attempts to expose some of the pitfalls and the weaknesses of the Oracle solution in its current form. The first part of the section discusses the inefficiency of the approach during the initial phase of a torrent's life, while the second part demonstrates the detrimental effects of the potential existence of hostile non-cooperating oracles.

Oracles and the Initial Phase of the life of a Torrent

When a new torrent is launched, the swarm typically consists of only one seed (the *initiator*), which owns the first replica of the shared file. The first clients which will join the swarm will have no pieces to contribute and they will have to heavily rely on the initiator to retrieve chunks of data. During this phase of the life of the torrent, it is important that the content is distributed to clients in all ASs as quickly as possible. Therefore, the system is bound to benefit from a random peer selection policy, which will guarantee that pieces of the file get scattered throughout the whole Internet fast enough. On the other hand, by biasing the construction of the overlay topology to encourage locality, the distribution of the pieces is restricted and the overall download performance of the swarm is harmed.

Simulations were conducted to measure the extent of the problem. For these experiments, it was assumed that the initial seed lied within AS6. With the beginning of the simulation, new clients joined the swarm without having pieces to contribute. This pattern of behavior can be characterized as a *flash-crowd*, following the terminology of Izal et al. [104]. Tests were run with and without the support of the oracles.

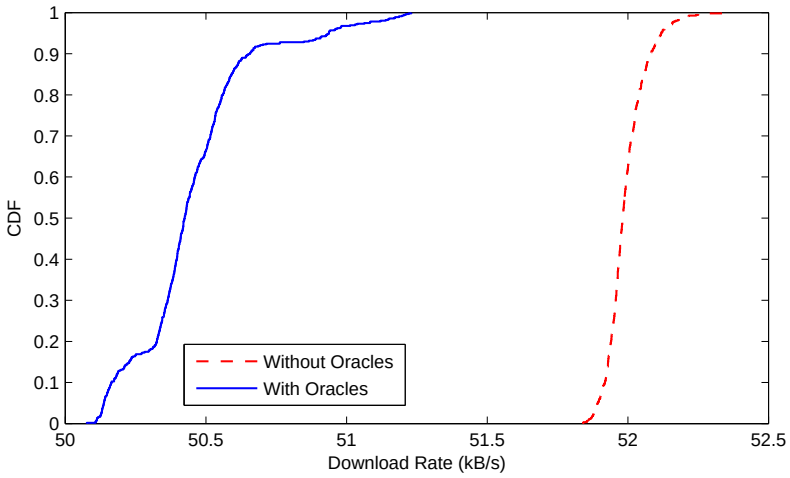


Figure 6.12: CDF of Download Rate in the initial phase of the life of the swarm.

The results showed that the introduction of the oracles still allowed the ISPs to greatly reduce the utilization of their links. The utilization of inter- and intra- AS links was found to decrease by 73% and 32% on average respectively. At the same time, the utilization of the most congested link of the network was reduced by 41%. This means that the results of the oracle solution are aligned with the interests of the ISPs, even in this scenario.

On the other hand, the Oracle approach was not found to be of benefit to the BitTorrent swarm. Figure 6.12 plots the CDF of the download rate of the clients. It can be seen that the introduction of the oracles did not improve the download performance. On the contrary, the average download rate of the whole swarm decreased by more than 3%. In addition, the coefficient of variation increased by 250%. This practically means that by using the guidance of the oracles, BitTorrent was not equally fair to all its clients anymore. Peers which were closer to the initiator received a much better treatment than the more distant peers. The distribution of the content strictly followed the patterns of the network topology; peers in ASs which were closer the the initiator completed the download of the file sooner than the peers in distant ASs. In other words, one could see the initiator as a 'transmitter' of content and the rest of the clients as 'receivers'; the content 'propagated' across the topology and local clients 'received' it faster than the distant ones.

This is obviously the result of the localized peering. A random peer selection would have allowed the tracker to 'pump up' the file and spawn new seeds in distant ASs more quickly. Obviously, the Oracle solution is not a Win-Win one in this case. The tracker may therefore decide not to employ it in this phase of the life of the swarm. Alternatively, it may consider that a small decrease in the performance (3%) is a negligible sacrifice that it is willing to make for the greater good (which is the reduction of congestion in the whole network in this case).

Malicious Oracles

Firstly, we investigate the case in which greedy ISPs abuse the oracles so as to promote their financial interests. In order to maximize its financial revenue, an ISP can have its Oracle provide peer recommendations which increase the traffic load on the transit links to its customer ASs. This behavior will boost up the utilization of these links and it may satisfy the incentives of the provider ISPs. However, it will also cause a series of problems to the other parties of the system. To investigate this problems, we conduct experiments with oracles which adopt such a ranking strategy. More specifically, we have the oracles of AS1, AS2, AS3 and AS5 (i.e. the ASs of our topology which are providers of other ASs) rank the peers so that they push as much traffic to their customers as possible.

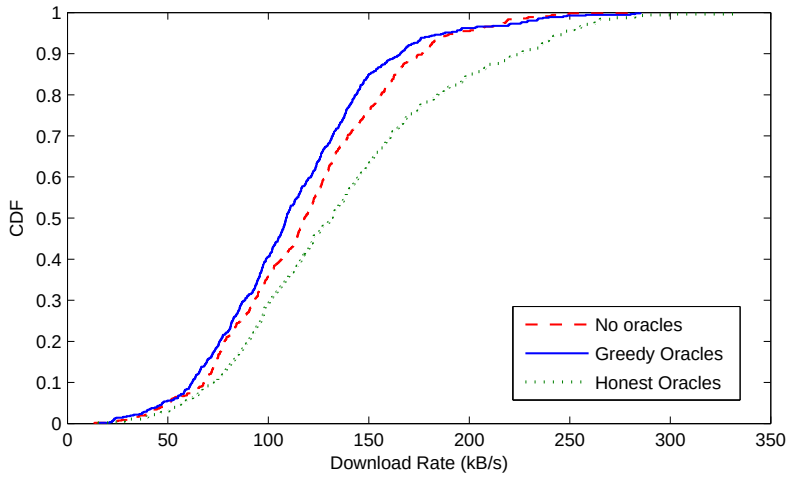


Figure 6.13: CDF of Download Rate for the scenario of revenue-driven oracles.

Firstly, as Figure 6.13 demonstrates, the download performance of the BitTorrent swarm does not benefit from the existence of these greedy oracles. On the contrary, the average download rate is reduced by 6.5% and the median rate by 9%, when compared to the scenario without oracles. This was to be expected, as BitTorrent clients are not able to fully enjoy the benefits of locality anymore. Only a small fraction of the clients manage to retain their high download rates and these are the clients in ASs which do not have customers (such as AS4).

Figure 6.14 depicts the utilization of the inter-AS links. As it can be seen, the utilization of these links is larger than it would be if the oracles were sincere and sometimes it is larger than the utilization in the scenario without oracles. To an extent, the negative influence of the greedy oracles in the provider ASs is moderated by the good behavior of the oracles of the customer ASs, but still in many cases the providers manage to make the utilization of the transit links explode (e.g. the link between AS2 and AS4). This may satisfy the provider ASs, but it is obviously not of benefit to the customer ASs. The problem is bound to scale when ASs which are victims of their providers become perpetrators in their turn and adopt similar strategies in order to extract money from their own customers. This will become a vicious circle, with the bottom-tier ASs being the most harmed ones (in financial terms).

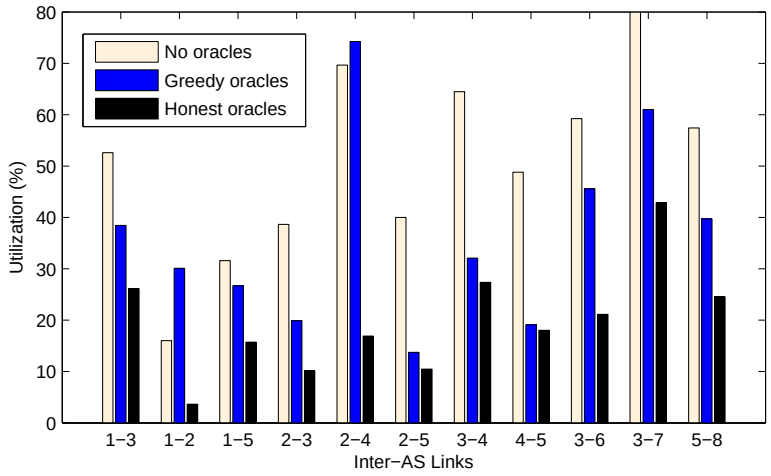


Figure 6.14: Utilization of inter-AS links in the presence of revenue-driven oracles.

On the whole, the existence of revenue-driven oracles threatens the success of the approach; thus, such oracles should be detected by the P2P applications and be isolated. But this can be one of the many types of dishonest oracles that may exist. As it was discussed in section 3.2.1, a variety of other malicious ranking strategies may be employed by the oracles. To demonstrate how this problem can scale, we consider that the oracles of our network adopt a variety of different malicious ranking criteria. Others generate DoS attacks, others deliberately recommend peers at distant ASs and others stick to promoting the financial interests of their ISPs. We run multiple experiments in a system with a variable number of swarms and study the results.

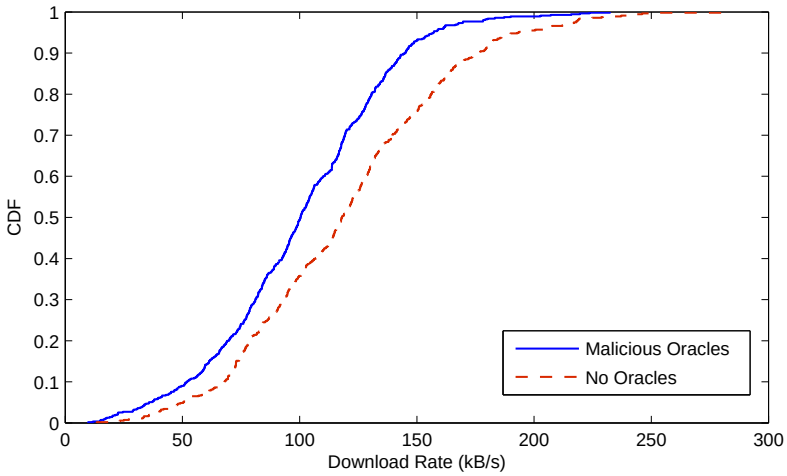


Figure 6.15: CDF of Download Rate when in the presence of malicious oracles.

Again, as Figure 6.15 demonstrates, the presence of this mosaic of malicious oracles has a detrimental effect on the performance of the BitTorrent application. The ALTO service led to a reduction of 19% in both the average and the median download rate of the clients. Consequently, it would have been better if the tracker had not considered the recommendations of the oracles and had remained loyal to its traditional random peer selection strategy. This observation highlights the imperative need for an introduction of a detection strategy to the system, which will be able to pinpoint and block oracles of this type.

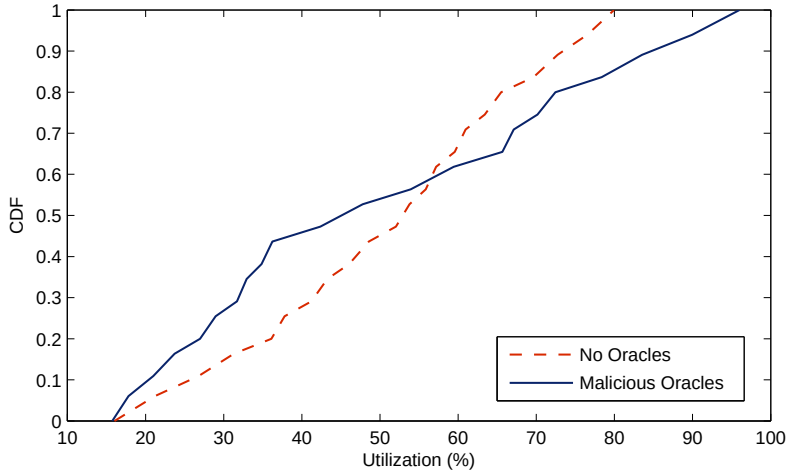


Figure 6.16: CDF of Utilization of inter-AS links in the presence of malicious oracles.

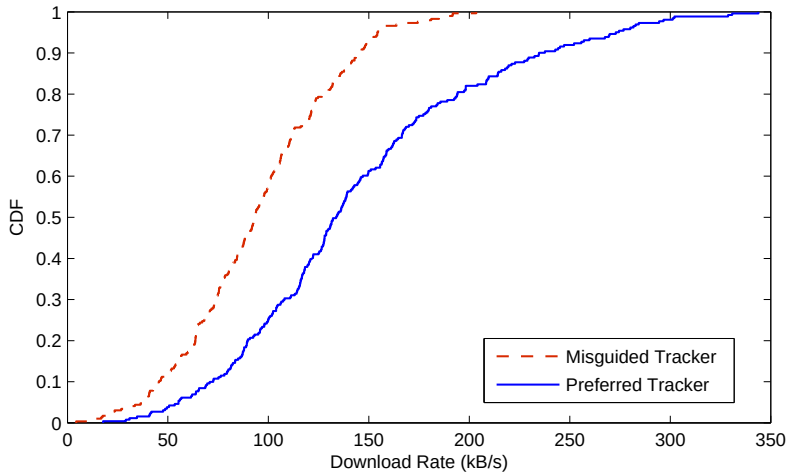


Figure 6.17: CDF of Download Rate when using oracles to express preferences among trackers.

Besides, the existence of malicious oracles also threatens the balance of the network as a whole. Figure 6.16 show that this hostile behavior augments the traffic load on the bottleneck inter-AS links. Congested links become even more loaded and under-utilized links become even less utilized. Thus, the balance of the traffic flow is jeopardized. The situation is quite similar for the intra-AS links. As a result of the hostile ranking criteria, the traffic on top of the most utilized intra-AS link increased by 7% on average. In some cases, this increase was seen to climb up to 20%. An obvious question could be why an ISP would operate an Oracle service which creates congestion problems instead of solving them. There are several answers to this question:

1. The ISP could benefit from this congestion on the basis of certain charging policies, as it has been already discussed.

2. The Oracle may be providing inefficient recommendations despite its will. For instance, it might have been 'hacked' by an intruder, which forces it to employ malicious ranking strategies.

3. The ISP might be obliged to misguide specific trackers, such as those which are known to manage the unauthorized distribution of copyright content.

4. The malicious ranking strategy causes congestion problems to distant ASs and not to the local one. If the local ISP is totally self-centered, it will not care for these problems.

