Hybrid and Anonymous File-Sharing Environments: Architecture and Characterisation

Juan Pablo Timpanaro

UNIVERSITÉ DE LORRAINE

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Hybrid and Anonymous File-Sharing Environments: Architecture and Characterisation

DISSERTATION

presented November 6th, 2013

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Juan Pablo Timpanaro

Composition of the jury

Rapporteurs:	Gabi DREO RODOSEK, Professor at Universität der Bundeswehr München Maryline LAURENT, Professor at Télécom SudParis
Examiners:	Guillaume DOYEN, Associate professor at Université de technologie de Troyes Bénédicte LE GRAND, Professor at l'Université Paris 1 Panthéon - Sorbonne René SCHOTT, Professor at Université de Lorraine, Telecom Nancy Olivier FESTOR, Research director at INRIA Nancy-Grand Est
Dissertation Director:	Isabelle CHRISMENT, Professor at Université de Lorraine, Telecom Nancy

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Acknowledgments

This thesis would not have been completed without the support and encourage of many people. First I would like to thank my thesis director, Professor Isabelle Chrisment, for her support and her dedication to my work these past 3 years. Without her guidance this dissertation would not have been possible.

I would also like to express my gratitude to Professor Olivier Festor, for allowing me the opportunity to be part of the Madynes team and its excellent group of people.

I would like to give a special thank to Thibault Cholez for his support and encourage during my first years of research.

I am especially grateful to my old friends, César, Martín and François for their help and fruitful discussions during my thesis. I also would like to thank to Gaëtan and Anthéa for making the office a better place to work.

At last, I would like to extend my gratitude to all the Madynes team, especially to Céline Simon for her endless patience, and to Frederic Beck and Alexandre Boeglin for their extensive technical knowledge and support during my research.

A mis padres, Nancy y Oscar

Por haberme brindado incondicionalmente su apoyo en todas las decisiones que he tomado. Por sus valores y su amor, que me han permitido ser una mejor persona.

A mi hermana, María Belén

Por haber creído en mi, por apoyarme y por las incontables experiencias vividas juntos todo este tiempo.

A mis hermanos, Lucio y Franco

Por estar siempre presentes y por las innumerables charlas y momentos vividos.

Table of Contents

Table of Figures

List of Tables

Abstract

Most of our daily activities are carried out over the Internet, from file-sharing and social networking to home banking, online-teaching and online-blogging. Considering file-sharing as one of Internet top activities, different architectures have been proposed, designed and implemented, leading to a wide set of file-sharing networks with different performances and goals. This digital society enables as well users' profiling. As Internet users surf the World Wide Web, every sent or received packet passes through several intermediate nodes until they reach their intended destination. So, an observer will be able to determine where a packet comes from and where it goes to, to monitor and to profile users' online activities by identifying to which server they are connected or how long their sessions last. Meanwhile, anonymous communications have been significantly developed to allow users to carry out their online activities without necessarily revealing their real identity.

Our contribution is twofold. On the one hand, we consider hybrid file-sharing environments, with a special focus on widely deployed real-world networks and targeting two defined goals. The first goal is to improve content indexation in the BitTorrent file-sharing environment, enabling BitTorrent content to be indexed in the Kad distributed has table and leading to a more robust BitTorrent system. The second goal is to improve content availability in the I2P file-sharing environment. We allow I2P users to anonymously access public BitTorrent content and we obtain a fully anonymous file-sharing environment, including anonymous content indexation and anonymous content distribution.

On the other hand, we focus on the understanding of anonymous environments through extensive monitoring. We characterise the I2P network, targeting the entire anonymous environment and its anonymous services. We consider different aspects of the network, such as the number of users, their characteristics as well as the number of anonymous services available. Through long-term measurements on the network and along with different correlation analyses, we are able to characterise a small group of users using a specific anonymous service, such as the users from a particular city performing anonymous file-sharing.

Introduction

Chapter 1

General Introduction

Contents

1.1 Context

Most of our daily activities are carried out over the Internet, from file-sharing and social networking to home-banking, online-teaching and online-blogging.

Considering file-sharing as one of Internet top activities, different architectures have been proposed, designed and implemented, resulting in a wide set of file-sharing networks with different performances and goals. These last years, these networks have been evolving, from centralised approaches to fully decentralised systems. The BitTorrent network [8] used central TCP-based trackers to coordinate peers sharing different files. Later on, the system moved from TCP-based trackers to UDP-based trackers so as to reduce network overhead. Finally, the entire BitTorrent architecture has recently shifted to a completely decentralised architecture, where two decentralised trackers are available in the network. The other widely deployed peer-to-peer network, namely the eDonkey network [9], has moved from a completely centralised architecture to a fully decentralised environment, where a distributed hash table provides a double-indexation mechanism, enabling a keyword-based search engine.

Digital societies bring with them the possibility of profiling users. As Internet users surf the World Wide Web, all data packets sent and received pass through several intermediate nodes until they reach their intended destination. Although it is not always possible to sniff data passing by an intermediate point, an observer can determinate where a data packet comes from and where it goes to. Thus, these intermediate nodes can monitor users, determine which server they connect to, how long their sessions last, where they connect from. This allows intermediate nodes to make a profile of users' online activities. Anonymous systems have been designed to enable users to carry out their online activities without necessarily revealing their *online identity*, like their IP addresses, in the process. Anonymous systems have become an intrinsic part of nowadays communications, driven by an enormous wave of governmental monitoring activities,

loose legislations regarding citizens' privacy and different privacy infringement cases in online activities. Country-based blocking [10, 11, 12], nation-wide monitoring [13, 14] and conflictive legislations against users' online privacy [15, 16] have fostered Internet users to move towards anonymous-based communications.

Our scientific content follows two axes, the wide set of file-sharing architectures, their performance characteristics and how these architectures can be exploited, on the one hand ; and that growing anonymous-based Internet community and how it can be monitored, on the other hand.

1.2 Problem Statement

Firstly, we consider different file-sharing architectures and their properties. Top popular file-sharing networks, such as the BitTorrent network or the Kad/Ed2k network have different performance properties. They have been studied from different angles and have been optimised for different purposes. The Kad/Ed2k network has an excellent Kademlia-based content indexation scheme, while the BitTorrent's download algorithm is known for its high performance. Therefore, the challenge is to determine how to connect them, thus exploiting the best of both networks.

Nowadays, *anonymous-aware* communications have led to the creation of different closed communities, such as dedicated anonymous file-sharing networks. However, one of the weaknesses of these systems is that they have been designed to preserve users' anonymity and not to access public content communities, such as the BitTorrent network. These two scenarios lead to the first point of our problematic, the study and analysis of *hybrid file-sharing architectures*, and the way we can take advantage of the different aspects of current file-sharing networks to build a stronger file-sharing environment.

Secondly, anonymous networks have been significantly developed [1] and more and more services are available within these systems. A proper characterisation of these systems is necessary to understand their real use in current communications and to determine whether these systems are widely deployed or used only by a particular set of users. In terms of security, it is important to determine whether this characterisation presents an anonymity risk for these systems. This leads to the second point of our problematic, the proper *characterisation of anonymous environments*.

1.3 Contributions

Our contribution is twofold. In the first place, we propose two hybrid file-sharing architectures, designed to improve content indexation within the BitTorrent file-sharing environment and to improve content availability in the anonymous I2P file-sharing environment. Then, we aim at characterising a chosen anonymous environment, targeting the I2P network. Figure 1.1 summarises these contributions and presents the relationship between them.

Hybrid peer-to-peer file-sharing environments

We focus on hybrid file-sharing environments, especially on widely deployed real-world networks, searching two defined goals. The first goal is to improve content indexation in the BitTorrent file-sharing environment by enabling BitTorrent content to be indexed in the Kad DHT.

1. The Tor network has tripled its user-base in the last three years, while the I2P anonymous network has doubled its user-base in the last year. Statistics from `https://metrics.torproject.org` and `http://stats.i2p.in/`, respectively. Last visited on 08/2013.

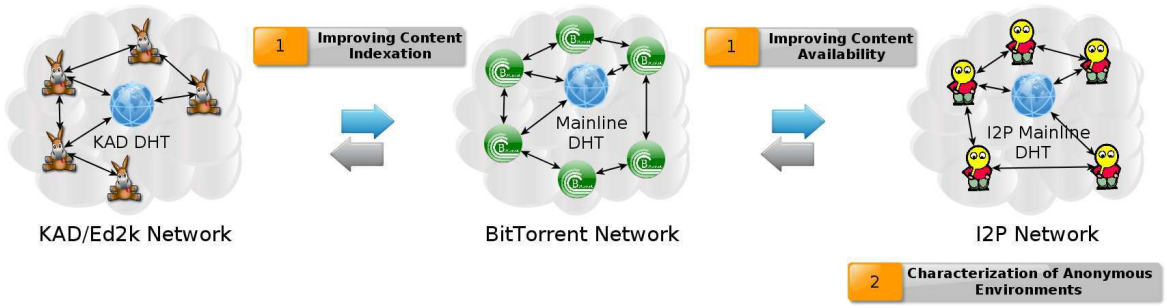


Figure 1.1 – An abstract view of our contributions

The second goal is to improve content availability in the I2P file-sharing environment by enabling I2P users to access BitTorrent content anonymously. This would lead to a fully anonymous file-sharing environment, including anonymous content indexation, as well as anonymous content distribution.

Improving content indexation in the BitTorrent file-sharing environment

The BitTorrent network has shifted to a fully decentralised architecture using a Kademlia-based DHT to support its operation. Although Kademlia is a well-studied and mature system, BitTorrent uses a highly vulnerable implementation which remains open to different attacks. Moreover, the current implementation of the BitTorrent's decentralised architecture enables a single level of indexation, storing peers that are sharing a given content, *i.e.* enabling a content-to-sources mapping.

At first glance, the Kad network has a two-level indexation mechanism, which is sufficient to store BitTorrent's content-to-sources mapping, while additionally providing an extra keywords-to-content mapping. This extra mapping would enable a fully-distributed keyword-based lookup service for the BitTorrent infrastructure, which is currently missing, and would provide a more secure indexation mechanism.

Considering this solution a bit more in detail,we can claim that Kad users would benefit from this scheme as well: the BitTorrent network counts tens of millions of users, who could interact with Kad users, improving their performance of download. Figure 1.2 illustrates this *hybrid* file-sharing architecture, where different *network meeting points* are used to interconnect both networks.

We propose the first hybrid file-sharing model for the BitTorrent and the Kad/Ed2K networks. We introduce a more robust content indexation mechanism within the BitTorrent system, while enabling a bidirectional interaction between Kad/Ed2k and BitTorrent users.

Improving content availability in the I2P file-sharing environment

Current communications have shifted to a more anonymous state, where anonymous file-sharing accounts for a considerable part. We focus on the I2P anonymous file-sharing environment and the interaction with a public environment, with the goal of improving content availability within this anonymous system.

We propose a hybrid file-sharing model, enabling the I2P file-sharing environment to interact efficiently with the BitTorrent network, thus improving content availability within the I2P network. This hybrid model guarantees a fully anonymous content indexation, as well as an

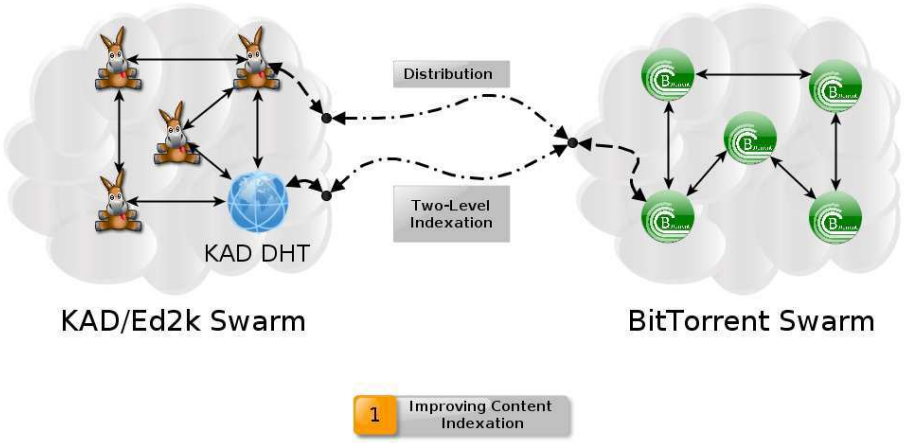


Figure 1.2 – Improving content indexation in the BitTorrent file-sharing environment

anonymous content distribution scheme, where I2P users access can BitTorrent content while maintaining their anonymity in the process.

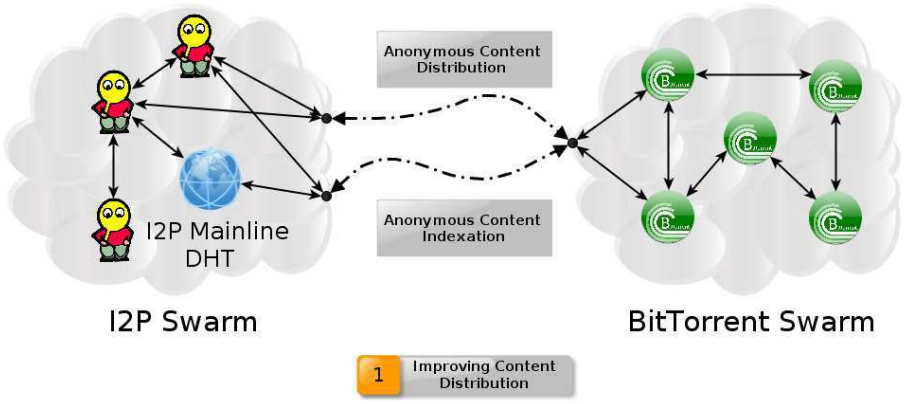


Figure 1.3 – Improving content availability in the I2P file-sharing environment

Figure 1.3 depicts our hybrid file-sharing model, where the same conceptual *network meeting points* are used to allow I2P users to interact with BitTorrent users. I2P users access these meeting points anonymously, achieving an anonymous content distribution. The I2P's Mainline DHT is used to index BitTorrent content, obtaining an anonymous content indexation, corresponding to the second step towards a fully anonymous file-sharing environment.

Characterisation of anonymous environments

We aim at understanding anonymous environments via extensive monitoring. These environments decouple a user's real identity from the assigned system's identity, thus enabling users to access different services anonymously, such as anonymous file-sharing or anonymous web surfing.

We target the entire I2P anonymous environment, which includes a wide range of anonymous services. We consider different aspects of the network, such as the number of users, the number of anonymous services available and the geographical characteristics of the users. Through different

correlation analyses, we were finally able to characterise a small group of users using a specific anonymous service, such as the users from a particular city performing anonymous file-sharing.

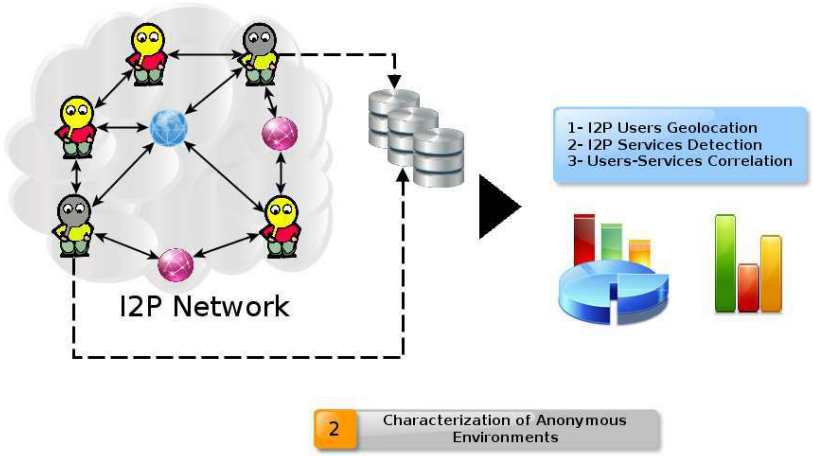


Figure 1.4 – Characterisation of the anonymous I2P environment

Figure 1.4 provides a glimpse of our contribution. Monitoring nodes are distributed throughout the system, constantly collecting basic network information, which is kept in different databases. Data is then processed to generate different network measurements, including correlation analyses between anonymous services and users.

We propose the first application-level study of the anonymous I2P network and provide the first group-based characterisation approach. Our findings have improved the I2P anonymous network to a more secure state, where I2P's designers modified different network parameters, hardening, yet not avoiding, our monitoring approach.

Manuscript organisation

This manuscript is organised in three parts. The first one corresponds to the state of the art, while the last two parts present our contributions.

1.3.1 First Part: State of the art

The first part of this document considers all the concepts and previous works necessary to understand our contributions and their framework. This section is divided into three chapters.

Chapter 2 describes Kademlia-based distributed hash tables, their main components and routing algorithms. We consider two foremost points within Kademlia-based distributed hash tables. On the one hand, we describe different monitoring techniques for these systems, from passive to active approaches, such as *distributed probes* or *crawling*. On the other hand, we deal with security issues within these systems, mainly focusing on two widely deployed implementations, namely the Kad network and the BitTorrent's distributed trackers.

Chapter 3 presents cooperation schemes between overlay networks, more specifically among file-sharing networks. We first introduce different interconnection models for general overlay networks, which generally aim at improving the routing performance. We describe the interconnection of heterogeneous file-sharing networks, *i.e.* networks operating over different protocols.

Finally, Chapter 4 presents anonymous environments currently deployed, like the Tor network and the I2P network. We consider a first approach where anonymity is fitted into a non-anonymous file-sharing network by means of an external anonymous system, *e.g.* by using the Tor network to anonymise the BitTorrent traffic. The second approach focus on dedicated anonymous file-sharing systems, such as Freenet or GNUnet. We describe different monitoring techniques of these anonymous systems, and the legal implications at stake when conducting this kind of monitoring.

Chapter 2 and 3 provide the background concepts for our contribution on hybrid peer-to-peer file-sharing architectures. Chapter 4 introduces the concepts underlying our second contribution: the characterisation of an anonymous environment.

1.3.2 Second Part: Hybrid peer-to-peer file-sharing architectures

The second part of this document describes our first contribution, namely the analysis and evaluation of a hybrid peer-to-peer file-sharing architecture and it is organised in two chapters.

Chapter 5 analyses a hybrid approach between the Kad/Ed2k and the BitTorrent networks, aiming at improving content indexation within the BitTorrent file-sharing environment. We first introduce a comprehensive analysis comparing the Kad DHT and the main BitTorrent's decentralised tracker. Then, we conduct a performance analysis comparing the algorithm of download of both Ed2k and BitTorrent networks. Finally, we bring forward our abstract hybrid file-sharing model, which is instantiated with both the Kad/Ed2k and the BitTorrent networks.

Chapter 6 describes the interaction between the I2P anonymous file-sharing environment and the public BitTorrent file-sharing environment, aiming at improving content availability within the I2P network. We instantiate our abstract hybrid file-sharing model using these two networks, enabling I2P users to access public BitTorrent content anonymously, while still having the capacity to index BitTorrent content within I2P's decentralised tracker, resulting in a fully anonymous file-sharing environment.

1.3.3 Third Part: Characterisation of anonymous environments

The third, and last part, of this thesis describes our second contribution, *i.e.* the characterisation of a widely deployed anonymous environment, and is divided into two chapters.

Chapter 7 introduces I2P's metadata database, a Kademlia-based distributed hash table and detail how we exploited it and placed a set of distributed monitoring nodes on the system. Finally, we show how this metadata was analysed to successfully characterise I2P's users and services.

Chapter 8 analyses the interaction between I2P anonymous and I2P BitTorrent-like clients by applying Pearson's correlation coefficient. Different case studies are conducted to determine to what extent users contributed to the activity of a given application.

Part I

State of the Art

Chapter 2

Kademlia-based hash tables: Principles, monitoring techniques and security issues

Contents

2.1 Introduction

A DHT or *distributed hash table* is a distributed storage system enabling the indexation of `<key,value>` pairs like in a regular hash table. This structure relies on distributed nodes to store data, where every node is responsible for maintaining a subset of all key-value mappings. Due to the dynamic nature of nodes in the system, values are replicated in several nodes to deal with continuous nodes' arrival and departure.

Distributed hash tables are widely used in peer-to-peer infrastructures to support content indexation. For instance, the widely deployed BitTorrent network [17] uses a DHT-based implementation [18] for indexing `<content,peers>` pairs, where every online peer in the network

becomes a node in the distributed hash table. Therefore, by querying the distributed hash table, we can determine which peers are sharing a specific content. The Kad network, another popular peer-to-peer network, is based on a DHT system to support its double-indexation mechanism, where `<keyword,contents>` and `<content,peers>` pairs are stored in the system, enabling a distributed keyword-based search system.

In this chapter, we first introduce the concept of distributed hash table and its principles, followed by a description of the best known DHT architectures, with a specific focus on Kademlia [19]. Then, we describe a set of passive and active monitoring techniques for Kademlia-based systems and different monitoring experiments on the Kad network and the BitTorrent's distributed trackers. We detail how these systems are exploited and which resources are needed to perform such monitoring. Finally, we present different security issues of Kademlia-based DHT.

2.2 Principles of a DHT

We first present the main components of a distributed hash table, its keyspace and the organisation of the participants. We then consider the procedure to store data within the system.

2.2.1 Main components of a distributed hash table

A distributed hash table has basically two main components: a *keyspace* and a *mechanism to organise the participants*.

Keyspace

The keyspace is the set of possible values used to identify the different elements (keys, values or nodes) in a DHT. It is given by the hashing function used to map keys to the associated values. For instance, a SHA1 hashing function [20] leads to a 160-bit keyspace whereas a MD5 function [21] produces a 128-bit keyspace. Within a DHT, the keyspace is split among all nodes on the system. Thus, every node is responsible for a subset of the entire keyspace and stores data related to that section of the keyspace. As participants are more or less dynamic within the system, the keyspace is constantly re-partitioned and re-assigned among current online nodes.

Organisation

DHT nodes are organised in an *overlay network* where a node maintains a *routing table*. To decide whether a node is neighbour with another node, a *distance function* is used. The Kademlia DHT uses the XOR function between two nodes' identifiers to determine how close these two nodes are. Once the nodes in the system are ordered according to their distance, a Kademlia node can maintain its routing table with a bigger set of closer nodes and just fewer distant nodes.

2.2.2 Data storage procedure

A simplified DHT storage and search procedure works as follows: let's assume the MD5 function is used and a video file `my_video.mpg` has to be stored. The MD5 hash of the filename is generated, producing a 128-bit long k key and then a message *STORE(k,my_video.mpg)* is dispatched to the closest node known in the DHT (in term of distance). This node forwards the message until it reaches the node responsible for indexing the key k, who finally stores the file. To search for the video file, the procedure is exactly the same except that a *GET(k)* message is issued, and the node responsible for the key k returns the video file.

To avoid loss of data due to nodes joining or leaving the DHT, some distributed hash table implementations use a *replication* scheme. Considering the previously mentioned example, the video file will be stored in a set of nodes instead of in a single node. This set consists in the x closest nodes to a given key k, where the value of x depends on the implementation of the DHT, normally oscillating within a range of tens of nodes. These characteristics make DHTs a good solution when seeking for a scalable and fault tolerant distributed storage system.

2.3 Kademlia distributed hash table

The first generation of peer-to-peer file-sharing networks was built on central directories and used costly flooding strategies to locate other nodes in the systems. More recent architectures are based on a distributed approach instead, employing distributed hash tables. The most relevant DHT designs include Kademlia, Chord [22], CAN [23] and Pastry [24]. We only study the Kademlia protocol, since it is the protocol of the peer-to-peer system under consideration in our thesis.

Kademlia is a distributed hash table based on the XOR logic function to compute distances between nodes. It was proposed in 2002 and since then it has been applied in different real-world systems, such as the Kad network.

2.3.1 Keyspace and k-buckets

Kademlia uses 160-bit long IDs for nodes, keys and values, which are obtained from the SHA1 hashing function. In order to compare two different identifiers, the XOR logic function is applied as follows: given ID_1 and ID_2, the distance between them is the $d(ID_1,ID_2) = ID_1 \oplus ID_2$ and interpreted as an integer value.

Kademlia's routing table is organised according to the distance between nodes' identifiers. Assuming a 160-bit keyspace and for every i, where $0 \leq i <160$, a node maintains different lists up to k of other nodes with distance between 2^i and $2^{(i+1)}$ from itself. These lists are called k-buckets. Nodes are widely spread in the keyspace and therefore lower k-buckets, *i.e.* for small values of i, contain fewer nodes, while higher k-buckets contain more nodes.

Figure 2.1 shows an example of a node's routing table. Let's consider a 3-bit keyspace, resulting in a maximum of 2^3 nodes and let's assume $k = 3$, obtaining *3*-buckets and a maximum of three nodes per bucket. We are analysing node **001**, which knows all nodes on the system, except for nodes **011** and **110**. Considering that the keyspace is 3-bit long and therefore $0 \leq i <3$, we have three different 3-buckets which we will call *3*`-bucket(0)`, *3*`-bucket(1)` and *3*`-bucket(2)`. The first bucket stores nodes' IDs with 0 bits in common with the node identifier ; the second bucket stores nodes' IDs having a *prefix* of `0XX` (only the first bit in common) ; finally the last bucket stores nodes' IDs with prefix `00X` (the first two bits in common).

Kademlia routing table is updated in an opportunistic manner: every time a node receives a message from a remote node **N**, it updates the information of **N** in the routing table. In our example, if node **001** receives a message from node **110**, which is not yet known, the routing table needs to be updated. Node **110** is a candidate for the *3*`-bucket(0)`. However, this bucket is full and assuming nodes **100**, **101** and **111** are always responding to `PING` messages, the new node **110** will not be inserted in the routing table.

2.3.2 Kademlia operations

There are 4 different messages in the Kademlia network, `PING`, `STORE`, `FIND_NODE` and `FIND_VALUE`.

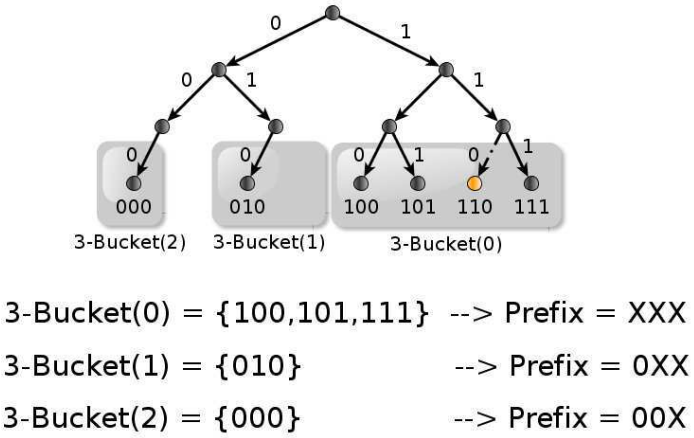


Figure 2.1 – An example of a Kademlia routing table and a three-bit long keyspace

The *ping* message checks whether a node is online or not. The *store* message informs a node when it has to store a value. A *find_ node* message allows a peer to retrieve a list of nodes close to a key. A Kademlia node receiving a *find_ value* message returns the same information as in a *find_ node* message, except if this node has previously stored a value for that key, in that case it returns the stored value.

Locating the closest nodes to a given key is the most important procedure in Kademlia and is called a *node lookup*. It consists in recursively searching for the k closest nodes to a specific key. To improve fault tolerance, Kademlia uses α parallel *find_ node* messages, where α, along with k, are system-wide values, normally $k = 20$ and $\alpha = 3$.

Kademlia's storage procedure is implemented on top of the node lookup process by conducting a node lookup first, and once a list of the k closest nodes is retrieved, a *store* message is issued to each node.

2.3.3 Management of nodes' arrival and departure

Kademlia has an effective and loose keyspace management, where the XOR logic function is used to compute distances and determine which set of nodes has to store a given key. This feature along with a regular republication of keys, which a user is responsible for, maintains a simple routing approach. There is no keyspace reassignment upon a node departure or arrival. However, by replicating data stored on the system, Kademlia efficiently deals with churn, greatly reducing the load of keyspace management [25].

However, in the unlikely case that all Kademlia nodes storing a given key fail, the key will be unreachable until a new re-publication takes place. For high-demanding peer-to-peer applications, an unreachable key can be a major inconvenient. An active mechanism to maintain the routing table updated is more suitable for critical distributed applications, where a key needs to be anyhow accessible. Therefore, Kademlia-based distributed hash tables are a good option for non-critical distributed systems, such as the BitTorrent's distributed trackers.

2.3.4 Current implementations

We take into consideration two widely deployed implementations of Kademlia-based systems.

The Kad network

The Kad network is probably the most known of Kademlia implementations. The Kad network [2] is a peer-to-peer file-sharing network which uses an UDP-based Kademlia implementation to support its double-indexation mechanism. Pairs `<keyword,contents>` and `<content,sources>` are stored in the network, where a user can search for a content using a keyword (first level of indexation) and then, once the desired content has been found, can search for the content's sources (second level of indexation).

The Kad network implements the Kademlia protocol through UDP messages, along with a 128-bit keyspace generated by the MD5 hashing function. One major difference with the original Kademlia protocol lies in an interactive node-lookup process, instead of the initially proposed recursive procedure. A second difference lies in a systematic process to update its routing table, where every k-bucket is checked once an hour.

The BitTorrent network

The BitTorrent network is another peer-to-peer network that employs a Kademlia-based implementation to support its indexation mechanism. The Mainline DHT [26] is the BitTorrent *trackerless* approach, which aims at replacing its central trackers. The Mainline DHT enables a single level of indexation, contrary to the Kad network, enabling only `<content,sources>` pairs to be indexed. This forces BitTorrent's users to use an *off-band* mechanism to replace the missing first level of indexation, where torrent files are retrieved according to keywords.

A second trackerless Kademlia-based implementation named Vuze DHT [27] is available as well. This DHT is similar to the Mainline DHT. However, the Vuze distributed hash table is only used by the Vuze client (formally named Azureus) and is therefore inaccessible to the rest of BitTorrent's clients.

2.4 Approaches for the monitoring of Kademlia-based DHTs

Due to the wide deployment of Kademlia-based systems, several monitoring techniques have been proposed. In this section, we review approaches for monitoring real-world Kademlia-based implementations: the Kad network and the BitTorrent's distributed trackers, aiming at evaluating how effective these approaches are and what kind of data can be retrieved from the network.

2.4.1 Passive monitoring techniques

In passive monitoring, network behaviour is not disturbed or modified by the monitoring itself. The monitoring nodes behave like normal network nodes and they do not generate extra network traffic nor disrupt the existing nodes.

This approach consists in one or more *distributed probes* collecting and storing network traffic as it passes through the nodes, such as the network messages, the seen nodes or the uptime of nodes. The main advantage of this approach is the *transparency* during monitoring. Since there is no active involvement from the monitoring nodes, monitoring can hardly be detected. However, the main disadvantage of this approach is that idle nodes do not send any kind of information and therefore are not detected during the monitoring, leading to a partial view of the network.

2. The Kad network is the distributed hash table supporting content's indexation for the eDonkey network. This content indexation/distribution scheme is used for different peer-to-peer client, such as aMule and eMule. Throughout this document we refer to this scheme as Kad/Ed2k or simply as the Kad network, even if it includes the eDonkey network as well.

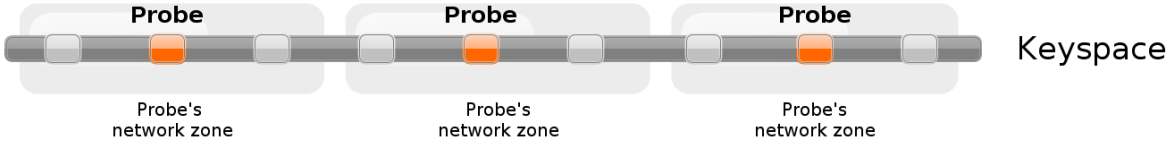


Figure 2.2 – Distributed probes and their assigned network zones

In a distributed environment like Kademlia, the only feasible manner to conduct a passive monitoring is to be part of the network itself. A straightforward example is the built-in population estimator of the aMule client, where an estimation of the number of users in the network depends on the density of nodes in the client's local routing table. Moreover, based on his/her routing table and in the normal protocol operation (like the regular *ping* messages issued by the client), the client determines the uptime of every seen node. Thus, the client computes a set of statistics about the network, only from information gathered by participating on the system.

Distributed probes

Figure 2.2 shows the main concept of distributed probes: each probe gathers information about its network zone. Thus, by placing a group of probes around the network, we can obtain a general view of it.

Distributed probes require that the monitored network allows nodes to be freely placed. Kademlia enables to freely choose a node identifier and therefore its *placement* within the distributed hash table. Thus, the distributed probes can be placed perfectly in particular network zones. Choosing a node's identifier freely in the network has a second advantage: probes can target specific network zones. In Kademlia, a distributed probe will gather information around its own part of the keyspace, of each *store* message it receives, for instance.

In the Kad network, Memon *et al.* [28] developed a distributed monitoring approach called *Montra*, based on distributed probes. Their approach aimed at reducing network disruption when capturing network traffic. These monitoring nodes, or *minimally visible monitors* according to the authors, are designed to target a specific point in the Kad keyspace by only answering messages from the target and ignoring the rest of messages from the network, thus becoming invisible to the rest of nodes.

Steiner *et al.* [29] proposed a less elaborated monitoring approach for Kad called *Mistral*. Mistral is similar to Montra in the sense that it uses a large number of distributed probes to capture network traffic. However, it lacks any analysis of nodes placement, resulting in twice as much monitoring nodes than Montra and producing a major disruption in the network.

Cholez *et al.* [30] conducted a content-based monitoring on Kad based in few ($\sim$ 20) distributed probes, using the PlanetLab testbed[3]. The authors targeted a specific key in the Kad keyspace, which was mapped to a value/content, similar to the Montra tool. As a result the authors were able to measure efficiently different characteristics for a specific content, such as the first publish message to successive request messages and download messages.

Falkner *et al.* [32] conducted a set of measurements on the Vuze DHT, providing insights about the churn, the messages overhead, the routing table inconsistencies, and the lookup performance, among others. The authors used, although briefly detailed, a distributed monitoring architecture, which included as much as 250 PlanetLab nodes.

3. PlanetLab is a research testbed with currently 1000+ distributed nodes [31].

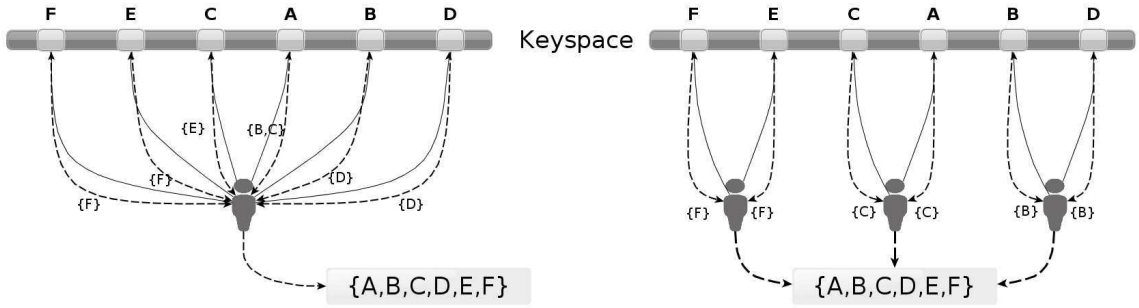


Figure 2.3 – Centralised/Distributed crawling

Increasing the number of distributed probes around the network increases the total network coverage and therefore improves the monitoring results. However, a large number of probes in the network changes the *density* of nodes in the network, altering the real image of the network as determined by Memon *et al.* [28]. Therefore, the number of distributed probes is a tradeoff between the network coverage sought and the network distortion we are willing to accept.

2.4.2 Active monitoring techniques

In active monitoring, dedicated nodes actively scan the network and probe every node. However, as the monitoring nodes do not behave as normal nodes producing a high number of outgoing messages and connections to external nodes, the collected data can be biased and may not represent the network state properly.

Crawling

Crawling is one of the best known active monitoring techniques. The monitoring nodes interactively discover new nodes through neighbours of known nodes. That way, exploring the network graph leads to a network *snapshot*: a precise view of the network at a particular point of time. The distribution of values among the nodes on the network is not disturbed by this active monitoring approach, because the data stored in every node is not re-assigned to other nodes. Yet, additional messages are issued from the monitoring nodes, thus increasing network traffic.

Crawling can be performed from a single machine or from several distributed machines. The first case is the most usual, since it does not require any type of aggregation among distributed nodes and generally achieves good performance in analysing sections of the network. On the contrary, distributed crawling can target several disjoint parts of the network at the same time, increasing network coverage but incurring in additional aggregation costs. An alternative of a full crawl is a *partial crawl*, where only a particular region of the network is analysed. Figure 2.3 illustrates a simplified view of these two approaches, where the keyspace is represented as a linear space only for ease of visualisation. In the centralised crawling, the crawler knows node `A` and contacts it to discover further nodes. Node `A` responds with two nodes, `B` and `C`. The crawler contacts these two nodes to discover further network nodes. This procedure is repeated until no new nodes are discovered. In a decentralised crawling, every distributed crawler performs the same procedure, but for a smaller part of the network.

Crawling has different requirements. On the one hand, it can only be done if the network nodes can be inquired remotely: Kademlia nodes accept a *find_ node* message, returning a list of

the closest nodes known for a given key, which allows a monitoring node to discover new nodes. Crawling cannot be carried out in a network where messages are accepted only from a particular subset of nodes, like in *friend-to-friend* networks. On the other hand, crawling is expected to generate a *snapshot* of the network. Therefore, if this process is slower than changes occurring in the network, such as its topology, the resulting snapshot might not be accurate enough. In that case, a distributed crawl or a partial crawl can be used to speed-up the process.

In Kademlia-based networks, crawling is achieved by iteratively issuing *find_ node* messages to discover nodes and gathering their neighbours. This process is repeated until no new nodes are found and every node has been queried. If the dynamic of peers is slower than the crawl itself, the resulting *snapshot* is a complete view of the network's users.

Regarding the Kad network, Stutzback *et al.* [33] conducted a complete study on the Kad network to characterise *churn*[4] in the network based on a distributed crawler called *Cruiser*. Cruiser operates a master-slave architecture, with different distributed instances of the crawler that analyse disjoint sections of the Kad DHT, improving crawling performance through parallelism.

Steiner *et al.* [34] conducted an active monitoring in the Kad network based on a straightforward crawler called *Blizzard*. They successfully crawled the entire Kad keyspace using two different instances of the same crawler from two different physical machines.

Considering the BitTorrent's distributed trackers, Wolchok *et al.* [35] developed *ClearView*, a centralised crawler for the Vuze DHT. ClearView deploys a large number of monitoring nodes in the network from a single physical machine targeting a specific network region. After a certain period of time (usually 30 minutes), these monitoring nodes are re-located in the network, targeting a new region, thus crawling the entire Vuze DHT from a single machine, in a hop-by-hop fashion.

Jünemann *et al.* [36] presented *BitMON*, a centralised partial crawler for the BitTorrent's Mainline DHT. BitMON repeatedly crawls a single partition of the network and determines the size of the network, the country of the users, and the most used ports, among others. BitMON automatically carries out an analysis of the data gathered and plots it.

Steiner *et al.* [37] extended their Kad crawler, *Blizzard*, so that it could operate with the Vuze DHT. They presented statistical values of the network, such as the uptime of nodes, the software versions and the geographical distribution of nodes.

2.5 Security issues on Kademlia-based DHTs

Security analyses have been extensively carried out in Kademlia-based DHTs, from single-machine attacks to more complex distributed attacks. Along with these studies, several solutions have been proposed to cope with these attacks. In this section, we start with briefly reviewing a general attack in the context of DHTs, namely the Sybil attack. Then we consider attacks on real-world implementations of the Kademlia protocol, namely the Kad network and the BitTorrent's distributed trackers.

2.5.1 The Sybil attack

The Sybil attack, first introduced in 2002 by Douceur [38], is probably the most best known attack in the literature against DHTs, even if it does not only target these systems. In its basic

4. *Churn* or *Churn rate* can be interpreted as the dynamic of peers, *i.e.* the arrival and departures of participants, and is one of the main characteristics in peer-to-peer environments [33].

form, a Sybil attack takes advantage of the lack of entity-to-identity management, where a physical entity can create several logical identities, taking control of a portion of the system, or even of the entire system. Douceur states that, without a central authority in charge of issuing identities, it is almost impossible to avoid this kind of attack on distributed environments such as peer-to-peer networks.

Urdaneta *et al.* [39] conducted a rather complete survey on defence mechanisms against the Sybil attack. The authors classified the defence mechanisms into six groups: centralised certification authorities, distributed certification mechanisms, identification schemes based on physical network characteristics, social networks, computational puzzles and game theory approaches.

One solution against a Sybil attack is the use of a central component for identity authentication. Centralised authorities are supposed to be trusted by every network participant and they should be able to ensure a valid entity-to-identity mapping on the system. Nevertheless, a central authority is an obvious target of attacks and, along with its high maintaining costs, it is not always suitable as a solution against Sybil attacks.

A distributed approach seems to better fit a DHT than a central authority but, due to the lack of mutual trust in a distributed environment and to the reduced view of the network, as stated by Urdaneta *et al.*, these defence mechanisms are unable to fully prevent a Sybil attack, only mitigate its effects. Dinger *et al.* [40] presented a distributed approach called *self-registration*, where node identifiers are assigned in a distributed fashion according to their IP addresses, yet leading to a probabilistic Sybil-resistant approach. Mashimo *et al.* [41] improved the former approach by employing a local trust mechanism, maintaining the Sybil protection even against an important increase of attackers.

Network characteristics, such as round-trip-times from different network nodes, can be used to assess whether two logical identities correspond to a single physical entity or not. Wang *et al.* [42] proposed the use of a *net-print* to identify a node within a network, where a local round-trip-times vector is computed and compared with another vector from random nodes. A cheating node can be detected by comparing both vectors.

The use of social information as an enhancing mechanism has been proposed to deal with Sybil attacks. These approaches bind a logical identity to a physical entity, avoiding multiple fake identities from a single entity. Yu *et al.* [43] proposed *SybilLimit*, an improved solution to their previous *SybilGuard* protocol [44], which uses existing social information (such as an extended version of the DBLP bibliography database) to develop a Sybil-resistant routing protocol.

Computational puzzles, also introduced as *Resource testing* by Levine *et al.* [45], aim at proving network nodes with highly-demanding computational puzzles, assuming that an attacker has less computational resources than a normal network node, since several logical identities are dependent on the same physical entity. However, the computational puzzles approach might only work if all identities are proven in a simultaneous way as stated by Douceur [38], and it requires that honest nodes continue to compute these puzzles in order to remain on the system.

Finally, game theory-based approaches [46, 47] pretend to impose a utility model, which often needs to use a sort of a currency in the system. Since the utilisation of a money currency within a distributed environment is highly complex, these approaches have remained theoretical and have not been deployed in any large distributed system.

The next two sections introduce real-world attacks on large deployed DHTs, along with their solutions.

2.5.2 Attacks in the Kad network

The Kad network has been studied from different angles, including its lookup protocol, content retrieval performance and security aspects. In this section, we focus on the security issues of the network, which have been widely investigated, leading to major changes in the protocol during the last years. We concentrate on three attacks, namely a *Spy* attack, an *Eclipse* attack and a *DDoS* attack.

A Spy attack can be considered as an improved passive monitoring, since it does not affect the behaviour of the network, nor the Kad nodes. It is only intended to gather information about a particular network zone, such as publish or search requests. An Eclipse attack, on the contrary, has an impact on the network. Sybil nodes are placed closer than any other node to a specific key in the Kad keyspace and, due to Kad routing protocol, every message converges to the Sybil nodes. These Sybil nodes can choose not to answer these requests, producing an eclipse on that particular key: normal nodes searching for that key will not get any answer. Finally, a DDoS attack can be achieved by placing Sybils in popular zones in the DHT (Sybils close to a popular movie ID, for instance) and including the IP address of the target in their routing response. That way, the target machine receives a high amount of Kad traffic, which can be regulated by shifting the Sybils to more or less popular network zones.

Steiner *et al.* [48] conducted a partial crawl in the Kad network for a six-month period and discovered an ongoing Sybil attack during their experiments. As mentioned before, the lack of an identifier [5] management scheme makes the network prone to that kind of attack. The authors deployed their own Sybil nodes from a single machine, where they achieved a Spy attack, an Eclipse attack and a DDoS attack.

Yu and Li [49] performed a DDoS attack in the network, highlighting the lack of identity management in the Kad's routing table. By firstly crawling the network and secondly spoofing different protocol messages, the authors successfully achieved a DDoS attack in their own target machine, with both TCP and UDP protocol connections. This study differs from the previous DDoS attack by Steiner *et al.*, since it does not use a set of Sybils nodes to announce the target IP address, but it crawls the network and spoofs different request messages. By simply spoofing a message, a new node (the target node) can be added in a remote node's routing table.

Wang *et al.* [50] presented a spoofing-based attack in the network, showing that an attacker can *overwrite* legitimate IP addresses in a victim's routing table by sending Kademlia *PING* messages with a spoofed identifier. Alternatively, the authors proposed a *reflection attack*, for which the victim's routing table is populated with the victim's own IP address. However, the study took into consideration a basic client version (eMule 0.48a), which does not have any protection mechanisms and enables a single-machine attack. In newer versions of the client, that attack is not possible.

Cholez *et al.* [51] measured Kad's latest protection mechanisms and how these mechanisms cope with the set of attacks previously introduced by [48] and [50], among others. They showed that those protection mechanisms, which include *flood protection*, *IP limitation*, and *IP verification*, can successfully prevent a single-machine attack, where a set of Sybil nodes would be deployed from a single physical machine. However, the authors showed that a distributed attack using the PlanetLab infrastructure is still possible.

Kohnen *et al.* [52] conducted an Eclipse attack in the network, considering the same clients as the former approach by Cholez *et al.* and their protection mechanisms. This Eclipse attack exploits the Kad's publishing process, where a set of distributed Sybil nodes are placed close to the target key, attracting every Kad request. Additionally, these Sybil nodess use a promotion

5. We use *identifiers* and *Kad IDs* interchangeably when referring to the Kad network.

process among themselves: they include the identifiers of other Sybil nodes in routing responses, ensuring that every Sybil node gets known while routing.

In continuation of their previous work, Cholez *et al.* [53] introduced an identifiers' distribution analysis, where they compared the real distribution of peers with the theoretical one in the Kad keyspace, successfully detecting attacking nodes in a network zone. They additionally proposed a set of countermeasures in order to progressively remove those attacking nodes in Kad's routing process, which requires minimum changes in the current client in order to be deployed in the real network. However, and as mentioned by the authors, their analysis, which considered a fix number of users in the network, does not represent a dynamic environment like the Kad network where the total number of users changes all along the day.

2.5.3 Attacks in the BitTorrent distributed trackers

Security attacks in the BitTorrent network have been widely studied [54, 55], mainly targeting its centralised architecture, which includes a central tracker to coordinate peers. However, this system has shifted to a decentralised architecture. We consider its decentralised trackers, namely the Mainline DHT and the Vuze DHT, and introduce the most relevant studies on these components. Due to the Kademlia-based nature of these decentralised trackers, some of the attacks proposed in Kad have already been deployed in the BitTorrent network with few modifications.

Wolchok *et al.* [56] conducted a Sybil attack on the Vuze DHT so as to defeat the Vanish system [57]. The Vanish system enables the creation of data that will become inaccessible or vanish after a specific time period by encrypting the data and storing the encryption keys in the DHT as with any normal `<key,value>` pair. The Vuze DHT requires that these pairs be periodically republished, or the pair will be otherwise dropped from the network. By intentionally not republishing those pairs, the encryption keys will be lost, and the data previously encrypted with these keys will be therefore inaccessible as well. Wolchok *et al.* used their ClearView crawler to implement their attack against the Vuze DHT, showing that a basic Sybil attack was enough to retrieve the encryption keys. They concluded that a public DHT, such as the Vuze DHT, was not suitable as a building block for security-sensitive applications.

Wang *et al.* [58] presented a monitoring study in the BitTorrent's Mainline DHT based on few distributed *honeypots*. Their study brought forward a large ongoing Sybil attack in the network and, even if the authors did not carry out an attack by themselves, they showed how this distributed tracker was under attack. They detected a set of Mainline nodes, corresponding to a large international ISP, performing traffic localisation[6]. They additionally detected another Mainline node injecting a great amount of Sybil nodes in the network to monitor content requests. Neither of these attacks damaged the network. However, the authors showed how weak the network was in terms of security protections.

The Kad DHT and the Mainline DHT are both based in the Kademlia DHT, however they present different security issues. The Kad DHT has been widely studied and different protection mechanisms have been presented. These protection mechanisms make the network resilient to different attacks from a single-machine attacker. A identifiers' distribution analysis has been proposed to deal with a distributed attack. BitTorrent's distributed trackers have been proven open to attacks, but no security solutions have been proposed.

6. *Traffic localisation* allows I2P operators to reduce inter-ISP traffic by encouraging peers to connect to other *local* peers, *i.e.* within the same ISP, as proposed by Varvello and Steiner [59], among other authors.

2.6 Conclusion

We presented different monitoring studies applied to Kademlia-based systems, including active monitoring techniques, such as *crawling*. Monitoring is possible in both Kademlia-based systems, namely BitTorrent's distributed trackers and the Kad network. Several authors analysed different aspects of both networks: the average size of network, the average uptime and geographical distribution of users, the churn rate and the messages overhead. In most cases, a single-machine *crawler* proved sufficient to analyse the network, although a distributed approach, like the work carried out by Stutzback *et al.*, could cover a wider part of the network.

Different attacks have been successfully carried out in the BitTorrent network. Even with the previous research works performed on the Kad DHT, not all protection mechanisms have been introduced in the system. Crosby and Wallach [60] stated that despite the security flaws known and the ease to launch an attack in both BitTorrent's distributed trackers, these two Kademlia-based systems were suitable for the BitTorrent infrastructure, and its peer discovering mechanisms. However, due to the recent shift of BitTorrent to a fully decentralised architecture, it is necessary to improve the BitTorrent's decentralised trackers and toughen their security protections.

Our thesis is focused on both aspects, security and monitoring, for both networks. In Chapter 5, we conduct a security and performance analysis of the Mainline DHT, aiming at improving BitTorrent's content indexation scheme. Our approach is based on a hybrid scheme, where the Kad network is used to index content, while the BitTorrent network is used to distribute content. The next chapter introduces different architectures for cooperative overlay networks and hybrid file-sharing architectures.

Chapter 3

Cooperative overlay networks and hybrid peer-to-peer file-sharing architectures

Contents

3.1 Introduction

More and more, software applications build up their own services on top of the Internet infrastructure, create their own overlay networks, such as peer-to-peer file-sharing networks. Considering a cooperation scheme among overlays is an interesting approach in order to maximise the overall network performance. A cooperation scheme where several file-sharing overlay networks are interconnected would enable users to discover different resources (different files, for instance) or to improve the download performance (more available peers, for example).

In this chapter, we study different overlay interconnection schemes, considering additionally heterogeneous networks. We first present interconnection architectures for general overlays and then focus on interconnection schemes among peer-to-peer file-sharing networks.

3.2 Cooperation among heterogeneous overlay networks

A wide set of applications uses overlay networks to support their operations, such as content distribution environment, peer-to-peer file-sharing networks or anonymous networks. Due to the

wide deployment of overlay networks, it becomes important to analyse whether it is feasible that these network cooperate among themselves, improving their own quality of service. The principal idea behind overlay cooperation is to exploit remote hosts with particular characteristics and use them among different overlays: a host from overlay A might find a specific content located only in overlay B, for example. However, overlays tend to be heterogeneous in terms of performance goals: a streaming-like overlay optimises latency, while a file-sharing network optimises bandwidth. Therefore, these heterogeneous performance goals need to be taken into account in an interconnection scheme.

This section introduces a general overview of overlay interconnection schemes. We first review three main network architectures aimed at easing the interconnection between possibly heterogeneous overlay networks. Then, we consider hybrid network models, which improve the organisation of overlays' interconnection.

3.2.1 Synapse

Liquori *et al.* [61] introduced *Synapse*, an information retrieval protocol based on the interconnection of heterogeneous overlay networks. The interconnection model is based on *synapses*, which are nodes acting as bridges between several overlays and forwarding data packets from/to the overlays they belong to. These overlay networks can be structured (such a Chord-based network), unstructured, or hybrid networks. The authors proposed two approaches for internetwork routing: a *white box* and a *black box* protocol. A white box protocol is used when interconnecting collaborative overlay networks, *i.e.* networks that can modify their routing protocol to include Synapse information (replication and routing strategies, for example). A black box protocol interconnects non-cooperative overlay networks. In this case, all the whole Synapse information needs to be carried in an additional control layer, the *synapse control network*. This is a DHT-based metadata overlay, serving as a temporary container of the protocol data.

The authors simulated their network, considering only the white box Synapse protocol and Chord-like structured networks as candidates for the interconnection. The Synapse protocol proved more performant when compared to a basic Chord-like implementation, considering latency, exhaustiveness (ratio of successful data lookups) and amount of messages generated by the Synapses. In this analysis, Liquori *et al.* showed that increasing the number of Synapses did not necessarily improve the lookup latency in the same proportion, which means that the Synapse architecture does not need a high number of Synapses to efficiently work.

Liquori *et al.* chose a highly dynamic scenario in terms of churn rate, which represents the current peer-to-peer file-sharing environments and showed that facing the same churn rate, a Chord-based network is less robust than a Synapse-based network: considering that any node in the system can fail with a 0.25 of probability, a Chord-based network can barely satisfy 10% of the requests, while this ratio increases to 50% in a Synapse-based network with two Synapses.

3.2.2 Sinergy

Kwon and Fahmy [1] studied diverse cooperation schemes among *co-exiting autonomous overlays*, considering heterogeneous networks, scalability problems, as well as security issues. The authors determined whether it was possible for autonomous overlays to cooperate, whether this interconnection improved the considered overlays and finally, whether a cooperation could serve as a basis to test and deploy new Internet services.

The authors proposed *Sinergy*, a proof-of-concept architecture that enables transparent internetwork cooperation based on a set of *agents*. These agents enable cooperative routing among

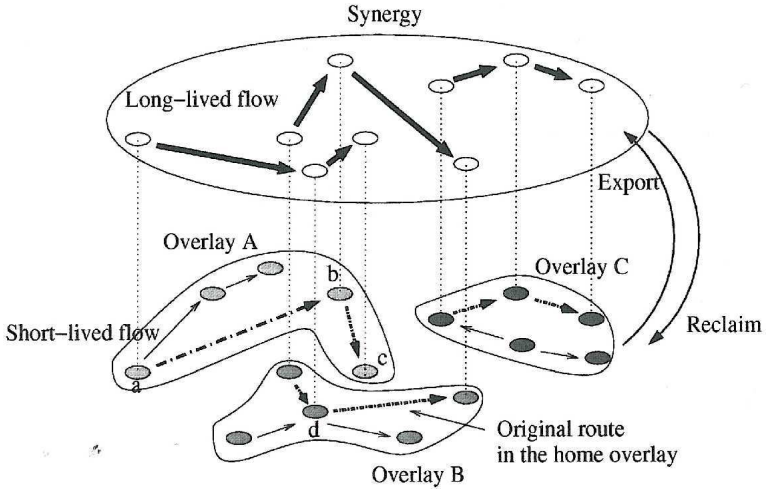


FIGURE 3.1 – Synergy architecture. Reprinted from [1]

overlay networks, in a way similar to the previously mentioned Synapses. Agents are chosen by each overlay according to different system heuristics (nodes with a high number of neighbours, well-located nodes within an overlay, etc.) and placed in an overlay mesh, the *agent network*. This overlay constitutes the core of Synergy and forms alternative and more efficient routing paths, which can be used instead of local overlay routing paths. Figure 3.1 introduces Synergy architecture: selected overlay hosts are *exported* to Synergy, forming the overlay mesh, where alternative routing paths are created. An exported host maintains two routing tables, one for its own ho.me overlay and one for Synergy.

Synergy computes its routing paths independently from the routing paths in the interconnected overlays, which allows overlays with different performance goals to cooperate. A Synergy agent maintains linking properties to every other agents with different metrics: throughput is suitable for file-sharing applications, while low-latency is more appropriate for real-time applications, for example.

Know and Fahmy conducted a performance evaluation on the PlanetLab testbed, deploying Synergy over 8 overlays (each of them containing 8 users). They showed that Sinergy introduced a minimum overhead (72 Kb/s for the conducted experiment), while improving latency, throughput and data packet loss for roughly half of the nodes on the system when compared with independent home overlays. The remaining 50% of the hosts showed slight improvements during their experiments.

Synergy takes into consideration heterogeneous overlays, where their routing goals clearly differ from one another, an important point when interconnecting overlays. However, the authors did not specify how non-cooperative overlays[7] could interact and how Synergy would cope with this scenario, contrary to Liquori *et al.*, who proposed an alternative routing protocol to deal with those non-cooperative overlays.

3.2.3 Network Symbiosis

Previous studies considered interconnection schemes in an *ad-hoc* manner, where network nodes establish connections with nodes from other overlays. However, it is not clear when, and

7. The term non-cooperative correspond to the definition by Liquori *et al.* [61].

under which circumstances, establishing these connections. The way a network evolves through time and how it deals with nodes' departures and arrivals are important issues to consider.

Wakamiya *et al.* [62] studied the interconnection of overlay networks from an abstract perspective, considering a biological-based model to delineate the interconnection scheme. The authors stated that cooperative overlays evolved through time, interacting among them and changing their internal structure (*i.e.* their network topology), eventually developing strong relationships and achieving a mature interconnection model or an *overlay network symbiosis*.

In their interconnection model, networks are dynamic. Each node decides to establish a connection with another node in a different overlay, contrary to the Synergy approach, where the network itself chooses its own interconnection nodes or *agents*. A node considers its surroundings and its own needs to dynamically establish these connections. In a file-sharing environment, for example, a peer holding a large number of files might decide to collaborate with the system by initiating new interconnections and making these files available. Additionally, and opposed to Synergy, an interconnection or *link* can be terminated by the interconnection node if the link does not provide the benefit required: a node with a considerable bandwidth consumption might decide to terminate the link. As stated by the authors, depending on which applications are deployed on top of the overlays, different forms of symbiosis can be considered: *mutualism* (both overlays benefit from the interconnection), *commensalism* (only one benefit while the other stays unaffected) and *parasitism* (one overlay benefits while the other gets negatively affected).

The authors evaluated their model by simulation. They concluded that an interconnection carried a considerable load in the internetwork nodes so that, to efficiently work, further mechanisms to reduce this load needed to be considered, such as caching data in interconnection nodes, as proposed by Konishi *et al.* [4].

3.2.4 Organising the interconnection architecture

The biologically-based study presented by Wakamiya *et al.* organised a network cooperation on decisions made by single nodes: they stated that, if a node considered it could help the system (for example by providing a high bandwidth channel), it could start an interconnection (as well stop an existing one). Although this feature improved the internetwork model, an organisation of interconnected nodes is still lacking. Placing these nodes into a mesh-like overlay, as Synergy does, is a first step.

The following studies propose organisation schemes, where structured and unstructured peer-to-peer networks are merged into a single model, bringing together the best of both approaches.

Two-tier hybrid model

Yang *et al.* [63] proposed a two-tiers data-sharing hybrid model. A structured Chord-based network served as the backbone of the system, while unstructured Gnutella-like networks were linked to the backbone and used it for communication among themselves, as illustrated in Figure 3.2.

The authors stated that the Chord-based backbone provided an efficient lookup service, while the unstructured networks a flexible lookup service based on a best-effort approach. When a data lookup is issued, it is first bounded to the backbone to find the corresponding unstructured network and then forwarded to this network, where a flooding-based search mechanism is used to find the correct result. The flooding-based (or alternatively a random walk) approach generates a considerable amount of network traffic. However, it greatly eases the departure and arrival of new nodes and provides a robust system against churn, an intrinsic property of current peer-to-

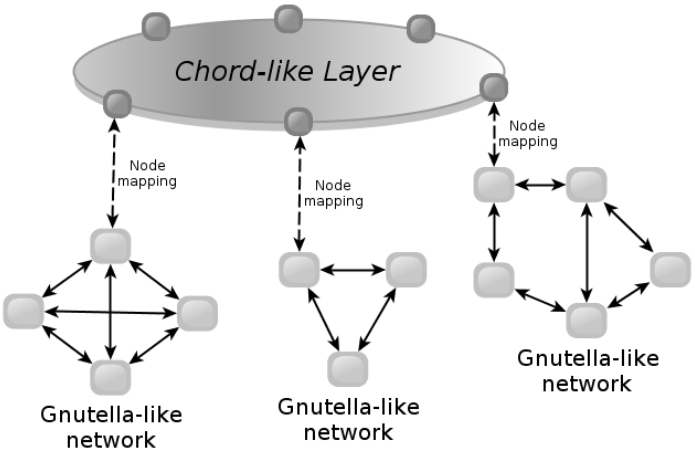


Figure 3.2 – Yang *et al.* system with a Chord-like upper layer and a Gnutella-like lower layer

peer systems. This model additionally includes a tunable system parameter, to determine the percentage of peers in each tier. The authors showed through simulations that 30% of the total nodes in the backbone was an optimal value, increasing data lookup efficiency and reducing data lookup latency.

The authors considered that a caching system could be a good solution to cope with requests for popular data. Additional improvements to this hybrid model are discussed (although not implemented), such as link heterogeneity (connecting a high capacity peer to multiple peers with low link capacities), topology awareness (matching the logical topology with the actual physical topology), content awareness (peers with the same content preferences can be placed in the same Gnutella-like network, thus improving data lookup), internetwork links (a temporary link can exist between Gnutella-like networks to alleviate the load on the Chord-based network) and BitTorrent-style networks (instead of Gnutella-like, a BitTorrent-like network is used, where the node belonging to the Chord-based network acts as a tracker).

Hierarchical Content Distribution Network

Jiang *et al.* [2] proposed a hybrid content distribution model based on a content delivery network (CDN for short) and a peer-to-peer network, named HCDN (Hierarchical Content Distribution Network), as shown in Figure 3.3. This model consists in a first layer of *central servers* and *edge servers* and a second layer of *user nodes*. Central servers behave as in the traditional CDN architecture, where the content to be distributed is firstly stored. Edge servers act as hot spots, where highly popular data is replicated and stored, the same was as in surrogate servers in CDNs (although within HCDN, an edge server can additionally share data with another edge server). Finally, user nodes connect to edge servers, as well as with other user nodes to form a peer-to-peer network and exchange data.

In the CDN layer, content is replicated from central servers to specific edge servers. When the number of replications reaches a threshold, content can be removed from central servers, increasing the storage capacity available. Existing replication strategies from CDN architectures, such as the heuristic distributed algorithm by Wauters *et al.* [64], can still be used in HCDN.

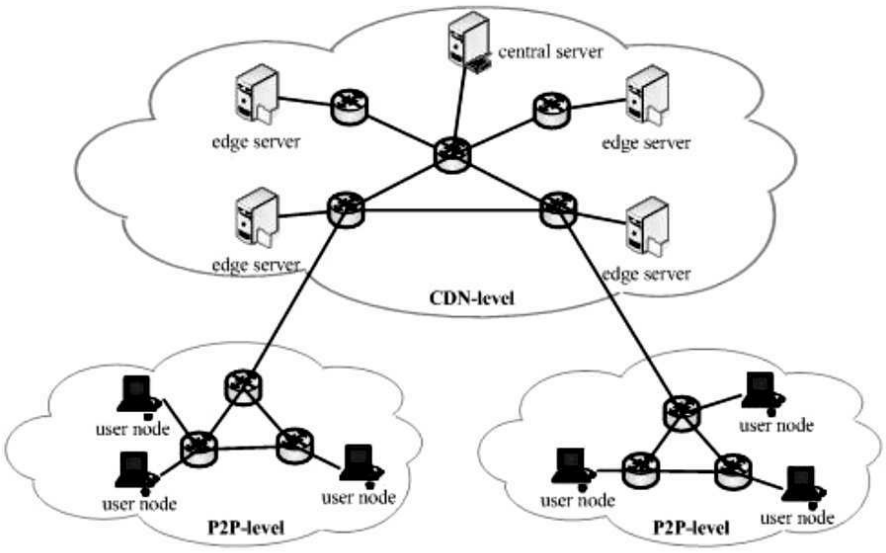


Figure 3.3 – Hierarchical Content Distribution Network (HCDN). Reprinted from [2]

In the P2P layer, current peer selection policies can be used, such as the BitTorrent tit-for-tat scheme. However, peers should only download from other peers when possible so as to lighten the load in edge servers.

The authors conducted a performance analysis comparing HCDN with a normal CDN and a P2P model. They showed that their model outperformed a normal CDN in terms of service capacity (measured in Kb/s), since a user connects to other users and edge servers. However, HCDN did not outperform the P2P model under study, achieving worse result in terms of network cost (measured in number of network hops) and it only got better performance than a P2P approach when the number of user nodes having the entire content in the system reached 70%.

3.3 Interconnection of heterogeneous file-sharing networks

Interconnection of peer-to-peer file-sharing overlays aims at improving data search within the system, the time to download time or bandwidth utilisation. One of the contributions of our thesis is the analysis of hybrid file-sharing architectures. This sections introduces different interconnection schemes for file-sharing networks.

3.3.1 A multi-layered interconnection scheme

Lloret *et al.* [3] developed an interconnection scheme for data search in peer-to-peer networks. *Super-peers* are used to interconnect several networks and forward traffic among them.

The interconnection scheme is based on a three-layer architecture as illustrated in Figure 3.4. The organisational layer is an *ad-hoc* network of super-peers, called *Onodes*, from different peer-to-peer networks. This layer is used to organise the interconnection between networks based on nodes of the lower layer, named distribution nodes or *Dnodes*. The distribution layer is composed of a group of Dnodes which, in fact are the ones forwarding data between networks. Finally, the lowest layer consists in the rest of the nodes of every interconnected network. Figure 3.5 depicts an interconnection example between networks A and B. A Dnode D_{a1} from network A needs to

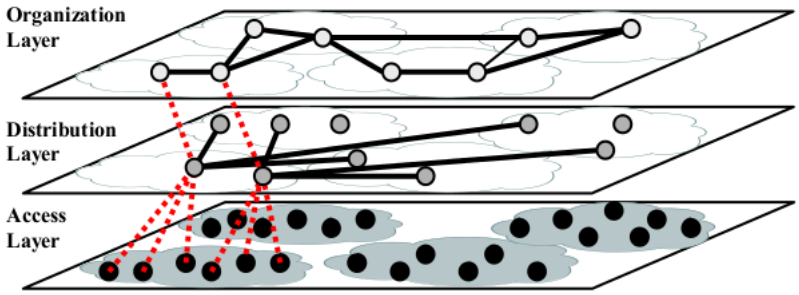


Figure 3.4 – Lloret *et al.*'s multi-layered interconnection scheme. Reprinted from [3]

request Dnodes from network B through one known Onode, in this case, O_a. By means of the organisational layer, O_a can contact O_b and retrieve two Dnodes from network B, D_{b1} and D_{b2}. Once D_{a1} retrieves a set of Dnodes from network B, it can start a distribution link.

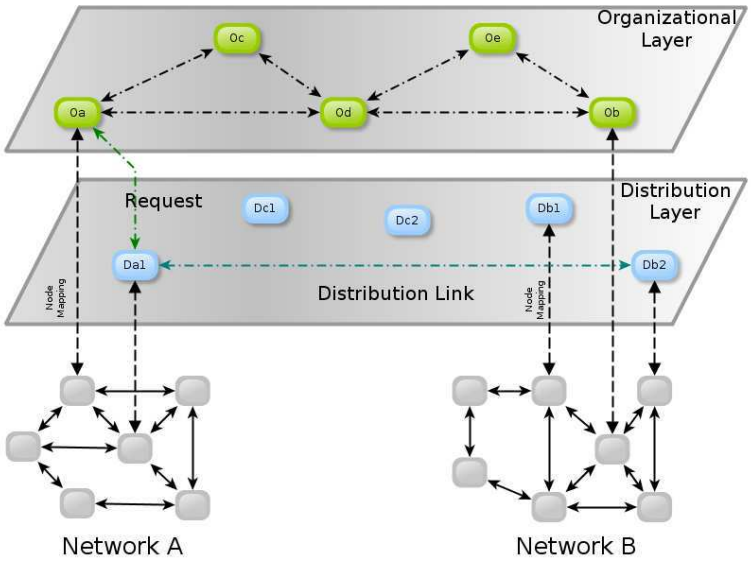


Figure 3.5 – Interconnecting networks A and B through Lloret *et al.*'s multi-layered interconnection scheme

The authors conducted a simulation analysis, for which they concluded that increasing the number of Dnodes improved the overall performance of the network, while maintaining an acceptable number of organisational messages, *i.e.* the messages necessary to maintain the interconnection scheme.

However, it is not clear how and when a Dnode should start a link with a Dnode from another network. The authors stated that a Dnode could request an adjacency by using an on-demand approach to upper Onodes. Additionally, the authors considered an active approach for Onodes leaving or joining the system, which required to re-run a routing algorithm in order to replace offline Onodes. Even if Onodes are elected by considering the most stable nodes in the system, the churn rate is considerably high in current peer-to-peer file-sharing environments. This can lead to a high load in the organisational layer due to the re-computation of this routing algorithm.

A comprehensive analysis considering real churn rates is missing and the performance values presented are tied to a static-network topology, which is not adequate for current file-sharing networks.

3.3.2 Interconnecting *pure* and *hybrid* file-sharing networks

Konishi *et al.* [4] introduced a cooperative scheme for *pure*[8] file-sharing peer-to-peer networks based on *cooperative peers* as seen in Figure 3.6. These cooperative peers act as a logical link, forwarding search messages and their responses, while the actual data transfer is carried out by normal peers in each network.

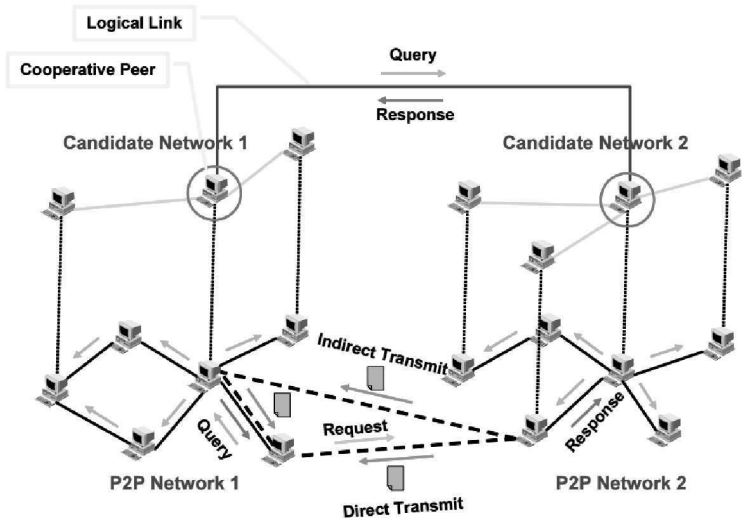


Figure 3.6 – Konishi *et al.*'s cooperation scheme. Reprinted from [4]

Networks nodes willing to improve their own quality of service[9] postulate to become cooperative peers. Their candidature or application is accepted on the basis of a distributed voting approach, where every voter locally computes its decision. Nevertheless, only *highly connected* and *separated peers* can become cooperative peers. *Highly connected* refers here to peers with a high number of neighbours, while the separation between two peers refers to the number of network hops. On the one hand, by selecting highly connected cooperative peers, the authors expect an effective message dissemination. On the other hand, by selecting separated cooperative peers, the concentration of messages is expected to be distributed along the whole network.

The authors analysed their scheme by interconnecting two networks with ten-thousand users and placing ten cooperative peers. They concluded that caching at the cooperative peers did not improve the search latency within the interconnection scheme, mainly because the size of the cache needs to be disproportionally big to achieve a high hit rate (>70%). The simulation additionally showed that choosing cooperative peers randomly did not improve the quality of service, but that it was all the same necessary to choose highly connected peers to enhance it. However, Konishi *et al.* only considered static peer-to-peer networks with a fixed number of users

8. In a *pure* file-sharing interconnection the implied networks use the same protocol, so there is no need to perform a protocol conversion.

9. The quality of service can be improved by introducing new peers into the peer-to-peer network or extending the search for a given content to different networks, for example.

and a fixed network topology, which is not representative of the current dynamic of peer-to-peer file-sharing networks.

Fu *et al.* [5] extended the work done by Konishi *et al.*. They considered *hybrid* peer-to-peer networks, as opposed to the contrary to pure file-sharing networks, where cooperative peers need to apply message conversions because of the incompatibility of protocols between networks. In the hybrid approach described in Figure 3.7 a *shared peer* enables the interconnection among two networks. This shared peer, *e.g.* the *cooperative peer*, is in charge of forwarding search messages as well as data packets between networks. Since networks might use different protocols, cooperative peers need to translate messages before forwarding them.

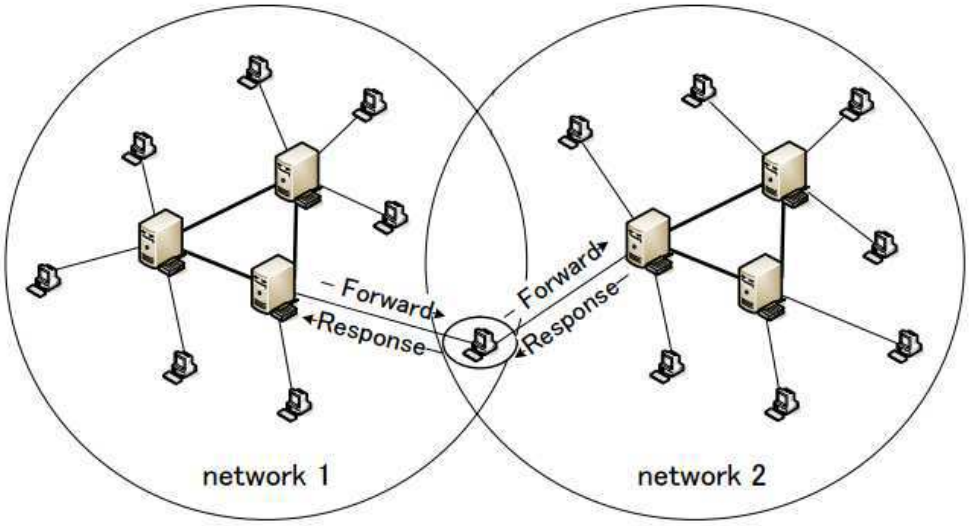


Figure 3.7 – Fu *et al.*'s *hybrid* cooperation scheme. Reprinted from [5]

The authors evaluated their interconnection scheme's performance through simulations and concluded that the application-level's quality of service was improved thanks to these cooperative peers, but one must still note that the load within these peers remains considerable. However, their simulations did not include message conversions, which is the main characteristic of hybrid interconnections, and so that the total load within cooperative peers cannot be considered as properly measured.

3.4 Conclusion

We reviewed different studies in the area of interconnection of overlay networks and cooperation schemes, including heterogeneous networks. A varied set of solutions have been proposed, for which the use of intermediate nodes or *shared nodes* is a common denominator. Table 3.1 shows a classification of the reviewed interconnection systems, highlighting their most important characteristics. *Synapse* is one of the most complete architectures, considering network churn and heterogeneous network protocols, proposing *black-box* and *white-box* protocol. We additionally described different organisational models, where structured and unstructured networks are merged. The structured network is usually the backbone of the model and it is composed of the minority of nodes in the system. The unstructured network serves as the distribution platform,

where peer-to-peer connections are normally employed among the participants, which account for the majority of nodes in the system.

Use	General use			Organisational models		Peer-to-peer file-sharing		
Model	Synapse	Synergy	Net. Symbiosis	Yang's scheme	HCDN	Lloret's scheme	Konish's scheme	Fu's extension
Interaction component	Synapses (Bridges)	Agents	Random connections	N/A	User nodes	Super nodes	Cooperative peers	Shared Peers
Organisation	*ad-hoc* overlay	*ad-hoc* overlay	*ad-hoc* overlay	2-Tiers (Structured& unstructured)	CDN layer	2-layers / *ad-hoc* overlay	*ad-hoc* overlay	*ad-hoc* overlay
Deployment	Simul. (With Churn)	PlanetLab (No Churn)	Simul. (No Churn)	Simul. (No Churn)	Sim. (No Churn)	Simul. (No Churn)	Simul. (No Churn)	Simul. (No Churn)
Topology	Dynamic	Random (Small size)	N/A	Barabasi-Albert (10000 nodes) / Random	Fixed	Random	Barabasi-Albert (1000 nodes)	Kazaa topology
Multi protocol	Yes	No	No	No	Undefined	Undefined	No	Yes, but untested
Start of interconnections	Network decision	Network decision	Local decision	Network decision	Network decision	Network decision	Network decision	Local decision

Table 3.1 – Classification of the reviewed interconnection systems

Yet, most of the solutions have been evaluated through simulation analyses. None of them considered real-world implementations of interconnection schemes taking into account popular file-sharing networks, neither considering an improved data search scheme nor considering an improved download protocol. Furthermore, most of the performed simulation analyses did not include the churn rate, a critical network aspect within peer-to-peer networks, and therefore their applicability in real-world system becomes questionable.

In our thesis, we consider the interconnection of widely deployed file-sharing networks. Chapter 5 presents a hybrid model between the BitTorrent and the Kad networks. This model improves BitTorrent's content indexation. Chapter 6 proposes a hybrid model considering anonymous networks, where the I2P network and the BitTorrent network are interconnected to improve I2P's content availability. In the next chapter, we detail different anonymous systems, with a special focus on the I2P network, as well as different anonymous file-sharing systems.

Chapter 4

Anonymous file-sharing networks: current approaches and monitoring techniques

Contents

4.1 Introduction

Nowadays, people perform most of their social and business activities over the Internet. Information, sent or received, goes through various intermediaries nodes, different autonomous systems, different networks, and can be recorded for a specific and targeted monitoring purpose. Either way, data we send over the *net* does not vanish easily nor rapidly. In the early days of the Internet, the notions of privacy and anonymity were vague and left aside. But, more and more users have moved to a *privacy-preserving* digital environment, mainly driven by different policies and data retention[10] directives such as the European Data Retention Directive.[11]

10. Also known as data *preservation*, which seems less acute but it is still as significant as retention.

11. The Data Retention Directive, or Directive 2006/24/EC. L105/54, from the *Official Journal of the European Union 2006*, roughly state that every telecommunication company needs to store users' data for a period between six to twenty-four months, to ease possible criminal investigations.

The concepts of anonymity and privacy are different, although they are interspersed. Anonymity allows a person to be unrecognisable among a greater group of people, while privacy stands for the right of a person to control his/her personal information to avoid any unwanted disclosure. In this *privacy-preserving* Internet, file-sharers have been facing various situations, such as monitoring and profiling of users or strict *anti-piracy laws*, for which several privacy-preserving file-sharing technologies have been developed to offer different degrees of anonymity to users. Fully-dedicated anonymous file-sharing networks have been proposed, as well as different mechanisms, to increase users' privacy, such as anonymising the BitTorrent traffic via a proxy or an anonymous layer.

Due to the high use of these anonymous systems, it is important to study and analyse them to determine their real use. Because of the very nature of anonymous communications, monitoring these systems is very challenging and requires further techniques.

Most of the current anonymous file-sharing systems are based on current anonymous systems. That is why, we first describe anonymous systems and their associated paradigms. Then, we describe current anonymous file-sharing environments. We focus on monitoring techniques, especially for widely deployed anonymous networks. We briefly consider the legal implications when monitoring public networks, as well as a few facts to keep in mind so as not to weaken users' privacy.

4.2 Anonymous communications

Anonymous communications mask the parties in a communication: the sender, the receiver or both. Within the Internet, an anonymous system hides, for example, the IP address of the sender when contacting a given web server. These anonymous systems serve as a basis for a wide range of applications, from censorship-resistant networks to anonymous file-sharing systems.

This sections presents two anonymous paradigms and different current anonymous systems.

4.2.1 Anonymous paradigms

Edman and Yener [65] conducted a survey on anonymous communication systems, where they proposed to classify these systems into *high-latency* and *low-latency* systems. Before outlining these paradigms, we first introduce the terminology when considering anonymous systems. Pfitzmann and Hansen [66] presented a study on terminology, aiming at standardising the various terms within the field.

Anonymity is the impossibility to identify a particular subject within a set of subjects, the *anonymity set*. This anonymity set is not fixed and depends on a possible attacker. If the attacker is powerful, it can narrow down the set, thus easing the identification of a particular subject. *Unlinkability* of two or more *items of interest* (IOI from an attacker's point of view is the impossibility to distinguish whether these items are related or not. Finally, *anonymity through unlinkability* is defined as the unlinkability between subjects and the actions of these subjects, such as sending or receiving a message. In a simplified version and from an attacker's point of view, a user is anonymous if he/she cannot be tagged as the sender or the receiver of a particular message.

High-latency systems

High-latency systems ensures strong anonymity in exchange of high delays and are only applicable to non-interactive applications, such as email applications.

David Chaum [67] wrote one of the first studies on anonymous communications, presenting the concept of *mix-nodes*, which is the basis for almost all current high-latency systems. Mix-nodes receive encrypted messages, decrypt them, add random padding to maintain their length and put them into a *batch*. A *batching strategy*[12] is used to determine when to send a received message to its actual destination. An attacker observing the incoming and outgoing messages of a mix-node will not be able to correlate them, since they were previously decrypted, producing a different set of bits. Additionally, the messages have been delayed and reordered within the mix-node.

Although ensuring some anonymity, a single mix-node is a single point of trust: a compromised mix-node can reveal the relation between incoming and outgoing messages. An alternative is to use a set of mix-nodes, where a message is routed through an ordered list of mix-nodes, thus increasing the overall anonymity, since every hop knows only the predecessor and the successor out of the total mix-nodes in the path. There are two classic mix-node topologies: *mix-cascades* and *free routes*. In the first one, a client's traffic is routed through one or more predefined routes ; in the second one, users select their own ordered list of mix-nodes to route their messages.

There have been different implementations of anonymous email systems using the concept of mix-node from Chaum. They are usually classified in *remailers* type I, II or III. The anonymous remailer type I, also known as *Cypherpunk remailer*, was initially presented by Hughes and Finney[13] and consists in a group of distributed mix-nodes. A header for every hop in the path is added by the sender before the message is sent. Each mix-node decrypts the incoming message with its private key, removes the header and forwards the still-encrypted data to the next hop, until data reaches the last mix-node. Anonymous remailers type I do not implement Chaum's padding strategy nor his batching or delaying strategies, and therefore they are sensitive to correlation attacks by a passive attacker. The type II remailer, or *Mixmaster*, was developed by Möller *et al.* [69]. It is an improved version of the previous type I remailer, which includes padding to maintain a fixed message-size, as well as a batching strategy to prevent correlation from a passive adversary. Finally, the type III remailer, or *Mixminion*, presented by Danezis *et al.* [70], is the latest implementation of an anonymous remailer and includes link encryption. The Mixminion remailer additionally ensures anonymity to both sender and recipient, contrary to remailers type I and II, which only ensure anonymity for the sender.

Low-latency systems

In a low-latency system, there is no reordering nor batching of incoming messages, with only traffic forwarding being performed. These systems fit better interactive applications, like web browsing. In a low-latency system, the equivalent of a mix-node is a *proxy*.

On the one hand, we have a single-proxy approach or *one-hop* approach, which uses an intermediate hop, namely a proxy server, to route a user's traffic. The receiver gets the intermediate hop's IP address and not the original user's IP address. The main disadvantage of this approach is that a proxy server is run by a single operator, which can be compromised, thus deanonymising any user using the service. On the other hand, most of the current low-latency anonymous systems use a set of proxies to form a *multi-hop* path, generally encrypting the data all along this path. The Crowds system, the Web Mixes approach and Onion routing are the most relevant low-latency systems.

12. A *batching strategy* specifies up to when the messages should be stored. "We keep up to 100 messages in the batch" or "Every minute the batch is cleared out", are some examples.

13. Parekh presented Hughes and Finney's work in his journal article *Prospects for remailers* [68].

Reiter and Rubin [71] designed the Crowds system, especially for anonymous Web transactions. In their mode, a user on the system forwards traffic following a forwarding-probability scheme: half of the time it forwards an incoming message to another user than its actual destination. An attacker participating on the system is not able to discover whether an incoming message was generated from the previous node or, if it has simply been forwarded. Thus, the system guarantees anonymity to the sender or *probable innocence*, as the authors classified their system.

Berthold *et al.* [72] designed a mix-based architecture for anonymous real-time Internet access. A user routes its traffic through one of several available *mix-cascades*. The last mix-node in the cascade forwards the traffic to a *proxy-server*, which forwards traffic to the Internet. The traffic within the mix-cascades is encrypted, thus hardening a traffic analysis attack.

Onion routing, firstly introduced by Goldschlag *et al.* [73], is based on the concept of Chaum's mix-cascades, where *onion routers* are used instead of mix-nodes. An onion router accepts a message, performs a cryptographic operation on the message and forwards it immediately to the next hop in the path, avoiding any batching or delaying strategies. A user selects a group of onion routes in the system and creates a *circuit*: a bidirectional path among different onion routers. During the creation of the circuit, each onion router in the circuit receives the identifier of the next hop (except of the last node) ; two encryption keys, one for forward traffic and one for backward traffic, and an encrypted payload to be forwarded to the next hop. Thus, every onion router in the circuit knows its predecessor, its successor, and which encryption keys need to be used during the data transfer.

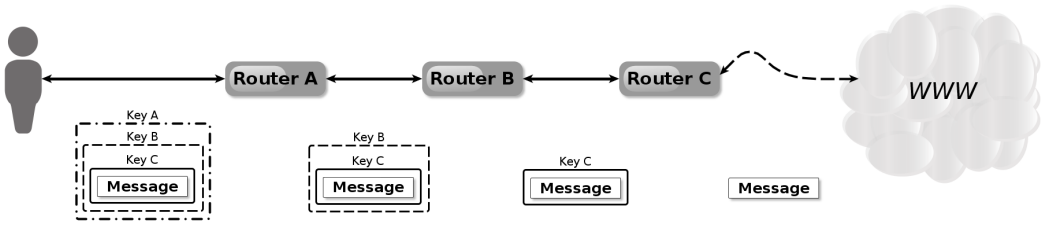


Figure 4.1 – An *Onion routing* example

Figure 4.1 depicts a scenario after the creation of a three-router circuit, where a user sends data over the circuit. The original message is encrypted with the forward key `C` of the last onion router, the resulting encrypted packet is re-encrypted with the forward key `B` of the middle router and finally with the forward key `A` of the first router, leading to a *layered encryption*. The onion routing consists in every onion router removing a layer of encryption, or *peeling off* a layer, to obtain further routing instructions and a still-encrypted payload. Router **C** gets the real message and forwards it to the Internet. When router **C** receives an answer, it encrypts the message with its backward encryption key and forwards it to router **B**. Every onion router performs the same operations until the message reaches the user. The user then peels off every layer of encryption to retrieve the actual message.

Onion routing does not include batching nor delaying strategies, as most of anonymous low-latency systems, although it performs padding in every message to maintain a constant message size.

Our classification of anonymous systems is illustrated in Figure 4.2. Among the alternatives previously presented for anonymous low-latency systems, we focus on *Onion routing*-based approaches. One of our contributions targets the I2P network, an anonymous system based on a

slight variation of Onion routing.

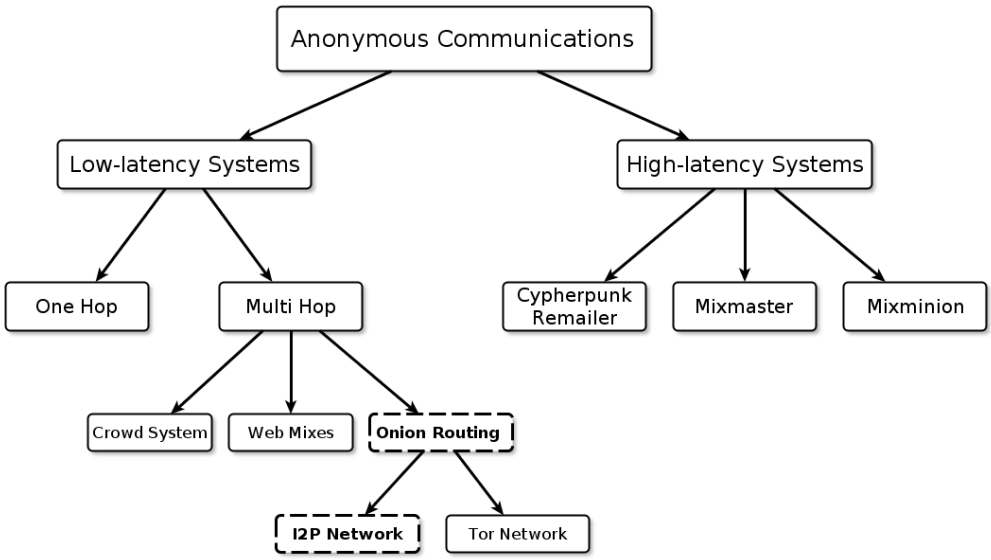


Figure 4.2 – Classification of anonymous systems

4.2.2 The Tor network

The Tor network is probably the most used and most widely deployed anonymous low-latency system, with five-hundred-thousand users in average [14]. This network enables anonymous TCP-based communications by tunnelling a user's traffic through a virtual three-nodes circuit before reaching the Internet.

The initial design was presented by Dingledine *et al.* [74] and it based on the original Onion routing design. However, it includes several variations, such as a central directory server and a *telescopic* approach for the creation of circuits.

The Tor network implements a central approach to distribute signed network information by means of trusted nodes, contrary to the original Onion routing design, which suggests flooding the network with routers' information. These trusted nodes are called *directory servers* and they maintain a synchronised directory of the all onion routers available in the network. Users periodically query these directory servers to obtain a list of onion servers.

In the original Onion routing design, a single message for tunnel creation is passed through every router, thus creating the intended circuit with only one message. The Tor network uses an interactive approach instead. In this approach, called a *telescopic circuit building approach*, the user creates initially the first hop of the circuit and then uses this one-router circuit to send another request to the second router in the circuit and so on for the rest of the routers. Therefore, the user uses the circuit as it is being created to tunnel further circuit building messages.

Since the Tor network is a volunteer-based network, anyone can run a Tor node and forward third-party traffic in the network. Additionally, a user can forward traffic outside the Tor network, to the Internet by running a Tor *exit node*. By using *exit policies* within an exit node, a user can select which type of traffic he/she willing to forward. Figure 4.3 shows an example, where `Alice`

14. Statistics available at `https://metrics.torproject.org/users.html`. Last visited on 08/2013.

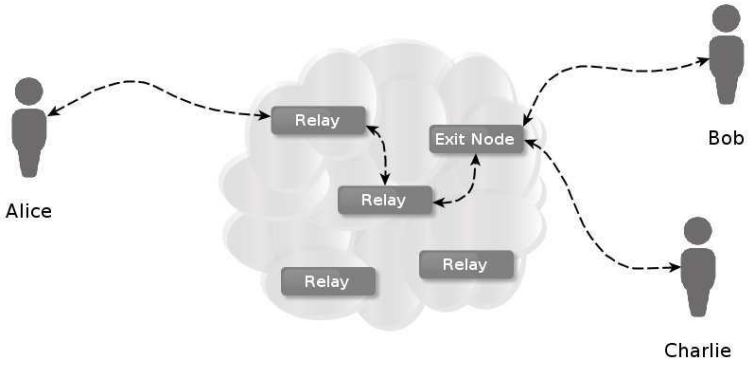


Figure 4.3 – The operation of the Tor network

uses the Tor network to contact `Bob` and `Charlie` using the same circuit, with the last node in the circuit acting as an exit node. Besides enabling anonymous TCP-based communications to the Internet, the Tor network provides a mechanism to deploy anonymous web sites or *hidden services* in the network, which are only accessible within the Tor network.

The Tor network has been widely studied and different improvements have been proposed, such as *entry guards* to cope with attacks on hidden services [75] ; an incentive-based approach to reward cooperative bridges (bridges are similar to Tor exit nodes) and improve the network overall performance [76], and a simplified used of hidden services on the network [77], among others.

However, it has been shown by McLachlan *et al.* [78] that, if the Tor network keeps growing up to a millionaire user-base like popular peer-to-peer services (such as Skype, for instance), Tor nodes will spend considerably more bandwidth in distributing the network directory than in actually forwarding traffic. The authors proposed a distributed directory service based on a Kademlia DHT to cope with the problem of scalability.

On the one hand, a distributed directory service would allow the Tor network to scale and support several million users. On the other hand, we consider that a network originally designed with a distributed database would be a better approach. The I2P network, which does not utilise a central component to coordinate the nodes in the system, but a distributed network database, fits our goal.

4.2.3 The I2P network

The *Invisible Internet Project*, or also known as I2P, is an anonymous network intended for low-latency anonymous UDP-like and TCP-like communications, mainly designed for anonymous web hosting and anonymous file-sharing. The system is designed as an anonymous network layer and does not ensures anonymity by concealing the originator or recipient of the communication, but by enabling users to communicate among themselves anonymously and in a secure way. Within I2P, both ends of a communication are unidentifiable to each other or to third-parties.