We examine the case in which an ISP uses the oracle to express preference in specific trackers and misguide others. Experiments showed that, by employing different ranking strategies, an oracle can increase the average download rate of a swarm by 22% and simultaneously decrease the average download rate of another swarm by 25%. This means that a malicious oracle has the ability to make the performance of a preferred swarm 60% higher than that of a swarm which has fallen into disgrace (Figure 6.17).

6.6 Discussion

Our experiments demonstrated that indeed the collaboration between the P2P applications and the ISPs can be of benefit to both sides. Even with the use of a trivial ranking criterion that simply segregates the candidate neighbors into interior and exterior ones, the ISP manages to localize the traffic within its network and offer a better Quality of Service to its customers. At the same time, P2P applications can also enjoy a better Quality of Experience by seeing their download rates increase significantly.

This improvement can be augmented with the use of additional ranking policies that allow for a fine-grained evaluation of intra-AS peers. However, we showed that not all of these policies are of equal benefit to both parties. Some work better for the ISPs, while others are more beneficial to the applications. Thus, an oracle should select its ranking strategy with great care, so that it balances the gainings of each side. Fortunately, the coalescence of individual criteria into elaborate ranking policies has proven to be an answer to this problem, since the composite strategies were shown to yield the benefits of their components.

Another interesting observation was that strategies which appeared to be beneficial in other studies were not seen to provide as good results in our system. As it has been explained, one reason for this is that some of the older studies failed to consider the heterogeneity of the topology and the general network conditions. Another reason is that by default a strategy cannot provide the same efficiency in all different scenarios and for all possible P2P applications. Indeed, we discussed that there are strategies which worked well for our BitTorrent-based protocol, but they may not be of benefit in other situations. To sum up, these results underscore the fact that the system should support the following two characteristics:

1. The oracle should be capable of implementing more than one ranking policies and should be trained to combine them appropriately.

2. The oracle should be able to tailor its recommendations to the needs of the specific application. This can be achieved either by letting the application express a preference on the ranking criterion, or by educating the oracle to be able to decide which strategy is the best fit for each P2P protocol.

Finally, the experiments demonstrated two weaknesses of the Oracle approach. Firstly, it is not a Win-Win solution during the initial phase of the lifetime of the swarm. Secondly, and most importantly, it is susceptible to the detrimental influence of hostile oracles. Not only does the existence of such oracles not improve the download performance of the P2P application, but it may also reduce it significantly. Moreover, although it may not be obvious at first sight, the malicious ALTO guidance causes problems to the ISPs as well (as one ISP harasses the other) and the balance of the whole network is threatened. Therefore, the need for a mechanism that will shield the system from these problems is imperative.

Chapter 7

Detection of Malicious Oracles

The previous chapter illustrated how the existence of malicious oracles can become a pitfall of the Oracle approach. So far, little work has been done towards the direction of trying to alleviate this problem. One of the most important contributions of this project is the design and the evaluation of strategies that can be employed by the Oracle (ALTO) clients and can shield them against insincere oracles. In this way, the project attempts to fill part of the gap in the existing bibliography and contribute to the improvement of the ALTO solutions that have been proposed. This chapter is dedicated to a detailed presentation of these strategies and the choices which governed their design. The first section describes the first detection strategy and the second one suggests enhancements or alternate versions of it. The third section describes a second detection strategy and the fourth one suggests how the first strategy can be adapted so that it can be utilized in an environment with multiple oracles per AS.

7.1 Between-Client Detection Strategy

This section describes the first strategy which can be used by the tracker of the distributed system of Chapter 3 for the detection of malicious oracles. The basic principle behind this strategy is the comparison between the download performance of clients receiving oracle-guided peer recommendations and the download performance of clients receiving random peer recommendations. As it is based on a comparison between groups of different clients, this algorithm will be subsequently referred to as the *Between-Client* Detection Strategy. Let us suppose that our goal is to determine the quality of the recommendations of the oracle of the Autonomous System X. The detection will be based on the performance of the BitTorrent clients which belong to this particular Autonomous System X. From now on, the term *clients* will be used to refer only to those clients located within AS X.

The *Detection Phase* begins at time t_0 and lasts for $t_{duration}$ sec. The clients which send a *Tracker Request* message to the tracker during this time period ($[t_0, t_0 + t_{duration}]$) are divided into two groups. The clients of the first group (Group 1) receive Tracker Response messages containing oracle-guided peer recommendations. The clients of the second group (Group 2) receive *Tracker Response* messages containing peers selected randomly by the tracker. The division into the two groups is done so that p% of the total clients belong to Group 1 and $(100 - p)$% of them belong to Group 2, where p is an appropriate percentage. The tracker must keep track of which group each client has been assigned to.

Every time the tracker receives a *Tracker Request* message during the Detection Phase, it follows the following steps:

1. Firstly, it checks whether the client has already been assigned to one of the two groups. If the

client has already sent a request during the Detection Phase, then it will have already received a classification. Otherwise, it will have to be classified to one of the two groups. More precisely, it is assigned to Group 1 with probability $p\%$ and to Group 2 with probability $(100 - p)\%$.

2. If the client belongs to Group 1, then the tracker consults the oracle and asks for a ranking of the candidate peers with respect to the client's IP address. Upon receiving a reply from the oracle, the tracker sends a *Tracker Response* message to the client containing the D highest ranking peers, where D is the number of peers included in the response. If the client belongs to Group 2, then it must receive a list of random peers. Thus, the tracker does not need to consult the oracle, but just picks D peers at random and embeds them in the response message.

So far, the tracker has ensured that a portion of the clients gets oracle-recommended peers and the rest of them get random peers. Now, all that remains to be done is to compare the download performance of the two Groups; this will allow for the evaluation of the oracle. If Group 1 gets better download rates than Group 2, this means that the Oracle recommendations are better than random and thus the swarm can benefit from the use of the Oracle. If Group 2 gets better download rates than Group 1, then the tracker can deduce that the oracle is malicious. The whole rationale behind the algorithm is illustrated in figure 7.1.

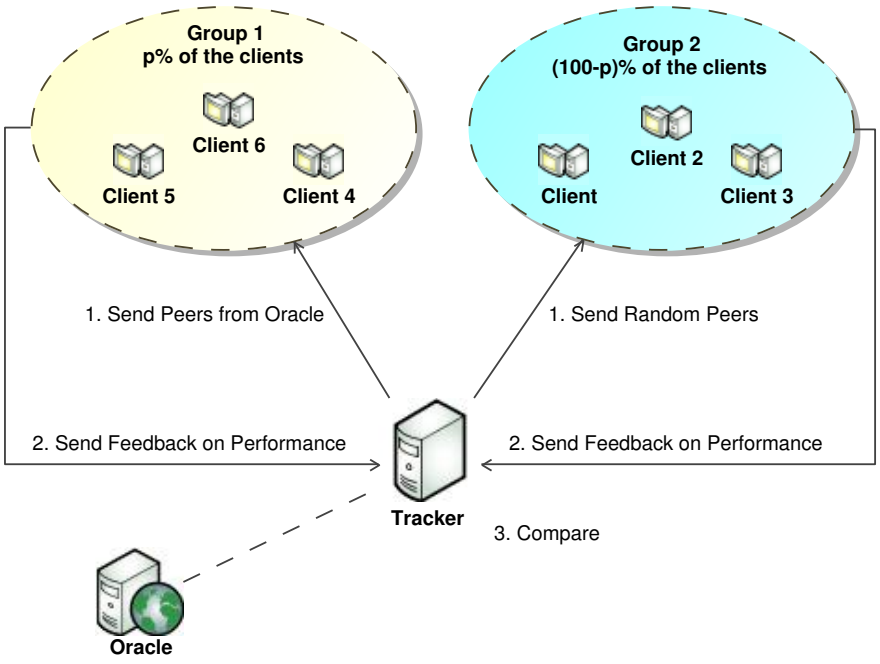


Figure 7.1: Between-Client Detection Strategy

In order to measure the download performance of the two groups, the tracker must collect metrics from the clients. As it has been discussed in Chapter 3, metrics of this type are already included in the request messages sent from the clients and they help the tracker keep statistics of the swarm. The only thing that the clients need to contribute to the Between-Client Strategy is to calculate the download rate they experience and embed it in the *Tracker Request* messages which they send to the tracker. The latter stores the download rate values which are sent during the Detection Phase and will base the evaluation of the oracle upon them. The tracker should ignore the download rate value contained in the very first *Tracker Request* message which a client sends during the Detection Phase. This is because this download rate is the result of the peer recommendation that the client received before the beginning of the Detection Phase.

When a suitable termination condition is met, the Detection Phase stops. This happens at time $t_0 + t_{duration}$, meaning that the detection lasted for $t_{duration}$ seconds. At this point, the tracker gathers all the metrics that it has collected from the clients during the detection phase and combines them to draw a conclusion on the sincerity of the oracle. Firstly, the tracker averages the rates of Group 1 and and Group 2 separately. If the average rate of Group 1 is larger than that of Group 2, the oracle is characterized as sincere. The tracker decides to trust it and consults it for all subsequent peer recommendations. If the average rate of Group 1 is lower than that of Group 2, the oracle is characterized as malicious. The tracker blocks the oracle, stops consulting it and proceeds with including randomly selected peers to all the *Tracker Response* messages. A new Detection Phase may be initiated T_{period} sec after the termination of the previous one. During this time period, the protocol behaves in its normal traditional fashion, with the oracle either completely blocked or completely trusted, according to the outcome of the detection.

As it will be shown in the following chapter,the Between-Client Detection Strategy manages to improve the performance of the whole swarm and it also contributes to the well being of the whole network.

During the time period ($[t_0,\ t_0 + t_{duration}]$), p% of the clients received oracle-guided recommendations and $(100-p)$% of the clients received randomly selected peers. If the tracker had not used the Detection Strategy and had chosen to trust the Oracle from the beginning, then 100% of the peers (i.e. the whole set) would have gotten oracle-guided recommendations during this same period. This practically means that if the oracle was a sincere one after all, the Between-Client Detection strategy made $(100 - p)$% of the clients perform worse than they would have performed if no detection was conducted during this time period. This is due to the fact that these clients received worse peers than they would have received in a scenario without detection and complete trust on the oracle. This degradation in the performance of these peers for $t_{duration}$ seconds is a necessary sacrifice that has to be made for the application of the suggested detection strategy.

The download performance of the clients during the time period ($[t_0, t_0+t_{duration}+T_{period}]$) i.e. between the end of the Detection Phase and the beginning of the next one depends on whether the strategy ultimately managed to correctly assess the intentions of the oracle.

1. If the oracle was a sincere one and this was identified by the tracker, then the clients will still perform as well as they would perform if no detection was conducted.

2. If the oracle was sincere and the tracker mistook it for a malicious one, it will be blocked although it should not be. This will negatively affect the performance of the clients, compared to how they would have performed if no detection was carried out. This means that it must be ensured that the detection strategy is accurate enough and does not make such false positive classifications.

3. If the oracle was a malicious one but this was not detected by the tracker, then the experience of the clients will be the same as it would have been if the detection strategy was not applied.

4. If the oracle was malicious and this was identified by the tracker, then the download performance of the clients will improve thanks to the detection strategy.

As it will be demonstrated in Chapter 8, the strategy is accurate enough and therefore the first and the fourth of the above scenarios are far more probable than the second and the third. This is the reason why the strategy manages to accomplish its purpose: the improvement of the performance of the overall system

The termination condition of the Detection Phase was deliberately not specified, since a number of different choices can be considered. The tracker may choose to terminate the detection when a certain time interval expires. Alternatively, it may select to stop detecting as soon as it considers that it has collected enough data to allow for an accurate decision. These termination conditions are actually

related to each other, as detecting for a longer time results in the accumulation of more data.

At this point, it must be noted that the larger the number of clients in the AS, the less long the Detection Phase needs to be. For small ASs with very few BitTorrent clients, the tracker will have to detect for a longer time in order to gather enough data that will enable a sufficiently accurate verdict. However, the detection phase should not be excessively prolonged, as it is also important that a decision is made as soon as possible. As it was already discussed, if the oracle is sincere, the detection comes with a cost in the performance of a portion of the clients. In addition, it is important that the decision is made soon so that the system can benefit the most from it (providing that it is accurate). For these last two reasons, the tracker should not be stuck in the detection phase for a long time. The tradeoff is obvious. If you detect for too long, you will maximize the accuracy of the outcome but you will sacrifice performance. If you terminate the detection early on, you will make the most out of the decision, but it is more probable that it will be an inaccurate one.

The percentage p is a very important parameter of the algorithm, so its selection should be conducted with great care and with respect to the selected termination condition. One choice is to have p be equal to 50. This practically means that it is equally probable for a new client entering the swarm during the Detection Phase to be assigned to any of the two groups. In such a case, when the detection terminates, the tracker will have two equally sized sets of data, one corresponding to oracle based recommendations and the other one related to random peer selections. This unbiased treatment of the two groups will maximize the probability that the detection decision is accurate.

On the other hand, if p is too large, only a small portion of the clients will receive random peer selections. This is bound to cause problems. For illustration, let us suppose that p is set to be equal to 95, meaning that only 5% of the clients will be assigned to Group 2. If the termination condition is the expiration of a time interval, the tracker will not have managed to collect a safe amount of data on the performance of Group 2 by the time the detection finishes. Chances are that these 5% of the clients which were assigned to Group 2 might not have been representative of the whole set of clients. This will eventually lead to a wrong decision. Now, let us consider that the termination condition is the accumulation of a sufficient amount of data,. This condition will ensure that enough information on the performance of Group 2 will also be collected. However, with only 5% of the clients being assigned to it, it will take an excessively large amount of time to gather this data. This will prolong the detection phase too much, with the negative effects that have already been discussed. Similarly, if p is too small, analogous problems will occur, this time for Group 1.

Setting p to be equal to 50 seems to be the best choice at this point. However, in many cases the use of a biased value for p may actually prove to help when used in conjunction with a termination condition that stops the detection phase when a certain time interval T_{exp} expires. On the basis of prior knowledge, the tracker may have reasons to believe that a certain oracle is a sincere one. For instance, it may have been behaving appropriately for a long time or it may be managed by a trusted party. For a detection phase lasting for T_{exp} sec, the tracker may choose to use a p that is larger than 50, but not too large to cause the problems mentioned in the previous paragraph. This helps because it reduces the percentage of the peers which are negatively affected by the detection phase, as it was explained earlier (provided that the oracle is indeed sincere).

For example, p can be set to 70. This means that 30% of the clients will get random peer recommendations and experience worse download performance than they would have if no detection was conducted. If p were 50, this problem would have been experienced by 50% of the clients. By reducing this percentage from 50% to 30%, this demerit of the strategy becomes less apparent. Nevertheless, this will also mean that at the end of the detection the tracker will have less data on the performance of Group 2. This might result in a small reduction in the accuracy of the technique. However, 30% is still a good percentage and it should not have a detrimental effect on the quality of the information

collected, which should still be representative enough.