I2P's anonymity

The I2P network layer is composed of participants known as I2P *nodes* or I2P *routers*[15]. The information regarding every particular I2P router, such as its IP address, is gathered in a special structure called *routerinfo*. An I2P node uses *tunnels*, similar to Tor's circuits, to communicate with other nodes, being these tunnels formed by others I2P nodes. Whenever an I2P router `A` wants to create a tunnel with an I2P router `B`, router `A` needs router `B`'s routerinfo.

Within the I2P network, a user's application is not identified via the tuple <IP address, port number>, but through a location-independent identifier which decouples a user's online identity and his/her actual physical location. This hash-like identifier is known as a *destination*. Every time a user deploys an I2P application, such as a file-sharing client, a destination is created for that application. This destination is used to receive incoming messages from third-parties, like others I2P file-sharing clients. The information concerning a particular destination is grouped in a special structure called *leaseset*. A remote I2P user needs that leaseset to establish a connection with the application.

The basis of I2P's anonymity is the unlinkability between leasesets and routerinfos. It is not possible to link a particular leaseset, representing an application, with a particular routerinfo, representing an I2P user. Let's illustrate that scenario with two I2P users, running two BitTorrent applications on top of the I2P network. Each user `A` and `B` has its own routerinfo ri_a and ri_b, respectively. Each BitTorrent application has its own leaseset ls_a and ls_b, respectively. Within the I2P network, it is not possible to link ri_a with ls_a or ri_b with ls_b and, therefore, an I2P user with a particular application.

A leaseset has a set of entry points or *gateways* to receive messages from third-parties. Ls_a will have one (or more) entry points, so remote applications, *e.g.* aap_b, can send messages. These entry points are the I2P nodes in the end of the tunnel of the user `A`, which are represented with different routerinfos. Therefore, an application aap_a will have a leaseset ls_a, where the entry points are ri_x and ri_y. The remote application aap_b will communicate with aap_b through ri_x and ri_y.

Distributed network database

I2P uses a distributed database to store its *network metadata*, that is, leasesets and routerinfos. The database is called the *netDB* and is a Kademlia-based DHT, composed of *floodfill* nodes. Floodfill nodes are normal I2P nodes with high bandwidth rates. All routerinfos and leasesets are stored within these floodfill nodes and are accessible by every node in the I2P network.

Considering the previous example, ri_a, ri_b, ls_a and ls_b are stored within the netDB. An I2P user `A` running an application app_a has a destination $dest_a$ and its associated leaseset ls_a. If an I2P user running an application aap_b wants to contact the application aap_a, he/she needs to search in the database the leaseset ls_a through the destination $dest_a$ (we can consider that, in a hash table, $dest_a$ is the key and ls_a is the value).

Thanks to a distributed database, the I2P network avoids a single point of failure, as opposite to Tor's central directory. Additionally, the I2P network uses a dynamic approach to deploy floodfill nodes and improve scalability.

15. The terms *I2P node* and *I2P router* are used interchangeably throughout this document.

Garlic routing

The I2P network uses a slight variation of the onion routing approach, called *garlic routing*. As previously mentioned, in Onion routing, a message is routed from its sender to the final endpoint through several intermediate nodes using layered encryption. The sender adds, to the message to be sent, an encryption layer for every node in the path. Each intermediate node *peels off* one of these layers, exposing routing instructions along with still-encrypted payload data. Finally, the last node removes the final layer of encryption, exposing the original message to the endpoint.

Garlic routing allows the sender to include several messages in a single *garlic message*. The main idea is to take advantage of the expensive initial encryption process to add further messages in the same encrypted outgoing message, which can be seen as the inside of a garlic, hence the garlic routing's analogy.

I2P currently uses this approach to include two extra messages into a single *garlic message*: a delivery status message and a netDB store message. Let's consider a communication between an I2P user and an anonymous web server hosted within the I2P network. The user sends a garlic message with the actual request to the web server, including these two extra messages. The first message is a delivery status message to be sent back to the I2P user once the web server receives the request, as an acknowledgement. The second message is the actual leaseset of the I2P user, which is used by the web server to send messages back to the I2P user. Thanks to this feature, every leaseset does not need to be published in the netDB.

Unidirectional tunnels

Contrary to Tor's circuits, I2P tunnels are unidirectional and formed by the gateway (entry point), a set of participants (intermediate nodes) and an endpoint. Two types of tunnels exist: *inbound* tunnels enable a user to receive data and *outbound* tunnels, to send data. Therefore, a fully bidirectional communication between two users involves four tunnels, as illustrated in Figure 4.4, which shows a simple connection between `Alice` and `Bob`. Alice's outbound gateway and inbound endpoint (the closest to Alice), as well as Bob's outbound gateway and inbound endpoint, are implemented by Alice's and Bob's routers, respectively.

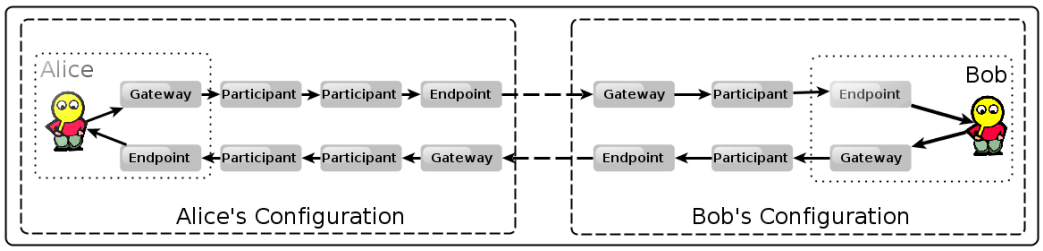


Figure 4.4 – Independent and unidirectional inbound/outbound tunnels

I2P leans on a constant local profiling mechanism of all seen I2P routers, regarding their performance, latency, and availability. The indirected behaviour of users is recorded and used to profile them, contrary to the Tor network, where users' claimed performance values are published in the central directory. No published performance information is used in I2P's local profiling mechanism, so as to avoid attacks from nodes claiming high performance values. An I2P router uses these profiles to select the most appropriate nodes for its tunnels.

Figure 4.5 depicts a simple view of the I2P network, where `Alice` communicates with `Bob` and `Charlie`. In this case, Alice's tunnels include three routers (Alice's router, a participant and a gateway/endpoint). Bob's and Charlie's tunnels have no participants and only two routers. Additionally, a same I2P node can be used for different tunnels, like the shared endpoint in Charlie's and Bob's outbound tunnel.

The tunnels used for the I2P applications are called *client tunnels* and are created as previously mentioned, considering the best profiled I2P routers. An I2P router additionally creates *exploratory tunnels*, which are formed by routers that have not yet been profiled. These exploratory tunnels are used to profile new routers.

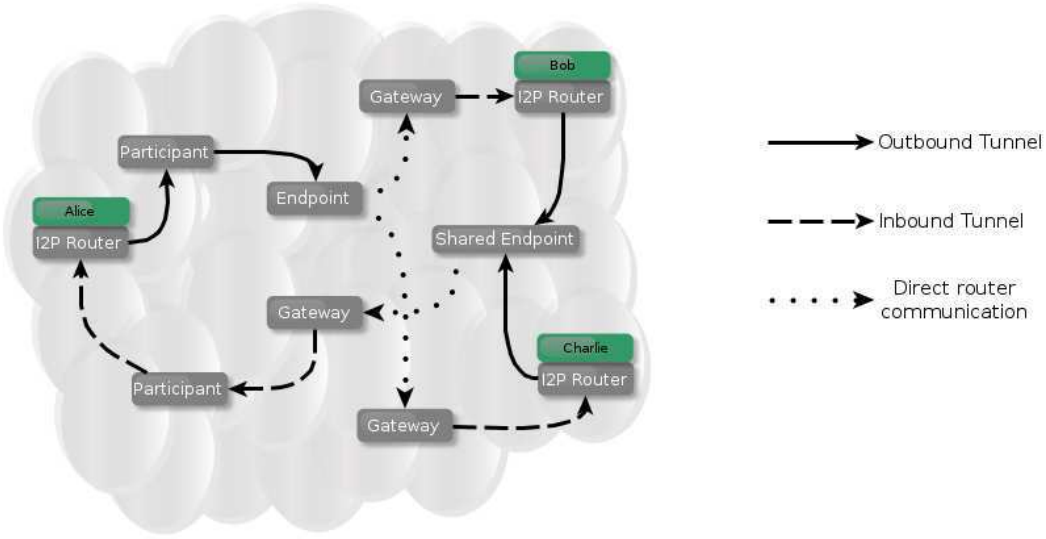


Figure 4.5 – Simple view of the I2P network

I2P applications

The I2P network is optimised for anonymous hosting, enabling not only the I2P user accessing the web server to stay anonymous, but the web server operator as well. Deploying a normal web server on top of an I2P router generates a leaseset which, once published in the netDB, can be recovered and used to access the web server. I2P provides a DNS-like service to support a name-mapping service with web sites names ending in `.i2p`.

I2P also provides a wide set of dedicated applications, such as chat clients (IRC-like and Jabber-like), file-sharing clients (BitTorrent-like, Kad-like and Gnutella-like), a blogging service, a distributed data store and newsgroups services, and has different libraries to develop applications on top of the network.

I2P security

The I2P network has not received as much academic attention as the Tor network, and therefore the amount of attacks and protections is not well documented. However, one important aspect to highlight is that in Tor, as the network is intended to route traffic to the Internet, a timing attack is always feasible. An attacker placing two nodes, one entry node and one exit node,

can correlate messages and de-anonymise a user accessing a certain service, such as a web server. I2P traffic rarely goes out the mix-net, and therefore that attack is much harder to achieve.

However, Herrmann and Grothoff [79] conducted an attack on I2P's anonymous web sites, called *eepsites*, where an attacker with moderate resources is able to de-anonymise the user hosting an anonymous web site. The authors exploited I2P's local profiling mechanism and biased the selection of peers towards their own malicious nodes. Thus, they increased the chances of placing more malicious nodes in the same I2P tunnel. Once a tunnel contains malicious nodes, it is more simple to measure messages going through the I2P nodes and determine the real host of a given anonymous web server.

Crenshaw [80] exploited the application layer of anonymous web servers within the I2P network to de-anonymise different users hosting these services. The author did not compromise the anonymity of the network *per se*, but showed that even with a strong underlying anonymity support like the I2P network, it was extremely important to consider privacy leaks in the application layer.

Mayzaud [81] modelled I2P's cryptographic protocols through the AVISPA project [82], where different tunnel configurations were tested. The author determined that I2P successfully protected a user identity on the system in all scenarios tested, except for the case of two-hops inbound tunnels. If an attacker is placed as the only participant within a two-hops inbound tunnel and if can determine that the previous node within the tunnel is published as an inbound gateway in the netDB, then he/she can conclude that the next hop within the tunnel is the creator of the tunnel. The author concluded that I2P's anonymity lies on the fact that I2P's tunnels are defined by every user and therefore their length remains unknown by a remote peer.

I2P and Tor

There are two major differences between the Tor network and the design of the I2P network. On the one hand, the I2P network does not enable to access the Internet like the Tor network does. Only one or two *out-proxies* [16] are available on the network, while several hundred exit nodes exist in the Tor network. On the other hand, every I2P node forwards traffic on behalf of the network, while only a small subset of the entire user-base participates in in the Tor network [17].

4.3 Anonymous file-sharing approaches

File-sharing has always contributed significantly to the overall Internet traffic and, due to the privacy-preserving environment we are involved in nowadays, more and more users are shifting towards anonymous environments. Anonymous file-sharing networks enable an anonymous content distribution scheme, where users connect themselves to exchange data, while not been recognised via their IP address. Some anonymous file-sharing networks additionally enable anonymous content indexation as well. Thanks to that latter feature, users do not have to share content's identifiers out-of-band, in a non-anonymous way.

This section reviews the best known anonymous peer-to-peer file-sharing approaches. We present different studies, showing that fitting anonymity in non-anonymous environments has several implications and that a user's identity can still be leaked. Fully dedicated anonymous file-sharing networks, on the contrary, have strong identity protection mechanisms, but the amount of

16. An out-proxy allows traffic to be routed outside the I2P network, to the normal Internet, as a Tor's exit node does.

17. At mid-2013 there are 500000+ Tor users and only 4000+ participants in the Tor mix-net, which represents the 8% of the entire user-base.

available content on the network is highly reduced when compared to public file-sharing networks, like the BitTorrent network.

4.3.1 Fitting anonymity in the non-anonymous BitTorrent environment

Two of the most widely deployed file-sharing networks, namely the BitTorrent and the Kad networks, were not designed to take into consideration anonymity issues. As these networks count several millions of users, different approaches have been considered to introduce anonymity within these systems.

Within a non-anonymous peer-to-peer file-sharing environment, peers communicate with each other to exchange data, usually in an *ad-hoc* manner: based on a peer-selection algorithm, a peer connects to remote peers, exchanges data and disconnects on a regular basis. These are direct peer-to-peer connections, and therefore a single attacker at one end can be able to monitor and identify other participant peers. A straightforward example is the active monitoring within BitTorrent swarms presented by Bauer *et al.* [83], where a monitor peer probes every known peer in a given swarm to determine whether a peer is actively sharing a given content.

Hence, to achieve some degree of anonymity, a peer might opt for *hiding* its identity [18], while connecting to remote peers. Proxy-based approaches or anonymising layers are the most used solutions.

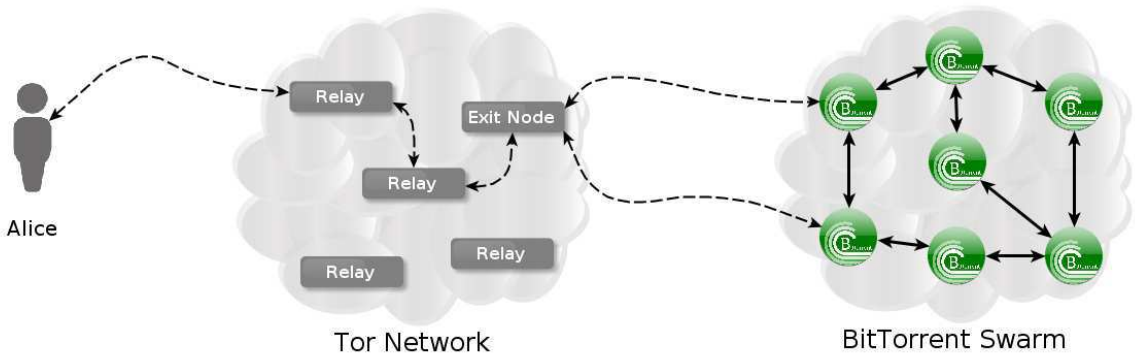


Figure 4.6 – Employing the Tor anonymising layer to access the public BitTorrent network

Figure 4.6 depicts a BitTorrent user routing its traffic through the Tor network while interacting with public users. This configuration has been chosen by a considerable number of BitTorrent users to anonymise their traffic, as stated by Le Blond *et al.* [84]. However, this approach is far from being secure in terms of ensured anonymity. Manils *et al.* [85] conducted an attack on BitTorrent users using the Tor network by exploiting Tor's lack of UDP support, among others. Since BitTorrent is mainly based on a distributed UDP-based tracker, users need to overpass the Tor network and connect directly to that service, revealing their public IP address. By correlating the information in this distributed tracker and the information logged by a malicious Tor exit node, an attacker can de-anonymise a user. However, this attack does not compromise the anonymity provided by the Tor network, and it only exploits an application-level problem: if BitTorrent users are able to anonymise their UDP traffic or encrypt their TCP traffic over Tor so as to prevent exit nodes from monitoring exiting traffic, this attack would no longer be possible.

18. Within Internet communications it is given by an IP address.

The same scenario is feasible when considering proxies or VPN (virtual private networks): if these services do not provide UDP support, a user's identity can be leaked by directly connecting to UPD trackers or decentralised UDP-based trackers. There are commercial solutions, such as the `BTGuard` proxy [19], that do not route UDP traffic, and however it is announced as a perfect solution to cope with anonymous downloads. `TorGuard` [20], on the contrary, is another commercial proxy which supports UDP trackers, thus avoiding the leakage of identity information when using a BitTorrent application. Nevertheless, a proxy is run by a single operator, who can always conduct a monitoring on users' activities or keep IP logs, leading to more difficult to encounter, but yet possible, anonymity problems.

4.3.2 Fully-dedicated anonymous environments

There are basically two different approaches when considering fully-dedicated anonymous file-sharing solutions: *brightnets* and *darknets*.

A brightnet allows users to freely establish connections with any other peer on the network. Its main characteristics lies in the fact that shared data is actually *meaningless* data. Therefore, there are no copyright laws that can be applied to it, thus detaching users from legal matters. Actual data is transformed into a meaningless set of bytes by combining and encoding different parts of real files into a single block of bytes, for a later regrouping and decoding. The brightnet concept does not ensure anonymity *per se*, since anyone joining the system can identify every other user on it. However, it is not possible to determine which files are being distributed on the system by simply observing it, and therefore there is no incentive to monitor the system. Albeit interesting, the brightnet concept has not been widely used in anonymous file-sharing systems, being the *Owner-Free File System* [86] the only semi-stable, although unmaintained, real implementation.

A darknet, or often called a *friend-to-friend* network, aims at providing anonymity to end-users by maintaining trust-based links among them. A user connects only to trusted peers (*i.e.* his/her friends) by means of specific application protocols. A file-sharing darknet can be seen as an instant messenger application with an added file-sharing layer, as suggested by Seeger [87]. The darknet notion has been more widely adopted within anonymous file-sharing networks. OneSwarm, Freenet, GNUnet and the I2P file-sharing environment are probably the best known and deployed anonymous file-sharing networks. We focus on the darknet approach in the following subsections.

OneSwarm

Isdal *et al.* [6] presented OneSwarm, a file-sharing architecture composed of three different file-sharing scenarios in a single framework: anonymous publication, anonymous download and friend-based file-sharing. Since every scenario has different tradeoffs between download efficiency and achieved privacy, we can obtain a complete anonymous file-sharing architecture by combining them. A high-level view of the architecture is illustrated in Figure 4.7. The principle is that OneSwarm users form an overlay, where any kind of content can be shared without restrictions (defined by the authors as *without attributions*), yet still *obscuring* users identity through source-address rewriting and multi-hops paths. New content can be introduced in the system by any OneSwarm peer connected to normal BitTorrent swarms (public sharing). Additionally, users can

19. `http://btguard.com/`. Last visited on 08/2013.
20. `http://torguard.net/`. Last visited on 08/2013.

establish *permissions* for certain data and only share it with a certain number of known peers (with permissions).

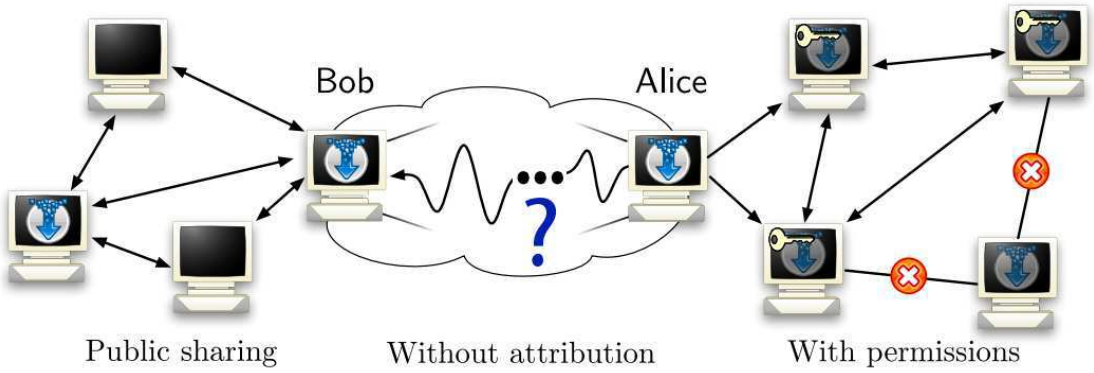


Figure 4.7 – OneSwarm architecture. Reprinted from [6]

Prusty *et al.* [88] highlighted OneSwarm's security issues by successfully conducting a *twin timing attack* and demonstrating that a collusion attack was highly effective. This collusion attacks allowed the authors to determine the real source of a given content, breaking the anonymous publication process. The twin timing attack takes advantage of OneSwarm's design, where artificial delays are introduced to cope with a single attacker. The authors stated that with 15% of attackers on the system, 90% of the system was vulnerable to this attack. Moreover, Prusty *et al.* additionally showed that the original software release included a hard-coded value for a probability value (p=0.95 instead of the p=0.5 declared value), which enabled a collusion attack to nearly affect the entire system with 25% of attackers. New versions of OneSwarm are now able resistant to this attack.

Cunche *et al.* [89] reinforced the collusion attack on OneSwarm by successfully launching an improved of it version called *indirect collusion attack*. This attack relies on a few monitoring nodes not necessarily connected to the targeted victim, but to the general OneSwarm overlay. The authors successfully detected the source of the content with a zero false negatives probability and barely a 2% false positives.

OneSwarm clearly steps forward in anonymous file-sharing by considering its different aspects: publishing, downloading and friend-based sharing. By replicating content downloaded from the public network into OneSwarm, this architecture allowed the BitTorrent content to be introduced into the system. However, this procedure needs to be manually carried out by a user. As the BitTorrent network already holds a considerable amount of content, it becomes unrealistic to perform up-to-date file-sharing. OneSwarm is not adequate if a user wants to download up-to-date content anonymously.

Freenet

Clarke *et al.* [90] introduced Freenet, a censorship-resistant distributed data storage being, arguably, the only currently deployed full darknet. It is based on distributed nodes that share unused disk space to contribute to the network's total storage capacity. Based on this distributed storage, services like anonymous hosting, anonymous chatting and anonymous messaging have been deployed on top of the system.

Whereas in normal peer-to-peer file-sharing architectures, a user owns data and makes it

available within the system for later sharing, Freenet divides data into encrypted chunks and stores it in several nodes throughout the system. This approach, called *data partition*, reinforces the anonymity of the original publisher, as well as data availability, since data is still accessible even when the original publisher goes off-line. By including encryption in the former approach, Freenet guarantees plausible deniability to users, since they do not only store chunks of the original data, but they encrypt them.

Figure 4.8 depicts a simplified approach of Freenet's storing and retrieving process. The producer stores a file which is divided in two parts, p_1 and p_2. Node **A** is responsible for one part, while node **C** is responsible for the other part. During the storing process, node **B** caches part p_1, while forwarding the request. The requester retrieves part p_1 directly from node **C**, while part p_2 is retrieved from node **A**, where node **C** caches this part in the process. That way, future requests for this file from nodes close to the requester will be answered faster, since both parts are now cached in node **C**.

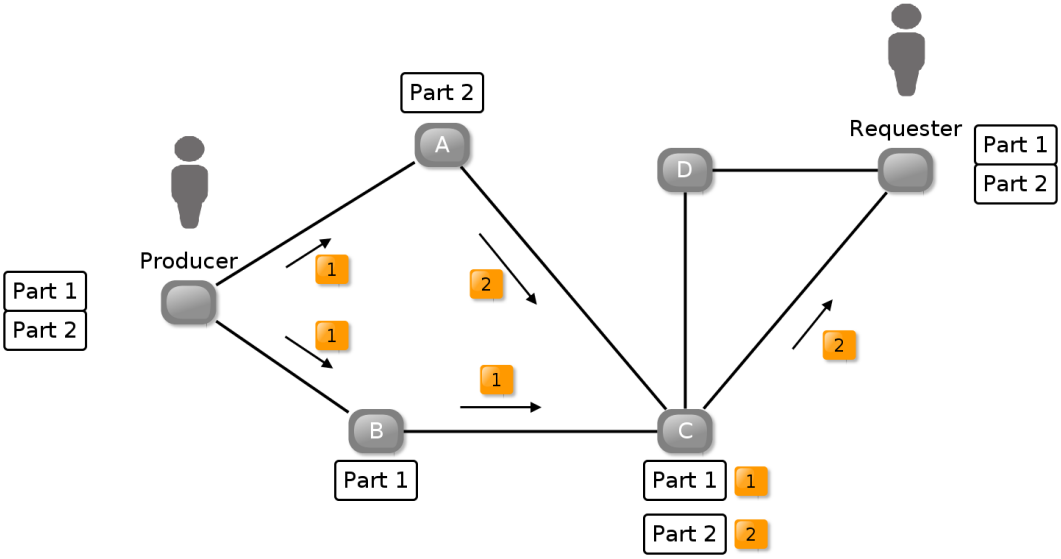


Figure 4.8 – Bird's eye view on Freenet's storing and retrieving process

Freenet nodes are organised as a *small-world* network[21], where most of the nodes on the system have few connections with other nodes. Yet, a small group of nodes is highly connected to other nodes. These networks take advantage of these well-connected nodes to achieve a small-length-path between any pair of nodes, as well as to increase fault-tolerance. In Figure 4.8, node **C** is a highly-connected node, which serves as a *hub* to reach almost every other node in the system. Freenet was originally designed as an opennet, where every node established a few connections with untrusted peers. In a recent release, an extension has been proposed so as to operate in dark mode, enabling peers to connect only with trusted peers, resulting in a fixed network.

Regarding Freenet's security, the authors stated that the network was susceptible to a wide range of attacks when operating in open mode, including correlation attacks and fully monitoring the network to determine who the original publisher of a given content was. Operating in dark mode highly increases the overall anonymity of the system. However, different studies have shown

21. Introduced by Stanley Milgram in his work, *The Small-World Problem* [91].

that even in dark mode, the network is still vulnerable to different attacks.

Evans *et al.* [92] demonstrated that Freenet's routing algorithm in dark mode was prone to a *clusterisation* attack. In dark mode, Freenet's nodes are randomly distributed, and due to receive the same amount of store requests, thus balancing the load on the network. The attack works by grouping nodes and, in that way, unbalances the distribution of nodes. The result is that a percentage of nodes do not receive any store request, while other ones receive a high number of requests, exceeding their storing capacity and consequently causing data loss. Schiller *et al.* [93] completed the work by Evans *et al.* by considering two extra attacks in the current Freenet's routing algorithm: a divergence attack (aimed at disrupting the routing) and a non-optimal distribution attack (aimed at disturbing the distribution of nodes within the system). The authors proved that all of these three attacks, including the original clusterisation attack, were possible on the system by exploiting a routing primitive.

Finally, Tian *et al.* [94, 95] proposed two attacks on Freenet, a *traceback* attack and a *routing table insertion* attack. In their model, a routing table insertion attacks aims at establishing connections between a target node and an attacker by considering the network topology around the attacker and predicting the routing paths for data insert/request messages. Freenet's routing algorithm maintains a dynamic topology so as to prevent static routing paths from forming, where a connection between two nodes **A** and **B** is maintained until a successful number of requests have been received by these nodes. Once a threshold is achieved, the connection is dropped and a new one is created. The routing table insertion attack takes advantage of this mechanism by enforcing a connection between two honest nodes to be dropped so as to establish a new connection between the attacker and one of those honest nodes. The routing table insertion attack enables the second attack, namely the traceback attack, which aims at determining the origin of a message, such as a data insertion message, thus de-anonymising the original publisher of a given content.

GNUnet

Bennett *et al.* [96] proposed GNUnet, a decentralised reputation-based network for anonymous-based peer-to-peer communications, with anonymous file-sharing being the main service built on top of the network. GNUnet operates in open mode, where nodes connect to untrusted parties. However, it additionally has a friend-to-friend mode to operate in dark mode, restricting connections only to trusted parties. In both cases, every connection between two nodes is encrypted by means of a SSH-like approach. GNUnet also includes a ranking mechanism to support its reputation-based communication, where nodes are ranked according to their behaviour within the system: nodes have to *pay* for queries (by decreasing their ranking value) and are *rewarded* when they respond them with valid replies (by increasing their ranking values). Eventually, high-ranked nodes have priority over low-ranked nodes, long-trusted nodes being thus favoured over recent untrusted nodes, in an overall attempt to avoid Denial-of-Service attacks.

A censorship-resistant file-sharing protocol [97] has been developed on top the current GNUnet system, achieving: 1) *plausible deniability* for intermediate nodes forwarding search queries and content ; 2) *content fragmentation* among several nodes to balance the load on the network ; 3) *encrypted regular-expression-based* search queries so as to hide what users are searching for intermediate-forwarding nodes. This file-sharing system is based on a *hash-tree* structure. A file is divided in multiple 1 *Kb* fixed-size blocks and different levels of indexation are used to maintain the file structure, including a file description and a keyword. Each block is encrypted and then stored in different nodes through the entire system in order to balance the load. The main disadvantage is that some blocks might be lost due to network churn, making it impossible to

reconstruct the entire file. The authors stated that the reduced size of the blocks makes their replication almost costless. However, the downside is the high number of queries needed to recover a file, which stresses the network. This hash-tree-based encoding is vulnerable to attacks where an attacker processes a given file, obtaining the set of blocks. If he/she then compares every seen block going through its GNUnet node with the set of blocks previously processed, he/she can filter out that specific content, leading a limited form of censorship. Since the entire system is designed to help users to introduce new content, the former attack is hard to avoid.

Kügler [98] also exploited the encoding scheme previously mentioned, conducting a *shortcut* attack, which is a variant of the intersection attack, in order to detect the initiator of a request. By exploiting a routing optimisation, known as *hot path routing*, the author was able to deanonymise the initiator of a request (a download request for instance).

I2P anonymous file-sharing environment

Lastly, we consider the I2P's anonymous file-sharing environment. Since the I2P anonymous network has already been previously introduced, we only focus on its anonymous file-sharing characteristics.

Unlike the previously mentioned file-sharing approaches, such as OneSwarm or Freenet, I2P provides a secure layer for applications to communicate anonymously among themselves. On top of this layer, different file-sharing applications were adapted to work with the concept of *destinations*[22], more like the censorship-resistant file-sharing protocol of GNUnet.

Three file-sharing clients are available within the I2P network: one Gnutella-based, named *I2Phex*, one aMule-based, called *iMule* and one BitTorrent-like, named *I2PSnark*. We mainly consider the I2PSnark because of its extended use within the system and, by contrast, the reduced user-base of the two other file-sharing clients. I2PSnark is a BitTorrent-like client, operating on top of I2P, maintaining every feature of BitTorrent-like clients. The only major difference is that remote peers are identified through *destinations*. The BitTorrent's tit-for-tat mechanism[23], its piece selection algorithm, and every characteristic of the protocol are maintained within the I2PSnark client.

Figure 4.9 shows the I2P BitTorrent-like file-sharing environment. On the one hand, every user communicates anonymously with other users on the network via the underlying I2P anonymising layer, leading to an *anonymous content distribution* scheme. As in the BitTorrent network, different swarms are formed within the I2P anonymous file-sharing environment to share content. On the other hand, there are different anonymous built-in trackers, as well as an additional decentralised tracker, which allow content to be indexed within the system, resulting in an *anonymous content indexation* scheme. The built-in trackers act as normal BitTorrent trackers, although they are contacted through the I2P network. The decentralised tracker is an adaptation of the BitTorrent's Mainline DHT, designed to work with the concept of destination instead of IP address.

I2P thus enables a fully anonymous *close* file-sharing environment, including anonymous content distribution and anonymous content indexation. It is a closed environment because it is not possible to access content from the regular BitTorrent network, but only content previously introduced in the I2P network. I2P BitTorrent-like users do not interact with public BitTorrent

22. A *destination* is the location-independent abstraction used by I2P to replace a tuple <IP address,port>. It has been previously introduced in Subsection 4.2.3.

23. BitTorrent uses a *tit-for-tat* approach to implement an incentive mechanism to optimise download rates [17].

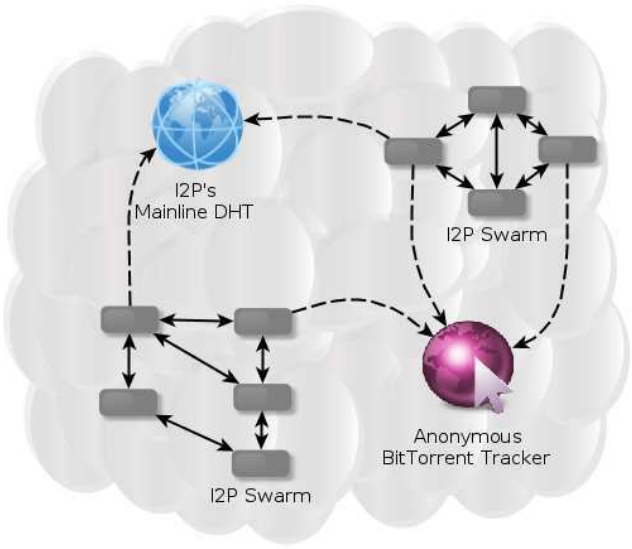


Figure 4.9 – I2P's BitTorrent-like environment

users. Therefore, the decentralised tracker is exclusively composed of I2P BitTorrent users and it only indexes I2P BitTorrent content.

Classification of anonymous file-sharing approaches

Figure 4.10 depicts our classification of the anonymous file-sharing systems reviewed in this section. We focus on anonymous up-to-date file-sharing, *i.e.* the possibility to anonymously access content in open file-sharing communities, like the BitTorrent network, which remain the major sources of content.

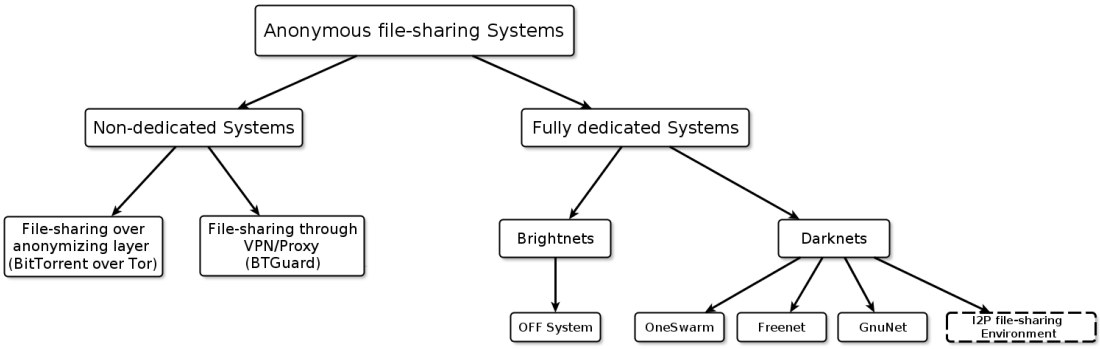


Figure 4.10 – Classification of anonymous file-sharing systems

Non-dedicated systems stand as a good option regarding content, since users can access open file-sharing communities while preserving their online anonymity. However, these approaches, such as using the Tor network to anonymise BitTorrent traffic, imply different anonymity risks, leading to application-level identity leaks.

Fully dedicated anonymous file-sharing approaches can ensure strong anonymity for end-

users, such as the I2P file-sharing environment. This approach makes possible anonymous content indexation and anonymous content distribution, leading to a fully anonymous file-sharing environment. Nevertheless, the amount of available content in the system is substantially less important than in non-anonymous environments such as the BitTorrent network.

In this thesis, we focus on I2P's file-sharing environment and propose a mechanism to enable I2P users to access public BitTorrent content, while preserving their anonymity.

4.4 Monitoring anonymous networks

As Chaabane *et al.* [99] stated in their monitoring study, a shared belief in the research community is that anonymous networks are used for the sake of freedom of speech and they should therefore remain unexplored or uncharacterised, so as not to reveal sensitive information. The authors later agreed on the fact that a monitoring study on anonymous environments would help understanding the network and would thus represent a mandatory step in the growing process of the system. We strongly agree upon this concept, as well as on the idea that any anonymity system that leaks sensitive information about users through a monitoring study is not mature enough and needs to be improved.

Monitoring an anonymous environment implies additional challenges when compared with a normal monitoring: 1) participants in an anonymous network do not necessarily use the network, but they might just forward traffic on behalf of the network ; 2) real users, such as file-sharers, might not forward traffic, but they are still anonymous ; 3) traffic is usually encrypted and a passive monitoring might therefore not be adequate.

We focus on the monitoring of anonymous low-latency systems, mainly targeting the Tor network and the I2P network. This section presents different monitoring studies on these networks. Our goal is to show which kind of information can be retrieved from a precise anonymous environment, how that information is retrieved and which monitoring architectures are more adequate when the network has a central directory, like the Tor network, or a distributed database, like the I2P network.

4.4.1 Monitoring the Tor network

On the Tor network, we have two types of monitoring approaches: a low-resource monitoring on the one hand, and a more aggressive approach, based on deep packet inspection on the other hand.

Low-resource monitoring

As Tor is a volunteer-based network, it becomes easy to place a monitoring node. Since the network is intended to route traffic to the Internet, an attacker can log every packet leaving the mix-net through his/her exit node by placing a monitoring node as an exit node. This feature, along with the fact that insecure protocols are used on the network, allows a low-resource monitoring to be highly effective. Figure 4.11 shows a simplified view of the Tor network, including relays, bridges, entry guard nodes and exit nodes, as well as all the possible monitoring points in the infrastructure.

Most of the analyses on the way to monitoring the Tor network take advantage of Tor's main feature: enabling an anonymous access to the Internet, where *exit nodes* are used. Another required feature for an easier monitoring is the use of a central component, such as the Tor

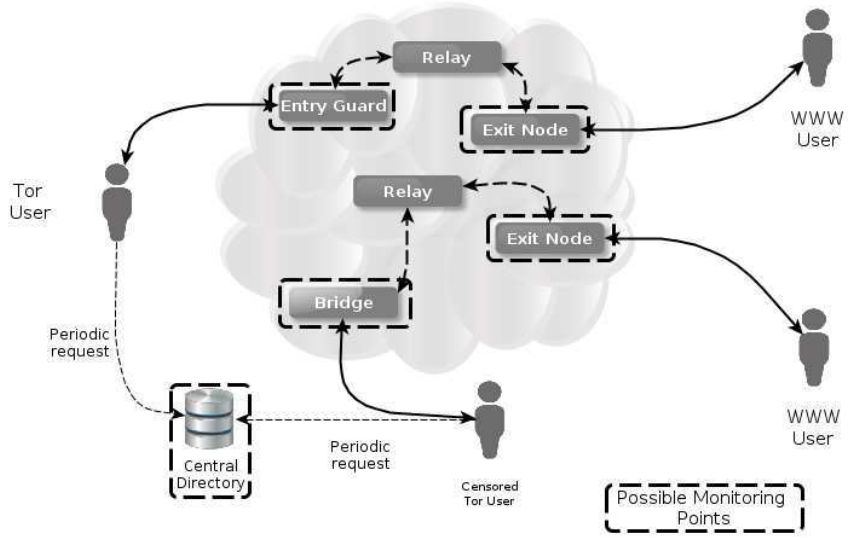


Figure 4.11 – Monitoring points in the Tor network's infrastructure

central directory, which can be publicly queried. Both of these features facilitate the networks' monitoring.

One of the first monitoring studies on the Tor network was performed by Egerstad and brought forward by different people [100, 101]. Egerstad obtained a large list of email credentials for embassies and different corporate agencies around the world. His monitoring consisted in placing a few exit nodes distributed worldwide, while logging every TCP packet that went in and out of his nodes. Many Tor users do not encrypt the traffic they relay to the network and therefore the last node in the anonymous path, *i.e.* the exit node, is able to log data packets in a plain-text form.

McCoy *et al.* [102] conducted a traffic characterisation study of the Tor network. The authors collected traffic regarding Tor clients, as well as exit traffic. By analysing the application layer of the data packets exiting the network, they were able to characterise different applications using Tor and showed that HTTP connections accounted for over 90% of the total detected traffic. These results indicated that interactive web traffic represented the main use of the Tor network. The authors additionally detected a considerable number of connections using insecure protocols, like POP or FTP, which allows the operator of the exit node to log different users credentials, *i.e.* the user's names and passwords. Moreover, the Tor network defines *entry guards*, which are nodes that are used for the first hop in a tunnel and are assigned by a central authority. By having a node assigned as an entry guard, an attacker can monitor different clients using the Tor network and determine their country of origin, for example.

Loesing *et al.* [103] also performed a statistical analysis on the Tor network, differentiating the users' country of origin and the port number of exiting traffic, where interactive web traffic was the most detected traffic once again. On the one hand, to determine the users' country of origin, the authors considered all the statistical information collected by *entry nodes* and *bridges*[24], which is uploaded to the central directories in a regular basis. That information consists in an

24. A bridge acts as a Tor relay, but it is not listed in the central directory, making more difficult to block it. This feature helps censored users to connect to the Tor network when Tor relays are blocked.

aggregation of users' countries of origin and not in their IP addresses, which somehow preserves a user's anonymity. On the other hand, to determine the port number of connections, Loesing *et al.* used a Tor exit node [25] to measure the target port and the amount of incoming and outgoing bytes per connection.

These previous research works clearly highlight the difference between privacy and anonymity. The Tor network anonymises a user's traffic by passing it through different nodes before it goes out of the network and therefore the end of the communication cannot determine the user's real IP address. However, Tor does not encrypt nor authenticate the traffic outside the network. If this data is in a plain-text form, the exit node can monitor the data passing through, thus attacking the users' privacy.

Mulazzani *et al.* [104] presented a simple statistical analysis on Tor servers, extending the functionality of the TorStatus project [26] to support long term measurements. The authors argued that having a detailed knowledge of Tor servers over time could help to detect anomalous behaviours, such as a country-wide blocking (all Tor servers from a given country suddenly stop working, for instance) or to improve the path selection algorithm (if the number of Tor servers for a given country decreases, users from that country might opt to build longer tunnels for example).

The use of the Tor network was approached from varied angles: McCoy *et al.*, as well as Loesing *et al.*, aimed at characterising the network based on the geographical distribution of users and the type of exiting traffic ; Mulazzani *et al.* however conducted a more simple analysis of the network, focusing only on Tor servers and their behaviour on the network.

Deep packet inspection

Chaabane *et al.* [99] conducted a more complex analysis on the Tor network by employing *Deep Packet Inspection* techniques on six Tor exit nodes. The authors discovered that around 30% of the total detected traffic was BitTorrent encrypted traffic which, along with the non-encrypted BitTorrent traffic, it accounted for more than 50% of the total detected traffic, making BitTorrent the biggest source of traffic on the Tor network. They additionally detected several Tor users using a one-hop circuit to obtain a SOCKS proxy instead of a full three-hop Tor circuit, which increases the throughput but greatly reduces the anonymity of the user. Finally, Chaabane *et al.* were able to *crawl* the Tor network so as to detect Tor bridges, which are not officially listed in the main central directory of the network. Their single-node crawler puts in evidence the lack of protection against automatic crawling in the central directory regarding Tor bridges, which reduces the purpose of non-listed Tor bridges.

4.4.2 Monitoring the I2P network

When considering the I2P network, the same techniques as in Tor cannot be applied due to its lack of a central directory and exit nodes. The I2P network is a *closed* network in the sense that no traffic leaves the network to the Internet. As previously mentioned, the I2P network is optimised for anonymous hosting and therefore most of the generated traffic remains in the I2P network.

There are two main statistical services for the I2P network. The first service, which is the

25. It is unclear whether a single node or multiples nodes were used in their study.

26. A project to maintain statistics about Tor servers, available at `http://www.torstatus.net/`. Last visited on 08/2013.

official statistics portal[27], provides approximate values for the number of users on the network, the number of applications deployed, as well as different parameters for I2P's tunnels, such as building success rates, building rejection rates and the number of average tunnels a normal I2P user participates in. These values are obtained from a single floodfill node deployed by the authors of the network. According to the number of current floodfill nodes in the network, the values of a single floodfill are extrapolated to have total approximate values for the entire system. The main issue with that service is that it only provides an estimation of the number of users and not the type of applications deployed on the network. All statistical values are extrapolated and therefore, only an *estimated view* is produced by this service.

The second service provides uptime statistics for I2P's anonymous web sites[28]. This service works by pooling all listed and known I2P eepsites to determine whether they are running or they are down. This service only contacts the eepsites listed in I2P's DNS-like service. If different eepsites have been created but not added to I2P's DNS-like service, they are not contacted and are thus excluded from the monitoring.

4.4.3 Legals aspects on network monitoring

Whenever a monitoring study is carried out, it is necessary to consider its implications regarding a user's privacy. By carefully monitoring, logging and analysing the network traffic, several conclusions can be drawn concerning the users in the network. When it comes to anonymous networks, those implications become more critical and further ethical guidelines must be considered.

This section introduces different aspects of anonymous network monitoring from a legal perspective. We consider that it is compulsory to take these aspects into account when monitoring an anonymous environment since it might not only engender anonymity problems for the users of the system but also for the researchers performing the monitoring.

Ohm *et al.* [105] showed that there was no special consideration nor *safe harbours* for academic research when conducting any kind of network monitoring and that it could therefore be considered violation of privacy. The authors mainly focused on federal laws in the United States but international were examined as well. The authors analysed tens of monitoring studies so as to evaluated the extent to which researchers contemplated users' privacy in their monitoring and up to which degree they sacrificed research on behalf of privacy. They reached the conclusion that due to the vagueness in the law[29], several authors sheltered their studies in well-known sentences like *"we have anonymised the data and therefore it is not illegal"* or *"capturing only headers does not imply a privacy violation"*. However, these affirmations are not sufficient to be exempt of any further legal implications, and even if there is not any fixed set of rules of thumb, Ohm *et al.* proposed different guidelines to minimise the liability, while conducting a monitoring study: 1) capture only the data needed for the study ; 2) distort the retrieved IP addresses if possible ; 3) if sensitive data (IP addresses, for instance) is stored, encrypt it whenever not used ; 4) restrict the monitoring to the smallest network required ; 5) be aware of filtering tools that might still keep the entire data packet on disk and 6) get a consent from users whenever possible.

This previous research work presented different aspects regarding the monitoring of non-anonymous networks. Loesing *et al.* [103] gave a set of *guiding principles* in their statistical

27. Accessible at `http://stats.i2p.in`. Last visited on 08/2013.

28. Accessible at `tino.i2p.in`. Last visited on 08/2013.

29. Although United States federal laws were considered, many international laws have been harmonised through conventions and treaties, as stated by [105], and therefore worldwide researchers can still find the authors' conclusions highly valuable.

study on the Tor network, which could be taken into consideration when conducting a monitoring study in anonymous networks. The authors considered that the legal requirements, *i.e.* the involved laws, depended upon which country the monitoring was conducted in and, along with the vague definition of these laws on the Internet community, it is eventually difficult to apply them correctly. User's privacy is one of the authors' main concerns and they argue that monitoring should not allow an attacker to extend its own logging capabilities. Finally, the *ethical approval* is another important point to consider, while conducting a monitoring study, regarding which Loesing *et al.* proposed to obtain informed consent from either the entirety of users on the system or from an Institutional Review Board which, in the field of computer science, is not always available.

When monitoring either a public network or an anonymous environment, certain aspects need to be considered so as to preserve users' privacy and comply with the applicable laws, while still maximising the monitoring.

4.5 Conclusion

We reviewed different anonymous systems, mainly focusing on anonymous file-sharing environments. We showed that fitting anonymity by means of external anonymity services, like anonymity layers or VPNs, could possible leak users' online identity. Dedicated anonymous file-sharing networks, on the contrary, ensure a strong anonymity for the end-user, but the lack of content in these systems can discourage users seeking up-to-date anonymous file-sharing, like in the case of the I2P's anonymous file-sharing environment. In Chapter 6, we consider the interconnection between anonymous and public file-sharing environments, aiming at improving content availability in the I2P's file-sharing environment.

We introduced different monitoring approaches regarding anonymous environments, where the low-latency anonymous Tor network has been widely characterised. Its central directory along with its *exit nodes* allow a low-resource monitoring to be highly accurate. The anonymous I2P network guarantees a strong anonymity for the end-user and has doubled its user-base in the last year. Yet, there have not been any comprehensive studies of the system nor a proper characterisation of it. Former monitoring techniques applied to the Tor network cannot be applied to this network, mainly because of I2P's lack of central directory or exit nodes. In Chapter 7, we present the first monitoring study of the I2P network, while in Chapter 8 we introduce the first study towards group-based characterisation in the I2P network.

The next chapter presents our work in the area of hybrid file-sharing networks and the way we managed to improve the indexation of BitTorrent's content via the Kad DHT.

Part II

Hybrid Peer-to-Peer File-Sharing Architectures

Chapter 5

Improving content indexation in the BitTorrent file-sharing environment

Contents

5.1 Introduction

Previous Internet studies [106, 107] reported that peer-to-peer (P2P) file-sharing applications contributed significantly to all Internet traffic. Currently, the situation has changed in favour of online streaming services, such as Youtube or Netflix [108]. However, in Europe 46% of the upstream traffic is still generated by P2P file-sharing applications, while in North America it accounts for the 42%, according to the same study. Therefore, P2P file-sharing applications remain still in the top positions when it comes to Internet traffic.

Considering P2P file-sharing applications, BitTorrent is the biggest active application being ranked first with millions of simultaneous users and available torrents. Legal issues involving copyrighted data in the network have yielded to the closure of major centralised tracker sites, such as The Pirate Bay or Mininova, leading to a *decentralised* architecture. This has accelerated the evolution of the network towards a fully distributed approach offering the same features as a central tracker. The BitTorrent network has two decentralised trackers, both based on the Kademlia DHT, as presented in Subsection 2.3.4. The Mainline DHT tracker is the most deployed

one, which has been embedded in most of the BitTorrent clients since 2006. Alternatively, a second distributed implementation has been developed exclusively for the Vuze client (formerly Azureus). Both decentralised trackers offer a single level of indexation, mapping torrent to peers, which allow users to find sources to download a particular content. The main advantage over a central tracker is availability, since in a decentralised tracker several peers store the information. However, the level of security in these decentralised trackers is reduced, enabling different types of attacks.

The eDonkey (Ed2k) network has ranked second [108] in the most used P2P file-sharing applications, accounting for a third of all P2P file-sharing activity (BitTorrent accounts for the remaining two-thirds). In 2004, the main client of the eDonkey network, namely the eMule client, introduced a new fully decentralised P2P network called Kad, which was designed to be backward compatible with eDonkey. Since the shutdown of the major eDonkey servers in the past years due to legal drawbacks, Kad has gained popularity. Thanks to its fully distributed architecture and the open-source nature of its clients (eMule and aMule), Kad has been widely studied and improved, as detailed in Subsection 2.5.2. Being that the BitTorrent network is the most used P2P application but its decentralised components have not been improved from a security perspective, we argue that the BitTorrent network could easily benefit from the performance and security features of the Kad network to index content. Additionally, Kad could benefit from the performance of the BitTorrent download algorithm, thus both networks taking advantage of their mutual collaboration.

This chapter introduces our hybrid file-sharing model, which is instantiated with both BitTorrent and Kad/Ed2k networks. We first conduct an analysis of BitTorrent's Mainline DHT [30] and the Kad network and compare them from a performance and security perspective. Then, we evaluate the performance of the download algorithm of BitTorrent and compare it against the download algorithm of Ed2k network. Finally, we instantiate our abstract hybrid model with these two networks and assess it through the implementation of a hybrid client called hMule.

5.2 Comparison of DHTs

We compare both DHTs, Mainline and Kad, from two different perspectives. On the one hand, we need to determine the level of protection of both Kad and Mainline DHTs against basic attacks. On the other hand, we assess whether the Kad DHT performs better in terms of publication time and messages overhead than the Mainline DHT.

We used the aMule client version 2.2.6 for our analysis of the Kad DHT and the Vuze client version 4.5 with its Mainline DHT plugin version 1.3.3.1 for our analysis of the Mainline DHT.

5.2.1 Security comparison

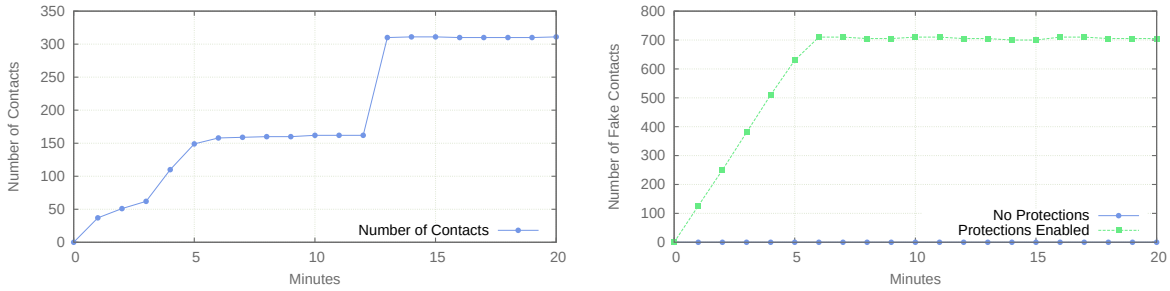
This section is organised as follows. Firstly, we evaluate the protection level of the Mainline DHT against a basic routing table attack, which is the basis for more complex attacks. Secondly, we bring forward a study of protection mechanisms in the Kad DHT, which prevent simple as well as complex attacks in the network.

30. The Mainline DHT has been adopted by major BitTorrent clients, while the Vuze DHT is only employed by a single client. Therefore, we only consider the former DHT for our comparative study. However, Crosby and Wallach [60] showed that both DHTs presented the same security properties.

Assessing the security features of the Mainline DHT

In order to evaluate the security level in the Mainline DHT, we consider the attack described by Steiner *et al.* [109], commonly known as a routing table poisoning attack, from a single attacker. With this basic attack an single attacker can take control of a remote peer's routing table by adding several fake routing contacts. Successfully achieving that attack will prove that the network is open to basic vulnerabilities, and more complex attacks can be easily performed.

Our first experiment consisted in positioning one Sybil per bucket. We conducted this attack by sending several `ping` requests (equivalent to a `hello request` in the Kad DHT) to the same target peer, using a unique IP address and different ports. A ping request contains the IP address and port of the receiver and the ID of the sender. In our case, every ping message contained a random ID. Figure 5.1(a) shows the number of contacts on the target peer's routing table. Firstly, we let the target peer bootstrap and reach a steady number of contacts, normally around 170 contacts. Secondly, we initiated the attack at the twelfth minute, where 160 ping requests were emitted in parallel [31]. As a result, the target peer's routing table got filled with almost all these fake contacts, reaching a total number of around 310 contacts. Because the target peer's first twenty buckets were already full with normal (not fake) contacts, new contacts could not be added in those buckets.



(a) Total number of contacts during the attack in the Mainline DHT (partial poisoning)

(b) Number of fake contacts during the attack in the Kad DHT (full poisoning). Reprinted from [7]

Figure 5.1 – Routing poisoning attack in the Mainline DHT & the Kad DHT

We performed a second experiment aiming at entirely filling each bucket, which means eight fake contacts per bucket [32]. Therefore, we sent 1280 (8 * 160) ping messages properly scheduled to the target peer. Notwithstanding, only one fake contact succeeded to enter a bucket, because the Mainline DHT does not allow two contacts in the same bucket to share the same IP address. This feature protected the routing table from a full poisoning (eight fake contacts per bucket), but not from a partial poisoning (one fake contact per bucket).

Security features in the Kad DHT

As previously mentioned in Subsection 2.5.2, Cholez *et al.* [7] studied the protections mechanisms of the Kad DHT, which have been progressively introduced in different versions of the client. These mechanisms include a flood protection mechanism, an IP limitation mechanism and a IP verification mechanism. A flood protection mechanism is achieved by keeping a history of all data packets received in the last 12 minutes, so as to avoid peers sending a high number of

31. Since we have 160 buckets, we send one message per bucket.

32. Routing table's description available at `www.http://bittorrent.org/beps/bep_0005.html`. Last visited on 08/2013.

Clients	Flood protection	IP limitation	IP verification
eMule 0.48a / aMule 2.1.3	No	No	No
eMule 0.49a / aMule 2.2.1	Yes	Yes	No
eMule 0.49b / aMule 2.2.2	Yes	Yes	Yes
The Mainline DHT	**No**	**Partial**	**No**

Table 5.1 – Enabled protections for each client version. Reprinted from [7]

messages as well as unrequested messages. The IP limitation mitigates the Sybil attack from an attacker owning a single public IP address by avoiding more than ten contacts from the same /24 subnet. Finally, an IP verification avoids identity spoofing (both for IP address and node's ID) by introducing a three-way handshake before adding a contact.

Figure 5.1(b) shows the behaviour of the Kad DHT against a routing poisoning attack. When there was no protection activated, the target peer's routing table was completely filled with fake contacts (full poisoning). However, when the target's peer activated all three protections, no fake contacts could be placed into its routing table.

Table 5.1 shows the different protections mechanisms in the Kad DHT (over the different clients) and in the Mainline DHT. These protection mechanisms mitigate an attack from a single IP address or from a /24 subnet. However, it could still possible to launch a distributed attack using several IP addresses. Cholez *et al.* [110] additionally proposed an analysis of the distribution of IDs in the Kad DHT, where the density of the IDs of peers is taken into consideration. Since an attacker aims at placing several attacking nodes in a particular DHT zone, it increases the normal density of nodes in that DHT zone and it can be therefore detected.

We have compared Kad and Mainline DHTs from a security perspective. The protection mechanisms included in the Kad DHT make the network resilient to a simple, but highly effective, routing poisoning attack, oppositely to the Mainline DHT, which is widely open to this simple attack. We continue our analysis from a performance point of view, to complete our comparison of both DHTs.

5.2.2 Performance comparison

For our performance analysis we consider three properties:
- **Publishing time:** How many seconds are required to publish a value in the DHT ?
- **Overhead during publishing:** How many messages are required to publish a value in the DHT ?
- **Lifetime of the stored information:** How long does a previous stored value last in the DHT ?

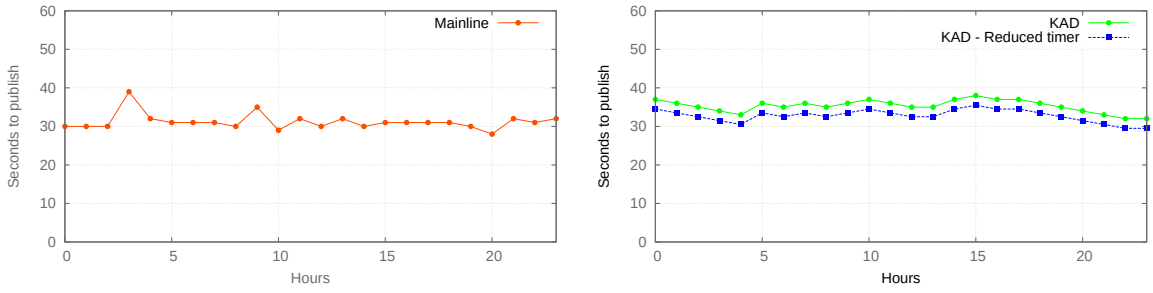
We have chosen to measure the performance regarding how long it takes to publish a value, but not how long it takes to search for it, since both operations are based on the same lookup mechanism and are expected to be symmetric. The Kad DHT and the Mainline DHT use the same routing procedure with a different request message for both, publishing and searching a value.

Publishing time

Publishing a value is one of the basic functions in any DHT. It consists in finding the adequate peers, *i.e.* the closest ones to the value, to later store the value. The time taken to publish a new

content is a good metric to measure the performance of the routing algorithm. An important difference between Kad and Mainline DHTs is the number of nodes a value will be stored in. In the Kad DHT a value is stored in the ten closest peers, while in the Mainline DHT a value is stored in the eight closest peers. To measure the publishing time, we consider the time it takes a full store in the Mainline DHT (in the eight closest nodes) and the time it takes to store a value in the eight (instead of ten) closest nodes in the Kad DHT.

During a period of twenty-four hours, we published one-thousand random values (one value per minute) in both DHTs, and measured the time to publish. Figure 5.2 shows the average time to publish for all published values. Both DHTs consumed between thirty and forty seconds to store a value, where the Mainline DHT performed a bit faster. The Kad DHT includes a fix three-second timer between the closest contacts are found and the actual publish request is emitted. Steiner *et al.* [111] showed that this three-second timer can be reduced to half a second without affecting the routing algorithm. Figure 5.2(b) depicts the time to published of the Kad DHT without this timer.



(a) Time to publish in the Mainline DHT

(b) Time to publish in the Kad DHT

Figure 5.2 – Time to publish in the Mainline DHT & the Kad DHT

Crosby and Wallach [60] conducted a performance analysis on the Mainline DHT, where they measured the time to publish around 60 seconds. They analysis was performed based on the Mainline BitTorrent client, while our analysis was performed using the Vuze client and its Mainline DHT extension[33]. Crosby and Wallach stated that different implementations of the Mainline DHT had different performances, which is visible in this case.

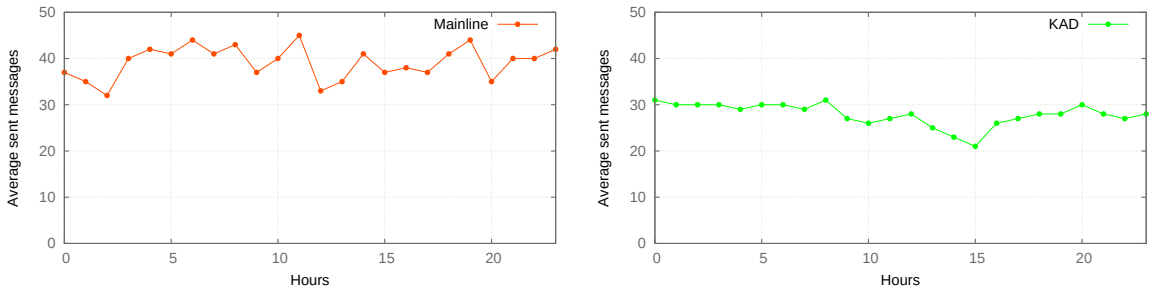
Overhead during publishing

We also measured the number of routing messages required to publish a value in both DHTs. Figure 5.3 depicts these values for both DHTs. The Kad DHT requires between 25 and 30 messages on average for each publication, while the Mainline DHT requires 40 messages on average.

To publish a value, we require first to search for the closest nodes and then to instruct them to store the value. This last part of the publishing process requires always the same number of messages: if we have a set of eight closest nodes, then we will send eight messages, one for each node. Therefore, the difference on the number of messages sent is due to routing messages to find the closest nodes. Thus, the Mainline DHT produces more routing messages than the Kad DHT.

As mentioned by Crosby and Wallach [60], the Mainline DHT implements the Kademlia

33. Plugin accessible at `http://plugins.vuze.com/`. Last visited on 08/2013.



(a) Messages sent in the Mainline DHT

(b) Messages sent in the Kad DHT

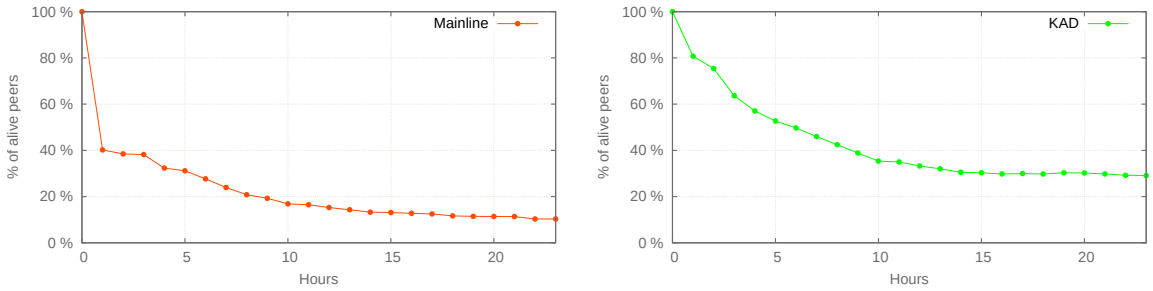
Figure 5.3 – Number of messages sent during the publishing process in the Mainline DHT & the Kad DHT

protocol with a *branching width*[34] of two, which only guarantees one bit closer at every lookup step. For that reason, the Mainline DHT uses more messages to converge to the set of closest nodes.

Lifetime of the stored information

Since churn is an intrinsic characteristic of DHTs, values previously published need a periodic republishing or they might be otherwise lost. The information lifetime gives us an approximation of the rate needed to maintain a value in the network. A DHT with a high churn rate requires a higher republishing rate.

In order to measure the lifetime of information in both DHTs, we kept the set of peers where we initially published the thousand random values and periodically checked whether these peers were alive or not. Figure 5.4 depicts the measured aliveness of these peers through time.



(a) Alive peers in the Mainline DHT

(b) Alive peers in the Kad DHT

Figure 5.4 – Percentage of alive peers in the Mainline DHT & the Kad DHT

Regarding the Kad DHT, we can observe that after the first 30 minutes, 84% of all peers are alive, and after five hours only 50% are alive. Finally, after a whole day only 28% of all peers are reachable. The Kad DHT adapts its publishing process accordingly by republishing keywords every twenty-four hours and files after five hours. Keywords are republished by all peers sharing files, and therefore a twenty-four hours republishing rate is suitable.

34. During the lookup process, a node finds systematically the closest nodes to store a value. The *branching width* indicates how fast the lookup process converges.

The Mainline DHT has higher churn levels. After the first thirty minutes, only 41% of all peers are alive. If we consider the initial set of eight nodes where a value is stored, it means that after thirty minutes, only three nodes remain online, decreasing the probability of finding a value. Finally, after twenty-four hours only 9% of the entire set of nodes is no longer reachable. The analysis by Crosby and Wallach [60] presents the same results, where they defined as *infant mortality events* those nodes that become unreachable after a short period of time. This explains the sudden drop in the curve of aliveness in the Mainline DHT.

The Kad DHT outperforms the Mainline DHT in both aspects, security and performance. This DHT includes three protection mechanisms to avoid different attacks, such as routing table poisoning attacks. It also requires less messages to publish a value and the churn rate in the network is lower when compared to the Mainline DHT.

5.3 The download algorithm of the BitTorrent and the Ed2k networks

The performance characteristics of the Kad DHT and its security features make it an excellent DHT: its routing algorithm achieves good lookup times and it is resilient to most known attacks. The churn level measured in the network is a key feature to take into consideration, since it strongly affects the final performance, even though it does not depend on the DHT design itself, but on users' behaviour.

Although the DHT level is important, the final users are more sensitive to the performance of the algorithm of download. The overall download can be overshadowed by a slow download algorithm, even if the indexing process operates extremely fast and well.

In order to determine the performance of the algorithm of download of the Kad/Ed2k and the BitTorrent networks, we measured how long it took for a Kad/Ed2k and a BitTorrent clients to download a seven-hundred mega-bytes random file. Our experiment was deployed using the PlanetLab testbed with a total of fifty distributed nodes. We considered a single-file download for our experiments, since that is a normal condition in file-sharing environment, to download a single file at a time.

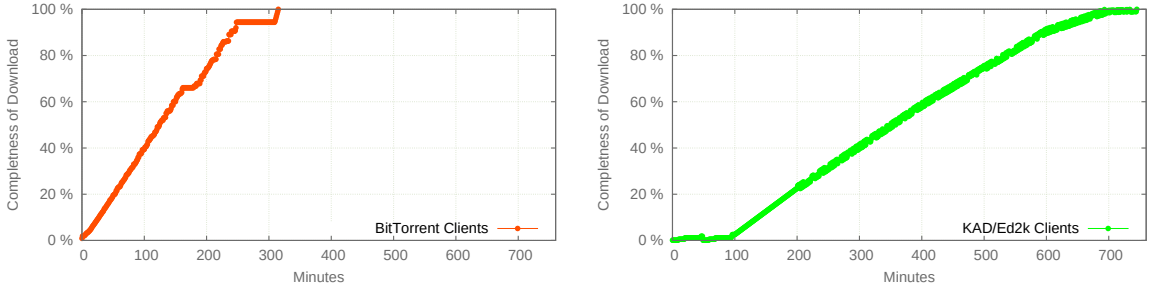
We consider two cases, with one initial seeder and with ten initial seeders[35]. For our experiments we used the same clients as for the comparison of DHTs in Section 5.2.

5.3.1 Download time with one seeder

Starting with one initial seeder is the normal behaviour, since the original publisher initially uploads itself as the only source and new peers eventually complete the download and become new seeders. Figure 5.5 shows a comparison between BitTorrent and Ed2k clients. While BitTorrent clients achieved a complete download of the file in 315 minutes, Ed2k clients achieved it in 745 minutes.

The algorithm of download of BitTorrent performs better and achieves, on average, a complete download in 42% of the total time of a Ed2k client. It is normal that the total number of seeders increases fast, therefore we conducted a second experiment with ten initial seeders and measured the final time to download.

35. Despite *seeder* being a BitTorrent's term, it represents in the Ed2k network a peer with the entire content.



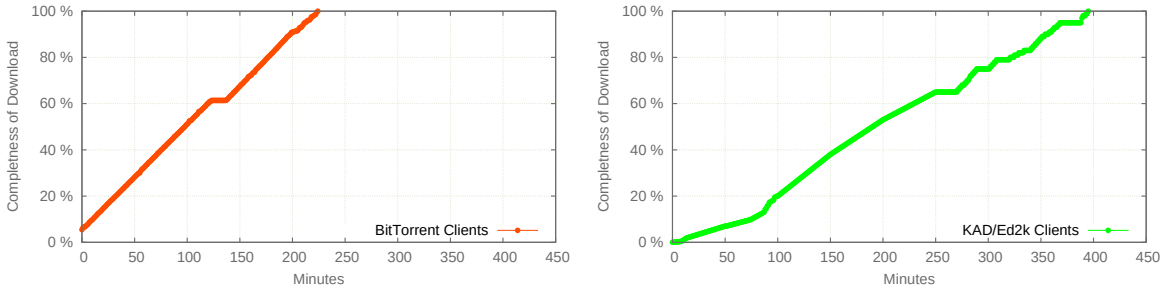
(a) Average time of download for BitTorrent clients with one initial seeder

(b) Average time of download for Ed2k clients with one initial seeder

Figure 5.5 – Time to download for BitTorrent & Ed2k clients with one initial seeder

5.3.2 Download time with ten seeders

New content tends to disseminate fast and new seeders become available in a short period of time. We considered that ten initial seeders adequately represent non-popular content. Figure 5.6 depicts the results of the experiment. In this case, BitTorrent clients took 224 minutes to complete the download, while Ed2k clients took 395 minutes in total. BitTorrent clients still perform better, completing the download in 57% of the total time to download of Ed2k clients.



(a) Average time to download for BitTorrent clients with ten initial seeders

(b) Average time to download for Kad/Ed2k clients with ten initial seeders

Figure 5.6 – Time to download for BitTorrent & Ed2k clients with ten initial seeders

BitTorrent's download algorithm overcomes Kad/Ed2K's algorithm when it comes to a single-file download, when 2% of the peers are seeders (1 source among 50 peers) and when 20% of peers are seeders (10 peers out of 50). In both experiments, BitTorrent clients completed the download in half of the time of Ed2K clients.

5.4 A hybrid model with the BitTorrent and the Kad/Ed2k networks

Our previous evaluation demonstrated that the Kad DHT outperforms the Mainline DHT and includes different protections mechanisms. These mechanisms make the network resilient to complex attacks, on the opposite to the Mainline DHT, which is open to a basic routing poisoning attack. However, in a single-file download scenario, the algorithm of download of BitTorrent performs faster than the Ed2k's algorithm

According to these results, we consider a combined approach. This approach aims at improving the indexation of content within the BitTorrent environment by using the Kad DHT to index BitTorrent's content, leading to a hybrid file-sharing network. In this section, we first introduce an abstract and hybrid model for file-sharing. Then, we instantiate this model with the BitTorrent network and the Kad/Ed2k network. Finally, we detail a hybrid client called **hMule** serving as the interconnection point between these two networks.

5.4.1 An abstract hybrid model for file-sharing

Figure 5.7 shows a high-level view of our interconnection model. We consider two networks, A and B, and an interconnection layer serving as the meeting point for users from both networks. *Hybrid nodes* form this interconnection layer and are responsible for forwarding messages and traffic between networks. Three types of interactions are possible: 1) among nodes from an interconnected network and hybrid nodes ; 2) among hybrid nodes ; and 3) among nodes from the same interconnected network.

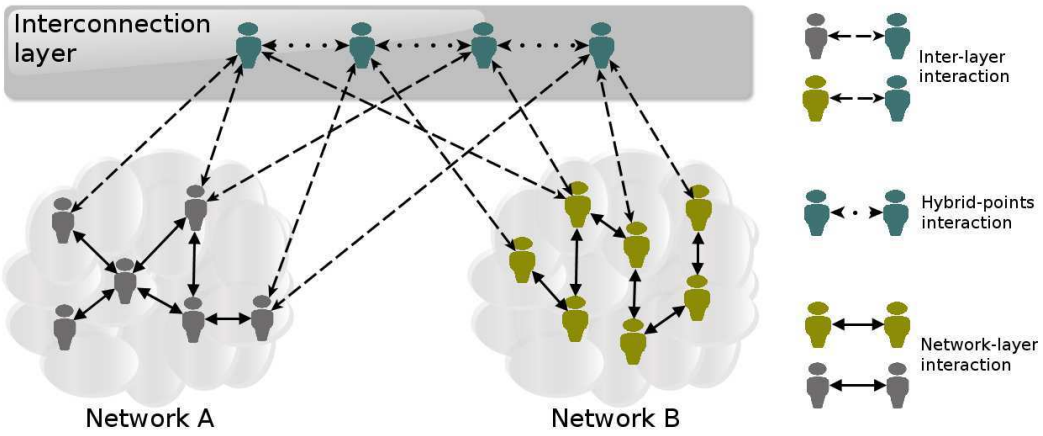


Figure 5.7 – Abstract hybrid file-sharing model.

Table 5.2 gives all the parameters of our hybrid model, to be defined when the model is instantiated.

- The parameter `interaction type` indicates whether both networks' users communicate between each other (through the interconnection layer) or if the communication is only in one way (from network A to network B, for instance).
- `Hybrid nodes deployment` denotes whether the hybrid nodes are deployed at the beginning of the interconnection or they dynamically deployed on-demand.
- The `number of hybrid nodes` is directly related to the number of users in both networks, the capacity of each hybrid node and the intended coverage of the interconnection, as denoted in Equation 5.1. U_a and U_b indicates the number of users in networks A and B respectively, while C_a and C_b represent the number of users a single hybrid node can manage in networks A and B, respectively. IC indicates the intended coverage and can take values in the range [0,1]. A value of one specifies that we want to interconnect every node in both networks at the same time.

$$\texttt{nb\_hybrid\_nodes} = \texttt{MAX}(\lceil U_a/C_a \rceil, \lceil U_b/C_b \rceil) * IC \tag{5.1}$$

Parameter name	Type of value	Description
Interaction type	Unidirectional/Bidirectional	Unidirectional (A->B or B->A) or Bidirectional (A<->B)
Deployment of hybrid nodes	On-demand/Fix	Are hybrid nodes deployed dynamically ?
Number of hybrid nodes	Numeric value	How many hybrid nodes do we need ?
Protocol translation	Yes/No	Do networks A and B use the same protocol ?
Interaction of hybrid nodes	Yes/No	Do hybrid nodes interact among themselves ?
Type of hybrid node	Passive/Active	Are hybrid nodes detectable by non-hybrid nodes ?

Table 5.2 – Parameters of the hybrid model

- A `protocol translation` is needed when both interconnected networks operate under different protocols, such as different file-sharing networks.
- The interaction of hybrid nodes is useful to exchange network and control messages. This interaction can improve the interconnection, but it increases the load among hybrid nodes.
- Finally, `type of hybrid node` indicates whether these nodes are perceived as different nodes on both networks A and B and therefore users are aware of the these hybrid nodes ; or hybrid nodes behave and operate exactly as normal nodes, thus being unidentifiable by the users.

5.4.2 An instantiation with the BitTorrent and the Kad/Ed2k networks

Figure 5.8 introduces a high-level-view of this hybrid scenario, including the Kad DHT and the BitTorrent's algorithm. A user queries the first level of indexation of the Kad DHT with different keywords to search for the desired content. Once the particular content (and its associated hash value) has been chosen, the user queries the second level of indexation of the Kad DHT to retrieve the list of sources for that content. This list of sources enables the user to join a particular BitTorrent swarm and download the required file through the BitTorrent's algorithm. All these steps are summarised in Table 5.3.

Steps	Description	Input	Output
1	Retrieve a set of possible content and select one	Keywords	Hash
2	Retrieve a list of sources for that Hash	Hash	Sources
3	Join the BitTorrent swarm	Sources	File

Table 5.3 – A hybrid content indexation/distribution mechanism.

Whenever a file has been downloaded through the normal BitTorrent network, it is published automatically into the Kad DHT. Thus, BitTorrent content becomes indexed in the Kad DHT, enabling users to publish or search torrents by keywords, a missing feature in the BitTorrent's decentralised trackers.

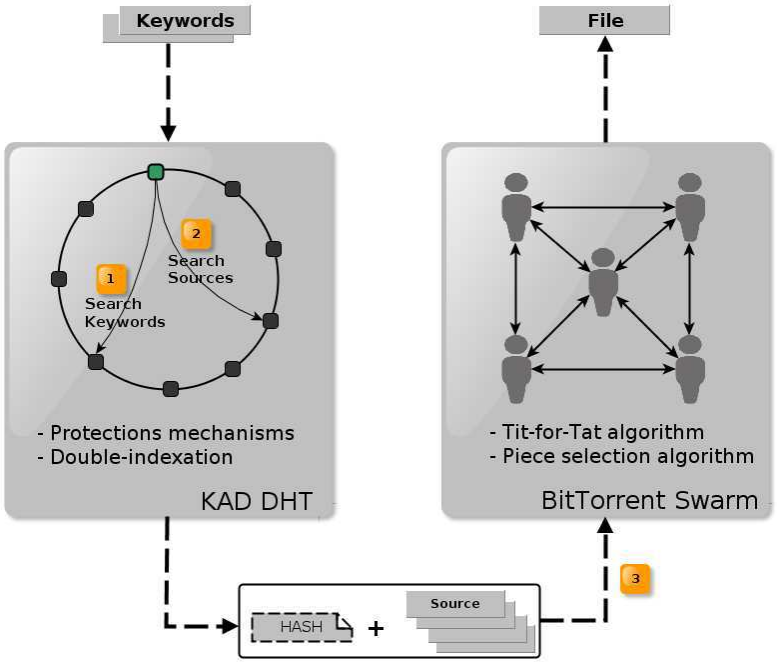


Figure 5.8 – BitTorrent-Kad/Ed2k hybrid approach

We instantiate this high-level-view scenario with our abstract model. Figure 5.9 depicts the instantiation of our model with both BitTorrent and Kad/Ed2k networks, where the interconnection layer is formed by hybrid nodes called **hMule**.

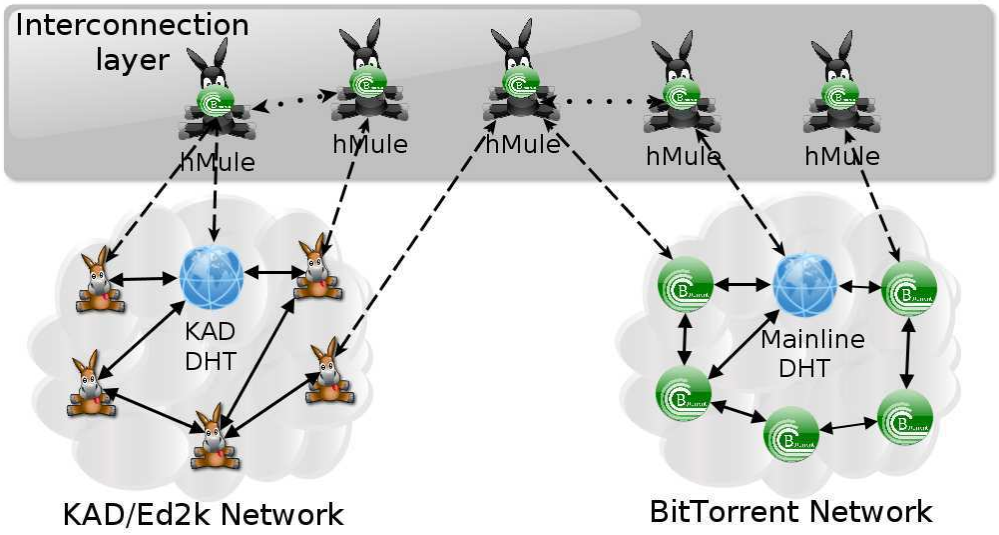


Figure 5.9 – Instantiation of our hybrid model with the BitTorrent and the Kad/Ed2k networks

The hMule client is the meeting point for both networks and implements the three-step process depicted in Figure 5.8. HMule is able to search in the Kad DHT for BitTorrent content based

on different keywords. Once the content has been found, it is downloaded using the BitTorrent's algorithm. Thus, hMule users can benefit from a secure and keyword-based indexing mechanism while using the BitTorrent's algorithm. Besides being able to exploit both networks, hMule clients are backward compatible and can connect to normal BitTorrent clients and download content from the pure BitTorrent network, as well as connect to normal Kad/Ed2k clients and download content from aMule/eMule clients.

Table 5.4 introduces the values for each of the parameters of the hybrid model for file-sharing.

Firstly, hMule clients are able to access the Mainline DHT and download content from the normal BitTorrent network. Once this content has been downloaded, it is automatically published in the Kad DHT, where remote aMule/eMule clients can search for it and download it through the hMule client (using the Ed2k's algorithm). Secondly, hMule clients can download content from the Kad/Ed2k network and publish it in the Mainline DHT, where normal BitTorrent clients can access it and download it from the hMule client (using the BitTorrent's algorithm). This is the best option to maintain a backward compatible client, while introducing our hybrid model. So, the `Interaction type` is bidirectional interaction.

Hybrid nodes are deployed by users, fully controlling the client. Users can start or stop the hMule client by themselves, leading to an `on-demand` deployment process. Therefore, it is not applicable to define a number of required hybrid nodes to support the interconnection, since there is no central coordinator to manage the interconnection.

Traffic do not flow from one network to the other one directly. The hMule nodes keep the content and make it accessible to both networks. So, a `protocol translation` is not required.

HMule nodes interact among themselves through the BitTorrent's algorithm to share BitTorrent content. Even if hMule nodes implement the Ed2k's algorithm, they always use BitTorrent's algorithm to share content among themselves.

Finally, hMule nodes are `passive nodes`. Normal aMule/eMule clients, as well as normal BitTorrent clients, cannot identify hMule nodes when interacting with them.

Parameter	Value
Interaction type	Bidirectional
Hybrid nodes deployment	On-demand
Number of hybrid nodes	Not Applicable
Protocol translation	Not Applicable
Hybrid-nodes interaction	Yes
Type of hybrid node	Passive

Table 5.4 – Parameters of the instantiation for the BitTorrent and the Kad/Ed2k networks

We defined the parameters to maximise the characteristics of both networks. We maintain a fully backward-compatible client, where BitTorrent and Kad/Ed2k clients can interact with hMule clients without any modifications.

5.4.3 The hMule client

The hMule node is the interconnection point between both the BitTorrent and the Kad/Ed2k networks, improving the indexation of content in the BitTorrent environment. Users are required to use this hybrid client to profit from our improved three-steps indexing mechanism, while still being able to access the BitTorrent and the Kad/Ed2k networks, resulting in a fully backward-

compatible client. The development of the hMule client was done in collaboration with Damian Vicino [112].

We now describe the implementation of the hMule client and its main characteristics.

Implementation of the hMule client

HMule is a modular C++ implementation, where we extend the aMule client to support the BitTorrent's algorithm as an additional algorithm to download. The aMule [36] open-source client was employed due to its modular architecture and its Linux-based implementation.

LibTorrent-Rasterbar [37] is the library used to support the BitTorrent functionality owing to its high-level interface to manage download sessions and individual torrents and peers. Additionally LibTorrent-Rasterbar supports a wide set of BitTorrent extensions, such as access the Mainline DHT, BitTorrent's metadata exchange [38], UPnP and IP filters.

The strategy to download and the hMule's referee

HMule uses both BitTorrent and Ed2k algorithms to connect to both networks. Whenever an hMule client intends to download content from the Kad/Ed2k network, it searches in the Kad DHT for content's sources. These sources can be, either normal aMule/eMule clients or hMule clients.

If the list of sources contains only aMule/eMule clients, the hMule client uses the original Ed2k's algorithm. However, if among the retrieved sources there are other hMule clients, the hMule client connects only to these hybrid clients and download the content using the BitTorrent's algorithm. This procedure is made by a component called *referee*, which decides to which remote sources connect to.

The diagram of sequence 5.10 depicts this interaction. The hMule client retrieves a set sources, both aMule/eMule and hMule clients from the Kad DHT. It then requests the content's filename, where each remote source responds with the filename. Remote hMule sources will respond additionally with the BitTorrent's infohash identifier of the content.

Once the hMule client has received a response from every remote source, it evaluates if the number of hMule clients accounts for more than a predefined threshold [39]. If so, the hMule client decides to download the content using the BitTorrent's algorithm and request the torrent file from the remote hMule clients. If the threshold is not reached, the hMule client downloads the content from the aMule/eMule sources. Thus, hMule dynamically decides whether to use the BitTorrent's algorithm or Ed2k's algorithm.

Conversion of hash identifiers

The Kad DHT uses the MD5 hashing function and therefore 128-bit identifiers are employed to index keywords and files. On the contrary, the Mainline DHT uses the SHA-1 hashing function and hence 160-bit identifiers. A problem arises when an hMule client publishes BitTorrent content, which contains a 160-bit identifier, into the Kad DHT, which uses a 128-bit identifier.

Before publishing BitTorrent content into the Kad DHT, the hMule client hashes the original 160-bit identifier of the Mainline DHT with the MD5 function, obtaining the required 128-bit

36. `http://sourceforge.net/projects/amule/`. Last visited on 08/2013.
37. `http://www.rasterbar.com/products/libtorrent/`. Last visited on 08/2013.
38. `http://bittorrent.org/beps/bep_0009.html`. Last visited on 08/2013.
39. Preliminary tests were conducted with a threshold of 50%, indicating that if half of the remote sources are hMule clients, then the content is downloaded through BitTorrent's algorithm.

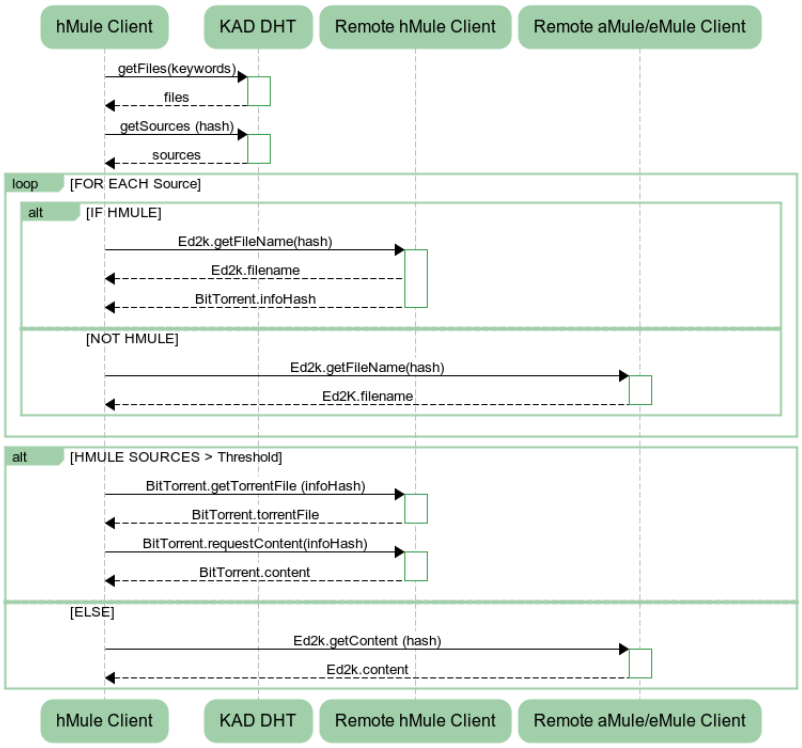


Figure 5.10 – The adaptive download mechanism of hMule

Kad identifier. This mapping is locally maintained by the hMule client and persisted through different sessions. Whenever the hMule client receives a filename's request for a given Ed2k's hash, it searches in its local map for the hash to determine whether it has been previously computed from a Mainline DHT's infohash. If so, it can additionally answer the request with the Mainline DHT's infohash.

This mapping solves the main incompatibility problem between the identifiers of the Mainline DHT and the Kad DHT. It only requires a extra message to communicate the Mainline DHT's infohash, which is implemented as an extension of the eMule protocol. The mapping between hash identifiers is extremely light in terms of memory use, with barely thirty-five kilobytes for a list of one-thousand `<Ed2k's identifier, Mainline's identifier>` pairs.

5.4.4 Evaluation of the hMule client

The hMule client is part of the hMule project [40], is fully implemented and freely accessible under GPL license.

The client has been deployed on the PlanetLab tested to corroborate the right integration in both BitTorrent and Kad/Ed2k networks. We successfully downloaded non-copyrighted content from the BitTorrent network, and published it into the Kad DHT, where further hMule clients downloaded it using the BitTorrent's algorithm. Additionally, aMule clients were able to search and download this content from hMule clients, using the Ed2k's algorithm.

We did not modify the algorithms of download and therefore they are expected to perform as previously measured. The process of indexation of the Kad DHT was not modified and there-

40. The hMule project is accessible at `http://hmule.gforge.inria.fr/`.

fore it behaves as previously shown. Table 5.5 positions our hybrid model with the reviewed interconnection models presented in Chapter 3. Our model in the only model considering widely deployed file-sharing networks and their dynamic topologies. We use an *ad-hoc* overlay to organise our hybrid nodes, a common denominator in hybrid interconnection models.

Use	General use	Org. models	Peer-to-peer file-sharing			
Model	Synapse	Yang's scheme	Lloret's scheme	Konish's scheme	Fu's extension	**hMule**
Interaction component	Synapses (Bridges)	N/A	Super nodes	Cooperative peers	Shared Peers	**Hybrid nodes**
Organisation	*ad-hoc* overlay	2-Tiers (Structured& unstructured)	2-layers / *ad-hoc* overlay	*ad-hoc* overlay	*ad-hoc* overlay	***ad-hoc* overlay**
Deployment	Sim. (With Churn)	Sim. (No Churn)	Sim. (No Churn)	Sim. (No Churn)	Sim. (No Churn)	**Real networks**
Topology	Dynamic	N/A	Random	Barabasi-Albert (1000 nodes)	Kazaa topology	**BT/Ed2k topology**
Multi protocol	Yes	No	Undefined	No	Yes, but untested	**Not applicable**
Start of interconnections	Network decision	Network decision	Network decision	Network decision	Local decision	**On-demand**

Table 5.5 – The hMule hybrid model compared with existing interconnection models

5.5 Conclusion

In this chapter we presented a comprehensive study of the Mainline DHT and the Kad DHT from a security and performance perspective. We further compared BitTorrent's algorithm and Ed2k's algorithm in a single-file download scenario.Through real-world experiments we put in evidence the lack of protections mechanisms of the Mainline DHT against basic attacks and demonstrated that the Kad DHT is resilient to the same attacks. Our experiments showed that BitTorrent's algorithm is approximately twice as fast than Ed2k's algorithm when considering a real-world single-file download scenario.

We analysed and proposed a hybrid approach for file-sharing, which uses the Kad DHT to index content and the BitTorrent's algorithm to download it. Our hybrid model is the first interconnection model to consider widely deployed file-sharing networks. We instantiated our model through a fully backward-compatible file-sharing client called **hMule**. This client enables a user to index BitTorrent content in the Kad DHT, still enabling the user to connect to both networks individually. The hMule client is fully implemented and freely accessible.

The next chapter presents the interconnection of anonymous and non-anonymous file-sharing environments. We use the same abstract model presented in this chapter, and we instantiate it with the anonymous I2P network and the BitTorrent network.

Chapter 6

Improving content availability in the I2P file-sharing environment

Contents

6.1 Introduction

Peer-to-peer file-sharing applications generate an important part of current Internet traffic, where the BitTorrent network produces, in average, a third of all European upstream traffic. Currently, the BitTorrent network hosts several millions of torrents and users simultaneously, making it the biggest content peer-to-peer delivery community.

Additionally, anonymous communications have been constantly increasing and Internet users are shifting to a privacy-preserving Internet. Users realise the importance of maintaining a certain degree of anonymity when accessing the Internet so as to keep their online ideas and their real identities separated. In anonymous communications, anonymous file-sharing whose goal is to maintain a user's identity hidden and avoid censorship or file-sharing profiling through data mining, while downloading different content.

However, and despite the wide range of anonymous file-sharing options, the available content on these systems is reduced and often out-dated. Public communities, on the other hand, are still the biggest source of content. Therefore, the problem becomes to anonymously access public content.

This chapter presents a hybrid approach for the BitTorrent and I2P networks, aiming at improving content availability in the anonymous I2P BitTorrent-like file-sharing environment. We propose a fully anonymous file-sharing environment, where content indexation, as well as content distribution are anonymous. We firstly show that I2P's file-sharing environment suffers from a considerable lack of content, barely containing 1% of all BitTorrent's available content. Secondly, we instantiate our abstract hybrid model presented in Subsection 5.4.1 with both BitTorrent and I2P file-sharing networks. Finally, we evaluate our instantiation through the implementation of a hybrid client, called **BiTIIP**, which allows I2P file-sharing users to access public BitTorrent content.