In a similar way, if the tracker has reasons to believe that a particular oracle is malicious (e.g. it has behaved inappropriately in the past or it is run by a suspicious party), then p can be set to a value lower than 50 (for example 30). This will increase the number of clients getting random recommendations during the detection phase. As these clients will experience better download times than those receiving the malicious service of the oracle, there will be a considerable benefit (provided that the oracle was indeed malicious).

Finally, we proceed to discuss an attack that a malicious oracle may execute in order to trick the detection scheme into drawing wrong conclusions. We also provide 3 remedies which can shield the tracker against this attack:

Attack: The oracle infers the time point at which the tracker begins its detection. To achieve this, it needs to keep track of a certain metric: the number of queries sent by that tracker per second. Changes in this number may indicate the beginning of a new detection with very high probability. For instance, when a tracker starts the assessment of a blocked oracle, the value of this metric rises. Similarly, when the tracker decides to re-evaluate a trusted oracle, this value decreases. As soon as the oracle determines that a new detection has begun, it can pretend to act honestly until it terminates. Again, the end of the detection can be identified by a change in the number of received queries. When this number is seen to increase, the oracle can deduce that the tracker has finished detecting and has been fooled into believing that the oracle is sincere. The oracle can know alter its ranking strategy and start sending bad recommendations. The tracker has chosen to completely trust this guidance and thus has fallen into the oracle's trap.

Remedies:

1. The tracker can send dummy queries to the oracle in order to let it jump to wrong conclusions with respect to the beginning and the end of the detection phase. For instance, close to the end of the detection the tracker can start sending queries for all its requesting clients without necessarily making use of the corresponding responses. This will lead the oracle to falsely deduce that the detection has finished. Meanwhile, the tracker can continue detecting as usual.
2. The tracker can compare the peer recommendations that the oracle provided for the same client during and after the detection. These recommendations should not differ significantly. If they do, the tracker may suspect that there has been a change in the behavior of the oracle since the last detection.
3. The tracker can have a small percentage of clients always receive random recommendations. This is the main idea behind the 'Sliding Percentage' variation which will be described in Section 7.2.

7.2 Alternatives of the Between-Client Strategy

This section provides a number of variations of the Between-Client Detection Strategy and some enhancements that can be used to improve its performance.

A sliding percentage of oracle-aided clients

Completely blocking or trusting an oracle may not always be the best approach. The behavior of the oracle (i.e. the ranking criteria it employs) is dynamic and may change at any point during the time period between the end of a detection phase and the beginning of the following one. The Between-Client

Strategy, as it was presented so far, will capture that change only at the end of the next detection. Meanwhile, the verdict of the previous detection will influence the peer recommendations sent to the whole set of peers in the AS and as this verdict ceases to be an informed and accurate one, it will cause significant harm to the the performance of the system.

As an example, let us consider the case in which a detection phase terminates, with the tracker accurately deducing that the oracle is malicious. According to the Between-Client Detection Strategy, the oracle is blocked and all peers are granted with randomly selected peers, at least until the end of the next detection. Let us also suppose that the oracle changes its ranking criterion and starts behaving sincerely a moment after the tracker decides to block it. The tracker will have to wait for the next detection phase i.e. for almost T_{period} seconds to release the block imposed on the oracle and start benefiting from its new strategy. In this way, a lot of time will be lost and a chance for an improvement in the performance of the system will be missed.

To alleviate the problem, the strategy can be reviewed and adapted. In this new variation, the detection never stops. In addition, an oracle is never totally blocked or completely trusted. The tracker always has p% of the peers receive oracle based recommendations and the rest of them get random ones. The initial value for p can be selected according to criteria analogous to those discussed in the previous section. Once again, the tracker collects information on the performance of clients of the two categories. When a certain time interval expires, the accumulated data is used to compare the average download rate of the two groups. If the oracle-guided clients perform better, p is incremented by a value k and becomes $p + k$. Otherwise, p is decremented by k and becomes $p - k$. This practically means that, as long as the oracle is sincere, the number of clients using its recommendations increases. If the oracle acts in a malicious way, the number of clients receiving recommendations from it decreases.

The percentage p has a maximum allowed value p_{max} and it cannot increase further than that. p_{max} is smaller than 100 and this indicates that a small number of clients in the AS will always receive random peer recommendations. Similarly, p also has a minimum allowed value p_{min}. p_{min} is larger than 0 and this indicates that a small number of clients in the AS will always make use of the service of the oracle. Hence, the tracker actually never ceases to detect and it always has the opportunity to evaluate the current behavior of the oracle. This allows for a prompt discovery of any change of the behavior of the oracle and the timely reaction to such a change. When p is very small (close to 0) or very large (close to 100), the tracker may wish to increase the duration of the time intervals between two successive assessments of the performance of the system, so that it is ensured that the new assessment of the group corresponding to the small percentage is based on a sufficiently large amount of data. On the other hand, this might not be necessary because a potentially inaccurate assessment caused by the small size of the sample will quickly be corrected.

The weakness of this strategy lies on the fact that the swarm will always have a small percentage of peers performing worse than the majority. As it will be demonstrated in Chapter 8, it works well if the oracle changes its behavior very frequently. Otherwise, the traditional Between-Client strategy should be preferred.

Adjusting the interval between successive detections

As it was described in the Section 7.1, the tracker uses the outcome of the detection in order to decide whether to block or trust a specific oracle. The tracker may choose not to re-detect and adhere to its decision. However, it is advisable that the Detection Phase is repeated periodically for two reasons. Firstly, the oracle may have altered its ranking strategy and the quality of its recommendations may have changed; thus, a malicious oracle might have decided to behave better and a previously sincere oracle may have decided to employ an obnoxious strategy. Secondly, the outcome of the original

detection might not have been an accurate one and the tracker might have blocked (trusted) a sincere (malicious) oracle. In both cases, a repetition of the detection phase is required so that the quality of the oracle is reassessed.

The time interval T_{period} between the end of one detection phase and the beginning of the following one may be selected to be constant. Nevertheless, if the tracker observes that numerous successive detections produce the same outcome, it might decide to run the strategy less frequently and thus increase this time interval. Detecting places additional computational overhead on the tracker and may also sacrifice the good download performance of a portion of clients. Thus, the tracker would prefer to minimize the conducted detections, especially if their outcome is a predictable one.

An example will be used to illustrate the idea described. Let us consider that the tracker completes the first Detection Phase and finds an oracle to be malicious. After T_{period} sec, a new detection is carried out and the oracle is again found to be malicious. Four more detection phases are completed and the oracle is found to be malicious in all of them. The tracker can decide that evaluating the oracle so frequently is futile and may proceed to increase the time interval between two successive evaluations to $2T_{period}$ sec. The algorithm may proceed in the same fashion, with the tracker increasing the time interval as long as the result of the detections remains the same, until it reaches a maximum value. If the outcome of a detection phase differs from that of the one preceding it, then the interval must be reset to its initial value. The allowed successive values of the interval may form an arithmetic progression with common difference T_{period} (T_{period}, $2T_{period}$, $3T_{period}$, $4T_{period}$, ...) or a geometrical progression with common ratio r (T_{period}, rT_{period}, r^2T_{period}, r^3T_{period}, ...). Algorithm 2 illustrates this strategy in its general form.

Algorithm 2 Adjusting the interval between successive detections

```
interval ← period
counter ← 0
previousOutcome ← −1

loop
  Perform a Detection Cycle and determine the outcome ∈ {0, 1}
  if outcome = previousOutcome then
    if interval < maxInterval then
      counter ← counter + 1
      if counter = N then
        counter ← 0
        Increase interval
      end if
    end if
  else
    counter ← 0
    interval ← period
  end if
  previousOutcome ← outcome
  Wait for interval sec
end loop
```

Warning/Forgiving the Oracle

It can be argued that the Between-Client Strategy is very unforgiving. As soon as it discovers that the other party (the oracle) is defecting (i.e. it is behaving maliciously), it retaliates by completely blocking it and not cooperating at all until the next scheduled Detection Phase. In the worst case, the tracker may choose not to re-detect at all and abide by its original decision. This will give the strategy a *Grim Trigger* character, ousting any opportunity for any future cooperation. Moreover, the oracle will not find out that the tracker is not satisfied with the provided service until it is too late and the block has been imposed.

An alternate approach is to have the tracker perform a number of shorter adjacent detections. If the outcome of a detection indicates that the oracle is malicious, the tracker can send a Warning message to the oracle in order to express its dissatisfaction. The communicated complaint may make the oracle reconsider its behavior and alter its ranking strategy. This change will be sensed by the tracker in the subsequent detections and ultimately the oracle will not have to be blocked. Otherwise, if the oracle continues to misbehave, the tracker can proceed to impose a block on it. Algorithm 3 illustrates how this variation of the Between-Client Strategy works.

Algorithm 3 Warning/Forgiving the Oracle

```
warningLevel ← 0

loop
  Conduct a detection sub-phase
  if the oracle is found to be malicious then
    warningLevel = warningLevel + 1
    if warningLevel = f then
      Block the oracle
      return
    else
      Send Warning message to the oracle
    end if
  else
    warningLevel = warningLevel − 1
    if warningLevel = −f then
      Place a trust on the oracle
      return
    end if
  end if
end loop
```

As it can be seen, the tracker may forgive the oracle up to $f - 1$ successive times. In other words, the oracle has at most $f - 1$ opportunities to correct its behavior. The value of f must be selected with great care. A too small f will lead to fast and possibly inaccurate conclusions. A very large f will prolong the total detection phase excessively and will harm performance.

The tracker can send a warning to the oracle via a new type of message which can be added to the existing Oracle protocol. Defining the exact format of the header and the payload of the message is out of the scope of this project. Alternatively, the warning can be communicated by embedding it within the oracle *Query messages*. This can be achieved, for example, by using some of the unused flags contained in the header of the messages whose format is defined by Akonjang et al. [36] in their Oracle Protocol Specification. The level of warning (i.e. the value of variable *warningLevel* in the algorithm)

can be included in the warning message, as an indication of how much unsatisfied the tracker is and how close it is to blocking the oracle. It is up to the oracle to decide how to react to a warning it receives.

This particular technique is susceptible to another type of attack from a hostile oracle. We give a description of it and suggest a possible remedy that can shield the tracker against it:

Attack: The oracle begins by sending unfruitful recommendations to the tracker. The tracker detects this behavior and sends a warning. The oracle switches to an honest ranking strategy. The tracker detects this improvement and recalls the warning. The oracle then returns to its original malicious policy. It can proceed in the same manner, switching between a hostile and a sincere behavior, while the tracker gets stack is a series of alternating warnings and recalls, without ever reaching a verdict.

Remedy: The tracker sets an upper limit in the maximum number of alterations of behavior that it can tolerate. If this threshold is exceeded, the oracle is considered to be malicious and is blocked. Alternatively, a limit can be placed on the maximum time that can be dedicated to detection. If this limit is surpassed, the oracle is again blocked.

Utilizing the clients' bandwidth

One of the first steps that the tracker has to perform is to decide whether each client will receive oracle-guided or random peer recommendations. In the algorithm that was described in the previous section, the clients were assigned to one of the two groups, with probability $p\%$ and $(100-p)\%$ respectively. This assignment was completely uninformed, oblivious of the network characteristics of the clients. Chances are that ultimately the two groups are not representative of the whole set of clients in the AS and do not capture its heterogeneity. For instance, Group 1 may end up with the faster clients or the clients that can be reached through routes with very fast links and Group 2 may mostly comprise clients which have slow access links or can be reached only via congested routes. Averaging the download performance of the clients of each group will not provide an accurate indication of the quality of the corresponding peer selection method. For the example given above, Group 1 will be seen to perform much better than Group 2. This will lead the tracker to deduce that the oracle is a sincere one, although it may in fact be malicious. The problem is bound to scale if $p \neq 50$. In this case, fewer peers will be assigned to one of the two groups and the synthesis of this group may end up being misleading.

The problem can be partly alleviated by making use of the value of the bandwidth of the clients. In most modern BitTorrent clients, the user is prompted to supply the (upload) speed of his/her line, as it is used by the software to configure parameters such as the number of upload slots or the maximum number of connections per torrent. For example, the uTorrent client [61] asks for this value during the Set Up phase. If this value is embedded within a *Tracker Request* message and sent to the tracker, it will help the latter counter-attack the problem described in the previous paragraph. Having knowledge of the clients' bandwidth, the tracker will be able to make assignments which guarantee that both groups comprise clients of all types. Alternatively, the values of the bandwidth can be used to normalize the values of the download rate. The download rate of each client can be divided by the corresponding bandwidth and the tracker can average these ratios, instead of averaging the rates.

Unfortunately, this practice cannot fully solve the problem. This is because judging a client solely by its bandwidth may lead to wrong conclusions. Other parameters also influence its download performance such as the congestion of the links leading to it or the number of active torrents running simultaneously on it. Obviously, the tracker does not have access to this type of information.

7.3 Within-Client Detection Strategy

This section discusses a different detection strategy, which attempts to nullify some of the weaknesses of the first one. The basic principle behind this second algorithm is the inspection of how one client performs for each of the two different types of peer recommendations. A client receives both random peers and peers suggested by the oracle. By determining which of the two groups of peers worked best for this client, the tracker has an indication of the intentions of the oracle. Conducting this experiment on multiple clients allows the tracker to finally draw an accurate conclusion. As this detection scheme is based on the assessment of how the two types of recommendations perform within the same client, it will subsequently be referred to as the *Within-Client Detection Strategy*.

During the Detection Phase, the clients are split into two groups. Group 1 contains $s\%$ of the clients and these are the ones which will be used for the evaluation of the oracle. In a sense, these clients can be seen as test subjects (or 'guinea pigs'), since the tracker will be experimenting on them. The first time the tracker receives a *Tracker Request* message from a particular client during the detection phase, it assigns this client to one of the two groups. With probability $s\%$, the client will be assigned to Group 1 and will become a test subject.