6.2 Content availability in the I2P network

It seems normal that popular content gets available first in public communities than in anonymous networks, mainly when considering content such as Movies, TV shows, Games or Music. Users download content from public file-sharing networks and then introduce it in anonymous file-sharing communities. Inevitably, the amount of content available in anonymous file-sharing environments should be considerably lower than in public communities. In order to assess the lack of content in the I2P anonymous file-sharing environment, we conducted a thirty-day measurement to determine the rate of new content introduced per day in the public BitTorrent community, and in the I2P file-sharing community.

We considered Torrentz [41], a major meta-search engine for BitTorrent content, which indexes torrent from various torrent sites, including The Pirate Bay [42] and BitSnoop [43]. Regarding I2P, we used the Postman tracker [44]., which is the biggest BitTorrent tracker available in the I2P network.

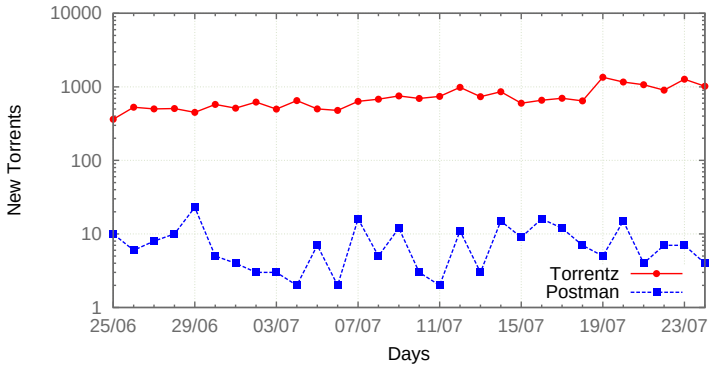


Figure 6.1 – New content introduced in the BitTorrent and the I2P file-sharing networks

Figure 6.1 depicts the amount of new content introduced every day in the BitTorrent public community and in the I2P Postman tracker. There are, in average, 720 new torrents in Torrentz and roughly 8 new torrents in the Postman tracker every day. We can also observe that Torrentz presents a more stable rate of new torrents per day, as opposed to the Postman tracker, where we can notice different peaks, most of them close to weekends (29/06, 07/07, 14/07 and 20/07).

41. `http://torrentz.eu/`. Last visited on 08/2013.
42. `http://thepiratebay.sx/`. Last visited on 08/2013.
43. `http://bitsnoop.com/`. Last visited on 08/2013.
44. `http://tracker2.postman.i2p/`. Last visited on 08/2013.

Trackers	Movies	TV Shows	Games	Music	Total
Torrentz	21.11%	47.45%	8.27%	23.15%	21648
Postman	48.30%	18.22%	2.96%	30.50 %	236

Table 6.1 – Categories of new content in the BitTorrent and the I2P file-sharing networks

We took into account four main content categories for our measurement: Movies, TV shows, Music and Games. Table 6.1 shows the amount of new content per category for every tracker and the total number of torrents measured. Barely the 1% of the torrents in Torrentz are present in the Postman tracker. Moreover, Torrentz reported 19 million active torrents, while the Postman tracker reported around 12000 active torrents. Even though the effort of different users to introduce BitTorrent content into the I2P network, we can assert the lack of new content in the I2P file-sharing environment.

We propose an *on-demand* mechanism, enabling I2P users to access BitTorrent content. The next section presents our interconnection model for the BitTorrent and I2P networks.

6.3 Interconnecting the I2P and the BitTorrent networks

We showed that the available content in the I2P file-sharing environment is reduced. We bring forward an interconnection scheme for both BitTorrent and I2P file-sharing networks, aiming at improving the content availability in the latter. Thus, we enable a fully-anonymous content distribution scheme, which includes anonymous content indexation as well as anonymous content distribution.

Interconnecting anonymous and non-anonymous environments needs to take into consideration which components of the interconnection have to be anonymous and which not. As previously mentioned in Subsection 4.2.1, the **anonymity of a subject** is the impossibility to discriminate the subject among a set, called the *anonymity set*. **Unlinkability** refers to the impossibility to link together two *items of interest* (e.g. a message to a user, for instance) from an attacker's point of view.

Hence, our main design goal for the BitTorrent-I2P interconnection is twofold. On the one hand, we aim at maintaining the anonymity of an I2P user, defined as UA_i, while interacting with a BitTorrent user, defined as UB_j, for the anonymity set defined in 6.1. On the other hand, we seek the unlinkability between an anonymous I2P user and the content he/she is downloading. From an attacker's point of view, it implies that it is not possible to bind successfully an I2P user with a public BitTorrent download.

$$\texttt{anonymity\_set} = \{UA_i, UB_j\}, \ \forall i, j \tag{6.1}$$

In this section, we first instantiate our abstract model for file-sharing, defined in Subsection 5.4.1 and its parameters accordingly. Then, we detail the hybrid client **BiTIIP** and its operation. Finally, we consider the components' anonymity of our model and their characteristics.

6.3.1 A hybrid file-sharing model for the BitTorrent and I2P networks

The BitTorrent network, as well as the I2P file-sharing network, are *swarm*-based, where different users share a particular content on the same file-oriented peer-to-peer network. Different swarms form the entire file-sharing network, where a user can take part in more than one swarm

at a time. We consider the interconnection among anonymous and non-anonymous swarms, thus we have different interconnections for different swarms.

Figure 6.2 introduces every component of our interconnection model. The interconnection layer is composed of hybrid nodes called *BiTIIP*. I2P users from a particular swarm interact with BitTorrent user from another swarm through our BiTIIP clients, thus forming a unique swarm and sharing the same content.

In our interconnection design, an I2P user needs to request specifically the BitTorrent content to download. This procedure is achieved in a single step in our design. The I2P user needs to request in our interconnection's eepsite [45], called *connectMe.i2p*, the desired BitTorrent content (through a magnet link or a torrent file) to download. With this single step, a new BiTIIP client will be started to respond to this demand and allows the I2P user to access the desired BitTorrent content. The new BiTIIP client will index the requested content in the I2P's Mainline DHT and in this way, the I2P user can subsequently query the I2P's Mainline DHT and find the content.

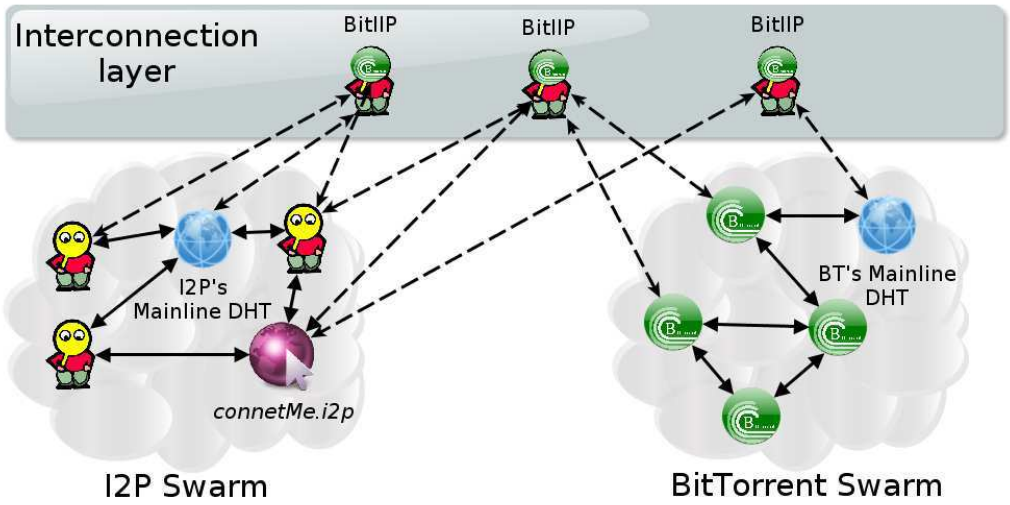


Figure 6.2 – Instantiation of our hybrid model with the BitTorrent and the I2P networks

A BiTIIP client connects to both BitTorrent and I2P swarms and allows the I2P users to access BitTorrent traffic, while enabling BitTorrent clients to request different pieces [46] of the content being downloaded from the I2P side. Therefore, we consider a bidirectional interconnection, since BiTIIP clients forward traffic from the BitTorrent network to the I2P network and vice-versa.

One important feature of our design is that every interconnection is started *on-demand*. The first I2P user willing to download a particular BitTorrent content will request it in our eppsite and a new BiTIIP client will be started. Once an interconnection has been started, further I2P users requesting the same BitTorrent content in the eepsite will be notify that the content has already been requested and therefore is already accessible in the I2P network.

A BiTIIP client has a limited capacity in terms of connections to remote clients. We analysed the code of different BitTorrent clients and determined that the maximum number of connections oscillates around one-hundred connections. We consider this value as the capacity for each BiTIIP

45. I2P enables anonymous web sites called *eepsites*. These have been previously presented in Subsection 4.2.3.
46. BitTorrent content is divided into fixed-size pieces for download, where clients request the pieces one-by-one as specified at `http://bittorrent.org/beps/bep_0003.html`. Last visited on 08/2013.

client, *i.e.* how many users a BiTIIP client can manage in every swarm and therefore in Equation 5.1 we have $C_a = C_b = 100$. However, current top BitTorrent swarms have more than ten-thousand users [47], while I2P swarms barely have one-hundred users [48]. For this reason, we only consider an interconnection between an I2P swarm and a fraction of a BitTorrent swarm [49], thus applying the *intended coverage* (IC) parameter only to the smallest swarm, the I2P swarm. As a result, equation 5.1 becomes `nb_hybrid_nodes =` $\lceil U_{i2p}/100 \rceil$ `* IC`.

The I2P and BitTorrent network use different protocols. The I2P network uses *destinations* instead of tuples *<IP address, port number>* and *tunnels* to support its anonymous communications. Therefore, BiTIIP clients need to translate I2P messages into BitTorrent messages and vice-versa.

In our instantiated interconnection model, BiTIIP clients do not interact among themselves to perform control operations nor to share pieces. BiTIIP clients do not hold the downloaded content, but just forward it. Every request to a BiTIIP client is forwarded and not resolved on the client. Connections between BiTIIP clients will only increase the load in the interconnection and therefore are not allowed.

Finally, BiTIIP clients behave exactly as BitTorrent clients in one side and I2P users in the other side, and therefore they are undetectable for the rest of the users. Table 6.2 summarises all the parameters of our instantiation for the I2P and BitTorrent networks.

Parameter	Value
Interaction type	Bidirectional
Hybrid nodes deployment	On-demand
Number of hybrid nodes	$\lceil U_{i2p}/100 \rceil$ * IC
Protocol translation	Yes
Hybrid-nodes interaction	No
Type of hybrid node	Passive

Table 6.2 – Defined parameters for the hybrid model instantiated for the BitTorrent and the I2P networks

This instantiation enables a full anonymous file-sharing environment. The indexation of the content is performed through the I2P's Mainline DHT, while the distribution of the content is achieved by means of the use of BiTIIP clients. The next section introduces how the BiTIIP clients operate and their interaction with I2P users.

6.3.2 Operation and interaction of BiTIIP clients

BiTIIP clients are dynamically started and deployed based on requests from the I2P users. Whenever an I2P user wants to download BitTorrent content, he/she needs to contact our eepsite. The sequence diagram 6.3 depicts the interaction between an I2P user and our eepsite.

The first step is to request the content in our eepsite using a magnet link or a torrent file. If the content has already been requested by other I2P user, *i.e.* a BiTIIP client has already been started, the eepsite notifies the I2P user that the content is already available. If the content is

47. Statistics available at `http://torrentz.eu/`. Last visited on 08/2013.
48. Statistics available at `http://tracker2.postman.i2p/`. Last visited on 08/2013.
49. We are aiming at introducing BitTorrent content into the I2P network and therefore we do not require to interconnect all BitTorrent users.

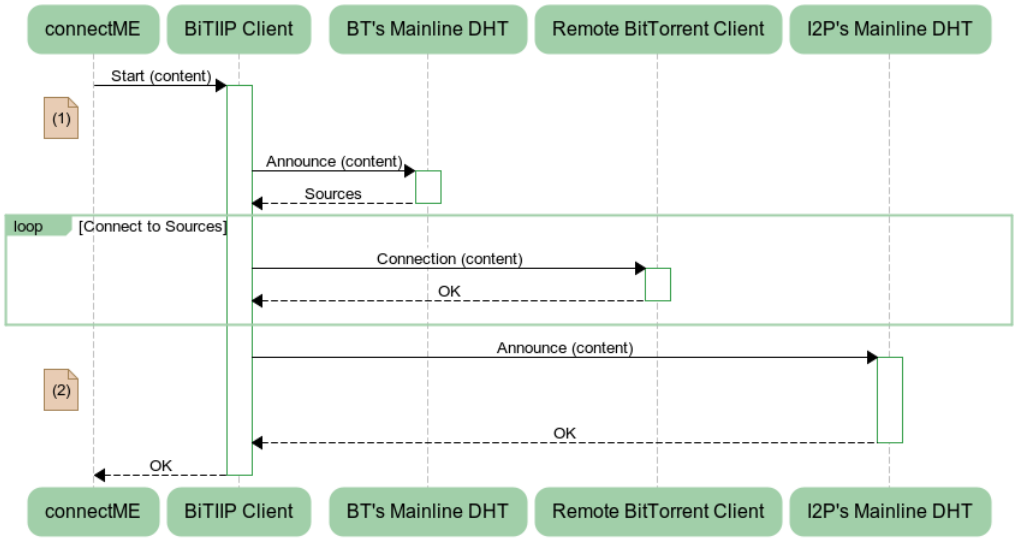


Figure 6.3 – Interaction between an I2P user and the *connectMe.i2p* eppsite

being requested for the first time, the eepsite notifies the users that a new BiTIIP client has been created.

The second step, performed by all I2P users, is to search the I2P's Mainline DHT for content's sources. In the event the content has already been requested, the I2P user will retrieve not only a BiTIIP client but others I2P users sharing the content.

The sequence diagram 6.4 depicts the interaction between our eepsite and BiTIIP clients.

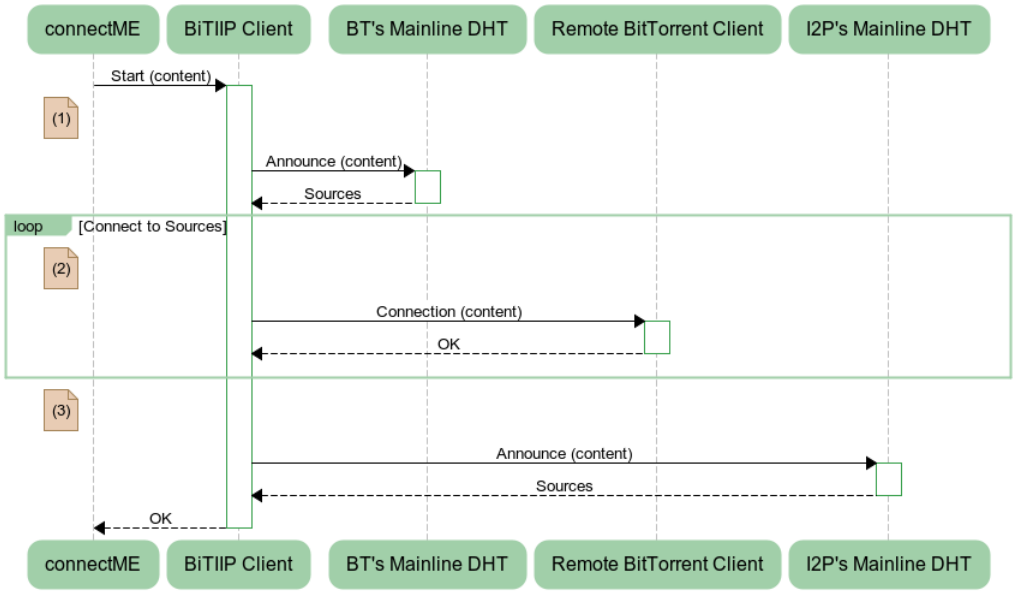


Figure 6.4 – Interaction between *connectMe.i2p* and a BiTIIP client

When the *connectME.i2p* eepsite receives a request message and no interconnections exist, a new BiTIIP client is started (1). The BiTIIP client will search the BitTorrent's Mainline DHT

for BitTorrent sources, and will connect to all retrieved sources to join the BitTorrent swarm (2). Finally, the BiTIIP client will index the requested BitTorrent content in the I2P's Mainline DHT. This last step allows I2P users to find sources for the requested BitTorrent content.

6.3.3 Interconnection layer's anonymity

We define an interaction between anonymous and non-anonymous environments, where our design goal is to maintain the anonymity of users in the I2P network. It also implies to maintain the unlinkability between an anonymous I2P user and a BitTorrent download.

As mentioned in Subsection 4.2.3, I2P uses unidirectional tunnels to ensure anonymous communications, where every user defines his/her own tunnels and therefore his/her own tradeoff between anonymity and performance: shorter tunnels achieve better performance but guarantee less anonymity on the contrary to longer tunnels.

Alice uses its own tunnels to connect to remote BiTIIP clients as depicts in Figure 6.5 and therefore her communications remain anonymous. BiTIIP clients use a *0*-hop tunnel configuration to connect to remote I2P users. They reduce their anonymity but increase their performance. As BiTIIP clients connect directly to BitTorrent clients, their anonymity will be reduced to zero in any case and a configuration with *0*-hop tunnels is adequate in this case.

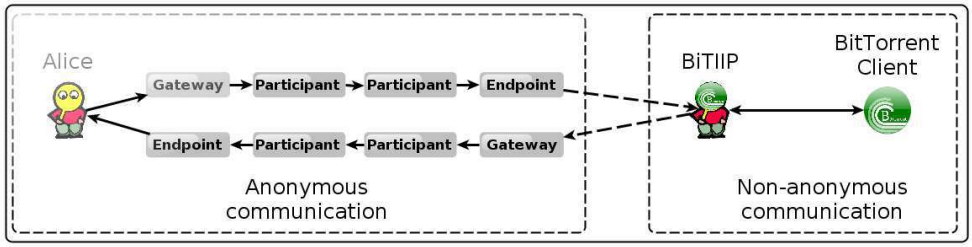


Figure 6.5 – Tunnel-based communication and BiTIIP's non-anonymous communications

This configuration allows us to achieve better throughput rates, while preserving I2P users' anonymity. From an attacker's point of view, it is only possible to link a public download with a particular BiTIIP client and not further into the I2P network. Even by placing a malicious BiTIIP client, an attacker will not able to de-anonymise an I2P user.

6.4 Evaluation of the BiTIIP client

BiTIIP is a Java-based client, using the Snark BitTorrent Library [50] and the I2PSnark file-sharing library. The Snark library enables BiTIIP to interact with BitTorrent clients, while I2PSnark enables BiTIIP to communicate with other I2P users performing file-sharing. BiTIIP serves as an interface between these two libraries and to synchronises requests from one network to the other one, while performing opportunistically caching of pieces.

Figure 6.6 depicts all the components on a BiTIIP client. The Snark library requires further extensions to support decentralised tracking through the Mainline DHT, as well as to support the BitTorrent extension protocol to exchange BitTorrent metadata [51], and therefore two components

50. The Snark project is available at `https://code.google.com/p/snark/`. Last visited on 08/2013.

51. Specification available at `http://bittorrent.org/beps/bep_0009.html` and `http://bittorrent.org/beps/bep_0010.html`. Last visited on 08/2013.

were included to the BiTIIP client to support these functionalities. On the contrary, the I2PSnark library enables decentralised tracking through the I2P's Mainline DHT. It additionally does not need an extension to support exchange of metadata, since in our interconnection model an I2P user accesses BitTorrent content and therefore only BitTorrent's metadata. The shared storage is the central component interacting with both libraries, receiving piece requests from one network, retrieving these pieces from the other network, and forwarding the requested pieces.

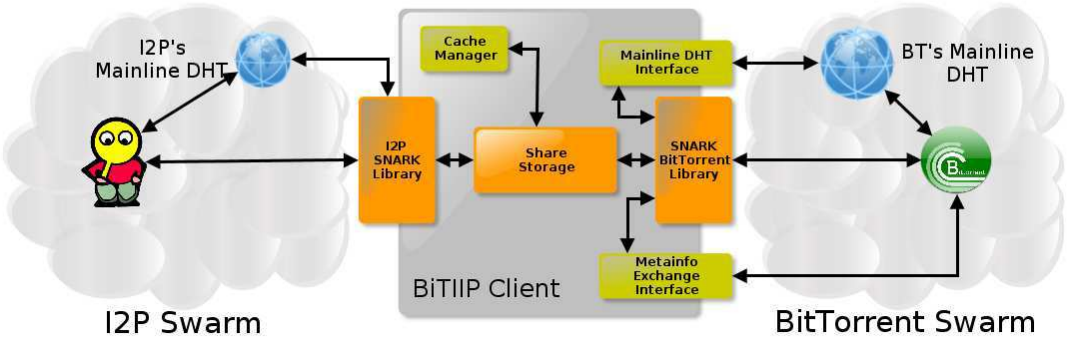


Figure 6.6 – Functional components of a BiTIIP client

This synchronisation requires time and affects the achieved download rates of the interconnection. This section analyses the performance achieved by our hybrid model when downloading public content from the I2P network. We conducted different experiments considering two configurations: a single BiTIIP client per swarm and multiple BiTIIP clients per swarm.

6.4.1 Download performance of the I2P network

We need to assess the current download performance of the I2P's file-sharing environment first, for a later comparison with our hybrid model.

We chose the top twenty torrents from the Postman tracker (regarding swarm size). We measured the download rates achieved, using the configuration depicted in Figure 6.5. We considered the overall swarm speed achieved, as well as the fastest peer in the swarm.

Figure 6.7 shows the results for the current performance of the I2P's file-sharing environment. The fastest peer in every swarm had an average rate of 9 KB/s, while the overall download speed of the swarms presented an average rate of 33 KB/s.

We observed that I2P's file-sharing environment did not exhibit a good performance. The *tunnel-based* communications of I2P penalise the final throughput. However, I2P's design goal is not high throughput but anonymous communications. Therefore, that is the tradeoff we need to accept for an anonymous download.

6.4.2 A single BiTIIP client

Once we assessed the performance of I2P, we evaluated our BiTIIP client. We took into account the configuration shown in Figure 6.8(a). A single I2P user connected to a BiTIIP client to download a random ten MB file hosted by a single BitTorrent client. This simple configuration

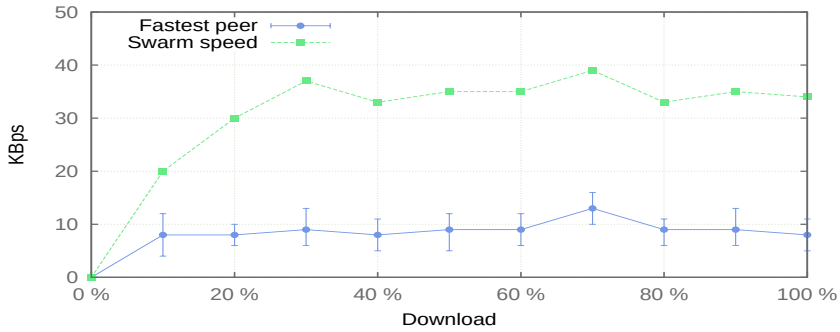
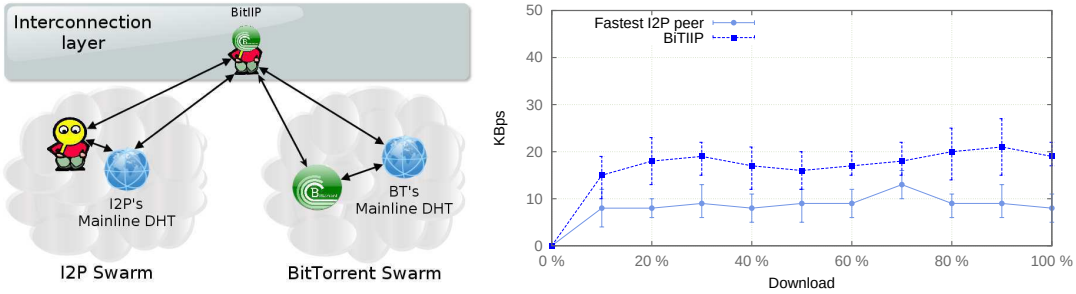


Figure 6.7 – Download performance of the I2P's file-sharing environment

enabled us to measure how fast our BiTIIP client performed as a connection point between these two networks.



(a) Configuration of the experiment

(b) Download performance of a single BiTIIP

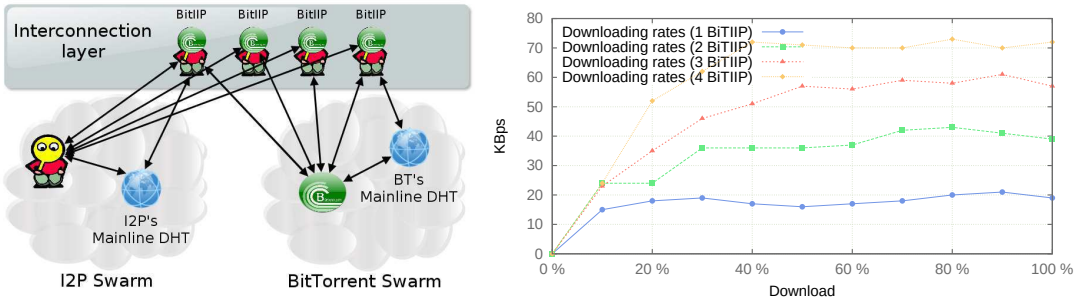
Figure 6.8 – Download performance of a single BiTIIP client

We performed the same download a of twenty-five times and obtained an average rate of 17 KB/s throughout our BiTIIP client. Figure 6.8(b) depicts our result and compares BiTIIP's performance against the fastest peer in I2P's swarms. As mentioned in Section 6.3.3, BiTIIP clients use a *0*-hop tunnel configuration, which reduces their anonymity in the I2P network, but considerable increases their performance and hence they achieve better rates than normal I2P users.

6.4.3 Multiples BiTIIP clients

A single BiTIIP client doubles the download performance of the fastest I2P peer detected. By including further BiTIIP clients within an swarm, we can increase its performance. We conducted a second experiment, where we added extra BiTIIP clients expecting to proportionally increase the performance. Figure 6.9(a) depicts the configuration of the experiment, where we used two, three and four clients. For ease of visualisation, only one BiTIIP client accessed both DHTs in the figure. However, every BiTIIP client had access to the I2P's Mainline DHT and the BitTorrent's Mainline DHT.

As expected, each BiTIIP client proportionally increased the throughput of the download as shown in Figure 6.9(b). Two BiTIIP clients provided around 37 KB/s, similar to the performance of the I2P's swarms.



(a) Configuration of the experiment (b) Download performance of multiple BiTIIP clients

Figure 6.9 – Download performance of multiple BiTIIP clients

We propose to increase the overall interconnection performance by adding, when possible, further BiTIIP clients to an existing interconnection with the only objective of increasing the overall performance.

It is important to consider the scalability of our design. As mentioned in Subsection 6.3.1, we aim at interconnecting an I2P swarm with a fraction of a BitTorrent swarm. Therefore, the required number of BiTIIP clients is proportional to the number of I2P users within an I2P swarm. Based on the number of I2P users in top I2P swarms, presented as well in Subsection 6.3.1 and the estimated capacity of a single BiTIIP client, we are able to interconnect an I2P swarm with any BitTorrent swarm with a single BiTIIP client.

6.4.4 The *connectME* project

The *connectME* project[52] encourages Internet users to contribute to the improvement of anonymous environments, with a special focus on anonymous file-sharing environments. The BiTIIP client can be deployed so as to increase our current set of available BiTIIP clients. As more BiTIIP clients become available, more content can be introduced in the I2P's file-sharing environment. The project's website brings together our implementation of the BiTIIP client, distributable under a GPL license. The BiTIIP client was developed in collaboration with Tarang Chugh [113].

Table 6.3 positions our hybrid model against the interconnection models previously presented. Like the hMule model, our hybrid model considers widely deployed networks and their real topologies. This model enables an *on-demand* mechanism, where I2P users can request BitTorrent content, where BiTIIP clients are deployed to meet this demand. There is no modification from the side of the I2P users, which enables a fully backward-compatible model with previous I2P clients. Our hybrid model is the first model to consider an interaction between anonymous and non-anonymous environments, where anonymous users can access public content without compromising their anonymity.

6.5 Conclusion

This chapter introduced a hybrid model for file-sharing aiming at improving the content availability in the I2P's file-sharing environment. We showed that barely 1% of all available BitTorrent content was available in the I2P network, containing mostly out-dated files. According

52. Accessible at `http://connectme.gforge.inria.fr/`.

Use	General use	Peer-to-peer file-sharing				
Model	Synapse	Lloret's scheme	Konish's scheme	Fu's extension	hMule	**BiTIIP**
Interaction component	Synapses (Bridges)	Super nodes	Cooperative peers	Shared Peers	Hybrid nodes	**Hybrid nodes**
Organisation	*ad-hoc* overlay	2-layers / *ad-hoc* overlay	*ad-hoc* overlay	*ad-hoc* overlay	*ad-hoc* overlay	***ad-hoc overlay***
Deployment	Sim. (With Churn)	Sim. (No Churn)	Sim. (No Churn)	Sim. (No Churn)	Real networks	**Real networks**
Topology	Dynamic	Random	Barabasi-Albert (1000 nodes)	Kazaa topology	BT/Ed2k topology	**BT topology**
Multi protocol	Yes	Undefined	No	Yes, but untested	Not applicable	**Yes**
Start of interconnections	Network decision	Network decision	Network decision	Local decision	On-demand	**On-demand**
Anonymous interconnection	No	No	No	No	No	**Yes**

Table 6.3 – The BiTIIP hybrid model compared with existing interconnection models

to that lack of content, we designed an interconnection model enabling I2P users to access public content.

Our hybrid model allows I2P users to access BitTorrent content while preserving their anonymity. This leads to a scheme, where an anonymous content indexation and distribution mechanisms are possible. We evaluated our model through a hybrid client called **BiTIIP**, which outperforms the fastest I2P user and preserves I2P's users anonymity. BiTIIP acts as the meeting point between I2P users and BitTorrent users, enabling a bidirectional connection. By means of I2P's tunnel-based communications, an I2P user can anonymously contact a BiTIIP client and access BitTorrent content without jeopardising its anonymity.

We considered the I2P anonymous network for our interconnection model according to its characteristics. The next chapter presents our characterisation of this network considering, among others, the number of users, the number of applications and the type of these applications.

Part III

Characterisation of Anonymous Environments

Chapter 7

Characterisation of the I2P network

Contents

7.1 Introduction

Anonymous communications have been acquiring more and more interest since the past decade, either for fighting against any type of censorship, passive attacks (third-parties sniffing, traffic analysis, user profiling) or for malicious purposes (copyrighted material downloads). Anonymous file-sharing is one of the most active fields in anonymous communications and is increasingly growing. This growth is partially due to the onrush of negative news on public file-sharing communities, including legal actions by governmental institutions, law-enforcement agencies to major file-sharing communities. Another reason is the rising concerns of privacy and anonymity in the Internet.

A large-scale monitoring analysis on a file-sharing community provides an overview of the network [114, 115, 116], enabling us to know which kind of content is distributed in the anonymous environment, the amount of users performing anonymous file-sharing, as well as determining which users are downloading a specific content. However, a large-scale monitoring on anonymous environments is very challenging and has not been extensively investigated. Most of the studies are essentially focused on the Tor network [99, 117].

This chapter introduces the first monitoring architecture for the anonymous I2P environment, providing deep insights about the utilisation of the network and its users. We demonstrate

the ability to detect most of I2P anonymous applications, including anonymous file-sharing applications, and determine their use on the network through time, as well as to geolocate I2P users. Firstly, we introduce I2P's DHT, called the *netDB*, which is the central component in our architecture and we explain how it can be exploited. Secondly, we detail the entire monitoring architecture and its components. Finally, we evaluate our monitoring architecture through a six-day experiment.

7.2 Exploiting the I2P network

In this section, we first complete the characteristics of the *netDB*, previously presented in Subsection 4.2.3. Then, we detail how the netDB is exploited to allow us to monitor the I2P network.

7.2.1 The netDB

The netDB is composed of *floodfill* nodes. These floodfill nodes are normal I2P nodes which have locally decide to become floodfill nodes. An I2P node evaluates the current number of floodfill nodes in the network, its available bandwidth rates and its *health*[53]. If the number of floodfill nodes is less than five-hundred peers[54], the current I2P user has more than 256 KB/s available and it is well-integrated in the network, it automatically becomes a floodfill node.

The distributed nature of the netDB and the local mechanism to decide whether to become a floodfill node hardens the I2P network. These characteristics avoid a single point of failure, since there is no central component to attack.

NetDB storing process

The netDB uses an *iterative lookup* procedure to publish and retrieve values, based on the Kademlia XOR metric. This metric determines the distance between two identifiers. On the I2P network, the metric is applied between a floodfill's *routing identifier* and the *value identifier* (either a routerinfo or a leaseset). Whenever an I2P user wants to publish a value in the netDB, it performs an iterative lookup as detailed in Subsection 2.3.4. It iteratively queries remote floodfill nodes, starting firstly by its known floodfill nodes, in order to find the closest floodfill nodes for the value to be published.

Figure 7.1 depicts an example, where an I2P user wants to store a routerinfo with a hash value of `13`. It starts by querying its only known floodfill, node `2`. Node `2` responds with node `7`. A second request is sent to node `7`, which responds with node `12`. A final request is sent to node `12`, which responds with itself as the closest node to the value `13`. A final store request is then sent to node `12`, which will store the value. In this example, the I2P user stores the routerinfo in a single floodfill node. However, in the current netDB implementation an I2P user stores a value in the closest five floodfill nodes. This parameter is known as the *replica set* and is used to improve fault tolerance against floodfill nodes going off-line.

53. A router's health indicates how well the router is integrated in the network.

54. The I2P design specifies that around 6% of the network should form the netDB, where with the current estimation of 24000 I2P users, there should be nearly fifteen-hundred floodfill nodes. However, a minimum value of five-hundred nodes is hard-coded and therefore the real percentage decreases to nearly 2%.

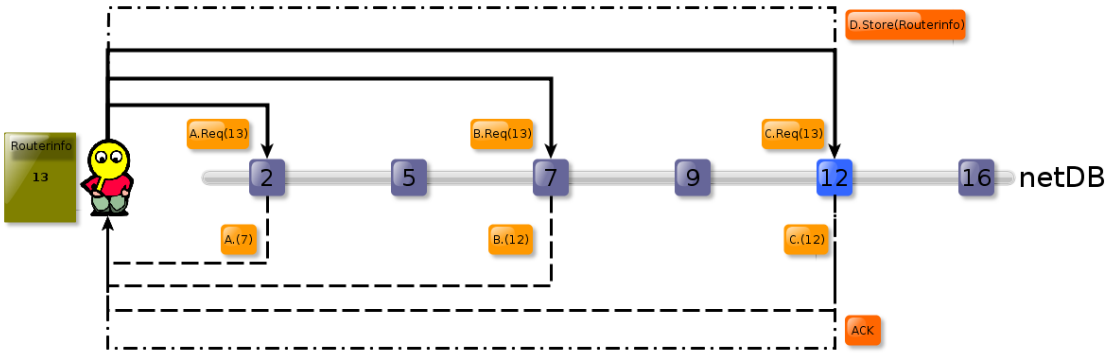


Figure 7.1 – NetDB's *iterative lookup*

Keyspace shifting

As every Kademlia-based DHT, I2P's netDB uses the XOR metric to determine in which nodes a value should be published in or retrieved from by comparing a node's identifier and a value's identifier. A node's identifier is determined during the first execution and normally remains unchanged throughout the entire life of the node.

Nevertheless, the netDB uses a *temporary identifier* instead of a fix identifier to compute the XOR distance. This temporary identifier, called a *routing identifier* as opposed to the *node identifier*, is obtained by appending the node identifier with the current date and hashing the result, as shown in 7.1. Therefore, the identifier used in the netDB is the routing identifier, which changes every day, while the node identifier remains fixed.

$$routing\ identifier = SHA256(node\_identifier||yyyyMMdd) \tag{7.1}$$

At midnight, every previously published value needs to be republished in another DHT location, since the routing identifier changes, as shown in Figure 7.2. I2P uses this approach to increase the cost of a localised Sybil attack in the netDB, since attackers are placed close to a target identifier in the DHT by forging their identifiers, aiming at receiving every publish and search request for the targeted identifier. However, as the DHT keyspace changes every day, Sybils' identifiers need to be daily recomputed, increasing the cost of the attack.

The major drawback of this approach is when the DHT's keyspace is shifting, most of the publish and search requests fail, until every value is accordingly republished.

7.2.2 Exploiting the netDB

We exploit the netDB by taking advantage of its mechanism to become a floodfill node. The netDB uses a local mechanism without any central authority nor a general consensus among all floodfill nodes. Any I2P user can volunteer as a floodfill node. This strategy allows us to deploy a set of *monitoring floodfill nodes* in the netDB.

We conduct a *passive monitoring*, as described in Subsection 2.4.1, where our monitoring floodfill nodes behave as *distributed probes*, targeting different zones in the netDB. These monitoring floodfill nodes act as normal floodfill nodes, forming the netDB along with other floodfill nodes and storing I2P's metadata.

We aim at gathering as much network metadata as possible. Whenever we receive a publish request, either for a routerinfo or for a leaseset, we process it as a regular floodfill node, and

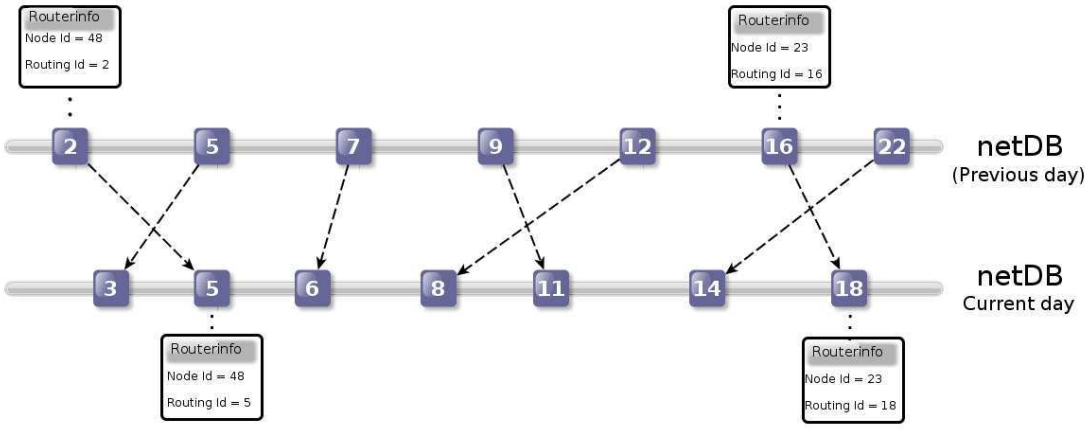
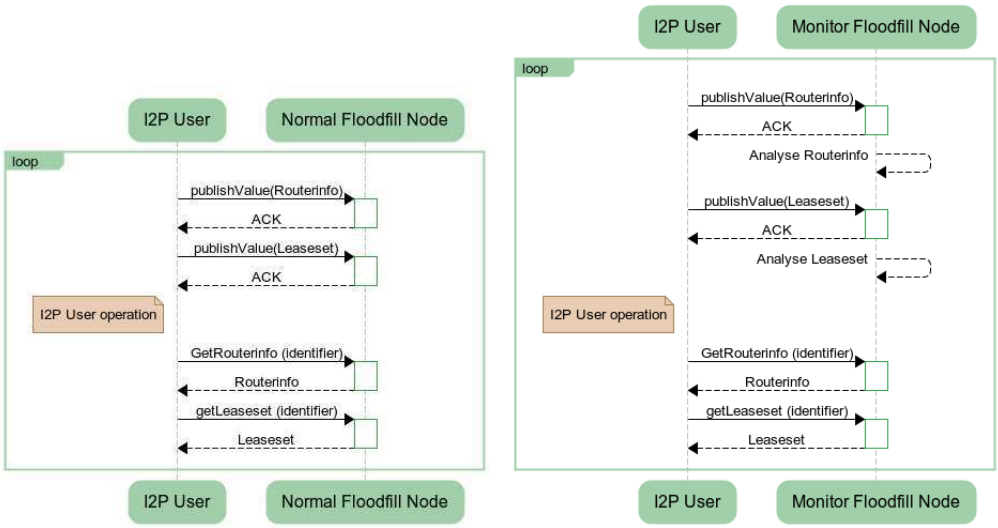


Figure 7.2 – NetDB's *daily shifting*

store it for a latter analysis. The sequence diagrams 7.3 show the difference between a normal non-monitoring floodfill node and our modified monitoring floodfill node. From an I2P user's point of view, there is no notable operational difference which make our monitoring floodfill nodes passive monitoring nodes.



(a) Interaction of a normal floodfill node (b) Interaction of a monitoring floodfill node

Figure 7.3 – Interaction of normal and monitoring floodfill nodes

By placing different monitoring floodfill nodes in the netDB, we are able to gather a vast amount of network metadata, analyse it and characterise the I2P network. Routerinfos are used to geolocalise I2P users and determine their online behaviour in the I2P network. Leasesets are used to determine which anonymous applications are deployed, as well as their uptime in the network.

7.2.3 Distribution of the monitoring floodfill nodes

A monitoring approach using distributed probes works better if the network to be monitored enables free placements of nodes, which is not the case of the netDB. The netDB uses a *node identifier* and a *routing identifier*. In the normal Kademlia design, the node and routing identifiers are identical, enabling a node to easily choose its position in the DHT. On the contrary, the netDB recalculates its routing identifier every day. Therefore, in order to position a node in the DHT, we need to compute a hash dictionary, retrieving a node identifier given its actual routing identifier. This is known as a *pre-image attack* [118] and it is practically infeasible with the SHA2 hashing function, which is netDB's hashing function.

As we are not able to place a monitoring floodfill node in a specific netDB position, a localised attack in the network would require an enormous computational cost. However, to retrieve as much I2P's metadata as possible, the monitoring floodfill nodes are required to be well distributed in the keyspace, *i.e.* not being grouped all together in a particular part of the keyspace.

As a cryptographic function, the SHA2 function presents an uniform distribution of all generated hash codes and therefore the current set of floodfill nodes' routing identifiers should theoretically present a uniform distribution. In order to evaluate the real distribution of the floodfill nodes, we monitored all floodfill nodes (which accounted for five-hundred nodes during our measurements) during one month and recorded their routing identifiers. At every new shifting of the keyspace we determined the new distribution of the routing identifiers and computed a daily average.

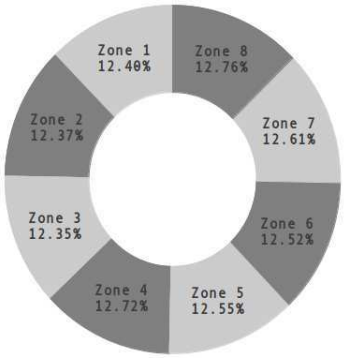


Figure 7.4 – Distribution of the floodfill node's routing identifiers in the netDB

Figure 7.4 depicts the results of our experiment, where we reduced the entire netDB keyspace to eight equal zones for ease of visualisation. We obtained a uniform distribution of the floodfills nodes. We were also able to detect around forty floodfill nodes that did not respect the shifting mechanism and stayed fixed during our measurements. Those floodfill nodes were deployed by a few IP addresses, which does not correspond to the normal operation of the network.

7.3 Monitoring architecture

This section describes our monitoring architecture and its components. It includes the analysis of routerinfos and leasesets, as well as an analysis of the required number of monitoring floodfill nodes to monitor the entire I2P network.

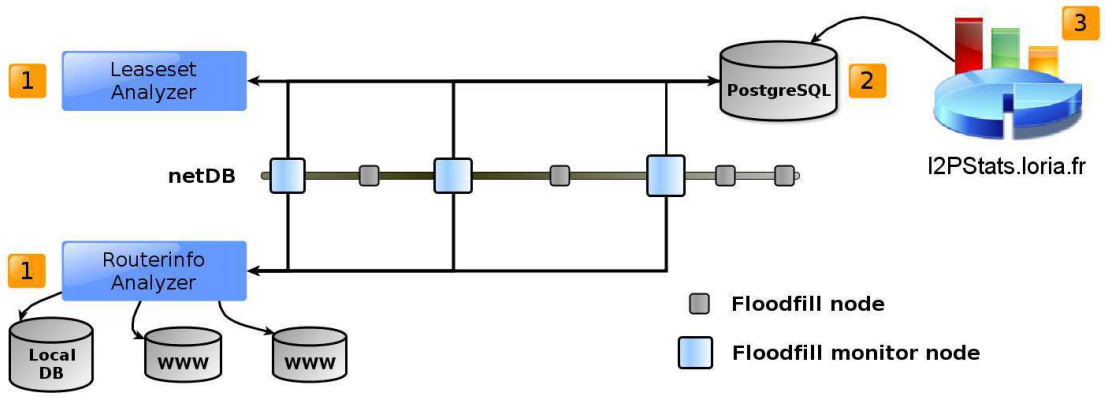


Figure 7.5 – A passive distributed monitoring architecture for the I2P network

7.3.1 Monitoring architecture overview

We have a passive distributed monitoring architecture, where monitoring nodes are distributed on the network and behave as normal nodes. Figure 7.5 depicts our architecture.

A set of monitoring floodfill nodes are placed in the netDB to collect I2P's routerinfos and leasesets. Once this metadata has been analysed, the results of the analyses are stored in a central database. Later, they are aggregated and displayed in a statistical website.

The deployment of monitoring floodfill nodes is completely flexible. These nodes can be dynamically added to the netDB, thus increasing the amount of network metadata retrieved and analysed. The next sections introduce both routerinfo and leaseset analysers and the required number of monitoring floodfill nodes to have a complete network coverage.

7.3.2 Analysis of Routerinfos and Leasesets

A routerinfo identifies an I2P user on the network by specifying its contact details, as shown in Figure 7.6. It includes a 2048-bit ElGamal encryption key, a 1024-bit DSA signing key, as well as a certificate, which is all defined as the *router identity*. The routerinfo includes a *contact address*, such as `contact.com:4567`, a set of *text options*, such as the capability flags of the router, and finally, a *signature* of the entire routerinfo. The ElGamal encryption key is used when contacting a router, while the DSA key is used to verify routerinfo integrity.

A leaseset provides the contact details for a client's anonymous I2P application, as shown in Figure 7.6. A leaseset includes a 2048-bit ElGamal encryption key, a 1024-bit DSA signing key, as well as a certificate, which is all defined as the *destination identity*. The leaseset also includes an *encryption public key* for end-to-end encryption, a *signature* of the entire leaseset and a set of *leases*, which are the real entry points for the anonymous application. Each entry point specifies a *gateway*, a *tunnel ID*, which is represented as a four-bytes number, and an *expiration date* for the lease.