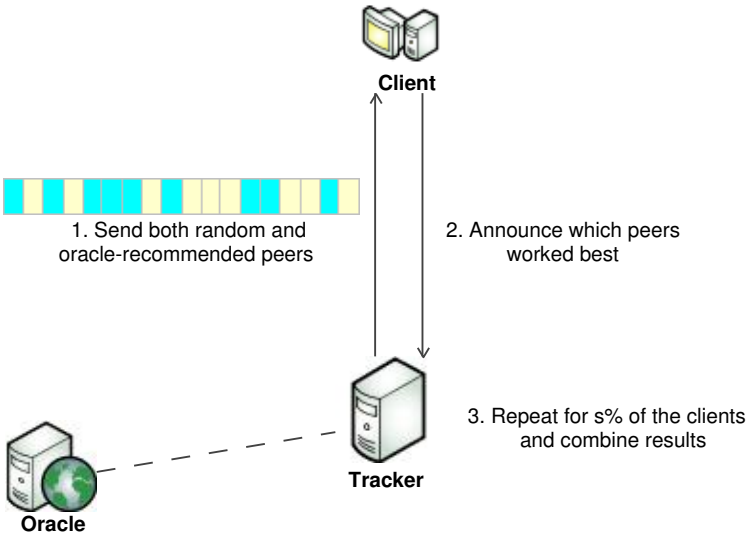


Figure 7.2: Within-Client Detection Strategy

Clients of Group 1 receive *Tracker Response* messages which contain both random peers and peers recommended by the oracle. More precisely, 50% of the D peers contained in the response are selected at random and the rest 50% of them are proposals of the oracle. A client attempts connections to peers from both categories and keeps track of the download rate experienced for each connection. This functionality is already part of the BitTorrent protocol, as it is required by the Choke/Unchoke algorithm. When the client re-announces to the tracker, it sends information which will help the latter deduce whether the random or the oracle-suggested peers worked better for this particular client. The tracker collects information of this type from all the clients which are test subjects. At the end of the detection phase, the tracker counts the number of clients num_o which voted for the oracle and the number of clients num_r which voted for the random peers. If $num_o > num_r$, the oracle is considered to be sincere and it becomes trusted. Otherwise, it is perceived as a malicious one and it is blocked, at least until a next scheduled detection phase. The rationale behind the Within-Client Detection Strategy is illustrated in Figure 7.2.

The clients of the second group are not part of the experiment and do not influence the outcome of the detection strategy in any way. They all receive either oracle recommendations or random peers, depending on which of the two alternatives was being used the moment the detection phase began. For instance, let us suppose that a detection terminates and concludes that the oracle is sincere. The tracker places trust on the oracle and uses its service for 100% of the clients in the AS, until the beginning of the next detection phase. When the new detection begins, s% of the clients become test subjects, while the rest $(100 - s)$% of them continue to get oracle recommendations. Similarly, if the previous detection had deduced that the oracle was malicious, the beginning of the new detection phase would find 100% of the clients getting random recommendations. The detection algorithm would make s% of the clients test subjects, while the rest $(100 - s)$% of them would continue to get randomly selected peers.

During the very first detection phase in the lifetime of the swarm, the tracker can use any background knowledge that it has in order to decide whether to grant oracle-guided or random recommendations to Group 2. If there is any type of evidence that the oracle might be malicious, Group 2 can start by getting random peers. Otherwise, it can be made to receive the suggestions of the oracle. Normally, the sincere oracles are expected to be more than the malicious ones. This is because (with the help of the detection strategies) the malicious oracles will soon be blocked by most distributed applications and they will finally perish. Thus, it makes sense for a tracker to have Group 2 initially get oracle-guided recommendations.

The value of s affects the accuracy of the algorithm. A larger s means a larger sample of peers to experiment on. However, increasing s also reduces the population of the second group. The latter can be good or bad, depending on whether the oracle has changed behavior since the last detection. As it was described in the previous paragraph, Group 2 will be getting recommendations which are in accordance with the outcome of the last detection strategy. If the oracle has changed its ranking policy since then, then it would be better if Group 2 were small (and s be large). On the other hand, if the oracle has retained its ranking scheme, there is an incentive for Group 2 to be large (and for s to be small). Therefore, the value of s must be selected with great care. If it is speculated that the oracle will change its behavior very frequently, s can be made to reach its maximum value (i.e. 100%). Otherwise, in a more stable environment, s should be given a value which guarantees sufficient accuracy, but which also does not reduce the size of Group 2 too much.

An alternate approach is to have the tracker constantly evaluate the behavior of the oracle. In this variation, detection never stops and a constant percentage s% of the total clients receives mixed oracle/random recommendations. The tracker collects feedback from these clients and uses the incoming information to keep continuous track of the sincerity of the oracle. Then, it can proceed to adjust the type of recommendations that are sent to the rest $(100 - s)$% of the clients accordingly. The value of s must be small, otherwise the performance of the swarm would be negatively affected by the constant presence of a large number of test subjects.

As it was described, the tracker needs to know whether the set of random peers or the set of oracle-suggested peers worked best for a client. This demands the exchange of information between the tracker and a client which is under test. Regarding what type of information has to be exchanged, two approaches exist. They both demand some additions in the existing Tracker Protocol (as it was described in Chapter 3):

1. The tracker lets the client know which set each of the peers belongs to. This piece of information can be easily embedded within the *Tracker Response* message. Instead of sending one list of D peers, two lists of $\frac{D}{2}$ peers are sent. The client tries peers from both lists and determines which list provided the best performance. When it re-announces to the tracker, it indicates the index of the winning list via a suitable parameter in the *Tracker Request* message.

2. The tracker does not inform the client of which set each of the peers belongs to. In other words, the client receives one list of peers as usual. Within this list, the tracker has included peers from both sets in a random order. The client calculates the download rate it experiences for each of the peers in the list (or at least the ones that it attempted to connect to). When it re-announces to the tracker, it sends tuples of the format (*peer*, *rate*). These tuples are embedded in the *Tracker Request* message. The tracker receives the client's request and splits the tuples into two groups, according to whether the peer in the tuple was a random or an oracle-recommended one (this means that the tracker also has to cache this type of information when it sends a *Tracker Request* message). The tracker then averages the rates of the peers in the two groups and compares them. This allows it to know which set of peers worked better for that particular client.

The second approach has two disadvantages. Firstly, it imposes a larger computational and memory overhead on the tracker. The latter has to perform additional calculations such as the averaging of the download rates. It also has to cache the peer recommendations that it sends to the clients. Secondly, the second approach demands the exchange of more bytes of information. This is due to the additional 'download rate' values that must be included in the *Tracker Request* messages. The main disadvantage of the first approach is that it exposes the detection strategy to the clients. It is desirable that a client is unbiased and objective in its evaluation of the peers. This can best be achieved if the client does not know if and how the peers are split into two sets.

The Within-Client Detection Strategy can be adapted to incorporate some of the alternatives that were presented for the Between-Client Detection Strategy. More precisely, Algorithm 2 can be used to adjust the time interval between two subsequent detections. Algorithm 3 can also be used as a mechanism of sending warning messages to the oracles. An attack like the one described in Section 7.1 could also be launched against the Within-Client strategy. However, analogous same remedies can be used for protection.

Comparing the two proposed Detection Strategies on a theoretical basis, it can be speculated that the Within-Client strategy outmatches the Between-Client strategy. This is due to two reasons:

1. In the Between-Client strategy, the comparison between the two types of recommendations (random ones and oracle-guided ones) is done by exercising them on two totally distinct groups of clients. As it was already discussed, it is impossible to guarantee that these two groups have the same synthesis. Even if the bandwidth of the clients is used as an indication, other factors will also determine the download performance that a client is expected to exhibit. Eventually, chances are that one of the groups will end up with better clients than the other and this will influence the outcome of the detection in favor of that group. This limits the accuracy of the Between-Client strategy. On the other hand, in the Within-Client strategy the two types of recommendations are tested in the context of the same client. Thus, they are evaluated under the same network conditions and the verdict is a more accurate one.

2. Let us consider the approach in which the oracle in never fully blocked or fully trusted, but a small percentage $e\%$ of the clients is always used for detection. In the Between-Client strategy, these $e\%$ clients will only receive recommendations which are of the less efficient type and they will have no other choice but to endure this phenomenon. In the Within-Client strategy, again $e\%$ of the clients get to receive peers of the less favorable type. However, the performance of these clients is not compromised, as half of the recommended peers are still of the good type. The clients can utilize the optimization features of the BitTorrent protocol (such as the Choke/Unchoke algorithm) to separate the wheat from the chaff, distinguish the good from the bad peers and stick to the former ones. So, eventually these $e\%$ of the clients will find their way and preserve their well-being.

The drawback that the Within-Client strategy has when compared to the Between-Client strategy is

that it demands more substantial changes to the existing BitTorrent protocol. BitTorrent clients may be reluctant to introduce these new features.

7.4 Discovery of the best oracle using detection

This section describes a variation of the Between-Client Detection Strategy which can be used by the tracker to discover the oracle providing the best service in an AS. It is assumed that there are n different oracles offering ALTO guidance to the hosts of a specific AS, with $n \geq 2$. The Quality of Service that each one of them provides is different and depends on the corresponding employed ranking strategy. Furthermore, it is assumed that the tracker is aware of the existence of the whole set of candidate oracles, thanks to the application of a suitable Oracle Discovery mechanism, such as the ones described in [105, 106, 107].

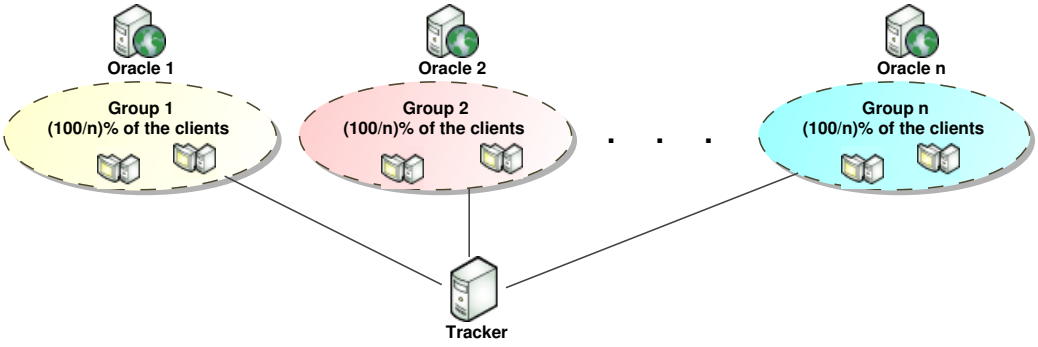


Figure 7.3: Discovery of the Optimum Oracle using Detection

Again, the basic principle is the same as the one behind the original Between-Client strategy. The tracker divides the clients of the AS into n equally sized groups, with each group corresponding to one of the oracles under test (see Figure 7.3). For the duration of the Detection Phase, the clients receive recommendations from the oracle of their group and the tracker collects information on their download performance. As soon as the detection terminates, the download rates of all the clients in a group are averaged and the groups are ordered according to this average. Hence, the tracker manages to rank the oracles on the basis of the performance experienced by the clients which receive their recommendations. The lower ranking oracles can be blocked and the higher ranking ones can be trusted.

The tracker may also choose to divide the clients into $n + 1$ equally sized groups, with the extra one receiving randomly selected peers. The performance of this latter group can be used as a comparison base and it can help determine which of the oracles are malicious and which are sincere. This can prove to be a valuable variation of the algorithm, especially in the scenarios in which all of the oracles are malicious. In such a case, simply selecting to trust the highest ranking oracle is a bad choice, as even the recommendations of this oracle will lead to a degraded download performance. By having $\frac{100}{n+1}\%$ of the clients get random peers, the tracker can diagnose this scenario, block all the oracles and operate in a traditional oracle-less fashion. Nevertheless, it is estimated that in most cases the majority of the oracles will be sincere ones. If this is true, sending random peer recommendations to a portion of the clients will cause their performance to be lower than it would be if they received the guidance of an oracle. Therefore, if the tracker has prior knowledge indicating that most of the oracles behave appropriately, it may decide not to make use of this practice.

As it was described, the tracker can conduct a single detection and use the collected information to rank the oracles and discover the optimal one. However, if the list of candidate oracles is large, the

detection phase will have to last for a very long time, so that the tracker collects enough data for each group of clients. In addition, fine grained decisions, involving oracles whose behavior does not differ a lot, may not be possible if n is very large. For example, let us consider the case in which $n = 10$. The clients are split into 10 groups, each one containing 10% of the total set of available clients. Let us consider that the two best oracles have very slight differences. Having the tracker determine the better of the two based on a fraction of 20% of the total clients will not be possible.

As a solution to the above problem, an alternative approach is proposed. The detection phase is divided into smaller sub-phases. During the first sub-phase, the clients are split into n groups. Each one of them contains $\frac{100}{n}$ of the total clients and receives recommendations from one of the n oracles. At the end of the sub-phase, the tracker uses the collected data to discover the worst ranking oracle and blocks it. During the second sub-phase, the clients are divided into $n-1$ groups. Each one of them contains $\frac{100}{n-1}$ of the total clients and receives recommendations from one of the $n-1$ remaining oracles. At the end of the second phase, another oracle (the one performing worst) is blocked. The procedure continues until the tracker has to decide between the 2 best oracles, based on groups containing 50% of the total clients.

Finally, it may not always be a good practice to direct all the queries to the optimum oracle and have all the others blocked. This is due to two reasons. Firstly, an excessive load will be placed on that oracle and this might introduce lag to the communication with it. Secondly, if the oracle alters its ranking criterion and starts behaving in a malicious way, the tracker will not be able to detect it immediately. The change will only be discovered after the next Detection Phase, which may be scheduled to happen after a long period of time. Meanwhile, the download performance of the clients will be detrimentally affected. To resolve this issue, the tracker should direct most of its queries to the highest ranking oracle (e.g. 80% of them) but also have a small portion of the clients receive recommendations from the second best oracle (e.g. 20% of them). This will ensure that the load is not concentrated towards one single destination and that a change in the behavior of the optimal oracle is pinpointed as soon as it happens.

Chapter 8

Evaluation of the Detection Strategies

This chapter is dedicated to the evaluation of the detection mechanisms which where presented in Chapter 7. We begin by enumerating the goals of our experiments. After that, a detailed presentation of the results is provided. We conclude with a summary and a discussion of our key findings.

8.1 Aims

The algorithms and the techniques which were discussed in Chapter 7 were encoded and were added to the model that was presented in Chapter 5. Extensive tests were conducted so that their performance could be evaluated. The goals of our experiments can be summarized as follows:

- Firstly, the two proposed detection strategies were tested on a large repertoire of oracles, both sincere and malicious ones. Through these experiments, we aimed at identifying the improvement that each strategy introduces to the behavior of the system. More precisely, we tried to measure the increase in the download rates of the P2P clients, as well as the potential reduction in the utilization of the links of the entire simulated network. This would allow us to compare and contrast the two Detection Strategies and pinpoint their benefits and drawbacks. In addition, we attempted to measure the accuracy of the strategies, in terms of how successful they are in discriminating between honest and hostile oracles
- Via additional experiments, we aimed at quantifying the extent to which each one of the variations that were proposed in Section 7.2 affects the performance of the system.
- Our final goal was to prove that in an AS with multiple distinct oracles, the multi-oracle Between-Client detection technique of Section 7.4 manages to cater for the interests of the P2P application as well as to promote the well-being of the AS in terms of congestion.