These structures are processed and analysed by every monitor floodfill node as shown below.

Routerinfo analysis

A routerinfo is analysed in order to know the geographical localisation of the I2P user it represents. A monitoring floodfill node uses different services to accurately determine the ge-

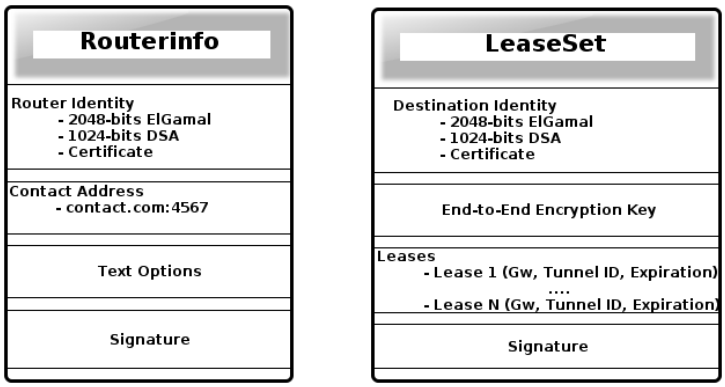


Figure 7.6 – I2P's metadata, a Routerinfo and a Leaseset

ographical localisation of a particular IP address, as shown in the Sequence Diagram 7.7. It employs a local database based on the MaxMind services [55] to determine the country, the region and the city of the given IP address. In the case that some fields are not retrieved, such as the city, the monitoring floodfill node uses two external web services, *Geo IP Tool* [56] and *Who is this IP ?* [57] to complete the geographical data.

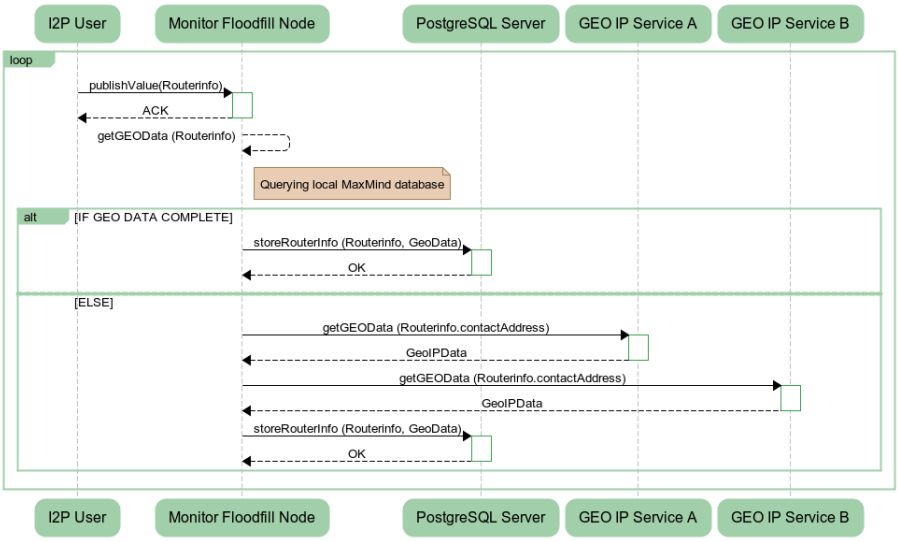


Figure 7.7 – Determining the geographical localisation of an I2P user through its routerinfo

Leaseset analysis

Analysing a leaseset requires a far more complex procedure when compared to analysing a routerinfo. Our goal is to determine which application is represented by a given leaseset, such as

55. Available at `http://www.maxmind.com/` under a *Creative Commons Attribution-ShareAlike 3.0 Unported License*. Last visited on 08/2013.
56. `http://www.geoiptool.com/`. Last visited on 08/2013.
57. `http://www.whoisthisip.com/`. Last visited on 08/2013.

a particular anonymous file-sharing client or an anonymous web server.

A destination is I2P's representation for an IP address and a port number, and therefore we can open a TCP-like socket through a destination or send UDP-like messages to it. The Sequence Diagram 7.8 depicts the five-step procedure performed by a monitoring floodfill node in order to *tag* a destination with a particular anonymous application.

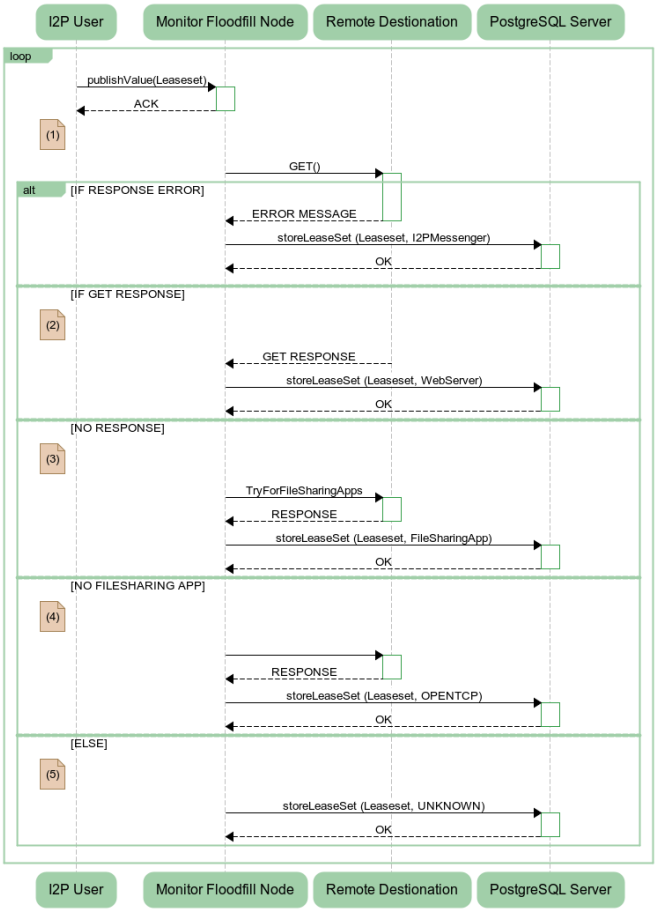


Figure 7.8 – Overall procedure for testing a leaseset

A monitoring floodfill nodes starts by checking whether the destination corresponds to the *I2PMessenger* anonymous messenger application by sending an `HTTP GET` message through a TCP-like socket. The default behaviour of this application when receiving an HTTP message is to respond with an error message indicating that the destination is not running an anonymous web server, but an anonymous I2P messenger application (1). If the destination is in fact running an anonymous web server, it responds to the GET message accordingly and it can be tagged an eepsite (2).

When there is no answer to the HTTP GET message, a monitoring floodfill node checks for anonymous file-sharing applications, including *I2PSnark* clients, *IMule* clients and *I2Phex* clients (3).

The iMule client operates a TCP-based destination and a UDP-based destination. Therefore, a TCP-like `HELLO` request and a UDP-like `Kademlia HELLO` request are sent.

If no response is received, a monitoring floodfill node checks for a I2Phex client by sending a

`GNUTELLA CONNECT` message through a TCP-like socket.

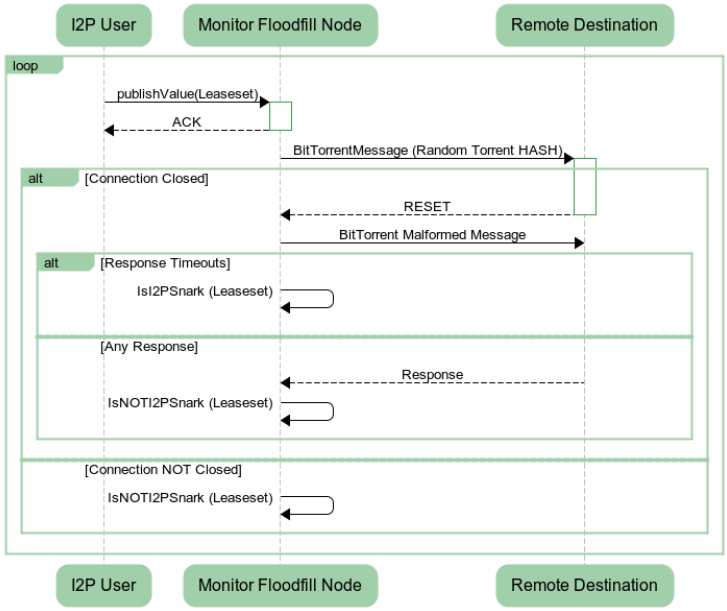


Figure 7.9 – Tagging a destination as an anonymous I2PSnark client

A two-step analysis is required when testing for an I2PSnark client, as illustrated in the Sequence Diagram 7.9. First, a well-formed BitTorrent message with a random torrent hash is sent. If the destination is actually running an I2PSnark client, it will immediately close the connection when receiving a well-formed BitTorrent message for a torrent that the client is not sharing. Then, if the connection is immediately closed, a second malformed BitTorrent message is sent. If the destination is actually running an I2PSnark client, it will not respond and therefore the response will timeout. This two-step analysis enables a monitoring floodfill node to determine whether a destination is running an I2PSnark client.

Finally, if all of the above analyses are unsuccessful, a monitoring floodfill node checks whether the destination corresponds to a TCP-like application by attempting to open a TCP-like socket (4). Otherwise, the destination is tagged as unknown and the analysis concludes (5).

The I2P network is optimised for anonymous web sites called *eepsites* and file-sharing applications are widely used in an anonymous environment. For these reasons, eepsites and file-sharing clients will be our target applications.

7.3.3 Deployment of monitoring floodfill nodes

It is important to consider the network coverage of our monitoring architecture, which depends on the current number of monitoring floodfill nodes deployed in the netDB and the *replica factor*. The latter indicates in how many nodes a value will be stored in.

Equation 7.2 denotes the minimum number of monitoring floodfill nodes to achieve a full coverage, with N as the total number of floodfill nodes and X as the current replica set. For the current netDB with five-hundred floodfill nodes and a replica set of five nodes, one-hundred monitoring floodfill nodes are required to have a complete network coverage. If we consider a total of five-hundred floodfill nodes and a replica set of five nodes, we can obtain a network coverage of 70% with seventy monitoring floodfill nodes, for instance.

$$\texttt{nb\_monitors} = \lceil\ \texttt{N / X}\ \rceil,\ \texttt{N= \#floodfills , X= replica factor} \tag{7.2}$$

The calculated number of monitor floodfill nodes is adequate when considering a perfect distribution in the netDB as shown in Subsection 7.2.3.

For our monitoring architecture we use the Planetlab testbed and due to technical reasons we are able to deploy seventy monitoring floodfill nodes. Our goal is to determine whether our monitoring floodfill nodes are well-distributed in the netDB and therefore we reach the intended network coverage of 70%. We conducted the same distribution experiment as in Subsection 7.2.3 only with our monitoring floodfill nodes.

During a month and at every keyspace shifting, we recorded the seventy routing identifiers. Table 7.1 shows the distribution of our monitoring floodfill nodes, where we divided the entire netDB's keyspace in eight zones for ease of visualisation. We can observe a good distribution, except for zone four that had less monitors during the thirty days.

Keyspace zone	% of monitors (daily average)
Zone 1	11.6279%
Zone 2	13.1782%
Zone 3	14.7286%
Zone 4	8.5271%
Zone 5	14.7286%
Zone 6	12.4031%
Zone 7	13.1782%
Zone 8	11.6279%

Table 7.1 – Distribution of monitoring floodfill nodes

According to these results, our seventy monitoring floodfill nodes cover approximately 70% of the network.

7.4 A real time view of the I2P network

This section presents the results obtained with our monitoring architecture. We first analyse the current geographical distribution of I2P users in the network, including country-based and city-based distributions. Then, we present all detected anonymous I2P applications, with a special focus on anonymous file-sharing clients and anonymous web servers. Finally, we describe an online statistics portal, which aggregates our results.

We deploy our monitoring architecture for a six-day period, from `2013-06-18 12:00:00 CEST` to `2013-06-24 12:00:00 CEST`, with a total of seventy monitoring floodfill nodes deployed on the PlanetLab testbed. During our measurement, the netDB had around five-hundred floodfill nodes. According to Equation 7.2, with covered 70% of the network with our seventy monitoring floodfill nodes.

7.4.1 I2P users

We introduce the overall number of detected I2P users throughout the monitoring measurement and classify them according to their geographical properties.

Detected number of I2P users

Figure 7.10 depicts the number of I2P users detected throughout the six-day monitoring, where we observe an average of nearly 28000 daily I2P users. We also observe different daily *high peaks* at `18h00`, indicating that I2P users were more active during the European afternoon, with 31000 users in average. We also notice daily *low peaks* at midnight, where the number of detected I2P users dropped to nearly 24000 I2P users. This was due to the I2P netDB's shifting mechanism, introduced in Subsection 7.2.1. All floodfill nodes, including our monitoring floodfill nodes, changed their position in the netDB and therefore started to receive new metadata, while previously stored metadata was no longer valid. This situation affected the operation of the network for a short period of time and therefore it affected our monitoring results.

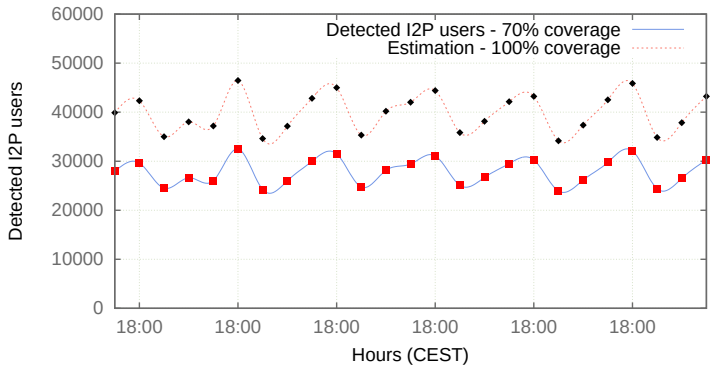


Figure 7.10 – Number of I2P users detected

I2P users' behaviour presents a similar pattern to peer-to-peer networks [119], depicting a sinusoidal pattern with a period of twenty-four hours and a maximum at `18h00` CEST. According to our 70% of network coverage, we estimated a daily average value of nearly 40000 I2P users, as depicted in Figure 7.10.

Our results present the same behaviour as the values published in the I2P's official statistics web site [58], where I2P users present a sinusoidal pattern. However, we detected a bigger number of I2P users, putting into evidence that the official statistics web site does not consider the entire network and its approximations do not fully represent the I2P network.

Country-based characterisation

We detected a total of 113433 widely distributed I2P users. Table 7.2 summarises the top ten countries detected, where Russia accounts for nearly the 40% of all users.

We observe that the I2P network had a considerable participation from Russian users. We detected a total of 159 countries, indicating that the I2P network is widely deployed.

City-based characterisation

Table 7.3 depicts the top fifteen cities out of the 13547 total cities detected. We observed that twelve out of the top fifteen cities are Russian cities. Moreover, we detected a total of 813 Russian cities out of the 1108 total Russian cities [120], which indicates that the I2P network is used in nearly the 75% of all Russia.

58. `http://stats.i2p.in/`

Country	Number of detected users	Overall percentage
Russian Federation	45299	39.93%
United States	9968	8.78%
Germany	8563	7.54%
Ukraine	5491	4.84%
Brazil	4940	4.35%
France	4073	3.59%
United Kingdom	3688	2.96%
Italy	3154	2.78%
Belarus	2370	2.08%
India	1859	1.63%

Table 7.2 – Top ten countries detected in the I2P network

City	Number of detected users	Overall percentage
Moscow	6542	5.76%
Saint Petersburg	2351	2.07%
Minsk	1191	1.04%
Krasnodar	1088	0.95%
Perm	1022	0.90%
Kiev	1020	0.89%
Chelyabinsk	984	0.86%
Omsk	920	0.81%
Nizhniy Novgorod	797	0.70%
Kazan	756	0.66%
Berlin	721	0.63%
Ufa	720	0.63%
Samara	668	0.60%
Yekaterinburg	623	0.54%
Voronezh	622	0.54%

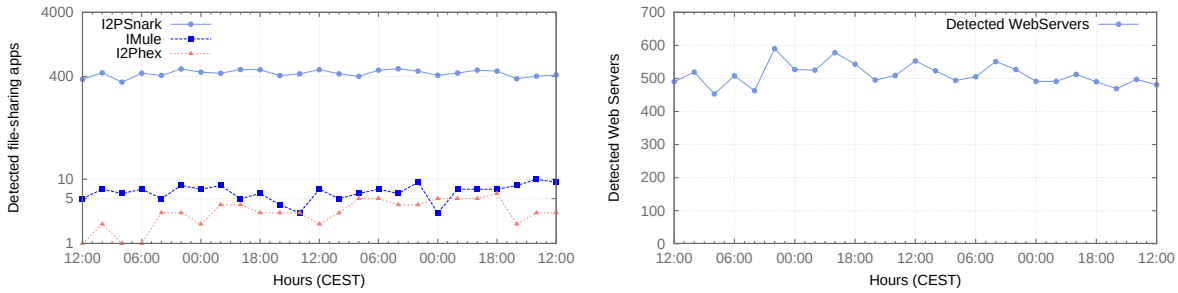
Table 7.3 – Top fifteen cities detected in the I2P network

7.4.2 I2P anonymous applications

As previously introduced, we are able to analyse different I2P anonymous applications, including different I2P anonymous file-sharing clients and anonymous web servers, hosting anonymous web sites or *eepsites*.

Anonymous file-sharing clients

Figure 7.11(a) depicts the number of detected I2P file-sharing clients, considering the three current available clients. We observe that I2PSnark is the most used application, with an average of 450 clients. IMule clients barely present any activity, with as low as 6 clients detected in average. Finally, I2Phex has an average value of 3 clients, with a slight increase of clients during the weekend (right part of the chart).



(a) Anonymous file-sharing clients detected.

(b) Anonymous web servers detected.

Figure 7.11 – Anonymous I2P applications detected.

The I2PSnark client is a built-in file-sharing client and therefore works out of the box, facilitating its use. On the contrary, both remaining anonymous file-sharing clients, iMule and I2Phex, are modifications to the regular file-sharing client and therefore their deployment in the I2P network is more complex, partially discouraging their use.

Anonymous web servers

Figure 7.11(b) shows the number of anonymous web servers detected, with an average of 510 web servers. We see that even anonymous web servers, which are intended to operate in a more stable way than file-sharing applications, present a subtle sinusoidal pattern as well. This indicates that I2P users hosting their anonymous web servers were still connecting and disconnecting from the I2P network as the rest of the users.

Contrary to the statistics service `Tino`[59], which contacts those anonymous web servers listed in I2P's DNS-like service, we consider every published anonymous web server in the netDB and therefore we provide a precise real time view of the network, even for unlisted eepsites.

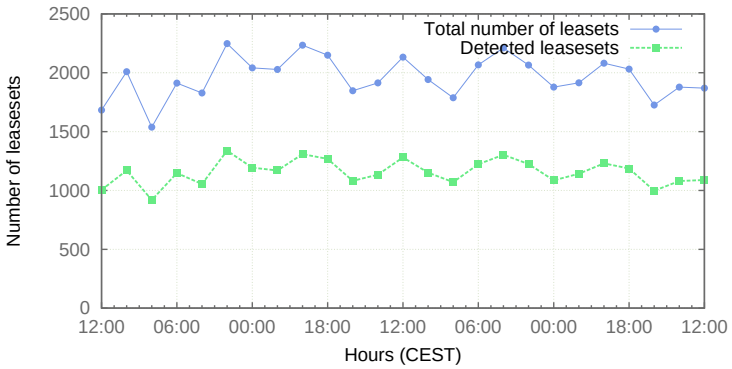


Figure 7.12 – Total number of leasesets detected

Figure 7.12 depicts the total number of leasesets detected during our measurement. We detected an average of 1960 leasesets, and we were able to *tag* 59% of them. It indicates that nearly two-thirds of the I2P network is performing anonymous file-sharing or hosting an anonymous web site.

59. Presented in Subsection 4.4.2 and accessible at `tino.i2p.in`. Last visited on 08/2013.

Estimation of the total network

As mentioned before, we cover nearly 70% of the entire network with our current monitor floodfill nodes. Figure 7.13 shows the total number of detected applications, both anonymous file-sharing clients and web servers, with an average value of 1154 applications. The figure additionally depicts our estimation of the total number of these two I2P applications for the six-day monitoring, with an average value of 1650 applications.

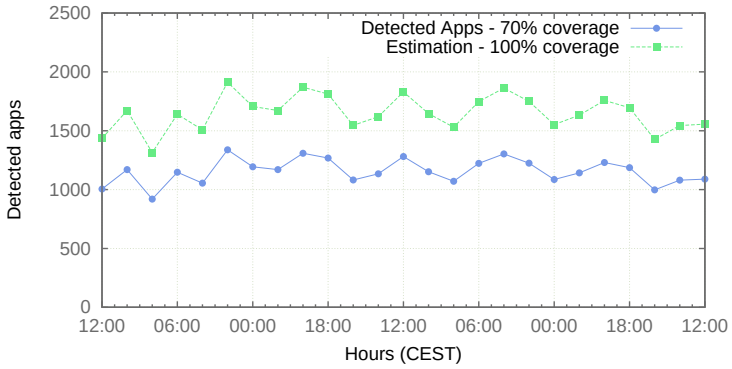


Figure 7.13 – Estimated number of anonymous file-sharing clients and web servers

I2PStats: The I2P statistics portal

All retrieved information is aggregated and displayed in our statistics web site[60], including a real time view of the network, as well as historical data. The estimation of the number of users, the I2P anonymous applications and the geographical distribution of I2P users is presented. This web site was developed in cooperation with Henry [121].

Our results reflected the normal operation of the network, which can be observed in our statistics web site.

7.5 Conclusion

We focused in the I2P anonymous environment. By exploiting I2P's distributed database, known as the *netDB*, we were able to deploy a set of passive distributed probes, thus collecting a vast amount of network metadata. By properly analysing this metadata, we determined different characteristics of the I2P users and the I2P anonymous applications.

We designed, implemented and successfully deployed the first large-scale monitoring architecture for the I2P network. We mainly considered anonymous file-sharing and anonymous eepsites, determining their real use in the network, leading to an accurately characterisation of the I2P anonymous environment.

We evaluated our monitoring architecture through a real-world analysis of the network and presented a six-day analysis of the system. We showed that Russia accounted for nearly the 40% of all I2P users detected, where 75% of the total number of Russian cities were identified. Additionally, the I2PSnark client presented an active behaviour, with an average value of 450 active clients. The number of detected anonymous web servers hosting eepsites was measured

60. Accessible at `http://i2pstats.loria.fr/`.

as well, with an average value of 510 servers. The statistics service `Tino` announced a total of 350 anonymous web sites and we were able to detect a total of 510 anonymous web sites. This indicates that there were several *unlisted* anonymous web sites in the I2P network.

Our monitoring architecture continuously operates, with an average of seventy distributed monitoring floodfill nodes deployed on the PlanetLab's testbed. According to our distribution analysis and the current amount of floodfill nodes in the netDB, this amount of monitoring floodfill nodes enables us to analyse approximately 70% of the entire I2P network. A real time analysis of the I2P network can be found in our statistics web site, including the current geographical distribution of I2P users and the use of different I2P anonymous applications.

The replica factor of I2P's netDB has changed based on our monitoring efforts, increasing the number of monitoring floodfill nodes required to have a complete coverage. The replica set was previously formed by a eight nodes, where a full coverage (considering the current number of five-hundred floodfill nodes) would have been achievable with sixty monitoring floodfill nodes instead of one-hundred monitoring nodes. By reducing the replica set in the netDB, I2P designers increased the number of monitoring floodfill nodes required to have a full coverage, but they also reduced netDB's fault-tolerance to churn.

The next chapter presents our approach to perform a *group-based characterisation* in the I2P network and to further improve the insights of the system.

Chapter 8

Group-based characterisation of the I2P network

Contents

8.1 Introduction

The previous chapter introduced a distributed monitoring architecture for the anonymous I2P environment, which allows us to characterise I2P users and I2P anonymous applications. Besides providing us with insights of the system, this application-level analysis certainly introduces an anonymity risk. By analysing the behaviour of a particular application and the behaviour of a set or users, we should be able to determine whether this group of users is responsible for a given application's activity on the system.

This chapter presents the first step towards group-based characterisation, where we target the entire I2P anonymous file-sharing network. We conduct a comprehensive correlation analysis based on data collected from our distributed monitoring architecture, considering users from the top detected cities, along with the I2PSnark application, which is the most used anonymous file-sharing application in the I2P network.

Pearson's coefficient is used to analyse this correlation and to determine whether these cities contribute, and in which measure, to I2P's file-sharing activity. Our goal is to show that despite a strong underlying anonymising layer, it is still possible to analyse users' activities in the network and establish whether their behaviour presents similar patterns with anonymous network applications, such as file-sharing applications.

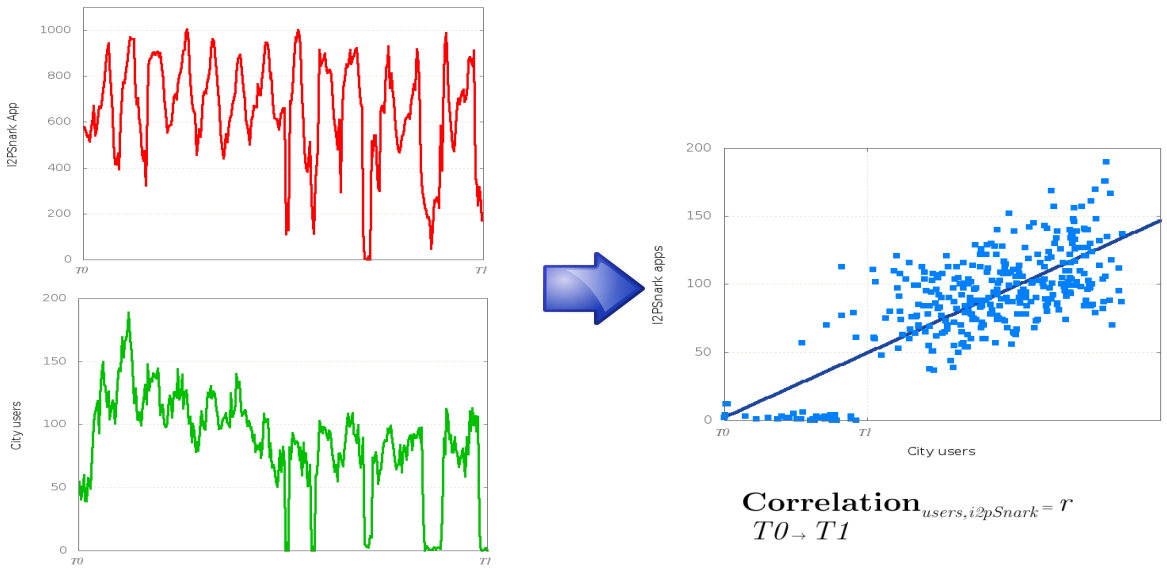


Figure 8.1 – Towards group-based characterisation in the I2P network

We first introduce our strategy for group-based characterisation and detail Pearson's correlation coefficient and its characteristics. Then, we present our experiments results through three different case studies. Finally, we address the complexity of a correlation analysis between a country and an application, as well as an analysis with further I2P applications.

8.2 Strategy for group-based characterisation

This section presents our strategy to perform a group-based characterisation in the I2P network and the correlation coefficient used for our analysis.

8.2.1 Strategy for characterisation

Figure 8.1 introduces, more precisely, our objective, considering data from the real network. We have the number of detected I2PSnark applications along with the number of detected users from Moscow for a period of fifteen days. Our goal is to establish in which measure this set of users contributes to the detected file-sharing activity for this period of time. A positive correlation between these two set of data would allow us to determine that these users were actually performing file-sharing on the network. Moreover, we can assert which part of the variance of file-sharing activity is explained from the variance of the number of users from Moscow.

We want to analyse two variables, I2P users and I2PSnark applications. With a *bivariate correlation analysis* we can determine if two variables present a *dependent relationship* and establish whether this dependency is positive or negative. We consider the number of I2P users from a particular city and the number of detected I2PSnark applications. This type of data is *ratio* data. Data of type *ratio* is the most descriptive type of data [122], where the intervals between every value are meaningful and a natural zero exists. Our data has a ratio type since the intervals are meaningful (the difference between twenty and ten users is the same as the difference between forty and thirty users) and a natural zero exists.

According to this type of data, we use *Pearson's correlation coefficient*, which gives us a measure of the *linear dependence* of two variables.

8.2.2 The Pearson's correlation coefficient

The Pearson's correlation coefficient value of two variables is defined as the covariance of the two variables divided by the product of their standard deviations, as presented in Equation 8.1.

$$\rho_{x,y} = \frac{\mathrm{Cov}(X,Y)}{\sigma X \sigma Y} \tag{8.1}$$

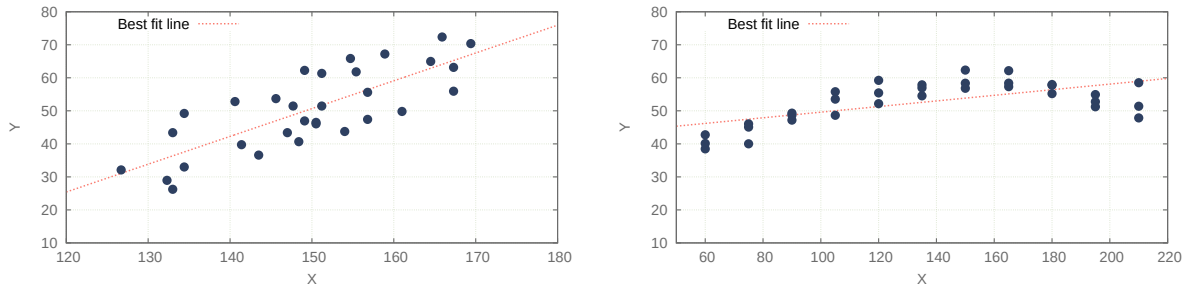
Equation 8.2 describes Pearson's correlation coefficient applied to a sample of the population, where the covariance and standard deviations are replaced for those of a particular sample of n elements.

$$r_{x,y} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{8.2}$$

Data requirements for Pearson's coefficient

The Pearson's coefficient is only applicable if the data under analysis fulfils four requirements:
- **Normal distribution:** data should follow an approximative normal distribution.
- **Type of data:** interval or ratio data is required.
- **Liner relationship**: the variables need to present a linear relationship.
- **Homoscedasticity:** data should present the same variance around the *line of best fit*, which means that it should be scatter in the same way around it.

Although data is not required to follow a perfect normal distribution, an approximative normal distribution is required so that the coefficient is applicable.

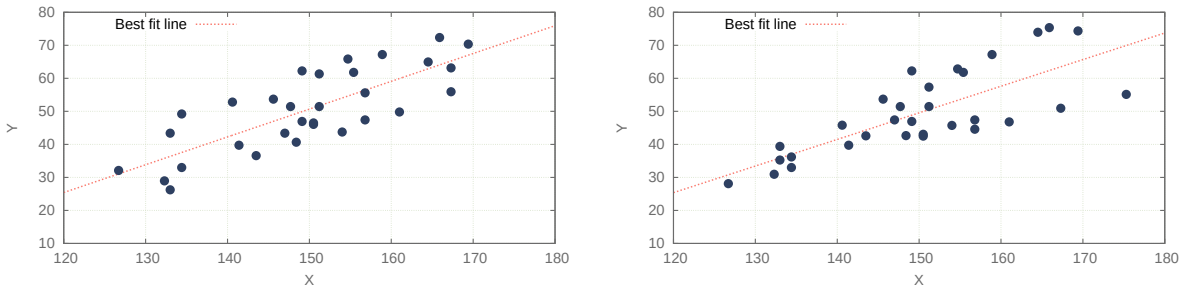


(a) Linear (b) Non-linear

Figure 8.2 – Linear and non-linear data

Non-linearity can be detected by plotting the data in a scatter plot and visually checking whether it follows a linear form or not, as presented in Figure 8.2. Non-linearity of data can lead to an erroneous prediction for values outside the data range.

Finally, non-homoscedasticity or heteroscedasticity occurs when the variance of the data along the line of best fit changes, which can result in a data distribution as shown in Figure 8.3. Non-homoscedasticity in the analysed data can bias the correlations towards the subset of data with the largest variance.



(a) Homoscedasticity (b) Heteroscedasticity

Figure 8.3 – *Homoscedasticity* and *Heteroscedasticity*

Interpretation of Pearson's coefficient

Pearson's coefficient provides an output between -1 and 1. According to Cohen [123], a correlation value above 0.50 (or less than -0.50) is considered strong, a value between 0.30 and 0.50 (or between -0.50 and -0.30) as moderate, and finally, a value between 0.30 and -0.30 as weak, like depicted in Figure 8.4. We only retain positive values in our analysis, since with more I2P users, more I2PSnark applications. We need to take into account an additional parameters to properly interpret our correlation analysis, the coefficient of determination, called r^2 (being r the Pearson's coefficient), and the significance of the correlation, called p.

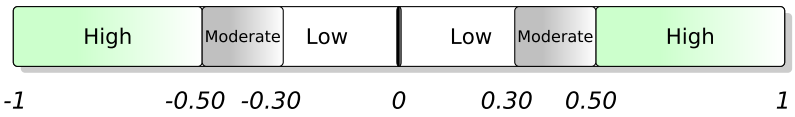


Figure 8.4 – Pearson's reference values

The coefficient of determination varies between zero and one and explains the variance of the data. It is extremely important in our analysis, since it determines in which measure the changes of a set of users are responsible for the changes of the number of detected I2PSnark applications. These changes correspond to the variance of the data and are what we consider the *activity* of users or file-sharing applications. Therefore, the coefficient of determination indicates us in which measure the changes in the number of users are responsible for the changes in the number of file-sharing applications, *i.e.* it explains the variance of the data.

The significance of the correlation, or p-value, gives the probability of obtaining a given correlation value for a non-representative sample, *i.e.* how true our correlation value is. The significance of the correlation is important when dealing with reduced sample sizes ($\sim$ 20% elements, for instance), where an incorrect selection of a sample (not-random, biased, etc.) can produce wrong correlation values.

Analysis of outliers

Outliers are extreme values in our dataset, which can have a considerable impact in our statistical measurements. Outliers can produce, for instance, non-normality in the data, violating the model's assumptions and producing deviant results, occasionally underestimating the outcome of our analysis, or what might be even more critical, overestimating our results. Ascombe

[124] classified outliers in two general groups, those coming from errors in the measurements and those coming from the inherent variability of the data itself. Therefore, it is highly important to properly analysed the data and determine whether these extreme values were indeed generated by errors in the measurements or they truly represent the data.

There are different ways to deal with outliers, where *robust statistical methods* are used. These methods are designed to resist outliers by either eliminating them from the analysis or rearranging them to comply with the model's assumptions, such as data normality. We consider a *trimmed estimator* and *Winsorising*, which are two of the best known robust methods to deal with outliers.

A trimmed estimator consists in ordering the data and removing a fixed percentage of it from both ends. If we apply a trimmed estimator at 10% to a list of one-hundred elements, we need to remove five elements from both ends of the ordered data. Winsorising applies the same mechanism as the trimmed estimator, although values are not removed but replaced with a specific percentile of the data. If we apply Winsorising at 20%, every value under the tenth percentile need to be set to the tenth percentile value, while every value above the nineteenth percentile need to be set to the nineteenth percentile value.

8.3 Experimental results and analyses

We use the Pearson's correlation coefficient in the I2P network in order to analyse the relationship between users from a particular city and the number of I2PSnark applications. We expect to detect a moderate-to-high correlation between the top cities and I2PSnark applications.

8.3.1 Experiment setup

A single-day correlation analysis does not represent the actual use of the network, since a day can present particularities and lead to biased samples and underrate or overrate correlation coefficient values. So, measurements were conducted over fifteen-day period, where we consider that three week-ends is a good time window to detect a long-lived correlation between a particular city and I2PSnark applications. Our correlation analysis uses data collected from the real I2P network through our monitoring architecture presented in the Chapter 7, for a period of fifteen days from *2013-03-15 00:00:00 CEST* to *2013-03-30 00:00:00 CEST*. Due to technical limitations we used only forty monitoring floodfill nodes during the period of analysis.

8.3.2 Methodology

For the following analysis, we first need to determine whether our data fulfil Pearson's requirements for a correlation analysis, which is data normality, data linearity and data homoscedasticity. Data normality is tested through the analysis of the frequency histogram (I2P users vs. I2PSnark applications) and how well it fits a normal distribution. A *line of best fit* is employed to determine whether the distribution of points follows a linear distribution. Homoscedasticity is corroborated by visually checking the plotted data. The type of data under analysis was previously determined as ratio data, which is adequate for a Pearson's analysis.

Our monitoring architecture does not cover the entire I2P network, and therefore we do not collect data for every available user and I2PSnark application. This produces a few situations where we detect a low number of either users or I2PSnark applications, leading to outliers.

We deal only with *extreme-low values* in our dataset. We might encounter a low number of I2PSnark applications due to a partial network coverage. Therefore, we modified the two robust

methods considered to analyse only the *low* part of the data, *i.e.* those values close to zero. A *Trimmed estimator at 10%* removes only every value under the tenth percentile of the ordered data. A *Winsorising at 10%* replaces only those values under the tenth percentile.

We considered the top cities in term of users detected in our period of analysis, as well as their daily activity during the entire fifteen-day measurement. We detected 16085 cities, where most of them presented less than ten users through out the entire fifteen-day period. Because of the active use of I2PSnark clients, those cities presenting a low number of users were not adequate for our correlation analysis. Table 8.1 depicts the most active cities.

City	Users	Overall percentage
Moscow	244223	8%
Saint Petersburg	106688	3.5%
Tokyo	29667	$\sim 1\%$
Yekaterinburg	28507	$\sim 1\%$
Kiev	26262	$\sim 1\%$
Novosibirsk	23090	$\sim 1\%$
Knoxville	17949	$< 1\%$
Krasnodar	17849	$< 1\%$
Nizhny Novgorod	17008	$< 1\%$
Paris	14837	$< 1\%$
Berlin	13471	$< 1\%$
Omsk	12814	$< 1\%$
Ufa	12056	$< 1\%$
Munich	5392	$< 1\%$

Table 8.1 – Most active cities detected for a fifteen-day period

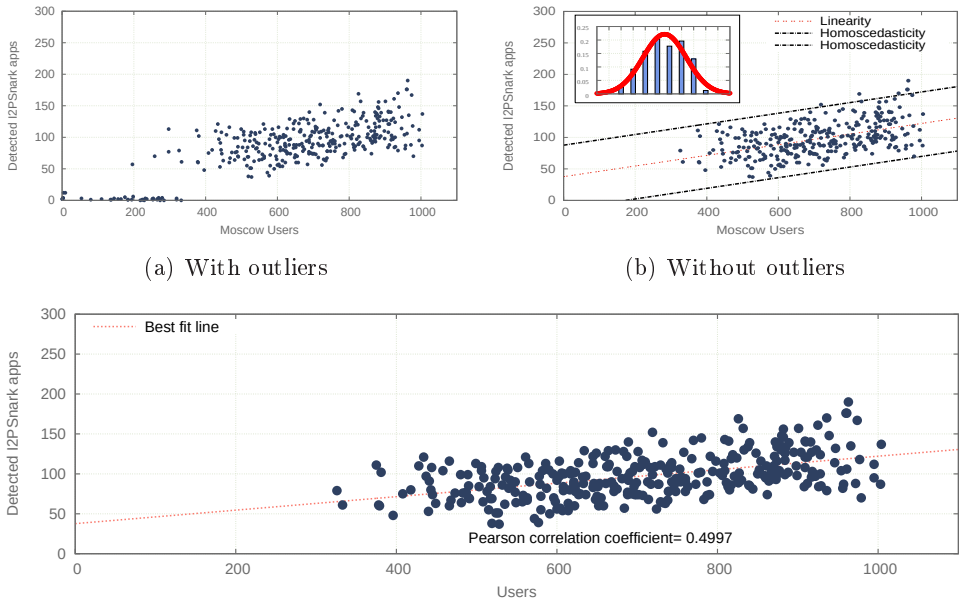
From all leasesets retrieved over the fifteen-day period, 31.7% were I2PSnark clients. According to the results presented in Subsection 7.4.2, we detected the same fraction of I2PSnark applications even for a smaller network coverage (forty monitoring floodfill nodes instead of seventy).

8.3.3 Case studies

Moscow and Saint Petersburg presented a high number of users detected, where the first one contributed to the 8% of the total users detected. We first considered those two cities for our correlation analysis as a case study, since they were the ones that contributed the most. Later on, we analysed another less significant city, and determined whether it represented a part of the file-sharing activity detected or not.

Correlation analysis for Moscow

Figure 8.5(a) plots Moscow's data with the number of I2P users in the x-axis and the number of I2PSnark applications detected in the y-axis. We observe a set of outliers in the bottom left corner of the chart. Figure 8.5(b) depicts Moscow's data after outliers' exclusion through our modified trimmed estimator, which is used for the correlation analysis.



(a) With outliers

(b) Without outliers

(c) Pearson's correlation coefficient analysis

Figure 8.5 – Pearson's analysis for Moscow/I2PSnark

The results shows that data approximately follows a normal distribution and additionally a straight line, presenting the required linearity. Finally, the data keeps a constant dispersion, indicating homoscedasticity, the final requirement to properly apply Pearson's coefficient.

Figure 8.5(c) shows the result of Pearson's correlation for the fifteen-day sample with a value of $r = 0.4901$. We see a **strong correlation** according to the references values presented in Subsection 8.2.2. This shows that I2P users from Moscow highly contributed to the I2P's file-sharing activity during the analysed period. The coefficient of determination is $r^2 = 0.2401$, indicating that the activity of users from Moscow explained a quarter of all detected file-sharing activity for this particular fifteen-day period.

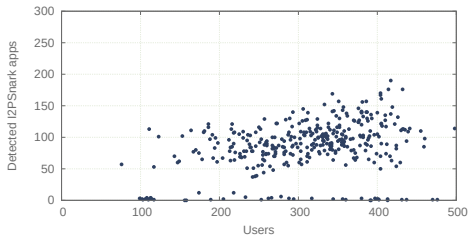
Correlation analysis for Saint Petersburg

Figure 8.6(a) plots Saint Petersburg's data and Figure 8.6(b) depicts this data after outliers' exclusion through our trimmed estimator.

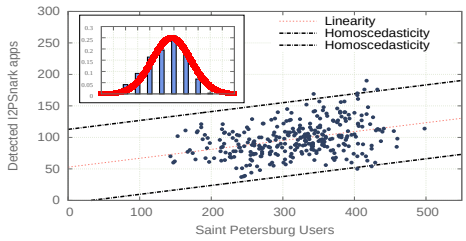
The results shows that the data approximately follows a normal distribution. Linearity as well as homoscedasticity were corroborated with the same mechanisms as with Moscow. Figure 8.6(c) shows the result of Pearson's correlation for the fifteen-day period, where a **moderate correlation** is observed with a value of $r = 0.3952$. In this case, the coefficient of determination indicates that the activity of users from Saint Petersburg accounted for 15.61% of the total file-sharing activity for our period of analysis.

Correlation analysis for Munich

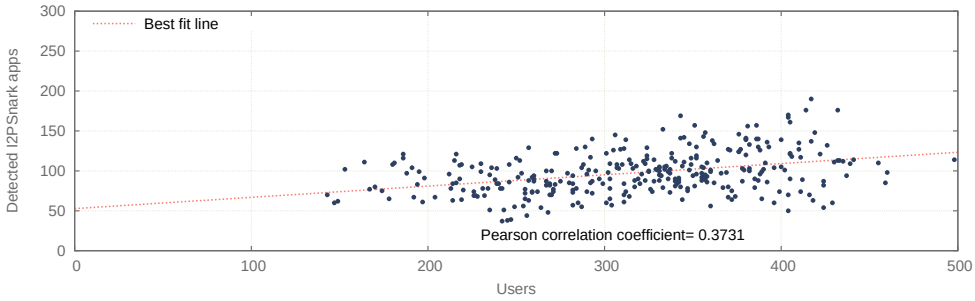
We saw that two of the most active cities presented a strong correlation value (for Moscow) and a moderate correlation value (for Saint Petersburg) with I2PSnark applications according



(a) With outliers



(b) Without outliers

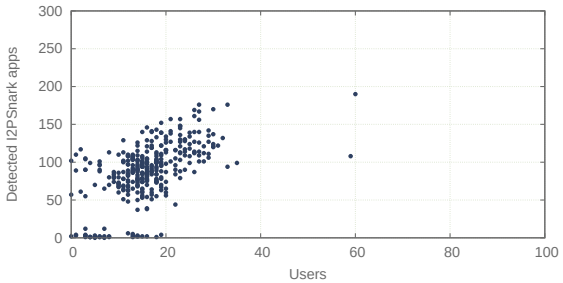


(c) Pearson's correlation coefficient analysis

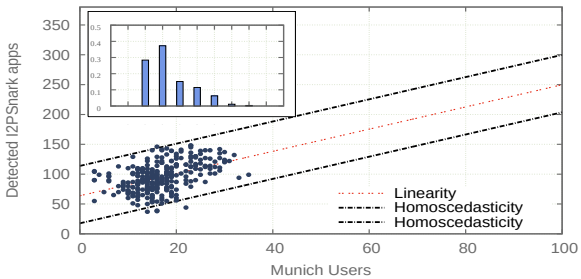
Figure 8.6 – Pearson's analysis for Saint Petersburg/I2PSnark

to Pearson's correlation coefficient. Let's consider a city such as Munich. This city has an active daily participation, however it barely contributes to 0.2% of the total number of users detected. Figure 8.7(a) plots our data for the city of Munich, where a possible correlation might not seem as clear as with the previous cities. Figure 8.7(b) depicts the data after outliers' exclusion.

The results show a lack of a normal distribution. Moreover, the data presents heteroscedasticity, where for bigger values of users, smaller is the variance, *i.e.* the data points are close to the line of best. In this case, Pearson's coefficient was not applicable, since data did not comply with the coefficient's assumptions.



(a) With outliers



(b) Without outliers

Figure 8.7 – Data distribution for Munich/I2PSnark

We could not determine whether users from Munich contribute to I2P's file-sharing activity. However, we can visually analyse the data. We observe that most of the points were concentrated

between 14 and 24 I2P users, while the number of I2PSnark applications detected varied from 50 to 150. This behaviour can be observed in the frequency histogram, where both peaks correspond to intervals [14,18) and [18,22) and account for the 65% of the total data points. This indicates that changes in the number of detected I2PSnark applications were not related with changes in the number of I2P users from Munich. According to this visual analysis, we can deduce that this set of users did not perform significant file-sharing in the I2P network.

8.3.4 Analysis of *low-end* outliers

We consider that a trimmed estimator is more adequate for our correlation analysis, since we discard outliers and keep only those representative values. Winsorising is adequate to deal with errors in the measurements, where low values are replaced with more suitable, and more likely values. However, in our case it would distort the dataset when replacing deviant values, either I2P users or I2PSnark values.

Table 8.2 shows a comparison of the different Pearson's correlation values for our fifteen-day period for the analysed cities. We took into consideration different configurations of our two adapted robust estimators. We also consider a visual exclusion of outliers. We observe that the correlation coefficient for data with a visual exclusion and with the use of a trimmed estimator at 10% are similar for both cities. This indicates that a visual exclusion of outliers is adequate when the dataset can be plotted and easily interpreted. Using the dataset without an outliers analysis leads to biased results, such as the case of Moscow, where the correlation coefficient increases up to a value of 0.76.

City	No Exclusion	Visual	Trim. 10%	Trim. 20%	Wins. 10%	Wins. 20%
Moscow	0.7657	0.4997	0.4901	0.4111	0.7418	0.5902
Saint Petersburg	0.3300	0.3731	0.3952	0.3370	0.2916	0.3419

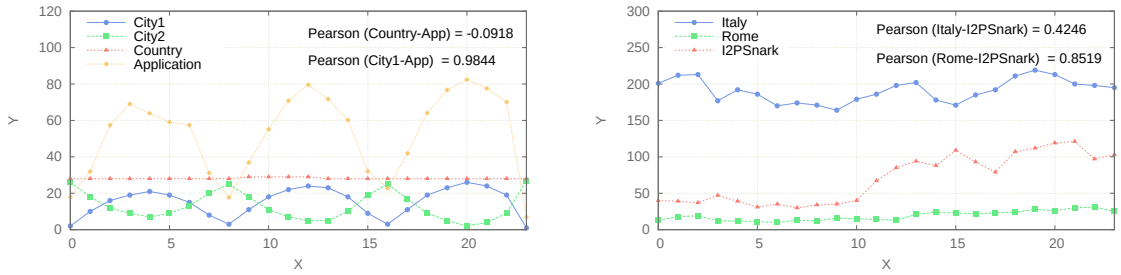
Table 8.2 – Pearson's coefficients for different robust estimators

8.4 Discussion

We introduced a correlation analysis on the I2P network, so as to determine which city contributed the most to I2P's file-sharing activity, thus characterising these I2P users in terms of use and deployment of an anonymous application. Here, we analyse a possible correlation between a country and I2PSnark applications.

We only considered I2P cities and I2PSnark applications, and left aside a possible analysis between countries and these applications. When combined into a single country, the correlation might not be noticed. An extreme example of this scenario is presented in Figure 8.8(a), where a strong correlation exists between an application and a particular city, but when considering the value for the country itself, including both cities, the correlation value considerable decreases. Figure 8.8(b) shows a real-case scenario considering Italy and Rome. There is a moderate correlation between Italy and I2PSnark applications. However, when considering Rome and I2PSnark applications the correlation value doubles. A pre-analysis needs to be carried out before considering a correlation analysis between countries and applications.

Furthermore, we only studied I2PSnark applications, since it is the most active file-sharing application on the network. The rest of the file-sharing clients reached a low daily participation



(a) An extreme case of a country-based correlation analysis

(b) A real case of a country-based correlation analysis

Figure 8.8 – A false negative case for a country' correlation

and therefore they were not adequate for our correlation analysis. Additional applications, such as anonymous web servers, did not present the necessary variation through time in order to correlate them.

8.5 Conclusion

We demonstrated that it is possible to simultaneously analyse the behaviour of I2P users and I2P applications. We showed that we were able to profile I2P users and determine which set of users contributed the most to I2P's file-sharing activity.

By accordingly applying Pearson's correlation coefficient to the data retrieved by our distributed architecture, we detected a strong correlation between I2P's top city, Moscow, and I2P's top file-sharing application, I2PSnark. A moderate correlation was as well detected between users from Saint Petersburg and I2PSnark applications. The variance of users from both cities explained a third of the total file-sharing activity in the I2P network for the studied period. We additionally showed that not every active city contributed to I2P's file-sharing activity, such as Munich.

Our analysis could be improved by improving our geographical localisation tool to obtain more precise geographical zones in a city. If we are able to confine a group of users to a particular region in a city and correlate them against I2PSnark applications, we could be able to determine which part of the city is more or less responsible of the detected anonymous file-sharing activity.

The I2P network is a complete anonymising layer, but it still requires further anonymity studies and peer-reviewed research to strengthen the overall provided anonymity. Our work contributed to the ongoing improvements on the network, for which we foresee an excellent option when considering anonymising services.

Conclusion

Chapter 9

General Conclusion

Contents

This chapter summarises our contributions and presents the final conclusion about our thesis. We introduce the limitations of our work, mainly considering the de-anonymisation of users based on large-scale monitoring. We end with our work perspectives in the field of anonymous networks.

9.1 Contributions summary

We presented two main contributions. On the one hand, we designed and evaluated two hybrid file-sharing architectures aiming at improving content indexation and content availability in BitTorrent-like file-sharing environments. On the other hand, we designed and evaluated a distributed monitoring architecture that allows us to characterise a widely deployed anonymous system.

9.1.1 Hybrid peer-to-peer file-sharing environments

We studied hybrid file-sharing environments from two angles. In the first place, we improved the content indexation scheme in the BitTorrent file-sharing environment via a hybrid BitTorrent-Kad/Ed2k approach, considering the best of both networks. In the second place, we improved the content availability in the I2P file-sharing environment, where anonymous I2P users can access public BitTorrent content through an on-demand approach.

Improving content indexation in the BitTorrent file-sharing environment

Peer-to-peer file-sharing applications still generate a considerable amount of all Internet traffic. Nearly half of all European upstream traffic is BitTorrent and Kad/Ed2k traffic, which indicates that both networks are highly used and deployed.

On the one hand, BitTorrent is the most used content distribution platform and is based on a vulnerable implementation of its Kademlia-based decentralised tracker, making the entire architecture vulnerable. The Kad/Ed2k network, on the contrary, uses a secure DHT to provide a double-indexation mechanism. On the other hand, the algorithm of download of BitTorrent reduces nearly 50% the time of download when compared to the Ed2k algorithm, in a single-file download environment. Therefore, we argue that both networks can benefit from one another, achieving a hybrid peer-to-peer file-sharing environment.

We designed, evaluated and implemented a hybrid file-sharing client named *hMule*. This client is able to index BitTorrent content in the Kad DHT, thus empowering the BitTorrent network to use the Kad's indexation service instead of its vulnerable decentralised tracker implementation. This hybrid scheme additionally enables a double-indexation scheme, where not only `<content,peers>` pairs are indexed, but `<keyword,contents>` pairs are indexed as well. This double-indexation introduces an extra indexation level in the BitTorrent network, a feature currently missing. The hMule client allows users to download content using the algorithm of download of BitTorrent, thus gathering the best of both networks, a strong content indexation scheme and an excellent content distribution platform.

The hMule client demonstrates how a hybrid interconnection scheme between two file-sharing networks can lead to an improved file-sharing environment, avoiding to reinvent the wheel and to create two similar Kademlia-based content indexation mechanisms. The hMule client has been tested in both the BitTorrent and the Kad/Ed2k networks and it has been proven functional and backward compatible. The hMule project [61] encourages Internet users to employ the hMule client and benefit from this hybrid cooperation.

Improving content availability in the I2P file-sharing environment

Anonymous communications are significantly growing and Internet users are shifting to a *privacy-preserving Internet*, drawing their attention to the concept of online anonymity and online privacy. In these anonymous communications, anonymous file-sharing environments are a major field, where dedicated file-sharing networks have been designed and deployed, such as the I2P anonymous file-sharing network. The I2P network provides a BitTorrent-like file-sharing environment on top of the network, which allows users to index and distribute content in an anonymous manner. However, the available content on the system is widely reduced, and non-anonymous public networks, such as the BitTorrent network, are still the major sources of content.

We designed and developed a hybrid file-sharing client named *BiTIIP*, enabling anonymous I2P users to access public BitTorrent content without compromising their anonymity. The BiTIIP client enables an *on-demand* approach, where I2P users specify the desired BitTorrent content and an interconnection between both networks is created to respond to this demand. The client indexes BitTorrent content in the I2P decentralised tracker, enabling an anonymous content indexation, while content is distributed through the I2P anonymous network, proving an anonymous content distribution. This hybrid scheme results in a fully anonymous file-sharing environment.

61. Accessible at `http://hmule.gforge.inria.fr/`.

The connectME project[62] provides the first step towards the interconnection of anonymous and non-anonymous networks, with a special focus on anonymous file-sharing environments. The project enables Internet users to install a BiTIIP client, in this way increasing the available number of BiTIIP clients.

9.1.2 Characterisation of anonymous environments

We conducted the first large-scale monitoring on the I2P anonymous system, characterising users and services running on top of the network. We first designed and implemented a distributed monitoring architecture based on distributed probes placed in the I2P's netDB, which allows us to collect a vast amount of network metadata. Then, we conducted a correlation analysis between the behaviour of users and the behaviour of anonymous file-sharing clients to achieve a group-based characterisation in the network, enabling us to determine which I2P city contributed the most to the anonymous file-sharing activity in the network.

Characterisation of the anonymous I2P environment

Anonymous systems have been gathering more and more users. The I2P anonymous network has doubled its user-base in the last year, but still there is no comprehensive analysis of the network nor monitoring studies characterising the system. The I2P network provides a wide set of services, such as anonymous file-sharing clients and anonymous web hosting. However, we cannot measure the use of these anonymous services on the system and determinate the real use of the I2P network.

Our distributed monitoring architecture allows us to determine the use of anonymous file-sharing applications and anonymous web servers or *eepsites* available in the I2P network. We established that the I2PSnark client is the most used anonymous file-sharing application, contrary to other file-sharing alternatives, such as iMule or I2Phex. We detected as well several *unlisted* eepsites on the system, which accounted for a 30% of the total number of eepsites detected. We additionally determined the geographical distribution of the I2P users and concluded that the I2P network is widely deployed, being Russia the most active country.

This large-scale monitoring provided us with the first insights of the I2P anonymous system, including the use of different anonymous services, as well as the geographical distribution of the I2P users. Our distributed architecture ease the understanding of this anonymous system, help us to understand in which direction the system is evolving and the main use of the system.

Group-based characterisation in the I2P anonymous environment

Our distributed monitoring architecture provides us with different insights about the I2P network, where different application-level analyses are performed to detect anonymous services, such as anonymous file-sharing clients or anonymous web servers. These application-level analyses allow us to detect the behaviour of particular applications, notably their period of activity. By considering the behaviour of a particular anonymous service along with a particular set of I2P users, we were finally able to determine in which measure this set of users was responsible for the activity of the anonymous service.

We applied Pearson's correlation coefficient to successfully established which I2P cities contributed the most to the file-sharing activity in the network. By correlating the behaviour of the I2P users from two top cities along with the behaviour of the I2PSnark client throughout a

62. Accesible at `http://connectme.gforge.inria.fr/`

particular period of time, we determined that the activity of users from those cities explained 38% of all detected file-sharing activity.

We demonstrated that a large-scale monitoring could not only provide us with different insights of an anonymous system, but it could as well led to a group-based characterisation. The behaviour of users on the system is as important as the anonymity guaranteed by the system itself and need to be considered whenever employing an anonymous system.

9.2 Limitations

The limitations of our work are twofold. One the one hand considering our hybrid file-sharing architectures and on the other hand considering the de-anonymisation of users based on large-scale monitoring.

9.2.1 Limitations on hybrid file-sharing architectures

We designed and evaluated two hybrid file-sharing architectures, enabling a better indexation of the BitTorrent content via the Kad DHT and improving the content availability in the I2P network. Both of our approaches work by introducing an interconnection layer, serving as the meeting point between the two intended networks to interconnect. This layer is formed by *volunteer* hybrid nodes, which are deployed by regular Internet users.

The disadvantage of our hybrid model lays on the volunteer-based approach, since if the number of volunteers is less than the required number of hybrid nodes, the interconnection is weak. However, as more Internet users offer to deploy hybrid nodes, the interconnection becomes stronger. This situation is likely to occur during the bootstrap phase, in which our hybrid clients, as well as the overall hybrid model, are not well-known. The solution is to manually deploy different hybrid nodes to support a weak interconnection until the hybrid model gets well-known and a few external hybrid nodes have been deployed.

9.2.2 Limitations on group-based characterisation through large-scale monitoring and de-anonymisation

We showed that a large-scale monitoring could provide us with valuable insights when considering anonymous systems. We established that anonymous file-sharing is widely deployed in the I2P network, mainly guided by I2PSnark clients. The I2P network is widely deployed, where Russia accounts for nearly the 40% of all I2P users detected. We employed this large-scale monitoring to determine the behaviour of users and anonymous services on the system. Through the Pearson's correlation coefficient we determined which I2P city contributed the most to I2P's file-sharing activity, leading to a group-based characterisation.

This group-based characterisation indicates us which subset of users, from a particular city in this case, contribute the most to a particular anonymous service, such as I2PSnark file-sharing clients. However, we are not able to link a particular I2P user with an anonymous service through that large-scale monitoring, mainly because we detect the behaviour of users and anonymous services through a particular period of time, which does not provide us with the required precision to bind a single user with a specific anonymous service.

In order to further de-anonymise a single user, we need to take into consideration the I2P network itself and its mechanisms to guarantee a user's anonymity on the system as described in the following section. By compromising those mechanisms, we could be able to determine the services a given user is running on top of the I2P network.

9.3 Perspectives

Our future work is focused on three aspects of the I2P network.

9.3.1 User de-anonymisation

Starting from our limitations to de-anonymise a particular I2P user, we studied I2P's unidirectional tunnels and the mechanism used to create these tunnels. We discovered a vulnerability in the mechanisms to create I2P's tunnels, which allows an attacker to detect whether he/she is the last participant in an inbound tunnel. With this knowledge, it would be possible to attack an I2P's eepsite in order to de-anonymise the eepsite's operator.

De-anonymising a single user in the I2P network would require two steps.

We first would need an automatic mechanism to reduce the set of *suspects*, *i.e.* which nodes we think are hosting an anonymous web site. Otherwise, it is practically infeasible to attack the entire I2P network.

The second step is a mechanism to attack I2P's local profiling algorithm, so as to position malicious I2P routers in an inbound tunnel.

9.3.2 Attack detection in I2P's netDB

I2P's netDB is a Kademlia-based DHT without any mechanism to certify identities and therefore it is susceptible to a Sybil attack. Although it contains a slight modification in how routing identifiers are calculated to avoid a localised Sybil attack, it is still possible to deploy more general attacks.

An attacker running several floodfill nodes could either monitor the network or attack it. By placing a big set of malicious floodfill nodes and gathering more I2P's metadata than a single floodfill node, an attacker could decide to stop answering requests for that metadata, producing a major disruption at a particular moment.

A full-space crawling in the netDB could give us every active floodfill node and their information, including routing identifiers and IP addresses. By inspecting this information, we could determine whether a single IP address has multiple routing identifiers associated.

Attack detection is the first step towards a more robust netDB, where a identities' management mechanism could lessen the impact of a Sybil attack. The netDB does not enable free placement of nodes and therefore a localised Sybil attack is practically infeasible. Local rules to avoid multiple floodfill nodes from a single node would prevent a full-space Sybil attack.

9.3.3 Content characterisation of I2P's *eepsites*

The I2P network is optimised for anonymous hosting, where different *eepsites* are available on the system. We have detected through our distributed monitoring architecture several unlisted eepsites, accounting for the 30% of the total available eepsites.

We consider that analysing every eepsite could allow us to characterise I2P's anonymous hosting environment, to determine which kind of content is hosted on the system. This *in-depth* characterisation would enable us to understand the system and its users, not only from a behavioural point of view, but from a preferences point of view.

Bibliography

[1] S. Fahmy M. Kwon. Synergy: an overlay internetworking architecture. In *Proc. of the 14th IEEE International Conference on Computer Communications and Networks*, pages 401–406, San Diego, CA, USA, October 2005.

[2] Hai Jiang, Jun Li, Zhongcheng Li, and Jing Liu. Efficient hierarchical content distribution using P2P technology. In *Proc. of the 16th IEEE International Conference on Networks*, pages 1 –6, December 2008.

[3] Jaime Lloret, Fernando Boronat, Carlos E. Palau, and Manuel Esteve. Two Levels SPF-Based System to Interconnect Partially Decentralized P2P File Sharing Networks. In *Proc. of The 2005 Joint International Conference on Autonomic and Autonomous Systems / International Conference on Networking and Services 2005*, Papeete, Tahiti, 2005.

[4] Junjiro Konishi, Naoki Wakamiya, and Masayuki Murata. Proposal and evaluation of a cooperative mechanism for pure p2p file sharing networks. In *Proc. of the 2nd International Conference on Biologically Inspired Approaches to Advanced Information Technology*, BioADIT '06, pages 33–47, Berlin, Heidelberg, 2006. Springer-Verlag.

[5] Hongye Fu, Naoki Wakamiya, and Masayuki Murata. A Cooperative Mechanism for Hybrid P2P File-Sharing Networks to Enhance Application-Level QoS. *IEICE transactions on communications*, 89(9):2327–2335, September 2006.

[6] Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas Anderson. Privacy-preserving P2P data sharing with OneSwarm. In *Proc. of the 2010 ACM SIGCOMM Conference*, SigComm '10, pages 111–122, New York, NY, USA, 2010. ACM.

[7] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Evaluation of Sybil Attacks Protection Schemes in KAD. In Ramin Sadre and Aiko Pras, editors, *Proc. of the 3rd International Conference on Autonomous Infrastructure, Management and Security*, volume 5637 of *AIMS '09*, pages 70–82, Enschede, Netherlands, 2009. University of Twente, Springer.

[8] BitTorrent.org. The BitTorrent Protocol. `http://bittorrent.org/`. Last visited on 08/2013.

[9] Oliver Heckmann, Axel Bock, Andreas Mauthe, and Ralf Steinmetz. The eDonkey File-Sharing Network. `ftp://www.kom.tu-darmstadt.de/papers/HBMS04-1-paper.pd`. Last visited on 08/2013.

[10] Jacob Appelbaum and Roger Dingledine. How Governments Have Tried To Block Tor. `http://archive.org/details/28c3howGovernmentsHaveTriedToBlockTorByJacobAppelbaumRoger`. Last visited on 08/2013.

[11] Ramesh Subramanian. The Growth of Global Internet Censorship and Circumvention: A Survey. `http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2032098. Lastvisitedon08/2013.`

[12] Margaret Coker. U.A.E. Puts the Squeeze on BlackBerry. http://online.wsj.com/article/SB10001424052748704702304575402493300698912.html. Last visited on 08/2013.

[13] Bill Marczak and Morgan Marquis-Boire. Researchers Find 25 Countries Using Surveillance Software. http://bits.blogs.nytimes.com/2013/03/13/researchers-find-25-countries-using-surveillance-software/. Last visited on 08/2013.

[14] James Bamford. The NSA Is Building the Country Biggest Spy Center (Watch What You Say). http://www.wired.com/threatlevel/2012/03/ff_nsadatacenter/. Last visited on 08/2013.

[15] Official Journal of the European Union. Directive 2006/24/EC of the European Parliament. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:105:0054:0063:EN:PDF. Last visited on 08/2013.

[16] Mike Rogers and Dutch Ruppersberger. Cyber Intelligence Sharing and Protection Act. http://thomas.loc.gov/cgi-bin/bdquery/z?d113:h.r.624:. Last visited on 08/2013.

[17] Bram Cohen. Incentives build robustness in bittorrent. Technical report, bittorrent.org, 2003.

[18] Andrew Loewenstern. DHT Protocol. http://www.bittorrent.org/beps/bep_0005.html.Lastvisitedon08/2013. Last visited on 08/2013.

[19] Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Revised Papers from the 1st International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK, 2002. Springer-Verlag.

[20] Donald E. Eastlake and Paul E. Jones. US Secure Hash Algorithm 1 (SHA1). http://www.ietf.org/rfc/rfc3174.txt?number=3174. Last visited on 08/2013.

[21] Ronald L. Rivest. The MD5 Message-Digest Algorithm (RFC 1321). http://www.ietf.org/rfc/rfc1321.txt?number=1321.

[22] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SigComm '01, pages 149–160, New York, NY, USA, 2001. ACM.

[23] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev.*, 31(4):161–172, 2001.

[24] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Middleware '01, pages 329–350, London, UK, UK, 2001. Springer-Verlag.

[25] Zhonghong Ou, Erkki Harjula, Otso Kassinen, and Mika Ylianttila. Performance evaluation of a kademlia-based communication-oriented p2p system under churn. *Comput. Netw.*, 54(5):689–705, 2010.

[26] BitTorrent Enhancement Proposals. Mainline distributed hash table speficication. http://bittorrent.org/beps/bep_0005.html. Last visited on 08/2013.

[27] Vuze Wiki. The Vuze distributed hash table. http://wiki.vuze.com/w/Distributed_hash_table. Last visited on 08/2013.

[28] Ghulam Memon, Reza Rejaie, Yang Guo, and Daniel Stutzbach. Large-scale monitoring of DHT traffic. In *Proc. of the 8th International Conference on Peer-to-peer Systems*, IPTPS '09, pages 11–11, Berkeley, CA, USA, 2009. USENIX Association.

[29] Moritz Steiner, Wolfgang Effelsberg, and Taoufik En-najjary. Load reduction in the kad peer-to-peer system. In *Proc. of the 5th International Workshop on Databases, Information Systems and Peer-to-Peer Computing*, Viena, Austria, 2007.

[30] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Monitoring and Controlling Content Access in KAD. In *Proc. of the 9th International Conference on Communications*, Capetown, South Africa, May 2010. IEEE.

[31] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proc. of the 1st HotNets Workshop*, Princeton, New Jersey, USA, October 2002.

[32] Jarret Falkner, Michael Piatek, John P. John, Arvind Krishnamurthy, and Thomas Anderson. Profiling a million user dht. In *Proc. of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 129–134, New York, NY, USA, 2007. ACM.

[33] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proc. of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 189–202, New York, NY, USA, 2006. ACM.

[34] Moritz Steiner, Ernst W Biersack, and Taoufik En Najjary. Actively monitoring peers in KAD. In *Proc. of the 6th International Workshop on Peer-to-Peer Systems*, Bellevue, United States, February 2007.

[35] Scott Wolchok and J. Alex Halderman. Crawling BitTorrent DHTs for fun and profit. In *Proc. of the 4th USENIX Conference on Offensive Technologies*, WOOT '10, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association.

[36] K. Junemann, P. Andelfinger, J. Dinger, and H. Hartenstein. BitMON: A Tool for Automated Monitoring of the BitTorrent DHT. In *Proc. of the 10th IEEE International Conference on Peer-to-Peer Computing*, pages 1 –2, Delft, The Netherlands, August 2010.

[37] Moritz Steiner and Ernst W Biersack. Crawling Azureus. Technical Report EURECOM+2495, Eurecom, June 2008.

[38] John R. Douceur. The Sybil Attack. In *Revised Papers from the 1st International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, 2002. Springer-Verlag.

[39] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. A survey of DHT security techniques. *ACM Comput. Surv.*, 43(2):8:1–8:49, 2011.

[40] Jochen Dinger and Hannes Hartenstein. Defending the sybil attack in p2p networks: Taxonomy, challenges, and a proposal for self-registration. In *Proc. of the 1st International Conference on Availability, Reliability and Security*, ARES '06, pages 756–763, Washington, DC, USA, 2006. IEEE Computer Society.

[41] Yo Mashimo, Masanori Yasutomi, and Hiroshi Shigeno. SRJE: Decentralized Authentication Scheme against Sybil Attacks. In *Proc. of the 12th International Conference on Network-Based Information Systems*, pages 220–225, Indianapolis, USA, 2009.

[42] Honghao Wang, Yingwu Zhu, and Yiming Hu. An efficient and secure peer-to-peer overlay network. In *Proc. of the 30th Conference on Local Computer Networks*, LCN '05, pages 764–771, Washington, DC, USA, 2005. IEEE Computer Society.

[43] Haifeng Yu, P.B. Gibbons, M. Kaminsky, and Feng Xiao. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. In *Proc. of the 2008 IEEE Symposium on Security and Privacy*, pages 3 –17, May 2008.

[44] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. In *Proc. of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SigComm '06, pages 267–278, New York, NY, USA, 2006. ACM.

[45] Brian Neil Levine, Clay Shields, and N. Boris Margolin. A Survey of Solutions to the Sybil Attack. Tech report 2006-052, University of Massachusetts Amherst, Amherst, MA, October 2006.

[46] Rupert Gatti, Stephen Lewis, Andy Ozment, Thierry Rayna, and Andrei Serjantov. Sufficiently secure peer-to-peer networks. *Workshop on the Economics of Information Security*, 2004.

[47] N. Boris Margolin and Brian N. Levine. Informant: detecting sybils using incentives. In *Proc. of the 11th International Conference on Financial cryptography and 1st International conference on Usable Security*, FC '07/USEC '07, pages 192–207, Berlin, Heidelberg, 2007. Springer-Verlag.

[48] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. Exploiting KAD: possible uses and misuses. *SIGCOMM Comput. Commun. Rev.*, 37(5):65–70, 2007.

[49] Zhoujun Li and Xiaoming Chen. Misusing Kademlia Protocol to Perform DDoS Attacks. In *Proc. of the 2008 IEEE International Symposium on Parallel and Distributed Processing with Applications*, ISPA '08, pages 80–86, Washington, DC, USA, 2008. IEEE Computer Society.

[50] Peng Wang, James Tyra, Eric Chan-Tin, Tyson Malchow, Denis Foo Kune, Nicholas Hopper, and Yongdae Kim. Attacking the Kad network. In *Proc. of the 4th international conference on Security and Privacy in Communication Netowrks*, SecureComm '08, pages 23:1–23:10, New York, NY, USA, 2008. ACM.

[51] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Evaluation of Sybil Attacks Protection Schemes in KAD. In *Proc. of the 3rd International Conference on Autonomous Infrastructure, Management and Security: Scalability of Networks and Services*, AIMS '09, pages 70–82, Berlin, Heidelberg, 2009. Springer-Verlag.

[52] Michael Kohnen, Mike Leske, and Erwin P. Rathgeb. Conducting and Optimizing Eclipse Attacks in the Kad Peer-to-Peer Network. In *Proc. of the 8th International IFIP-TC 6 Networking Conference*, Networking '09, pages 104–116, Berlin, Heidelberg, 2009. Springer-Verlag.

[53] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Efficient DHT attack mitigation through peers ID distribution. In *Proc. of the 24th IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, Atlanta, Georgia, USA, 2010. IEEE.

[54] Marti Ksionsk, Ping Ji, and Weifeng Chen. Attacks on BitTorrent - An Experimental Study. In Xuejia Lai, Dawu Gu, Bo Jin, Yongquan Wang, and Hui Li, editors, *Proc. of 3rd International Conference on Forensic Applications and Techniques in Telecommunications, Information and Multimedia*, volume 56 of *e-Forensics '10*, pages 79–89. Springer, 2010.

[55] Guillaume Urvoy Keller, Soufiane Rouibia, Jonathan Vayn, and Olivier Beauvais. Early stage denial of service attacks in BitTorrent: an experimental study. In *Proc. of the 2008 Workshop on Collaborative Peer-to-Peer Systems*, Roma, Italy, June 2008.

[56] Scott Wolchok, Owen S. Hofmann, Nadia Heninger, Edward W. Felten, J. Alex Halderman, Christopher J. Rossbach, Brent Waters, and Emmett Witchel. Defeating Vanish with Low-Cost Sybil Attacks Against Large DHTs. In *Proc. of the 17th Network and Distributed System Security Symposium*, San Diego, CA, USA, 2010. The Internet Society.

[57] Roxana Geambasu, Tadayoshi Kohno, Amit A. Levy, and Henry M. Levy. Vanish: increasing data privacy with self-destructing data. In *Proc. of the 18th USENIX Security Symposium*, SSYM '09, pages 299–316, Berkeley, CA, USA, 2009. USENIX Association.

[58] Liang Wang and J. Kangasharju. Real-world sybil attacks in BitTorrent Mainline DHT. In *Proc. of the 2012 IEEE Global Communications Conference*, pages 826–832, Anaheim, CA, USA, 2012.

[59] Matteo Varvello and Moritz Steiner. Traffic localization for DHT-based BitTorrent networks. In *Proc. of the 10th International IFIP TC 6 Conference on Networking*, Networking '11, pages 40–53, Berlin, Heidelberg, 2011. Springer-Verlag.

[60] Scott Crosby and Dan Wallach. An Analysis of BitTorrent's Two Kademlia-Based DHTs. Technical Report TR-07-04, Rice University, 2007.

[61] Luigi Liquori, Cédric Tedeschi, Laurent Vanni, Francesco Bongiovanni, Vincenzo Ciancaglini, and Bojan Marinkovic. Synapse: A Scalable Protocol for Interconnecting Heterogeneous Overlay Networks. Technical report, INRIA, 2009.

[62] Naoki Wakamiya and Masayuki Murata. Overlay Network Symbiosis: Evolution and Cooperation. In *Proc. of the 1st Conference on Bio-Inspired Models of Network, Information and Computing Systems*, pages 1 –5, Lugano, Switzerland, December 2006.

[63] Min Yang and Yuanyuan Yang. An Efficient Hybrid Peer-to-Peer System for Distributed Data Sharing. *IEEE Trans. Comput.*, 59(9):1158–1171, 2010.

[64] Tim Wauters, Jan Coppens, Filip De Turck, Bart Dhoedt, and Piet Demeester. Replica placement in ring based content delivery networks. *Computer Communications*, 29(16):3313 – 3326, 2006.

[65] Matthew Edman and Bülent Yener. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Comput. Surv.*, 42(1):5:1–5:35, 2009.

[66] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf, 2010. v0.34.

[67] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[68] Sameer Parekh. Prospect for remailers: where is Anonymity heading on the Internet ?. First Monday, Volume 1, Number 2 - 5 August 1996. `http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/476/397`. Last visited on 08/2013.

[69] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol - Version 3. IETF Internet Draft, 2003.

[70] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *Proc. of the 2003 IEEE Symposium on Security and Privacy*, SP '03, pages 2–, Washington, DC, USA, 2003. IEEE Computer Society.

[71] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1:66–92, 1998.

[72] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In *Proc. of the 2001 International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pages 115–129, Berkeley, California, USA, 2001. Springer-Verlag New York, Inc.

[73] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information. In *Proc. of the 1st International Workshop on Information Hiding*, pages 137–150, London, UK, 1996. Springer-Verlag.

[74] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *Proc. of the 13th conference on USENIX Security Symposium*, SSYM '04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[75] Lasse Overlier and Paul Syverson. Locating Hidden Servers. In *Proc. of the 2006 IEEE Symposium on Security and Privacy*, SP '06, pages 100–114, Washington, DC, USA, 2006. IEEE Computer Society.

[76] Tsuen-Wan Johnny Ngan, Roger Dingledine, and Dan S. Wallach. Building incentives into tor. In *Proc. of the 14th International Conference on Financial Cryptography and Data Security*, FC '10, pages 238–256, Berlin, Heidelberg, 2010. Springer-Verlag.

[77] Lasse Øverlier and Paul Syverson. Improving efficiency and simplicity of Tor circuit establishment and hidden services. In *Proc. of the 7th International Conference on Privacy Enhancing Technologies*, PET '07, pages 134–152, Berlin, Heidelberg, 2007. Springer-Verlag.

[78] Jon McLachlan, Andrew Tran, Nicholas Hopper, and Yongdae Kim. Scalable onion routing with torsk. In *Proc. of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 590–599, New York, NY, USA, 2009. ACM.

[79] Michael Herrmann and Christian Grothoff. Privacy-implications of performance-based peer selection by onion-routers: a real-world case study using I2P. In *Proc. of the 11th International Conference on Privacy Enhancing Technologies*, PETS '11, pages 155–174, Berlin, Heidelberg, 2011. Springer-Verlag.

[80] Adrian Crenshaw. Darknets and hidden servers: Identifying the true IP/network identity of I2P service hosts. In *Black Hat DC 2011*, DC, March 2011.

[81] Anthéa Mayzaud. Analyse des vulnérabilités du réseau pair à pair anonymisé I2P. Master's thesis, ESIAL, Villers-lès-Nancy, 2011.

[82] The AVISPA Project. Automated Validation of Internet Security Protocols and Applications. `http://www.avispa-project.org/`. Last visited on 08/2013.

[83] Kevin Bauer, Damon McCoy, Dirk Grunwald, and Douglas Sicker. BitStalker: Accurately and efficiently monitoring bittorrent traffic. In *Proc. of the 1st IEEE International Workshop on Information Forensics and Security*, London, UK, 2009.

[84] Stevens Le Blond, Pere Manils, Abdelberi Chaabane, Mohamed Ali Kaafar, Claude Castelluccia, Arnaud Legout, and Walid Dabbous. One bad apple spoils the bunch: exploiting P2P applications to trace and profile Tor users. In *Proc. of the 4th USENIX Conference on Large-scale Exploits and Emergent Threats*, LEET '11, pages 2–2, Berkeley, CA, USA, 2011. USENIX Association.

[85] Pere Manils, Chaabane Abdelberi, Stevens Le-Blond, Mohamed Ali Kâafar, Claude Castelluccia, Arnaud Legout, and Walid Dabbous. Compromising Tor Anonymity Exploiting P2P Information Leakage. *CoRR*, abs/1004.1461, 2010.

[86] The OFF Project. The OFF System. `http://offsystem.sourceforge.net/`. Last visited on 08/2013.

[87] Marc Seeger. *The current state of anonymous file-sharing* Bachelor's thesis, Studiengang Medieninformatik Hochschule der Medien, Stuttgart, Germany, July 2008.

[88] Swagatika Prusty, Brian Neil Levine, and Marc Liberatore. Forensic investigation of the OneSwarm anonymous filesharing system. In *Proc. of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 201–214, New York, NY, USA, 2011. ACM.

[89] M. Cunche, M.A. Kaafar, Jiefeng Chen, R. Boreli, and A. Mahanti. Why are they hiding? Study of an anonymous file sharing system. In *Proc. of the 1st Conference on Satellite Telecommunications*, pages 1 –6, Rome, Italy, October 2012.

[90] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Proc. of the 2001 International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pages 46–66, Berkley, CA, USA, 2001. Springer-Verlag New York, Inc.

[91] Stanley Milgram. The Small World Problem. *Psychology Today*, 2:60–67, 1967.

[92] Nathan S. Evans, Chris GauthierDickey, and Christian Grothoff. Routing in the Dark: Pitch Black. In *Proc. of the 23rd Conference on Computer Security Applications Conference*, pages 305–314, Miami Beach, Florida, USA, 2007.

[93] Benjamin Schiller, Stefanie Roos, Andreas Hofer, and Thorsten Strufe. Attack Resistant Network Embeddings for Darknets. In *Proc. of the 30th IEEE Symposium on Reliable Distributed Systems Workshops*, SRDSW '11, pages 90–95, Washington, DC, USA, 2011. IEEE Computer Society.

[94] Yingfei Dong and Zhenhai Duan. A Traceback Attack on Freenet. Submitted to INFOCOM, 2013.

[95] Yingfei Dong and Zhenhai Duan. A Routing Table Insertion Attack on Freenet. Submitted to the annual ASE/IEEE Cyber Security Conference., 2012.

[96] Krista Bennett, Tiberius Stef, Christian Grothoff, Tzvetan Horozov, and Ioana Patrascu. The GNet Whitepaper. Technical report, Purdue University, 2002.

[97] Krista Bennett, Christian Grothoff, Tzvetan Horozov, and Ioana Patrascu. Efficient sharing of encrypted data. In *Proc. of the 7th Conference on Information Security and Privacy*, pages 107–120, Melbourne, Australia, 2002.

[98] Dennis Kügler. An Analysis of GNUnet and the Implications for Anonymous, Censorship-Resistant Networks. In Roger Dingledine, editor, *Revised Papers of the 3rd Internationl Workshop on Privacy Enhancing Technologies*, volume 2760 of *PETShop '03*, pages 161–176, Dresden, Germany, 2003. Springer.

[99] Abdelberi Chaabane, Pere Manils, and Mohamed Ali Kaafar. Digging into Anonymous Traffic: A Deep Analysis of the Tor Anonymizing Network. In *Proc. of the 4th International Conference on Network and System Security*, NSS '10, pages 167–174, Washington, DC, USA, 2010. IEEE Computer Society.

[100] The Sydney Morning Herald. The hack of the Year. `http://www.smh.com.au/news/security/the-hack-of-the-year/2007/11/12/1194766589522.html?page=fullpage#contentSwap1`. Last visited on 08/2013.

[101] Bruce Schneier. Anonymity and the Tor Network. `http://www.schneier.com/blog/archives/2007/09/anonymity_and_t_1.html.Lastvisitedon08/2013.`

[102] Damon Mccoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining Light in Dark Places: Understanding the Tor Network. In *Proc. of the 8th International Symposium on Privacy Enhancing Technologies*, PETS '08, pages 63–76, Berlin, Heidelberg, 2008. Springer-Verlag.

[103] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. A case study on measuring statistical data in the Tor anonymity network. In *Proc. of the 14th international Conference on Financial Cryptograpy and Data Security*, FC '10, pages 203–215, Berlin, Heidelberg, 2010. Springer-Verlag.

[104] Martin Mulazzani, Markus Huber, and Edgar R. Weippl. Anonymity and monitoring: how to monitor the infrastructure of an anonymity system. *Trans. Sys. Man Cyber Part C*, 40(5):539–546, 2010.

[105] Douglas C. Sicker, Paul Ohm, and Dirk Grunwald. Legal issues surrounding monitoring during network research. In *Proc. of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 141–148, New York, NY, USA, 2007. ACM.

[106] Sandvine. Global Internet Phenomena Report 2011. `http://www.sandvine.com/downloads/documents/05-17-2011_phenomena/Sandvine%20Global%20Internet%20Phenomena%20Report.pdf`. Last visited on 08/2013.

[107] Klaus Mochalski and Hendrik Schulze. Ipoque Internet Study 2008/2009. *Africa*, 2009.

[108] Sandvine. Global Internet Phenomena Report 2012. `http://www.sandvine.com/downloads/documents/Phenomena_2H_2012/Sandvine_Global_Internet_Phenomena_Report_2H_2012.pdf`. Last visited on 08/2013.

[109] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. Exploiting KAD: possible uses and misuses. *Computer Communication Review*, 37(5):65–70, 2007.

[110] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Efficient DHT attack mitigation through peers' ID distribution. In *Proc. of the 7th International Workshop on Hot Topics in Peer-to-Peer Systems*, Atlanta, United States, 2010. IEEE International Parallel & Distributed Processing Symposium.

[111] Moritz Steiner, Damiano Carra, and Ernst W Biersack. Evaluating and improving the content access in KAD. *Springer "Journal of Peer-to-Peer Networks and Applications", Vol 3, NÂ°2, June 2010*, June 2010.

[112] Damian Vicino, Juan Pablo Timpanaro, Isabelle Chrisment, and Olivier Festor. hMule: an unified KAD-BitTorrent file-sharing application. Research Report RR-7815, INRIA, 2011.

[113] Tarang Chugh. Bridging the two BitTorrent Worlds: I2P and P2P. Research report, INRIA, 2012.

[114] Georgos Siganos, Josep M. Pujol, and Pablo Rodriguez. Monitoring the Bittorrent Monitors: A Bird's Eye View. In *Proc. of the 10th International Conference on Passive and Active Network Measurement*, PAM '09, Seoul, Korea, April 2009. Springer-Verlag.

[115] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. A global view of Kad. In *Proc. of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, San Diego, California, USA, October 2007. ACM.

[116] Anirban Banerjee, Michalis Faloutsos, and Laxmi Bhuyan. The P2P War: Someone Is Monitoring Your Activities! In *Proc. of the 6th International Networking Conference*, Networking '07, Atlanta, GA, USA, May 2007. Springer.

[117] Damon Mccoy, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the Tor network. In *Proc. of the 8th Privacy Enhancing Technologies Symposium*, PETS '08, Leuven, Belgium, July 2008. Springer.

[118] Network Working Group. Attacks on Cryptographic Hashes in Internet Protocols. `http://tools.ietf.org/html/rfc4270`. Last visited on 08/2013.

[119] Amir Alsbih, Thomas Janson and Christian Schindelhauer. Analysis of Peer-to-Peer Traffic and User Behaviour. `http://archive.cone.informatik.uni-freiburg.de/pubs/ITA11_bittorrent_alsbih_janson_schindelhauer.pdf`. Last visited on 08/2013.

[120] Russian Federal Service of State Statistics (Rosstat). Russian Census (2002). `http://www.perepis2002.ru/index.html?id=87`. Last visited on 08/2013.

[121] Jérôme Henry. Supervision du réau anonyme I2P. Research report, INRIA, 2012.

[122] S. S. Stevens. On the Theory of Scales of Measurement. *Science*, 103(2684):677–680, 1946.

[123] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences (2nd Edition)*. Routledge Academic, 2 edition, 1988.

[124] Frank Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17-21, 1973.

Publications

International conferences

- **Juan Pablo Timpanaro**, Isabelle Chrisment and Olivier Festor, `Improving Content Availability in the I2P Anonymous File-Sharing Environment`, in *Cyberspace Safety and Security* (CSS) 2012 - Melbourne, Australia
- **Juan Pablo Timpanaro**, Isabelle Chrisment and Olivier Festor, `A Bird's Eye View on the I2P Anonymous File-sharing Environment`, in *Network and System Security* (NSS) 2012 - Wuyishan, China
- **Juan Pablo Timpanaro**, Thibault Cholez, Isabelle Chrisment and Olivier Festor, `BitTorrent's Mainline DHT Security Assessment`, in *New Technologies, Mobility and Security* (NTMS) 2011 - Paris, France

International workshops

- **Juan Pablo Timpanaro**, Thibault Cholez, Isabelle Chrisment and Olivier Festor, `When KAD meets BitTorrent - Building a Stronger P2P Network`, in *Workshop on Hot Topics in Peer-to-Peer Systems* (HotP2P) 2011 - Anchorage, Alaska
- **Juan Pablo Timpanaro**, Isabelle Chrisment and Olivier Festor, `I2P's Usage Characterisation` (short paper), in *Traffic Monitoring and Analysis* (TMA) 2012 - Vienna, Austria

Poster sessions

- **Juan Pablo Timpanaro**, Isabelle Chrisment and Olivier Festor, `Monitoring Anonymous P2P File-Sharing Systems`, in *IEEE Peer-to-Peer* (IEEE P2P) 2013 - Trento, Italy
- **Juan Pablo Timpanaro**, Isabelle Chrisment and Olivier Festor, `Distributed Monitoring On Anonymous Environments`, in *Summer School on Communication Networks* (RESCOM) 2013 - Porquerolles, France

Research reports

- **Juan Pablo Timpanaro**, Isabelle Chrisment and Olivier Festor, `Monitoring the I2P network`, Research report RR-7844, 2011
- Thibault Cholez, **Juan Pablo Timpanaro**, Guillaume Doyen, Isabelle Chrisment, Olivier Festor and Rida Khatoun, `Vulnérabilités de la DHT de BitTorrent & Identification des comportements malveillants dans KAD`, Projet ACDA-P2P, Délivrable 2 (in French), 2011

Most of our daily activities are carried out over the Internet, from file-sharing and social networking to home banking, online-teaching and online-blogging. Considering file-sharing as one of Internet top activities, different architectures have been proposed, designed and implemented, leading to a wide set of file-sharing networks with different performances and goals. This digital society enables as well users' profiling. As Internet users surf the World Wide Web, every sent or received packet passes through several intermediate nodes until they reach their intended destination. So, an observer will be able to determine where a packet comes from and where it goes to, to monitor and to profile users' online activities by identifying to which server they are connected or how long their sessions last. Meanwhile, anonymous communications have been significantly developed to allow users to carry out their online activities without necessarily revealing their real identity.

Our contribution is twofold. On the one hand, we consider hybrid file-sharing environments, with a special focus on widely deployed real-world networks and targeting two defined goals. The first goal is to improve content indexation in the BitTorrent file-sharing environment, enabling BitTorrent content to be indexed in the Kad distributed has table and leading to a more robust BitTorrent system. The second goal is to improve content availability in the I2P file-sharing environment. We allow I2P users to anonymously access public BitTorrent content and we obtain a fully anonymous file-sharing environment, including anonymous content indexation and anonymous content distribution.

On the other hand, we focus on the understanding of anonymous environments through extensive monitoring. We characterise the I2P network, targeting the entire anonymous environment and its anonymous services. We consider different aspects of the network, such as the number of users, their characteristics as well as the number of anonymous services available. Through long-term measurements on the network and along with different correlation analyses, we are able to characterise a small group of users using a specific anonymous service, such as the users from a particular city performing anonymous file-sharing.

La plupart de nos activités quotidiennes sont aujourd'hui rythmées et régies par Internet, qu'il s'agisse de partage de fichiers, d'interaction sur les réseaux sociaux, de banques et de cours en ligne, ou encore de publication de blogs. En ce qui concerne le partage de fichiers, l'une des activités les plus pratiquées sur Internet, diverses solutions ont déjà été proposées, créées et implantées, pour constituer des réseaux de partage de fichiers aux performances et objectifs parfois très différents. Cette société du numérique permet le profilage des utilisateurs. Chaque information envoyée ou reçue sur Internet va ainsi traverser une série de noeuds intermédiaires jusqu'à atteindre sa destination. Un observateur pourra ainsi déterminer la provenance et la destination des paquets et de ce fait surveiller et profiler les activités des utilisateurs en identifiant le serveur auquel ils se connectent ou la durée de leur session. Parallèlement, les communications anonymes se sont développées afin de permettre aux utilisateurs d'utiliser Internet sans que leur identité ne soit nécessairement révélée.

Notre contribution se veut double. Nous nous intéressons tout d'abord aux environnements de partage de fichiers hybrides en nous focalisant sur des réseaux réels largement déployés. Nous visons, au travers de cette étude, deux objectifs. Le premier consiste en l'amélioration du système d'indexation de contenu au sein de l'environnement de partage de fichiers BitTorrent. Il s'agit, plus précisément, de renforcer le système BitTorrent par le biais d'une indexation de son contenu dans la table de hachage distribuée Kad.