8.2 Comparison and evaluation of the Detection Strategies

We begin by studying and comparing the performance of the Between-Client and Within-Client strategies. Both strategies are tested on numerous different distributions of malicious oracles across the network. We study cases in which all oracles are sincere, as well as scenarios with a strong presence of non-cooperating oracles. Furthermore, the algorithms are applied on all different types of malicious oracles which have been discussed (e.g. oracles promoting the financial interests of their ISPs, oracles launching DoS attacks etc). It is assumed that both strategies detect for the same amount of time and

that they repeat their detections with the same frequency. More precisely, we assume that in both algorithms, the tracker dedicates 20% of the time to detecting. For the Between-Client strategy, the simulations begin with $p = 70$. This means that during the first detection 70% of the clients receive the recommendations of the oracle, while the rest 30% of them receive lists of random peers. Similarly, for the Within-Client strategy, simulations begin with $s = 30$. Again, this means that 70% of the clients are guided by the oracle, with the rest 30% of them being test subjects and getting both types of recommendations. Hence, the two strategies are tested under the same conditions.

Firstly, the two techniques are compared on the basis of their accuracy. Both of them can be seen as binary classifiers whose purpose is to diagnose the hostility of oracles. They are analogous to the classifiers used in fields like medicine, signal processing and machine learning. Therefore, they can be evaluated with the help of the metrics which are traditionally used in statistical classification, such as the *Accuracy*, the *Recall*, the *Precision* and the *F1 metric* [108]. In order to calculate these measures, tests are conducted on numerous oracles. The outcome of such a test can be either positive (if the oracle was found to be malicious) or negative (if the oracle was found to be sincere). Moreover, depending on whether the result of the classification corresponds to the true nature of an oracle, it can be characterized as a true or a false one. Hence, on the whole, the results of the experiments can fall into one of the following four categories:

- True Positives (TP): These are the malicious oracles which were diagnosed as malicious.
- True Negatives (TN): These are the sincere oracles which were diagnosed as sincere.
- False Positives (FP): These are are the sincere oracles which were diagnosed as malicious.
- False Negatives (FN): These are are the malicious oracles which were diagnosed as sincere.

The number of true/false positives/negatives can be used to calculate the aforementioned classification metrics: More precisely:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{8.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{8.3}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{8.4}$$

Recall describes the completeness of the detection, while precision measures its exactness. F1 combines the previous two metrics into a single one. The results of our experiments are summarized in Table 8.1.

As the table illustrates, the Within-Client strategy is a more accurate detection algorithm than the Between-Client strategy. The reason is the one which was already outlined in Chapter 7. The Between-Client strategy applies the two alternative types of recommendations (random and oracle-guided ones) on two distinct groups of clients. These groups will never have exactly the same dynamics and the same capabilities. Chances are that one of them comprises superior peers and this will lead the Between-Client algorithm to vote in favor of that group. On the other hand, the Within-Client strategy tests

	Between-Client Strategy	**Within-Client Strategy**
Accuracy	90%	94%
Recall	94%	96%
Precision	87%	92.3%
F1	90.4%	94.1%

Table 8.1: Performance of the Detection Strategies

the two types of recommendations on the same client. This allows for an unbiased and ultimately more accurate outcome.

Another interesting thing to notice is that for both strategies the recall is larger than the precision. This means that it is more common to mistake a sincere oracle for a malicious one than the other way round. This is due to two reasons:

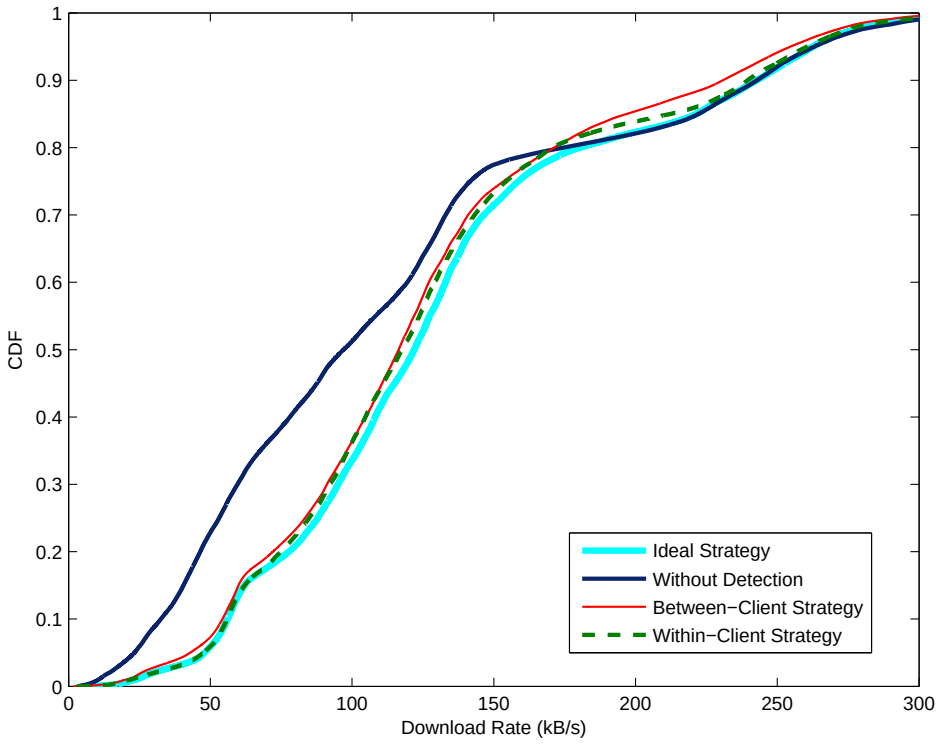
1. It is very unlikely that the random peer selections will be worse than some of the most vicious recommendations supplied by a malicious oracle. For example, let us consider the case of an oracle which launches DoS attacks, by always assigning the highest rank to a set of specific peers. There is very small probability that a series of random recommendations will perform worse than this. On the contrary, it is more probable that a random selection manages to reach the quality of the advice of a sincere oracle. For example, the randomly selected peers may happen to be seeds with fast access links (both in the local and in distant ASs). That is why false positives are more common than false negatives.

2. The actions of a malicious oracle may make an oracle of another AS appear as hostile, although in reality it may not be. We will illustrate this via an example. Let us assume that the oracle of AS1 generates DoS attacks to a portion of the clients in AS6. Let us also suppose that the oracle of AS6 is trying to be an honest one by favoring interior peers. Nevertheless, the clients of AS6 may probably not benefit from the recommendations of this oracle, as in this case a portion of the local nodes are harassed by the DoS attacks of the oracle in AS1 and will not be good peering choices. On the contrary, a random peer selection may prove to be more suitable for the clients of AS6 which are not facing the DoS attack. Hence, the tracker may conclude that the oracle of AS6 is hostile, although it is not. Its intentions are honest, but eventually its recommendations do not turn out to be beneficial, due to the intervention of the oracle of AS1. Therefore, this oracle will be blocked. Fortunately, this error is bound to be corrected by a new detection phase in the future. When the oracle of AS1 receives a long-lasting block for its malicious behavior, the recommendations of the oracle of AS6 will start to demonstrate their quality and this oracle will finally gain the tracker's trust.

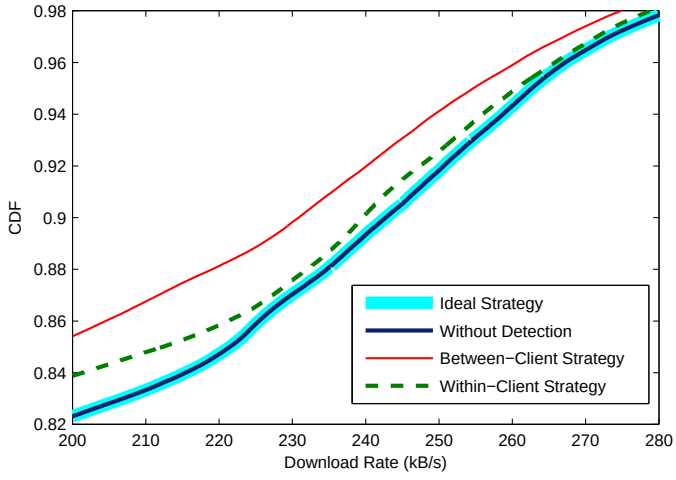
We now present the effect that the application of the detection schemes had on the download performance of the BitTorrent swarm. Figure 8.1(a) plots the CDF of the download rate for the two detection strategies, as well as for the scenario without detection, in which the tracker blindly trusts all oracles. It also depicts the CDF for an ideal optimum detection strategy, i.e. a detection strategy which detects for the same amount of time as the other two but:

- has a perfect accuracy of 100%.
- imposes no harassment on the clients during the detection phase.

Firstly, we should comment that both detection techniques experience a behavior that is very close to the ideal one. This is a proof of their quality and their accuracy. With the Between-Client strategy, the average download rate of the swarm was seen to increase by 11.5%. With the Within-Client strategy, the improvement climbed up to 15%. With the ideal improvement being 18%, this can be seen as a



(a) A graph of the CDF in its entire range



(b) A detail of Figure 8.1(a)

Figure 8.1: CDF of the Download Rate of the swarm for various detection schemes.

very good achievement. Within-Client is found to be superior to Between-Client for two reasons:

- Firstly, and most importantly, the Within-Client strategy is more accurate than the Between-Client one, as it has been thoroughly analyzed.
- As it was described in Chapter 7.3, the clients which are used as test subjects in the Within-Client strategy finally manage to separate the wheat from the chaff, recognize the peers of the good type and achieve a good download performance. On the other hand, in the Between-Client strategy the peers which get the recommendations of the bad type are doomed to tolerate a degraded performance. In a sense, one can think that the Within-Client technique generally places a smaller overhead on the swarm during the detection.

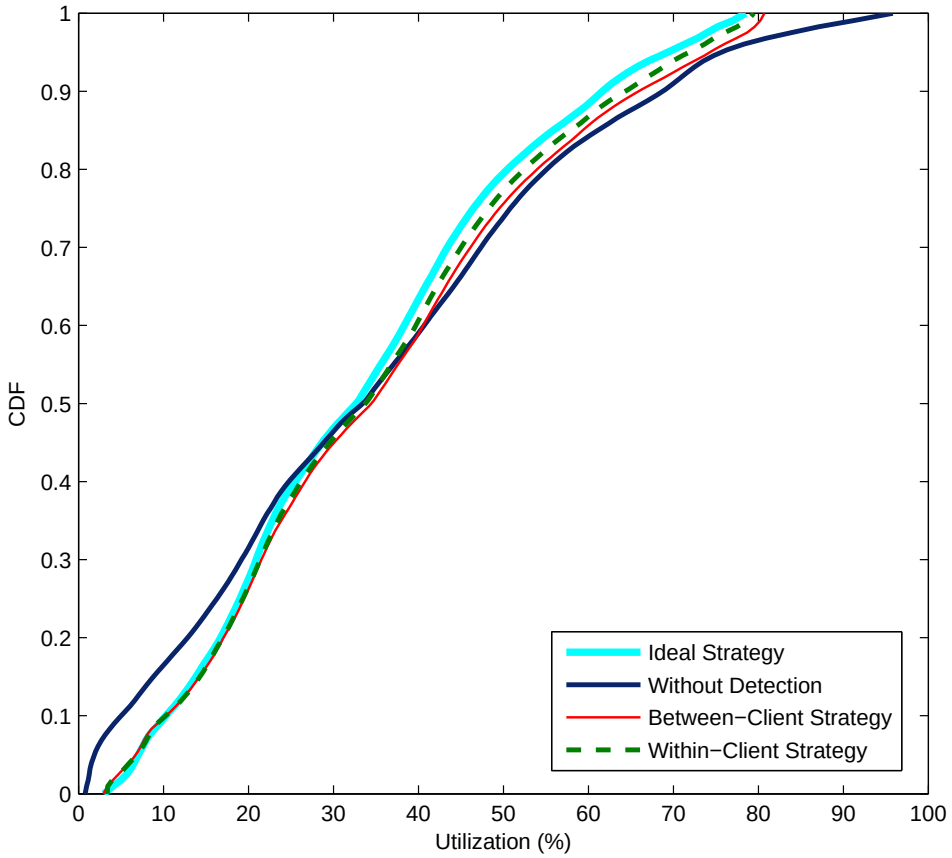
As Figure 8.1(a) illustrates, the detection algorithms which we propose are of significant benefit for approximately 80% of the clients. This is the reason why the lines for the two strategies lie to the right of the 'Without Detection' line for percentiles up to 80. However, the two detection schemes can be seen to cause a small portion of the clients to experience worse performance than the one they would have experienced if no detection were applied. This can be seen in Figure 8.1(b), which is a detail of figure 8.1(a) for percentiles between 82 and 98. This small sacrifice of performance for these few clients is due to two reasons:

- Both strategies lead to a small number of False Positives. This means that few oracles are blocked, although they should not be. The clients of these ASs will obviously have worse completion times, due to this (rare) unsuccessful detections.
- Let us consider the case of an honest oracle. During the detection phase, a portion of the clients receive random peers and thus experience degraded performance. If no detection were applied for this particular oracle, these clients would have been better off. As it has been discussed, in the Within-Client strategy these clients are less affected than in the Between-Client strategy, as half of the peers that they receive still come from the sincere oracle. That is the reason why the line for the Within-Client strategy lies to the right of the line for the Between-Client Strategy in Figure 8.1(b).

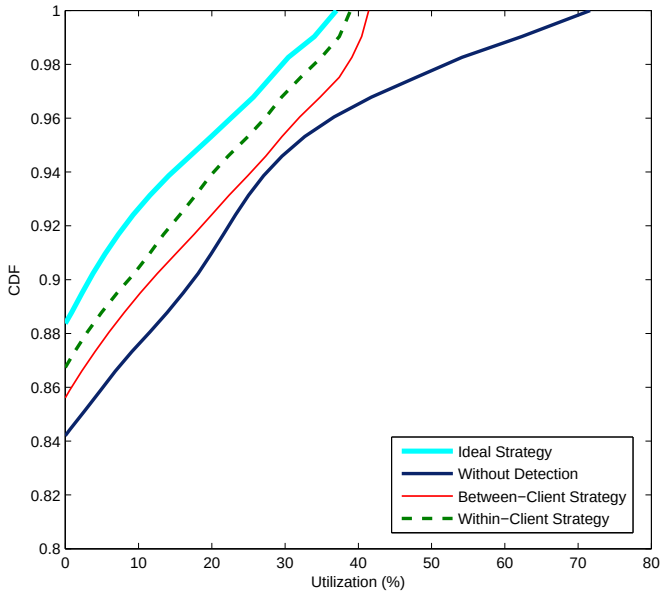
Of course, the ideal detection strategy would not exhibit any of the two features which were described above, since it would have an accuracy of 100% and would not impose any overhead on the clients of ASs with sincere oracles. This is the reason why in Figure 8.1(b) the line for the ideal strategy lies on top of the line which corresponds to the scenario without detection. Nevertheless, this decrease in the rates is very small and it only applies to a small fraction of the clients. At the same time, the rest 80% of the clients still benefit significantly from the detection. At this point, we should point out that both proposed detection algorithms reduce the coefficient of variation by about 25%. This practically means that they manage to increase the fairness of the protocol.

More interestingly, the detection schemes applied by the tracker do not only serve its own interests; they also work for the greater good of the network. Figure 8.2(a) depicts the CDF of the utilization of all the links of the network (calculated using the 95 percentile model as usual). Although the average utilization does not change with the use of the detection, the distribution of the load on the links is seen to improve. Under-utilized links now become of use, while the heavily utilized ones are relieved from a portion of their load. Again, the Within-Client strategy is seen to be superior to the Between-Client one, for the same reasons.

More specifically, the Between-Client strategy improves the utilization of the bottleneck link by 15% and the Within-Client strategy by 17%, when compared to the scenario without detection (also see Figure 8.2(b) which is a detail of Figure 8.2(a)). Furthermore, the standard deviation of the utilization decreased by 13% for the Between-Client and by 18% for the Within-Client strategy. This means that, thanks to the detection, the load was balanced more appropriately among the links. This is due to



(a) A graph of the CDF in its entire range



(b) A detail of Figure 8.2(a)

Figure 8.2: CDF of the Utilization of all the links of the network for various detection schemes.

the fact that, as it was discussed in Section 6.4, the presence of malicious oracles leads to the overloading of specific links. For instance, the traffic on some of the inter-AS links was seen to explode. By detecting and replacing the malicious recommendations with random ones, the load is more fairly distributed. As an example, let us consider an oracle which pushes as much traffic as it can to the customer ASs. This will lead the corresponding inter-AS links to experience a high utilization. By blocking the oracle and using random recommendations instead, these links enjoy a fairer treatment.

Finally, we must comment that once again the behavior of the two detection strategies is very close to the ideal, as far as the utilization of the links is concerned. This theoretical optimum detection strategy reduces the utilization of the bottleneck link by 18% (see Figure 8.2(b)) and the standard deviation of the utilization of all the links by 22%. As it is obvious, these values are very close to the ones which we obtained from the Within-Client Detection strategy.

In our experiments, the termination condition of the Detection Phase was chosen to be the expiration of a time interval T. Numerous different values for the duration of this interval were tested. The effect on the download performance of the swarm is presented in Figure 8.3. The graph applies to both detection strategies. As it obvious, neither a very short nor a very long detection is beneficial. Very short detections lead to inaccurate conclusions and result in the blocking of sincere oracles and/or the trusting of malicious ones. The tracker must be given enough time to collect sufficient data so as to ensure that its verdict is correct. On the other hand, the detection cannot be prolonged forever, as the swarm needs to benefit from its outcome as soon as possible. Getting stuck in the detection for a long time is undesirable.

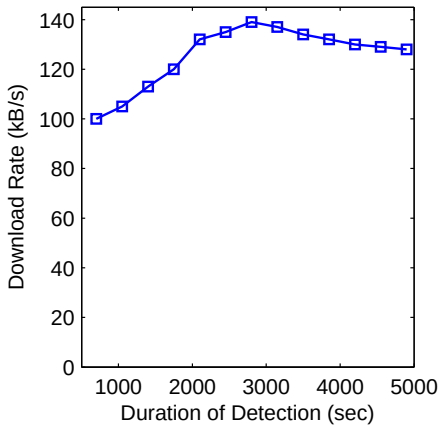


Figure 8.3: Download performance in relation to the duration of the detection.

8.3 Comments on the Alternative approaches

Sliding percentage of oracle-aided clients

We compared the traditional Between-Client Detection Strategy, which completely blocks (trusts) an oracle that was found to be malicious (sincere), with the variation of the sliding percentage of oracle-aided clients, which was described in Section 7.2. We studied the effect of these two detection techniques on the download performance of the BitTorrent swarm. The results agreed with the analysis that was already provided. Always having a small portion of the clients receive recommendations of the less favorable type helps the tracker keep constant track of the behavior of the oracle. Therefore, it can pinpoint changes in this behavior promptly and it can react on time and adjust its policy accordingly.

On the other hand, this small portion of clients will experience degraded download performance, damaging the average download rate of the swarm. We will demonstrate this tradeoff via an example.

Let us consider the experiment of Figure 8.4. The figure plots the average download rate of the whole swarm over time for the two scenarios under study: the traditional Between-Client strategy and the variation with the sliding percentage. We assume that the traditional strategy detects for 1 hour. When the detection terminates, it makes a decision and either completely blocks or totally trusts the oracles under test. It then reschedules the next detection 9 hours later. In other words, the tracker devotes 10% of its time to detecting. The second technique begins by having 50% of the clients receive random recommendations and the rest of them get the guidance of the oracles. Every 20min, the tracker assesses the performance of the two groups of clients and adjusts the percentage accordingly. In our experiments, we assume that the maximum and minimum allowed values for this percentage are 80% and 20% respectively. Finally, we assume that the oracles begin as malicious and at $t = 260min$ they change their strategy and become sincere.

The figure illustrates the merits and the demerits of the second technique. Its advantage is that it manages to respond to the change in the behavior of the oracles as soon as it takes place at $t = 260min$. During the time period 400-600min, this technique manages to fully exploit the fruitful recommendations of the oracles, while the traditional Between-Client algorithm continues to blindly block them. The drawback of the sliding percentage can be noticed in the time periods 100-200min and 700-800min. During the first of them, the traditional strategy is totally relieved from the negative influence of the malicious oracles, while the sliding-percentage variation still imposes this burden on 20% of its clients. Similarly, during the period 700-800min, the traditional algorithm has all its clients benefit from the honest oracles, while in the second scenario 20% of them will still receive random recommendations.

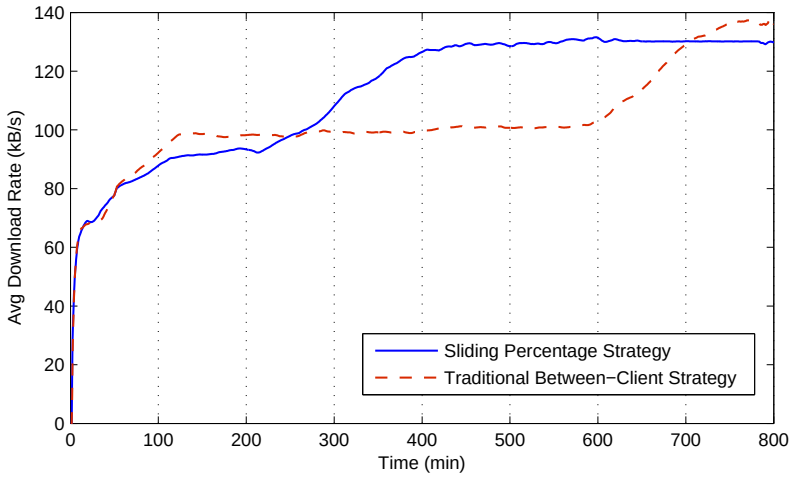


Figure 8.4: Comparison between the traditional Between-Client strategy and the sliding-percentage alternative.

On the whole, the success of this variation is determined by the frequency with which the oracles alter their behavior. If the changes are frequent, a sliding percentage will permit prompt reactions to them. If they are not frequent, this algorithm will handicap the performance of the application. Experiments showed that in a scenario with oracles which never alter their ranking policy, the traditional Between-Client strategy outmatched the sliding-percentage technique, as it guaranteed that the average download rate was 6% higher. On the contrary, in a scenario with oracles flipping their behavior every 250 min, the second strategy allowed for an improvement of more than 20% in the average download rate. It is up to the tracker to decide which alternative is most preferred in the

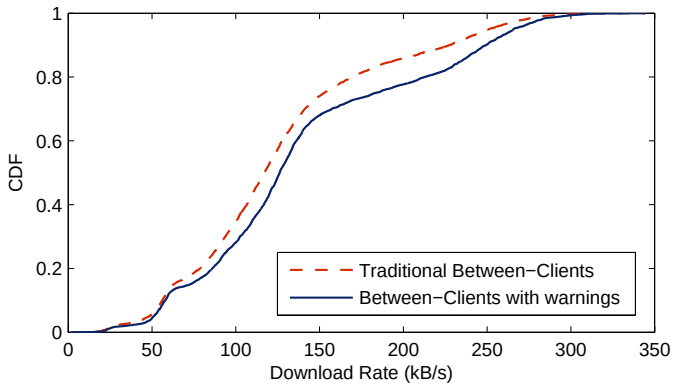


Figure 8.5: CDF of Download Rate when warnings/forgiveness are used. It is assumed that eventually half of the malicious oracles cooperated.

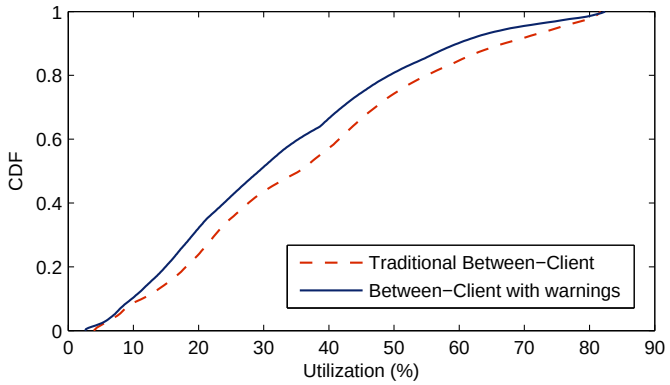


Figure 8.6: CDF of the Utilization of the links when warnings/forgiveness are used. It is assumed that eventually half of the malicious oracles cooperated.

environment where it operates.

Warning/Forgiving the Oracle

Experiments were conducted for the evaluation of Algorithm 3, which was presented in Section 7.2. The algorithm was tested in conjunction with both the Between-Client and the Within-Client strategies. In these experiments, we assumed that half of the oracles of the network were sincere by default and the other half were malicious. We also assumed that half of the malicious oracles considered the warning messages that they received from the tracker and chose to switch to an honest ranking strategy. The other half remained loyal to their hostile policy. Finally, the f parameter of the algorithm was selected to be equal to 3. This means that each oracle was forgiven for a maximum of 2 times.

The results showed that, in this context, the use of Algorithm 3 brought additional improvement to the BitTorrent application. More precisely, with the use of this form of '*reinforcement learning*', the average download rate of the swarm was seen to increase by 10%, when compared to the performance of the simple detection without the use of warnings. Figure 8.5 shows the CDF of the download rate for the simple traditional Between-Client strategy and the enhanced one which uses warning/forgiving.

Moreover, again with half of the oracles finally deciding to cooperate, this strategy succeeded in improving the traffic flow across the entire network. As seen in Figure 8.6, although the utilization of the bottleneck link did not decrease, the average utilization improved by about 9%.

Others

Finally, experiments led to the following observations:

- The use of the clients' bandwidth for the normalization of the statistics collected from them, as it was described in Section 7.2, managed to improve the accuracy of the Between-Client strategy by about 2.5%. Obviously, this is not a significant improvement. Judging the client only from the capacity of its access link is not a safe approach. It fails to consider parameters such as the client's load and the bandwidth and the utilization of the rest of the links which form the routes to this client. Unfortunately, this is information that is not available to the tracker. If it were, then the tracker would not need the service of an oracle in the first place. Hence, ultimately the best way to maximize the accuracy of the Between-Client strategy is to gather statistics from adequately large samples of clients.
- Adjusting the interval between successive detections, as it was presented in Algorithm 2 of Section 7.2, can improve the average download rate of the swarm by a further 3% (when compared to the simple detection algorithm). Of course, this improvement is observable only in cases when the behavior of the oracle is stable and does not change frequently. If the oracle constantly adapts its ranking criterion, the Algorithm will ensure that the interval between the detections has its smallest possible value. Thus, in such cases, the Algorithm will not result in any improvement at all (but it will not cause any harm either). Furthermore, the maximum allowed value for the time interval must be selected with extra care. If $maxInterval$ is selected to be very large, then a potential change in the ranking criterion of the oracle during this time will not be diagnosed promptly enough, and the variation may end up causing harm to the average download rate of the swarm (a decrease of 4% was measured in such scenarios).

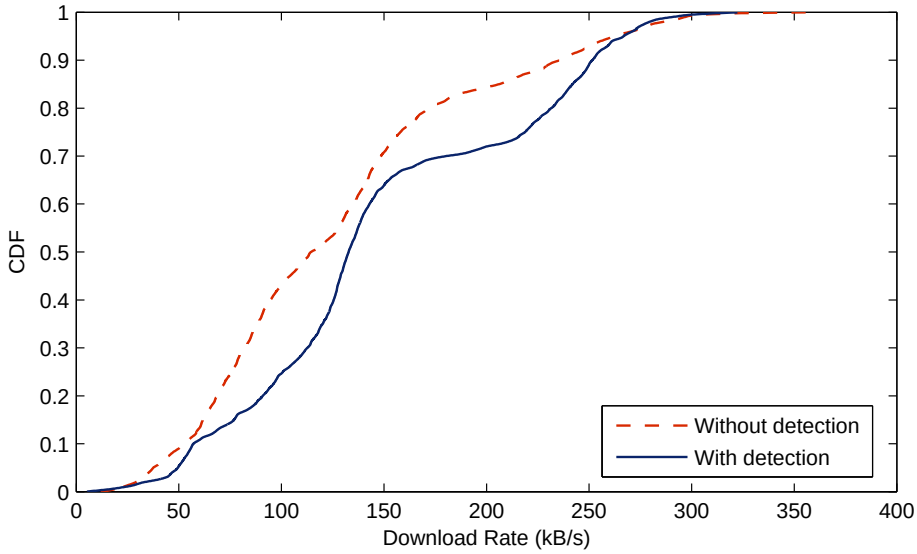
8.4 Discovery of the Optimum Oracle

The purpose of this section is to demonstrate the effectiveness of the technique which was described in Section 7.4. This technique is based on the principles of the Between-Client strategy and aims at identifying the optimum oracle, in cases of ASs with multiple deployed oracles.

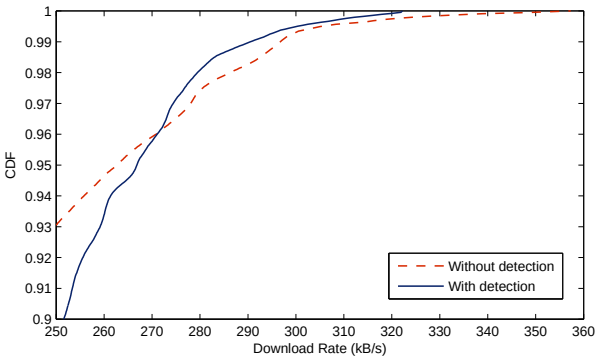
We assume that AS2 of our topology has 4 oracles offering ALTO guidance. It is also assumed that two of these oracles are malicious. The first one is promoting the financial interests of the ISP and pushes all the traffic to the customers. The second one attempts to generate DoS attacks towards particular nodes in AS6. The other two oracles are honest ones. The first one of them is employing the naive coarse-grained strategy of simply segregating the candidate peers according to whether they are internal or external. The second one adopts a more elaborate ranking criterion and ranks internal peers on the basis of the cost of the routes which lead to them. Finally, we assume that there is only one swarm operating in the network and that the tracker is aware of the existence of all four oracles. The ALTO discovery mechanisms which were used by the tracker for the acquisition of the IP addresses of the oracles is out of the scope of this study.

We run experiments on two alternative scenarios:

- Scenario 1 - Without detection: Whenever the tracker wishes to receive ALTO guidance, it randomly selects an oracle from the list of candidate ones and consults it.
- Scenario 2 - With detection: The tracker applies the detection algorithm of section 7.4 to discover the optimum oracle. The oracle which emerges as the best one is then used for all subsequent recommendations, until the beginning of a new detection. Random recommendations are used



(a) A graph of the CDF in its entire range



(b) A detail of Figure 8.7(a)

Figure 8.7: CDF of Download Rate for scenarios without and with detection of the optimum oracle.

(as described in Section 7.4) to ensure that the winning oracle is never a malicious one. At the beginning of a new detection, the tracker divides the clients of the AS into 5 equally sized groups. Each group receives the guidance of a specific oracle, with the 5th one getting random recommendations. The tracker periodically checks the download performance of each group and dismisses oracles, until it converges to the optimal one. We assume that the tracker spends 25% of its time detecting and the rest 75% 'enjoying' the guidance of the oracle which eventually won the competition. Experiments were run for multiple different values of the duration of the detection, so as to push the accuracy of the algorithm to its limits.

As Figure 8.7(a) demonstrates, the use of the detection strategy greatly enhanced the download performance of the clients within AS2. The average and the median download rate in the scenario with the detection improved by about 17%, when compared to scenario without detection. Hence, it is obvious that the adoption of the algorithm is of significant benefit to the P2P application. In an AS with more oracles, this improvement is bound to scale.

One thing that we must note is that a very small fraction of the clients performed better when no detection was applied. These are the clients which correspond to the percentiles above 95 (see the detail of Figure 8.7(b)). The distribution in the scenario without detection is also seen to have a

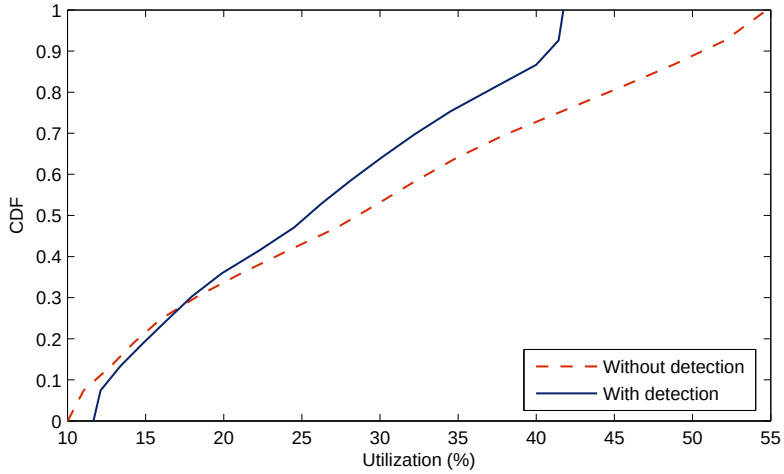


Figure 8.8: CDF of Utilization of interior links of AS2 for scenarios without and with detection of the optimum oracle.

longer tail. The clients which were better off when our algorithm was not applied were the ones which received recommendations from the best oracle by chance. With all the other clients getting worse recommendations, these few lucky clients managed to exploit the good guidance they received to the maximum and blossom. In the scenario with detection, the merits of this optimal ALTO guidance became available to the whole set of clients and not only to a small elite. As a result, this elite would not experience as large download rates as before, as it would not have the exclusive right of enjoying the optimum peer recommendations and the whole environment would generally be more competitive. Obviously, this is not a problem. On the contrary, it is indicative of how our technique managed to promote fairness and guarantee a good experience for every client.

Finally, the detection algorithm was not of benefit only to the BitTorrent application. It also worked for the greater good of the AS and its ISP. Figure 8.8 plots the CDF of the utilization of the links of AS2 for the two scenarios. As one can deduce, our algorithm managed to reduce the average link utilization by 12%. More importantly, it redistributed the traffic across the links in a more appropriate way, placing higher load on the under-utilized ones and relieving the bottleneck link by almost 25% of its load.

8.5 Discussion

The experiments which were presented in this Chapter demonstrate that the two proposed detection algorithms are guaranteed to enhance the performance of the system on many different levels. They improve the download rates of the BitTorrent clients and they increase the fairness of the protocol. At the same time, they contribute to the well-being of the entire network, by redistributing the traffic flow and reducing the utilization of the bottleneck links. Although they are typically designed to cater for the interests of the BitTorrent application, they work for the greater good of the Internet. This is due to the fact that even if malicious oracles may not harm their local ASs, they are self-centered and pay no respect to the balance of their neighbors (e.g. they try to generate DoS attacks, maximize the incurred charges etc.). By identifying and blocking such oracles, the tracker indirectly protects the ASs from this behavior.

The Within-Client Detection Strategy was found to perform better than the Between-Client Strategy. A question that could be asked is why we should even bother considering the Between-Client Strategy.

The answer lies on the fact that it has two advantages, when compared to the Within-Client algorithm. Firstly, as we demonstrated, it can be very easily adapted so that it can be used for the identification of the optimum oracle. Secondly, it can be employed without any changes to the existing BitTorrent protocol. On the other hand, as it has been discussed in Section 7.3, the Within-Client strategy requires the exchange of additional information between the clients and the tracker; thus, it demands an adaption of the BitTorrent protocol.

Finally, as far as the alternative variations of the algorithms are concerned, we can summarize our findings by stating that their success depends on the nature and the behavior of the hostile oracles. Different techniques may or may not be of benefit, depending on whether the oracles change their ranking policies frequently or not, as well as on whether they are willing to cooperate in case they receive negative feedback from the tracker. As the Oracle approach has not been widely deployed on the Internet yet, it is not possible to construct a concrete model for the behavior of the oracles. The future will show what quality of service real-life oracles will offer and how willing to cooperate they will be.

Chapter 9

Conclusions and Future Work

This chapter concludes the project by summarizing the results and the contributions of our work. In addition, Section 9.2 provides suggestions for future work on this area.

9.1 Conclusions

This project studied the ISP-P2P collaboration as a solution to the Application Layer Traffic Optimization problem. It explored alternative realizations of this proposal, identified some of its pitfalls and suggested novel techniques that can maximize its efficiency. The achievements of the project can be summarized as follows:

- Firstly, a detailed study of the problem space was provided. A research of the existing literature allowed us to formulate a taxonomy of the proposed solutions. The numerous different approaches were compared and the merits and demerits of each one of them were elevated. We discussed how the cooperation between ISPs and P2P applications has emerged as the most promising proposal, as it can prove to be a 'Win-Win' solution for both parties. Moreover, the basic principles behind these oracle-related approaches were identified and were used as a basis for the design of a distributed system comprising a P2P application which receives ALTO guidance by dedicated oracle servers.

- For the requirements of our study, the functionality of the system was encoded within the SSFNet simulator environment. Novel parameterizable models of the BitTorrent and Oracle Protocols were developed, thus augmenting the existing model library of SSFNet. The protocols were tested on a topology which was inspired by the architecture of the Greek Internet. The selection of the simulated topology was another one of the challenges of the project, as it required the accumulation of data from a number of resources and the careful research of the dynamics governing the relationships between real ASs.

- Several ranking criteria which can be employed by the oracles have been proposed in the bibliography. However, to our knowledge, so far they have not been evaluated and compared under the same networking conditions. Our work has focused on conducting experiments on a large repertoire of ranking strategies. Each one of them was shown to have its own benefits and drawbacks. Some were found to serve the interests of the ISP while others boost the performance of the P2P application. We demonstrated that simpler ranking criteria can be appropriately combined to form elaborate strategies which are of benefit to both cooperating parties.

- Through extensive experiments, we shed light on some of the weaknesses of the oracle approach, whose severity has not been recognized so far. More precisely, the Oracle Protocol was shown to introduce unfairness and inefficiency to the P2P application when applied during the initial phase of the lifetime of a swarm. More importantly, we demonstrated that the presence of malicious oracles jeopardizes the balance of the system and may ultimately cause it to collapse. Instead of being an improvement, the oracle solution may end up reducing the download rates of the application, while at the same time increasing the utilization of the bottleneck links. Hence, the need for a detection mechanism for the identification of hostile oracles was underlined.
- We invented two original detection strategies for the discovery of malicious oracles. The two algorithms were described in detail and were accompanied by a number of extensions and add-ons that can enhance their performance. The Between-Client strategy is a simple but effective detection scheme that can be easily deployed within the existing file sharing protocols. It was shown to improve the download performance of the P2P application by 11.5% in an environment with hostile oracles. Meanwhile, it was also seen to contribute to the well-being of the entire network, by reducing the utilization of the bottleneck links by up to 15%. The second algorithm, the Within-Client detection strategy, was shown to be more robust and accurate. Although it requires more modifications to the existing protocol, it manages to increase the download rates by up to 15% and reduce the utilization of the bottleneck links by up to 17%.
- Finally, we proposed an adaption of the Between-Client strategy which can aid in the discovery of the optimum oracle in a multi-oracle environment. Tests demonstrated that our proposal is able to relieve the bottleneck link by almost 25% of its load and boost the download performance of the application by 17%.

9.2 Future Work

The following possible pathways for future study can be identified:

- The newly developed detection strategies should ideally be tested on the real Internet. Although a simulation framework provides flexibility and allows the researcher to have absolute control over the experiment, it can never capture the full heterogeneity of the real networks. In many cases, simulations have been found to produce over-optimistic results. For instance, Ledlie et al. [109] have demonstrated that, when deployed on a real system, network coordinate systems did not manage to reach the efficiency which they exhibited within controlled simulation environments. Hence, in the future the two detection strategies should be cross-tested in the context of an actual BitTorrent system. Beginning with experiments on Planetlab [110] should be a good start, but even the utilization of such a realistic testbed may not prove to provide the required accuracy (as it is discussed in [32]). Ultimately, the algorithms will have to be embedded within a BitTorrent client like Vuze [33] and be evaluated 'in the wild'.
- The performance of the new Detection Strategies should also be tested in more diverse scenarios. Firstly, they should be evaluated across a richer repertoire of network topologies, some of which should preferably span ASs in multiple continents. In addition, future study could attempt to determine whether these algorithms are affected by different patterns of churn and free-riding. The existing literature provides a variety of proposed models that can be utilized to simulate churn (such as the model of Stutzbach et al. [111]) and free-riding (such as the model of Li et al. [112]). These models could be introduced to the designed system and allow for the further evaluation of the detection schemes.
- So far, the performance of the Oracle approach has mainly been evaluated on traditional P2P

file sharing applications such as BitTorrent and Gnutella. This project followed this paradigm and tested the two developed detection algorithms and their alternatives within the context of the BitTorrent protocol. It still remains to be seen how the Oracle approach in general and the proposed detection strategies in particular behave when applied onto some of the rest of the use cases of the ALTO problem. With regard to the detection ideas that we discussed, some of them may prove to perform as efficiently as in the case of BitTorrent. Others might have to be adjusted, while others may not be applicable at all. For instance, it would be interesting to determine how the Between-Client strategy can operate within the context of a streaming video application like Joost. Another system which could benefit from the detection of a malicious ALTO service is an online gaming community. It would be useful to explore how such systems can (or cannot) benefit from the work of this project.

- Another pathway is to explore how the more accurate Within-Client Detection Strategy can be used for the discovery of the optimum ALTO guidance in an AS with multiple oracles. Section 7.4 demonstrated how the Between-Client strategy can serve this purpose. However, the use of the Within-Client strategy is trickier, as having the same peer receive mixed recommendations from n different oracles (where n can be larger than 3) may have it confused and the statistics that it provides will not allow for an accurate ranking of these n oracles. Other alternatives will have to be explored. For instance, each peer could be used for the comparison of a subset of the oracles. The results from peers that compared different subsets could then be combined by the tracker to generate a complete evaluation of the performance of the n oracles.
- Finally, one aspect of the problem which has not been fully explored yet is the interaction between the oracles for the exchange of information which could assist them in providing the best service to the clients within their ASs. As we have described, in [27] Aggarwal et al. proposed a Global Coordinate System which would involve the cooperation between the oracles of different ISPs. The ranking of peers within the network of an ISP would be performed by the oracle of this ISP, while for ranking peers belonging to other ISPs, the oracle would have to consult directly the corresponding oracles. Akonjang et al. [36] have stated that a new protocol for this interaction would have to be specified, but they have not proceeded to define such a protocol yet. Future work could focus on simulating this interaction and determining whether it improves the performance of the system. This new element would give birth to a number of new issues. Now an oracle might be the party which would like to detect whether its peer-oracle is malicious or not.

Bibliography

[1] G. Kortuem, J. Schneider, D. Preuitt, T. Thompson, S. Fickas, and Z. Segall, "When peer-to-peer comes face-to-face: Collaborative peer-to-peer computing in mobile ad hoc networks," *p2p*, p. 0075, 2001.

[2] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "File-sharing in the Internet: A characterization of P2P traffic in the backbone," *University of California, Riverside, USA, Tech. Rep*, 2003.

[3] "Napster." `http://www.napster.co.uk`, 2010.

[4] "Kazaa." `http://www.kazaa.com`, 2010.

[5] "eMule." `http://www.emule-project.net`, 2010.

[6] C. Index, "Forecast and Methodology, 2009–2014," *Cisco Systems*, 2010.

[7] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is p2p dying or just hiding?," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04*, pp. 1532–1538.

[8] T. Mennecke, "DSL broadband providers perform balancing act." `http://www.slyck.com/news.php?story=973`.

[9] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider portal for applications," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 351–362, 2008.

[10] J. Seedorf and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement," *Request for Comments: 5693*, 2009.

[11] R. Alimi, R. Penno, and Y. Yang, "ALTO Protocol ," *draft-ietf-alto-protocol-05.txt (work in progress)*, 2010.

[12] S. Kiesel, S. Previdi, M. Stiemerling, R. Woundy, and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements ," *draft-ietf-alto-reqs-05.txt (work in progress)*, 2010.

[13] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An information plane for distributed services," in *Proc. of OSDI*, pp. 367–380, 2006.

[14] H. Xie, A. Krishnamurthy, A. Silberschatz, and Y. Yang, "P4P: explicit communications for cooperative control between P2P and network providers," *P4PWG Whitepaper*, 2007.

[15] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPS and P2P users cooperate for improved performance?," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, p. 40, 2007.

[16] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pp. 1–12, ACM, 2009.

[17] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should internet service providers fear peer-assisted content distribution?," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, p. 6, USENIX Association, 2005.

[18] V. Aggarwal, O. Akonjang, and A. Feldmann, "Improving user and isp experience through isp-aided p2p locality," in *Proceedings of IEEE Global Internet Symposium*, 2008.

[19] IPOQUE, "Internet Study 2007." `http://www.ipoque.com/resources/internet-studies/internet-study-2007`, 2010.

[20] D. Xu, S. Kulkarni, C. Rosenberg, and H. Chai, "Analysis of a CDN–P2P hybrid architecture for cost-effective streaming media distribution," *Multimedia Systems*, vol. 11, no. 4, pp. 383–399, 2006.

[21] D. Pakkala and J. Latvakoski, "Towards a Peer-to-Peer Extended Content Delivery Network," in *Proceedings of 14th IST Mobile and Wireless Communications Summit*, 2005.

[22] C. Huang, A. Wang, J. Li, and K. Ross, "Understanding hybrid CDN-P2P: why limelight needs its own Red Swoosh," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 75–80, ACM, 2008.

[23] S. Ren, L. Guo, and X. Zhang, "ASAP: an AS-aware peer-relay protocol for high quality VoIP," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, p. 70, Citeseer, 2006.

[24] D. Choffnes and F. Bustamante, "Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 363–374, 2008.

[25] N. Anderson, "Hammer drops at last: FCC opposes Comcast P2P throttling." `http://arstechnica.com/old/content/2008/07/hammer-drops-at-last-fcc-opposes-comcast-p2p-throttling.ars`, 2010.

[26] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "IDMaps: A global Internet host distance estimation service," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 5, pp. 525–540, 2001.

[27] V. Aggarwal, A. Feldmann, and R. Karrer, "An Internet Coordinate system to enable collaboration between ISPs and P2P systems," in *Proceedings of the 11th International ICIN Conference,(Location: Bordeaux, France)*, Citeseer, 2007.

[28] T. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *IEEE INFOCOM*, vol. 1, pp. 170–179, Citeseer, 2002.

[29] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 15–26, ACM New York, NY, USA, 2004.

[30] M. Costa, M. Castro, A. Rowstron, and P. Key, "PIC: Practical Internet coordinates for distance estimation," in *International Conference on Distributed Computing Systems*, vol. 24, pp. 178–187, Citeseer, 2004.

[31] B. Wong, A. Slivkins, and E. Sirer, "Meridian: A lightweight network location service without virtual coordinates," in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, p. 96, ACM, 2005.

[32] E. Marocco, V. Gurbani, V. Hilt, I. Rimac, and M. Tomsu, "Peer-to-Peer Infrastructure: A Survey of Research on the Application-Layer Traffic Optimization Problem and the Need for Layer Cooperation," in *IETF Workshop on Peer-to-peer Infrastructure*, 2008.

[33] Vuze. `http://www.vuze.com`, 2010.

[34] V. Aggarwal and A. Feldmann, "ISP-Aided Neighbor Selection for P2P Systems," 2008.

[35] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving traffic locality in BitTorrent via biased neighbor selection," 2006.

[36] O. Akonjang, V. Aggarwal, A. Feldmann, J. Jiang, and P. Xie, "The Oracle Protocol Draft v0.," 2008.

[37] OpenP4P, "P4P - Enable ISP & P2P to Work Together." `http://www.openp4p.net`, 2010.

[38] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 367–378, 2004.

[39] e. a. Griffiths, "RFC 5632: Comcast's ISP Experiences in a Proactive Network Provider Participation for P2P (P4P)," 2009.

[40] R. Paul, "Verizon embraces P4P." `http://arstechnica.com/old/content/2008/03/verizon-embraces-p4p-a-more-efficient-peer-to-peer-tech.ars`.

[41] D. Saucez, B. Donnet, and O. Bonaventure, "IDIPS: ISP-Driven Informed Path Selection," *draft-saucez-idips-01.txt (work in progress, expired May 2009).*

[42] O. Bonaventure, D. Saucez, and B. Donnet, "The case for an informed path selection service," *draft-bonaventure-informed-path-selection-00.txt (work in progress, expired August 2008).*

[43] D. Saucez, B. Donnet, and O. Bonaventure, "Implementation and preliminary evaluation of an ISP-driven informed path selection," in *Proceedings of the 2007 ACM CoNEXT conference*, ACM New York, NY, USA, 2007.

[44] J. Seedorf, S. Kiesel, and M. Stiemerling, "Traffic Localization for P2P-Applications: The ALTO Approach," in *Ninth International Conference on Peer-to-peer Computing (IEEE P2P 2009), IEEE*, 2009.

[45] "ALTO Working Group Charter." `http://www.ietf.org/html.charters/alto-charter.html`, 2010.

[46] S. Shalunov, R. Penno, and R. Woundy, "ALTO Information Export Service," *draft-shalunov-alto-infoexport-00.txt (work in progress, expired).*

[47] R. Penno and Y. Yang, "ALTO Protocol ," *draft-penno-alto-protocol-01.txt (work in progress, expired September 2009).*

[48] O. Akonjang, A. Feldmann, S. Previdi, B. Davie, and D. Saucez, "The PROXIDOR service," *Internet Engineering Task Force, Internet-Draft draft-akonjang-alto-proxidor-00, (work in progress, expired March 2009).*

[49] S. Previdi, "Routing Proximity Services," IETF 73, November 2008.

[50] S. Kiesel and M. Stiemerling, "ALTO H12," *draft-kiesel-alto-h12-02 (work in progress)*, 2010.

[51] E. Leonardi, M. Mellia, A. Horvart, L. Muscariello, S. Niccolini, and D. Rossi, "Building a cooperative P2P-TV application over a wise network: the approach of the European FP-7 strep NAPA-WINE," *IEEE Communications Magazine*, vol. 46, no. 4, pp. 20–22, 2008.

[52] J. Quittek, "Traffic Localization for Cooperative P2P Content Distributionl," in *2nd Japan EU Symposium on the New Generation Network and the Future Internet, (Tokyo, JP)*, ACM, October 2009.

[53] V. Jacobson, D. Smetters, N. Briggs, M. Plass, P. Stewart, J. Thornton, and R. Braynard, "VoCCN: voice-over content-centric networks," in *Proceedings of the 2009 workshop on Re-architecting the internet*, pp. 1–6, ACM, 2009.

[54] D. Cheriton and M. Gritter, "TRIAD: A New Next-Generation Internet Architecture, July 2000."

[55] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: routing on flat labels," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, p. 374, 2006.

[56] M. Sarela, T. Rinta-aho, and S. Tarkoma, "RTFM: Publish/subscribe internetworking architecture," *ICT Mobile Summit, Stockholm*, 2008.

[57] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, p. 192, 2007.

[58] B. Cohen, "The BitTorrent Protocol Specification." `http://www.bittorrent.org/beps/bep_0003.html`, 2009.

[59] TheoryOrg, "Bittorrent Protocol Specification." `http://wiki.theory.org/BitTorrentSpecification`, 2006.

[60] K. Katsaros, V. Kemerlis, C. Stais, and G. Xylomenos, "A BitTorrent module for the OMNeT++ simulator," in *Proc. 17th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), London, Great Britain*, 2009.

[61] uTorrent, "a (very) tiny BitTorrent client." `http://www.utorrent.com`, 2010.

[62] P. Michiardi, K. Ramachandran, and B. Sikdar, "Modeling and Analysis of Seed Scheduling Strategies in a BitTorrent Network,"

[63] M. P. Fabrizio Frioli, "A BitTorrent module for Peersim," 2008.

[64] Cisco, "OSPF Design Guide." `http://www.cisco.com/application/pdf/paws/7039/1.pdf`, 2009.

[65] A. Kaya, M. Chiang, and W. Trappe, "P2P-ISP Cooperation: Risks and Mitigation in Multiple-ISP Networks," 2009.

[66] M. Piatek, H. Madhyastha, J. John, A. Krishnamurthy, and T. Anderson, "Pitfalls for ISP-friendly P2P design," in *The Eighth ACM Workshop on Hot Topics in Networks (HotNets-VIII), New York City, NY, USA*, Citeseer, 2009.

[67] M. Stiemerling and S. Kiesel, "ALTO Deployment Considerations ," *draft-stiemerling-alto-deployments-04.txt (work in progress)*, 2010.

[68] S. Floyd and V. Paxson, "Difficulties in simulating the Internet," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 4, p. 403, 2001.

[69] CAIDA, "The Cooperative Association for Internet Data Analysis." `http://www.caida.com`, 2010.

[70] Robtex, "Swiss Army Knife Internet Tool." `http://www.robtex.com`, 2010.

[71] ADSLgr, "Greek ADSL Online Community." `http://www.adslgr.com`, 2010.

[72] OnTelecoms. `http://www.on.gr`, 2010.

[73] Forthnet. `http://www.forthnet.gr`, 2010.

[74] CYTA. `http://www.cyta.gr`, 2010.

[75] GR-IX, "The Greek Internet Exhange." `http://www.gr-ix.gr`, 2010.

[76] "Level 3 communications." `http://www.level3.com`, 2010.

[77] "Telecom italia." `http://www.telecomitalia.com`, 2010.

[78] Syzefxis. `http://www.syzefxis.gov.gr/`, 2010.

[79] "Space hellas." `http://www.space.gr/`, 2010.

[80] Lemontel. `http://www.lemontel.com/`, 2010.

[81] " National Statistical Service of Greece - Population census of 18 March 2001."

[82] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of wide-area internet bottlenecks," in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, p. 114, ACM, 2003.

[83] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S. Diot, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE network*, vol. 17, no. 6, pp. 6–16, 2003.

[84] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering," in *Proceedings of the 3rd international workshop on Economics of networked systems*, pp. 7–12, ACM, 2008.

[85] "OPNET Modeler Product Documentation, Product Release 10.0."

[86] "OMNET++ Community Site." `http://www.omnetpp.org`, 2010.

[87] "The Network Simulator - ns2." `http://www.isi.edu/nsnam/ns`, 2010.

[88] SSFNet, "Scalable Simulation Framework." `http://www.ssfnet.org/homePage.html`, 2010.

[89] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai, "A survey of peer-to-peer network simulators," in *Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool, UK*, Citeseer, 2006.

[90] "The x-kernel Protocol Framework." `http://www.cs.arizona.edu/projects/xkernel`.

[91] B. Leong, B. Liskov, and E. Demaine, "Epichord: parallelizing the Chord lookup algorithm with reactive routing state management," *Computer Communications*, vol. 29, no. 9, pp. 1243–1259, 2006.

[92] J. Postel, "Transmission Control Protocol-DARPA Internet Protocol Program Specification," *Internet Engineering Task Force*, 1981.

[93] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," Request for Comments (Draft Standard) RFC 1771, Internet Engineering Task Force, Mar. 1995," *Obsoletes RFC1654*.

[94] A. Feldmann, A. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, p. 313, ACM, 1999.

[95] J. Jenkov, "Is Dependency Injection Replacing the Factory Patterns?." `http://tutorials.jenkov.com/dependency-injection/dependency-injection-replacing-factory-patterns.html`.

[96] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-wesley Reading, MA, 1995.

[97] R. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2003.

[98] T. McCabe, "A complexity measure," *IEEE Transactions on software Engineering*, pp. 308–320, 1976.

[99] S. Le Blond, A. Legout, and W. Dabbous, "Pushing bittorrent locality to the limit," *Arxiv preprint arXiv:0812.0581*, 2008.

[100] TRWN, "BGP Protocol Presentation." `http://www.trwn.gr/downloads/BGP_part1.pdf`.

[101] M. Scanlon, A. Hannaway, and M. Kechadi, "A Week in the Life of the Most Popular BitTorrent Swarms," in *5th Annual Symposium on Information Assurance (ASIA'10)*, 2010.

[102] A. Odlyzko, "Internet pricing and the history of communications," *Computer Networks*, vol. 36, no. 5-6, pp. 493–517, 2001.

[103] B. Dijker, "95th Percentile Calculation." `http://www2.arnes.si/~gljsentvid10/pct.html`, 2010.

[104] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, and L. Garces-Erice, "Dissecting bittorrent: Five months in a torrent's lifetime," *Passive and Active Network Measurement*, pp. 1–11, 2004.

[105] M. Stiemerling and H. Tschofenig, "A DNS-based ALTO Server Discovery Procedure," *draft-stiemerling-alto-dns-discovery-00.txt (work in progress)*, 2010.

[106] H. Song, M. Tomsu, G. Garcia, Y. Wang, and V. Pascual, "ALTO Service Discovery," *draft-song-alto-server-discovery-03 (work in progress)*, 2010.

[107] S. Kiesel, M. Tomsu, N. Schwan, M. Scharf, and M. Stiemerling, "Third-party ALTO server discovery," *draft-kiesel-alto-3pdisc-03 (work in progress)*, 2010.

[108] M. P. et al., *Machine Learning - Computer Based Coursework Manual* . 2010.

[109] J. Ledlie, P. Gardner, and M. Seltzer, "Network coordinates in the wild," in *Proc. of NSDI*, 2007.

[110] PlanetLab, "An open platform for developing, deploying, and accessing planetary-scale services." `http://www.planet-lab.org`, 2010.

[111] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, p. 202, ACM, 2006.

[112] M. Li, J. Yu, J. Wu, *et al.*, "Free-riding on bittorrent-like peer-to-peer file sharing systems: Modeling analysis and improvement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 7, pp. 954–966, 2008.