Notre second but est d'améliorer les conditions d'accès au contenu ainsi que sa disponibilité au sein de l'environnement I2P de partage de fichiers. Nous permettons aux utilisateurs d'I2P d'accéder anonymement au contenu public de BitTorrent et nous aboutissons ainsi à un environnement de partage de fichiers totalement anonyme, indexation et distribution du contenu comprises. Nous centrons ensuite notre analyse sur une meilleure compréhension des environnements anonymes par le biais d'une surveillance à grande échelle. Nous définissons les caractéristiques du réseau I2P, en visant l'intégralité de l'environnement anonyme et son large champ d'activités et de services anonymes. Nous analysons les différents aspects du réseau, comme le nombre des utilisateurs, leurs caractéristiques ainsi que le nombre de services anonymes disponibles. Au travers des mesures et évaluations réalisées à long terme sur le réseau, couplées à différentes analyses de corrélation, nous avons réussi à identifier et caractériser un petit groupe d'individus exécutant un service anonyme spécifique comme, par exemple, les habitants d'une même ville utilisant une application anonyme de partage de fichiers.

UNIVERSITÉ DE LORRAINE

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Environnements Hybrides et Anonymes pour le Partage de Fichiers : Architecture et Caractérisation

THÈSE

présentée et soutenue publiquement le 6 Novembre 2013

pour l'obtention du

Doctorat de l'Université de Lorraine

(spécialité informatique)

par

Juan Pablo Timpanaro

Composition du jury

Rapporteurs :	Gabi DREO RODOSEK, Professeure à Universität der Bundeswehr München Maryline LAURENT, Professeure à Telecom SudParis
Examinateurs :	Guillaume DOYEN, Maître de Conférences à l'Université de Technologie de Troyes Bénédicte LE GRAND, Professeure à l'Université Paris 1 Panthéon-Sorbonne René SCHOTT, Professeur à l'Université de Lorraine Olivier FESTOR, Directeur de Recherche Inria Nancy-Grand Est
Dissertation Director:	Isabelle CHRISMENT, Professeure à TELECOM Nancy, Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Table des matières

Table des figures

Résumé

La plupart de nos activités quotidiennes sont aujourd'hui rythmées et régies par Internet, qu'il s'agisse de partage de fichiers, d'interaction sur les réseaux sociaux, de banques et de cours en ligne, ou encore de publication de blogs. En ce qui concerne le partage de fichiers, l'une des activités les plus pratiquées sur Internet, diverses solutions ont déjà été proposées, créées et implantées, pour constituer des réseaux de partage de fichiers aux performances et objectifs parfois très différents. Cette société du numérique permet le profilage des utilisateurs. Chaque information envoyée ou reçue sur Internet va ainsi traverser une série de noeuds intermédiaires jusqu'à atteindre sa destination. Un observateur pourra ainsi déterminer la provenance et la destination des paquets et de ce fait surveiller et profiler les activités des utilisateurs en identifiant le serveur auquel ils se connectent ou la durée de leur session. Parallèlement, les communications anonymes se sont développées afin de permettre aux utilisateurs d'utiliser Internet sans que leur identité ne soit nécessairement révélée.

Notre contribution se veut double. Nous nous intéressons tout d'abord aux environnements de partage de fichiers hybrides en nous focalisant sur des réseaux réels largement déployés. Nous visons, au travers de cette étude, deux objectifs. Le premier consiste en l'amélioration du système d'indexation de contenu au sein de l'environnement de partage de fichiers BitTorrent. Il s'agit, plus précisément, de renforcer le système BitTorrent par le biais d'une indexation de son contenu dans la table de hachage distribuée Kad.

Notre second but est d'améliorer les conditions d'accès au contenu ainsi que sa disponibilité au sein de l'environnement I2P de partage de fichiers. Nous permettons aux utilisateurs d'I2P d'accéder anonymement au contenu public de BitTorrent et nous aboutissons ainsi à un environnement de partage de fichiers totalement anonyme, indexation et distribution du contenu comprises. Nous centrons ensuite notre analyse sur une meilleure compréhension des environnements anonymes par le biais d'une surveillance à grande échelle. Nous définissons les caractéristiques du réseau I2P, en visant l'intégralité de l'environnement anonyme et son large champ d'activités et de services anonymes. Nous analysons les différents aspects du réseau, comme le nombre des utilisateurs, leurs caractéristiques ainsi que le nombre de services anonymes disponibles. Au travers des mesures et évaluations réalisées à long terme sur le réseau, couplées à différentes analyses de corrélation, nous avons réussi à identifier et caractériser un petit groupe d'individus exécutant un service anonyme spécifique comme, par exemple, les habitants d'une même ville utilisant une application anonyme de partage de fichiers.

Chapitre 1

Introduction

Sommaire

1.1 Contexte

Nous évoluons chaque jour dans l'ère de l'électronique et du numérique. La plupart de nos activités quotidiennes sont aujourd'hui rythmées et régies par Internet, qu'il s'agisse de partage de fichiers, d'interaction sur les réseaux sociaux, de banques et de cours en ligne, ou encore de publication de blogs, ces exemples ne constituant qu'un très mince échantillon des innombrables possibilités offertes par la toile. En ce qui concerne le partage de fichiers, qui reste sans aucun doute l'une des activités les plus pratiquées sur Internet, diverses solutions ont déjà été proposées, créées et implantées, pour constituer ainsi un large ensemble de réseaux de partage de fichiers aux performances et objectifs parfois très différents. Ces réseaux ont évolué au fil du temps, passant d'une approche plus centralisée à des systèmes aujourd'hui totalement décentralisés.

Le réseau BitTorrent, d'une part, utilisait des *trackers* (traqueurs)[1] basés sur le protocole de contrôle de transmissions TCP pour coordonner des pairs partageant différents fichiers. Le système a ensuite évolué vers le protocole UDP afin de réduire la charge du système. Au final, c'est toute l'architecture de BitTorrent qui a été récemment modifiée jusqu'à être totalement décentralisée avec l'emploi de tables de hachage distribuées (DHTs) à la place des traqueurs. Un utilisateur d'Internet peut ainsi accéder à un large éventail de contenu simultanément avec des dizaines de millions d'autres utilisateurs à travers le monde, et ce, via le protocole de téléchargement de BitTorrent, tout en utilisant un mécanisme d'indexation du contenu *off-band* comme des liens magnets[2] pour interroger la DHT.

De son côté, le réseau eDonkey s'est également tourné vers une architecture complétement décentralisée, où une DHT rend possible un mécanisme de double indexation permettant une recherche de fichiers à partir de mots clés. L'algorithme de téléchargement Ed2K demeure néanmoins moins étudié.

Cette ère du numérique s'inscrit également sous l'angle de la surveillance et du profilage puisque chaque information envoyée - ou reçue, par un internaute va passer par une série de nœuds

1. Entités fournissant une liste de sources pour les fichiers recherchés.
2. Format permettant de désigner et référencer les fichiers à télécharger.

intermédiaires jusqu'à atteindre sa destination. Même s'il s'avère possible de ne pas pouvoir détecter le passage d'une information au niveau d'un nœud intermédiaire, un observateur pourra toujours en déterminer sa provenance et sa destination et ainsi surveiller l'utilisateur en identifiant le serveur d'où il s'est connecté mais aussi la durée de sa session, pour finalement être en mesure de pouvoir dresser un *profil* des activités *online* de cet utilisateur. Parallèlement et pour contrer ce phénomène se sont développés des aires et des systèmes de communication anonymes, créés pour permettre à leurs utilisateurs de pouvoir s'adonner à leurs activités sur Internet sans que leur identité, comme par exemple leur adresse IP, ne soit nécessairement révélée au cours du processus. Les systèmes anonymes sont devenus une caractéristique intrinsèque à notre mode de communication virtuelle actuel, renforcés par une vague de surveillance politique [?, ?, ?, ?, ?], de lois floues sur le respect de la vie privée [?, ?] et de divers cas de violation de cette même vie privée lors d'activités en ligne. Au final, les internautes se sont sensiblement tournés de plus en plus vers ces modes de communication anonymes.

Notre étude s'articule autour de deux axes : d'un côté le large éventail d'architectures de partage de fichiers existant, leurs performances et la manière avec laquelle elles peuvent être exploitées ; de l'autre, cette communauté anonyme grandissante sur Internet et la manière avec laquelle elle peut être contrôlée.

1.2 Problématique

Nous évaluons tout d'abord différentes architectures de partage de fichiers et leurs propriétés. Les réseaux les plus populaires, comme BitTorrent ou KAD/Ed2k, ont été optimisés pour répondre des objectifs différents. Le réseau KAD/Ed2k présente un excellent système d'indexation du contenu basé sur Kadmelia, tandis que l'algorithme de téléchargement BitTorrent est connu pour sa haute performance. Notre objectif consiste ainsi trouver un moyen de connecter ces deux réseaux en exploitant les points forts de chacun d'entre eux.

La volonté exprimée de nos jours par un certain nombre d'utilisateurs de pouvoir communiquer au sein d'un environnement respectant la vie privée a conduit la création de plusieurs communautés fermées mais aussi l'émergence de réseaux dédiés au partage de fichiers anonyme. Si ces systèmes ont été améliorés pour préserver l'anonymat de leurs utilisateurs, ils présentent l'inconvénient de ne pas pouvoir accéder des communautés partageant publiquement du contenu, tel que le réseau BitTorrent. Ces deux facteurs nous amènent au premier point de notre problématique, savoir l'étude et l'analyse d'architectures hybrides de partage de fichiers afin de pouvoir tirer parti de certaines propriétés de différents réseaux actuels de partage de fichiers pour construire un environnement de partage plus solide.

Nous centrons ensuite notre analyse sur les réseaux anonymes, qui n'ont cessé de prendre de l'importance[3] et au sein desquels de plus en plus de services deviennent disponibles. Une caractérisation détaillée de ces systèmes est nécessaire pour nous permettre d'en comprendre l'usage réel dans les communications d'aujourd'hui, de déterminer si ces systèmes sont largement déployés ou utilisés par un certain type d'utilisateurs seulement et d'analyser leur évolution dans le temps. Ce qui constitue donc ici notre challenge est de pouvoir analyser ces systèmes sans compromettre l'anonymat de leurs utilisateurs, nous amenant directement au second point de notre problématique, à savoir une caractérisation correcte et détaillée des environnements anonymes.

3. Le nombre d'utilisateurs du réseau Tor a triplé au cours de ces 3 dernières années, pendant que celui du réseau I2P a doublé durant la seule année passée. Statistiques à `https://metrics.torproject.org` et `http://stats.i2p.in/`.

Chapitre 2

Contributions

Sommaire

2.1 Amélioration de l'indexation du contenu dans l'environnement de partage de fichiers BitTorrent

Des études antérieures sur Internet [?, ?] ont montré que la proportion d'applications de partage de fichiers pair-à-pair (P2P) était considérable en terme de trafic Internet. La situation a récemment évolué en faveur des services de *streaming* tels que YouTube ou Netflix [?]. Malgré cette tendance, 46% du trafic Internet en Europe et en Amérique du Nord sont toujours générés par les applications de partage de fichiers P2P, qui sont, par conséquent, toujours prédominantes.

Parmi ces applications de partage de fichiers P2P, BitTorrent demeure la plus importante et la plus active, classée au premier rang grące à ses dizaines de millions d'utilisateurs simultanés et ses millions de *torrents*[4] disponibles. Ces dernières années, le système a évolué vers une approche totalement distribuée avec l'introduction de tables de hachage distribuées pour remplacer les traqueurs. Ces DHTs, basées sur le protocole Kademlia[?], offrent un unique niveau d'indexation, reliant les *torrents* à leurs pairs et permettant ainsi aux utilisateurs de trouver les sources nécessaires au téléchargement d'un contenu particulier. Une des failles de la décentralisation réside néanmoins dans la réduction du niveau de sécurité avec différents types d'attaques qui sont ainsi rendus possibles au sein des DHTs. Le réseau eDonkey (ed2k) se classe en seconde position dans le classement des applications de fichiers P2P les plus utilisées, représentant un tiers de toute l'activité de partage de fichiers P2P (les deux autres tiers étant réalisés par BitTorrent). En 2004, le principal client du réseau eDonkey, à savoir eMule, a présenté un nouveau réseau P2P totalement décentralisé appelé Kad, conçu pour être compatible avec eDonkey. Son architecture totalement distribuée et la nature "open source"de ses clients (eMule et aMule) ont fait de Kad un réseau largement étudié et par là même amplement amélioré.

4. Metadonnées sur les fichiers à télécharger.

Nous pensons que l'application BitTorrent pourrait aisément profiter des performances et des fonctions de sécurité de Kad pour l'indexation du contenu et que Kad pourrait à son tour bénéficier des performances de l'algorithme de téléchargement de BitTorrent ; les deux réseaux tirant ainsi profit d'une possible collaboration.

2.1.1 Comparaison des DHTs

Les deux DHTs à l'étude, à savoir Mainline et Kad, sont ici évaluées sous deux angles différents. Nous devons d'une part déterminer leur niveau de sécurité face à différentes attaques, mais aussi établir leurs performances en termes de temps de publication et du nombre de messages échangés.

Comparaison du niveau de sécurité

Nous étudions tout d'abord le niveau de protection de la DHT Mainline contre une attaque de la table de routage, qui constitue la base pour de futures attaques plus complexes. Nous décrivons ensuite les mécanismes de protection mis en place dans la DHT Kad pour empêcher les attaques dans le réseau.

Afin d'évaluer le niveau de protection de la DHT Mainline, nous considérons l'attaque décrite par Steiner *et al.* [**?**], et communément appelée "empoisonnement de la table de routage". Nous laissons d'abord le nœud pair cible atteindre un nombre stable de contacts, normalement autour de 170. Nous lançons ensuite l'attaque, durant laquelle 160 requêtes sont envoyées simultanément. Il en résulte un remplissage de la table de routage du nœud pair cible avec presque entièrement des faux contacts, dont le nombre total est d'environ 310. Cholez *et al.* [**?**] a mené une étude sur les trois mécanismes de sécurité de Kad, mécanismes qui ont été progressivement introduits dans les différentes versions. Chaque nouveau mécanisme de sécurité vise la table de routage de Kad et rajoute des restrictions lors de l'ajout de nouveaux nœuds pairs. Lorsque le nœud pair cible active ces 3 mécanismes de protection, aucun faux contact ne peut être placé dans sa table de routage. La DHT Kad présente ainsi une barrière de protection permettant à un nœud de résister à une attaque certes simple, mais très efficace, d'empoisonnement de sa table de routage, contrairement à la DHT Mainline, qui est largement vulnérable face à ce même type d'attaque.

Comparaison des performances

Même si les deux DHTs sont basées sur Kademlia, une certaine liberté dans leur mise œuvre a conduit à d'importantes variations de performance, comme déjà démontré par Crosby et Wallach [**?**]. Pour notre analyse, nous nous concentrons sur trois caractéristiques :

- ***Temps de publication*** **:** combien de secondes sont-elles nécessaires à la publication d'une valeur dans la DHT ?
- **Charge réseau pendant la publication :** combien de messages sont-ils nécessaires à la publication d'une valeur dans la DHT ?
- **Durée de vie de l'information stockée :** combien de temps une valeur précédemment stockée dans la DHT va-t-elle durer ?

Les deux DHTs ont un comportement similaire lorsqu'il est question des temps de publication, avec environ entre 30 et 40 secondes pour stocker une valeur. La DHT Mainline semble a priori plus rapide de quelques secondes. Cependant, Kad dispose d'une fenêtre de 3 secondes entre l'extraction des pairs les plus proches et la demande de publication, fenêtre dont la durée pourrait être ramenée à une demi-seconde sans affecter l'algorithme de routage, comme l'ont démontré Steiner *et al.* [**?**].

Pour chaque publication, la DHT Kad génère en moyenne entre 25 et 30 messages, tandis que ce nombre monte jusqu'à 40 messages pour la DHT Mainline. Il est important de noter que ces valeurs incluent les messages de routage ainsi que les messages de service. Les messages de routage permettent de localiser les pairs les plus proches et, une fois ces pairs localisés, un message de service précis est émis. Comme un seul et unique message de service est envoyé, la différence entre le nombre de messages nécessaires aux deux DHTs provient donc des messages de routage et, par là même, la surcharge réseau durant la publication s'explique par le protocole de routage de la DHT Mainline.

Afin d'évaluer la durée de vie de l'information dans les deux DHTs, nous avons conservé l'ensemble de pairs où nous avions initialement publié des milliers de valeurs aléatoires tout en vérifiant périodiquement si ces pairs/nœuds étaient toujours actifs. Pour la DHT Kad et après les 30 premières minutes, nous avons constaté que 84% des nœuds étaient toujours actifs ; ce chiffre baissant jusqu'à 50% après une durée de cinq heures. Vingt-quatre heures plus tard, seulement 28% des nœuds étaient accessibles. La DHT Mainline présente un taux de déperdition plus important, avec 41% de nœuds actifs après 30 minutes et seulement 9% après 24 heures.

La DHT Kad surpasse ainsi la DHT Mainline en termes de sécurité et de performance. La table de hachage distribuée Kad, qui met en jeu trois mécanismes de protection, permet d'éviter des attaques diverses et variées telles que l'empoisonnement de la table de routage. De plus, elle requiert moins de messages de routage lors de la publication d'une valeur et présente un taux de *churn* moins important.

2.1.2 Algorithmes de téléchargement de BitTorrent et Ed2k

Les caractéristiques de la DHT Kad en matière de performance, alliées à ses fonctions de sécurité, en font une excellente table de hachage distribuée. Cependant, l'aspect performant de cette table est éclipsé aux yeux de l'utilisateur par l'efficacité de l'algorithme de téléchargement. L'utilisateur est, en effet, plus sensible à la durée globale de téléchargement d'un fichier, indépendamment de l'algorithme d'indexation. Afin de pouvoir évaluer l'efficacité des algorithmes de téléchargement de Kad/Ed2k et de BitTorrent, nous avons mesuré le temps nécéssaire pour télécharger un fichier de 700 MB. Nous avons, pour notre expérience, utilisé le banc d'essai PlanetLab, avec 50 nœuds distribués et un même scénario répété 10 fois. Nous avons conservé un schéma classique de téléchargement dans l'environnement du partage de fichiers, à savoir le téléchargement d'un seul fichier à la fois.

Temps de téléchargement avec un seule *seeder*

Commencer avec un seul et unique *seeder* (fournisseur) semble logique, puisque le pair d'origine se définit au départ comme étant la seule source, pour être éventuellement rejointe ensuite par de nouveaux pairs qui vont compléter la procédure de téléchargement et ainsi devenir de nouveaux *seeders*. Nos expériences ont montré que les clients BitTorrent ont pu télécharger leur fichier dans son intégralité en 315 minutes, tandis qu'il aura fallu environ 7 heures de plus, à savoir 745 minutes au total, aux clients Kad pour télécharger le même fichier. L'algorithme de téléchargement de BitTorrent est ainsi plus performant et permet d'effectuer, en moyenne, un téléchargement en 42% du temps requis par Kad/Ed2k. Comme le nombre de fournisseurs augmente très rapidement lorsqu'il s'agit d'un contenu partagé populaire, nous avons mené une seconde expérience mettant en jeu 10 sources initiales afin d'évaluer le temps de téléchargement finalement requis.

Temps de téléchargement avec dix *seeder*

Bien que la normalité consiste en un seul fournisseur de fichiers à partager, les nouveaux contenus tendent à être diffusés très rapidement et, avec eux, l'apparition logique de nouveaux "seeders". Nous avons donc établi que 10 fournisseurs initiaux pouvaient être à l'origine du partage d'un contenu. Dans ce cas précis, le téléchargement des clients BitTorrent a pris 224 minutes, tandis que celui des clients Kad/Ed2k aura pris 395 minutes au total. Les clients BitTorrent n'auront ainsi eu besoin que de 57% du temps nécessaire à ceux de Kad/Ed2k pour télécharger leur fichier. Considérant les résultats de ces expériences, il apparaît que l'algorithme de téléchargement BitTorrent surpasse clairement celui de Kad/Ed2k, qu'il s'agisse du téléchargement d'un seul fichier pour lequel 2% des nœuds/pairs sont *seeders* (1 nœud fournisseur sur 50) mais également dans le cas où 20% des nœuds/pairs sont *seeders* (soit 10 nœuds sur 50). Dans les deux cas, le temps de téléchargement pour les clients BitTorrent ne représente que 50% de celui pour les utilisateurs de Kad/Ed2k.

Nous avons évalué les algorithmes de téléchargement Ed2k et BitTorrent et démontré que l'algorithme BitTorrent est plus performant que l'algorithme Ed2k lorsqu'il s'agit de télécharger un fichier unique. Nous nous proposons ici de créer un modèle hybride en exploitant le meilleur des deux réseaux, avec une indexation du contenu réalisée par la DHT Kad, tandis que la publication du contenu sera réalisée via l'algorithme de téléchargement BitTorrent.

2.1.3 Un modèle de partage de fichiers hybride BitTorrent & KAD/Ed2k

Notre précédente étude a montré que la DHT Kad offre plus de sécurité et met en jeu différents mécanismes de protection qui la protège d'attaques complexes, contrairement à la DHT Mainline, vulnérable face à une simple attaque d'empoisonnement de la table de routage. Cependant, pour le téléchargement d'un fichier unique, l'algorithme de téléchargement BitTorrent a permis un gain de temps de près de 50% comparé à l'algorithme Ed2k. Sur la base de ces résultats, nous avons envisagé une approche combinée : le graphique 2.1 représente notre modèle d'interconnexion, créé avec les deux réseaux BitTorrent et Kad/Ed2k, où la couche d'interconnexion est formée par des nœuds hybrides appelés hMule.

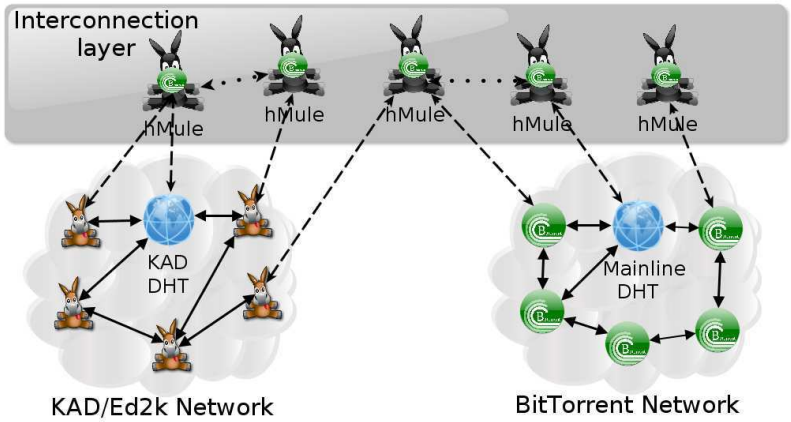


Figure 2.1 – Instanciation du modèle hybride avec les réseaux BitTorrent et Kad/Ed2k.

Le client hMule est le point de rencontre des deux réseaux et il est capable de rechercher du contenu BitTorrent dans la DHT Kad avec des mots clés différents. Une fois le contenu trouvé,

il est téléchargé par le biais de l'algorithme de téléchargement BitTorrent. De cette manière, n'importe quel utilisateur de hMule peut bénéficier d'un mécanisme d'indexation sécurisé basé sur des mots clés tout en utilisant l'algorithme de téléchargement BitTorrent. En plus de pouvoir exploiter les deux réseaux, les clients hMule sont "retro-compatibles" : ils peuvent se connecter à des clients BitTorrent classiques et télécharger du contenu sur le seul réseau BitTorrent, mais aussi se connecter à des clients Kad/Ed2k et télécharger du contenu provenant de clients aMule/hMule.

Le nœud/client hMule est le point d'interconnexion entre les réseaux BitTorrent et Kad/Ed2k, améliorant l'indexation de contenu au sein de l'environnement BitTorrent. Les utilisateurs sont tenus à l'utilisation de ce client hybride pour pouvoir profiter de notre mécanisme d'indexation amélioré tout en pouvant toujours accéder aux réseaux BitTorrent et Kad/Ed2k classiques, avec pour résultat un client totalement retro-compatible.

2.1.4 Synthèse

D'une part, l'environnement BitTorrent a évolué vers une architecture totalement décentralisée, où des DHTs basées sur le système d'indexation à un seul niveau Kadmelia sont utilisées, jouant le même rôle qu'un traqueur mais de manière distribuée. D'autre part, Kad/Ed2k présente une architecture de partage de fichiers pair-à-pair totalement décentralisée, au sein de laquelle la DHT Kad avec son mécanisme de double indexation offre un service de recherche par mot-clé ainsi qu'une indexation de contenu distribuée.

Au travers de nos expérimentations, nous avons mis en évidence les failles et carences de la DHT principale de BitTorrent en matière de mécanismes de protection, la rendant vulnérable face à des attaques basiques, tout en montrant, à l'inverse, que la DHT Kad pouvait, elle, résister à ce même type d'attaques. Après avoir évalué ensuite les performances des algorithmes de téléchargement en se plaçant dans un environnement réel de chargement d'un fichier unique, nous avons pu montrer que l'algorithme de téléchargement BitTorrent était approximativement deux fois plus rapide que celui de Ed2k. C'est la raison pour laquelle nous avons proposé une application de partage de fichiers hybride qui allient les performances de BitTorrent en matière de rapidité à la résistance de Kad du point de vue sécurité.

2.2 Amélioration de la disponibilité du contenu dans l'environnement de partage de fichiers I2P

Les communications anonymes sont en constante augmentation et l'Internet est en train d'évoluer vers un environment *privacy-aware*. Les utilisateurs prennent conscience de l'importance de maintenir un certain degré d'anonymat lorsqu'ils naviguent sur le Web afin de séparer leurs opinions en-ligne de leur réelle identité. Le partage de fichiers pair-à-pair occupe une grande partie des communications réalisées anonymement sur Internet. Il est important pour un utilisateur de maintenir son identité secrète afin d'éviter la censure ou le profilage pendant le téléchargement de données en ligne. Malgré le large éventail d'options de partage de fichiers anonymes, le contenu disponible dans ces systèmes est réduit et souvent obsolète ; les réseaux publics restent encore la principale source de contenu. Par conséquent, l'enjeu devient d'accéder au contenu public anonymement, soit par l'introduction systématique des nouveaux contenus publics dans les environnements anonymes, soit en permettant aux utilisateurs anonymes d'accèder à ce contenu en préservant leur anonymat.

2.2.1 Disponibilité du contenu dans le réseau I2P

Les contenus populaires sont en premier lieu disponibles dans les réseaux publics avant de l'être dans les réseaux anonymes. Inévitablement, la quantité de contenus disponibles dans les environnements de partage de fichiers anonymes est significativement faible comparée à celle des réseaux publics. Nous avons procédé à des prises de mesure s'étalant sur trente jours pour déterminer le taux de nouveaux contenus introduits dans le réseaux BitTorrent et dans le réseaux de partage de fichiers I2P. Par le biais de ces expériences, nous avons pu observer une moyenne de 720 nouveaux *torrents* ajoutés quotidiennement dans Torrentz[5] contre environ 8 dans le traqueur Postman[6]. Nous avons également remarqué des disparités concernant l'évolution de ces taux d'ajouts dans ces deux réseaux. En effet, Torrentz présente un taux d'ajouts relativement stable au cours du temps alors que des pics d'ajouts peuvent être observés sur Postman, généralement à l'approche des week-ends. Torrentz compte plus de 19 millions de *torrents* actifs, tandis que Postman en compte environ 12000 et seulement 1% des *torrents* disponibles dans Torrentz sont aussi présents dans le réseau I2P. Malgré les efforts des différents utilisateurs, le manque de nouveaux contenus dans le réseau de partage de fichiers I2P reste un fait avéré, appuyant ainsi notre idée d'instaurer un mécanisme *on-demand* pour accéder aux contenus publics.

2.2.2 Interconnexion entre les réseaux I2P et BitTorrent

L'ensemble des contenus disponibles dans le réseau de partage de fichiers I2P est de taille réduite. Pour cette raison, nous présentons un schéma d'interconnexion entre le réseau BitTorrent et le réseau I2P, visant à améliorer la mise en disponibilité des contenus dans ce dernier. Ce schéma permet en effet de mettre en place un systême d'indexation et de distribution des contenus de façon totalement anonyme. La figure 2.2 présente l'ensemble des composants de notre modèle d'interconnexion, où la couche d'interconnexion est formée par l'intermédiaire de nœuds hybrides appelés *BiTIIP*. Les clients BiTIIP permettent à un *swarm* (essaim)[7] I2P d'interagir avec un *swarm* BitTorrent, et de former ainsi un *swarm* unique partageant un même contenu et indexé dans la DHT Mainline de I2P. Cette indexation est anonyme et distribuée par le réseau I2P, conduisant donc à une distribution anonyme du contenu partagé.

L'anonymat dans la couche d'interconnexion

Nous définissons une interaction entre environnements anonymes et non anonymes dans laquelle nous souhaitons maintenir l'anonymat des utilisateurs issus du réseau I2P. Le réseau I2P emploie des tunnels unidirectionnels pour acheminer anonymement les communications entre ses utilisateurs. Chaque utilisateur définit la longueur de ses propres tunnels, et donc son propre compromis entre anonymat et performance : des tunnels courts sont plus performants que des tunnels longs (moins de chiffrement), mais rendent l'anonymat plus faible.

Les clients BiTIIP n'ont pas besoin eux même d'anonymat. Ils ont effet une interface sur le réseau public et se connectent directement aux clients BitTorrent. De ce fait, ils utilisent des tunnel à *0-hop* pour se connecter aux utilisateurs I2P privilégiant ainsi l'aspect performance. Par contre, l'anonymat des utilisateurs I2P est préservé car ils établissent avec les clients BiTIIP des tunnels à plusieurs hops (3 par défaut).

5. Torrentz est un méta-moteur de recherche pour le contenu de BitTorrent, qui indexe les *torrents* de divers sites, dont The Pirate Bay et Bitsnoop.
6. Postman est le traqueur le plus important dans le réseau I2P.
7. Ensemble de tous les pairs qui partagent le même torrent.

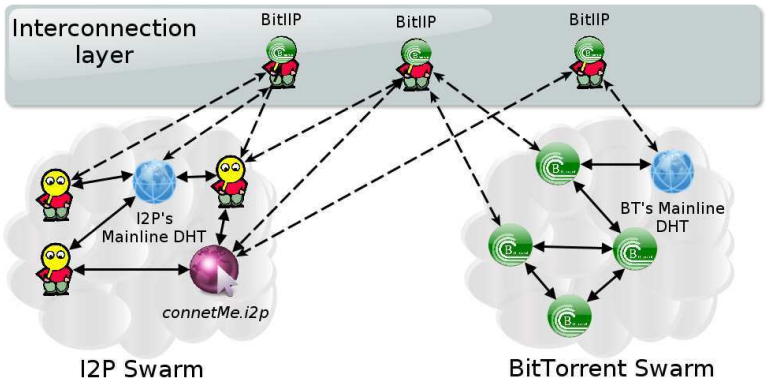


Figure 2.2 – Instanciation de notre modèle hybride avec les réseaux BitTorrent et I2P.

Du point de vue d'un attaquant, il est seulement possible de lier un téléchargement public à un client BiTIIP particulier. Même si un client BiTIIP malicieux est intégré dans la couche d'interconnexion, il ne ne pourra pas désanonymiser un utilisateur I2P.

2.2.3 Performance du client BiTIIP

BiTIIP est un client développé en Java qui utilise la bibliothèque BitTorrent Snark [8] et la bibliothèque de partage de fichiers I2PSnark. La bibliothèque Snark permet au client BiTIIP d'interagir avec les clients BitTorrent, alors que la bibliothèque I2PSnark lui permet de communiquer avec d'autres utilisateurs I2P. BiTIIP instaure un interfaçage entre ces deux bibliothèques, synchronisant ainsi les demandes d'un réseau à l'autre, tout en effectuant la mise en cache des *pieces* (parties d'un fichier) de façon opportuniste. Sur la base de nos expérimentations, un client BiTIIP unique atteint une vitesse de téléchargement de 17 Kbps en moyenne. Chaque client BiTIIP supplémentaire augmente ce débit de façon proportionnelle. Il est donc tout à fait possible d'ajouter de nouveaux clients BiTIIP à une interconnexion existante afin d'augmenter la performance globale de cette liaison, notamment lors des téléchargements.

2.2.4 Synthèse

Nous avons montré qu'à peine 1% des contenus disponibles dans le réseau BitTorrent le sont aussi dans le réseau I2P. Il s'avère de plus que le réseau I2P contient principalement des fichiers obsolètes. Ce constat nous a poussé à rendre possible l'accès aux contenus disponibles dans BitTorent depuis I2P. Le modèle hybride que nous proposons dans cette optique permet aux utilisateurs I2P d'accéder à ces contenus publics tout en préservant leur anonymat, menant ainsi à une indexation et une distribution des contenus publics de façon totalement anonyme. Nous avons validé notre modèle par l'intermédiaire d'un client hybride appelé BiTIIP. Ce client offre des performances satisfaisantes en termes de débit comparées à celles d'un client standard dans le réseau I2P, et permet également de préserver l'anonymat de ses utilisateurs. Un client BiTIIP se présente comme un point de liaison entre les utilisateurs I2P et ceux de BitTorent en offrant une connexion bidirectionnelle. Par l'usage de tunnels propres aux communications I2P, un utilisateur du réseau anonyme peut contacter un client BiTIIP et ainsi accéder aux contenus publics de BitTorrent, sans pour autant mettre en péril son anonymat.

8. Le projet Snark est disponible sur `https://code.google.com/p/snark/`.

2.3 Caractérisation de l'environnement anonyme I2P

Une analyse de suivi à grande échelle d'une communauté de partage de fichiers fournit une vue d'ensemble du réseau [**?**, **?**, **?**], permettant de déterminer le type de contenu distribué au sein de l'environnement anonyme, le nombre d'utilisateurs partageant des fichiers mais aussi le type d'utilisateurs téléchargeant un contenu spécifique. La surveillance et l'étude à grande échelle des réseaux et environnements anonymes n'ont cependant jamais été menées en profondeur jusqu'alors, et se sont focalisées principalement sur le réseau Tor [**?**, **?**]. Nous présentons ici la première architecture de supervision de l'environnement anonyme I2P et analysons l'utilisation faite de ce type de réseaux, ainsi que le profil de ses utilisateurs. Nous sommes ainsi parvenus à identifier la plupart des applications I2P anonymes, y compris les applications anonymes de partage de fichiers. Nous avons aussi déterminé leur usage au cours du temps sur le réseau et géo localisé leurs utilisateurs.

2.3.1 La netDB d'I2P

Le réseau I2P permet à deux entités de communiquer anonymement par le biais d'une couche d'abstraction qui brise l'association entre l'utilisateur d'une application et son identité. Les utilisateurs d'I2P sont en mesure de déployer des applications au dessus du réseau I2P. Ils peuvent être contactés au moyen d'un identifiant indépendant d'une quelconque localisation et désigné par le terme *destination* ; l'anonymat de l'utilisateur ayant à l'origine déployé ces applications restant ainsi préservé. Les informations relatives à un nœud I2P sont représentées par une *routerinfo* et les informations relatives à une destination sont représentées par un *leaseset*. Ces informations sont stockées dans la table de hachage distribuée I2P, appelée *netDB*. La netDB est une DHT basée sur Kadmelia et formée par un sous-groupe de pairs I2P appelés *pairs floodfill*, bien intégrés au réseau, dont la bande passante disponible est supérieure à 256 KB/s. Le nombre connu de pairs floodfill est d'environ 500 ; n'importe quel pair I2P standard pouvant automatiquement devenir un nœud floodfill. Cette approche distribuée, couplée à une décision locale des pairs I2P de devenir des nœuds floodfill, renforce le réseau I2P qui, n'étant pas caractérisé par la présence d'un élément central principal, contrairement au répertoire principal de Tor, ne présente pas un point unique de vulnérabilité.

2.3.2 Exploitation de la netDB d'I2P

Nous exploitons la netDB en tirant parti de son mode de fonctionnement afin de recueillir le plus de métadonnées I2P possible. N'importe quel pair I2P pouvant se présenter comme un nouveau nœud floodfill, nous avons donc déployé un ensemble de nœuds floodfill au sein de la netDB afin de procéder à une surveillance passive, pour laquelle nos nœuds floodfill se comportent comme des sondes distribuées, ciblant différentes zones de la netDB. Ces nœuds floodfill "espions" se comportent comme des nœuds floodfill standard, venant s'ajouter à d'autres nœuds floodfill pour former la netDB et stockant des métadonnées I2P. Nous avons ainsi pu collecter une grande quantité de métadonnées réseau, les analyser et établir les caractéristiques du réseau I2P. Les *routerinfos* sont utilisées pour géolocaliser les nœuds I2P et analyser leur comportement en ligne au sein du réseau I2P. Les *leasesets* sont utilisés pour identifier les applications déployées et déterminer leur durée de fonctionnement au sein du système.

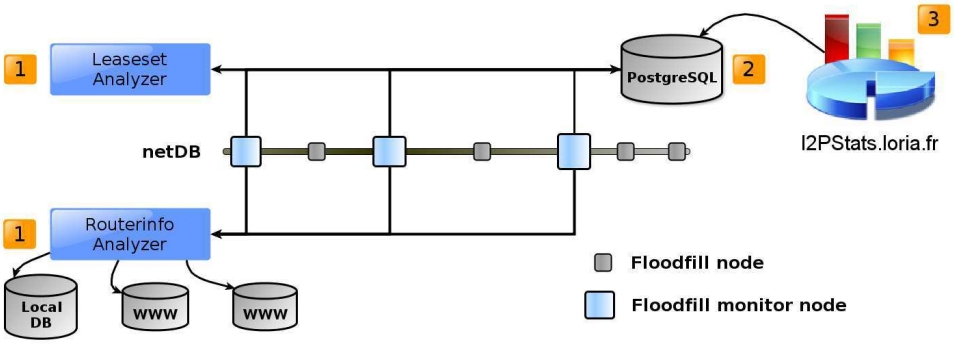


Figure 2.3 – A passive distributed monitoring architecture for the I2P network.

2.3.3 Architecture de surveillance

Nous avons détaillé la manière avec laquelle nous avons exploité avec succès la netDB d'I2P Nous avons proposé une architecture de supervision distribuée passive, dans laquelle des nœuds de surveillance sont distribués au sein du réseau tout en y restant indétectables puisqu'ils se comportent de la même manière que des nœuds standard. Le graphique 2.3 représente notre architecture : un ensemble de nœuds de surveillance floodfill sont placés au sein de la netDB afin de collecter des routerinfos et des leasesets I2P. Une fois ces métadonnées analysées, les résultats sont stockés dans une base de données centrale. Ils sont ensuite regroupés et affichés sur un site web statistique.

Le déploiement des nœuds de surveillance floodfill est totalement flexible puisque ces nœuds peuvent être ajoutés de manière dynamique à la netDB, augmentant ainsi la quantité de métadonnées réseau pouvant être extraites et analysées. Nous nous sommes ensuite intéresses aux moyens d'analyse des routerinfos et des leasesets, ainsi qu'au nombre de nœuds de surveillance floodfill nécessaires à une couverture totale du réseau.

Analyse des Routerinfos et Leasesets

Une *routerinfo* permet d'identifier un nœud I2P au sein du réseau grâce à ses coordonnées de contacts. Elle est analysée dans le but de d'obtenir la localisation géographique du pair I2P qu'elle représente en utilisant différents services de géolocalisation. En comparaison, analyser un *leaseset* requiert une procédure plus complexe. Une *destination* est une représentation I2P d'une adresse IP et d'un numéro de port, par le biais de laquelle nous pouvons envoyer des messages TCP ou UDP. En fonction des réponses reçues, nous essayons de déterminer quelle application est représentée par tel *leaseset*, comme par exemple une application de partage de fichiers, une application Messenger ou un site web anonyme.

Déploiement des nœuds de surveillance floodfill

Il est important de prendre en compte la couverture réseau de notre architecture de supervision. Pour calculer le nombre de nœuds de surveillance floodfill requis dans la netDB, nous considérons le nombre total de nœuds floodfill dans la netDB et le *replica factor* (facteur de réplication) . Ce dernier nous indique le nombre de nœuds où une valeur va être stockée. L'équation 2.1 représente le nombre minimum de nœuds de surveillance floodfill requis pour assurer une couverture complète du réseau, avec N le nombre total de nœuds floodfill et X la variable

actuelle du facteur de réplication. Pour notre netDB en cours d'analyse, avec $N = 500$ et $X = 5$, 100 nœuds de surveillance floodfill sont nécessaires à une couverture totale du réseau.

$$\texttt{nb\_monitors} = \lceil\ \texttt{N / X}\ \rceil,\ \texttt{N= \#floodfills , X= replica factor} \tag{2.1}$$

2.3.4 Le réseau I2P en temps réel

Afin de procéder à une évaluation de notre architecture de surveillance, nous présentons les résultats obtenus au cours d'une période de 6 jours, du 18 juin 2013 12h00 CEST au 24 juin 2013 12h00 CEST, avec un total de 70 nœuds de surveillance floodfill déployés sur le reseau PlanetLab (nombre maximal possible dans le cadre de l'expérimentation).

Détection des utilisateurs d'I2P

Nous avons pu détecter en moyenne presque 28 000 utilisateurs quotidiens. Nous avons observé des pics récurrents autour de 18h00, nous indiquant que les utilisateurs d'I2P sont plus actifs en fin d'après-midi - heure CEST - avec 31 000 utilisateurs en moyenne. A l'inverse, nous avons constaté une chute quotidienne du nombre d'utilisateurs autour de minuit CEST, avec des pics inversés tournant autour de 24 000 utilisateurs seulement. Ce phénomène est dû au *shifting mechanism* ou "mise à jour", de la netDB d'I2P, qui se produit tous les jours à minuit. Chaque nœud de surveillance floodfill, ainsi que tout nœud floodfill standard, change de position au sein de la netDB et commence ainsi à recevoir de nouvelles métadonnées, tandis que les métadonnées stockées précédemment ne sont désormais plus valides. Ce phénomène affecte le fonctionnement du réseau pendant une courte période, au cours de laquelle certaines requêtes netDB peuvent échouer, ayant pour conséquence d'affecter également les résultats de notre surveillance autour de minuit. Le comportement des utilisateurs d'I2P présente un profil similaire à celui des réseaux P2P [**?**], décrivant, sur une période de 24 heures, une courbe sinusoïdale dont le pic se situe à 18h00 CEST. En se basant sur le nombre de nœuds floodfill actifs lors de notre surveillance, nous avons pu couvrir presque 70% de tout le réseau I2P avec 70 nœuds de surveillance floodfill. Sur la base de ces résultats, nous estimons donc le nombre réel d'utilisateurs quotidiens d'I2P à une moyenne d'environ 40 000.

Caractérisation à l'échelle du pays

Nous avons détecté un total de 113 433 utilisateurs d'I2P disséminés à travers le monde. De tous ces utilisateurs, 40% proviennent de Russie, nous indiquant une participation considérable des utilisateurs russes au réseau I2P. Nous avons également recensé 159 pays, nous indiquant que le réseau I2P est bel et bien déployé à l'échelle mondiale, comme le montre le graphique 2.4.

Caractérisation à l'échelle de la ville

Nous avons pu comptabiliser 13 547 villes. Nous avons pu constater que 12 villes sur le top des 15 villes étaient des villes russes. De plus, sur les 1108 villes russes officiellement recensées [**?**], nous en avons détecté 813 lors de notre analyse, indiquant que le réseau I2P est utilisé dans presque 75% du pays.

Clients du partage de fichiers anonyme

Nous avons pu observer que I2PSnark était l'application la plus utilisée, avec en moyenne 450 clients. Les clients iMule représentent une part peu significative de l'activité, avec une moyenne

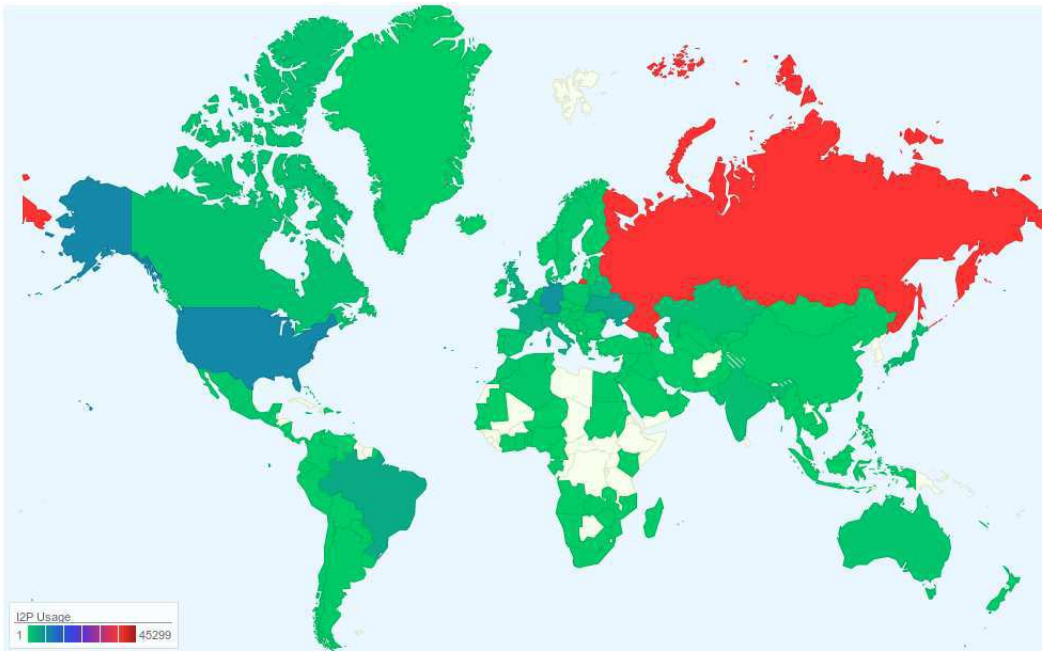


Figure 2.4 – Répartition mondiale des utilisateurs I2P.

descendant parfois jusqu'à 6 clients seulement. Pour finir, I2Phex compte en moyenne 3 clients, avec une légère augmentation d'activité durant les week-ends. Le client I2PSnark est déjà intégré au réseau et, par conséquent, accède directement à l'application, ce qui facilite son utilisation. À l'inverse, les deux autres clients du partage de fichiers anonymes résultent de modifications de clients non-anonyme de partage de fichiers, ce qui rend leur déploiement au sein du réseau I2P plus complexe et son utilisation finalement plus contraignante et donc plus décourageante.

Les serveurs web anonymes

Nous avons pu établir une moyenne de 510 serveurs web anonymes. Nous avons également observé que ces serveurs, pourtant conçus pour fonctionner de manière plus stables que les applications de partage de fichiers, présentent eux-aussi un schéma sinusoïdal. Cela nous indique que les utilisateurs d'I2P hébergeant leurs serveurs web anonymes se connectent et se déconnectent du réseau I2P, tout comme le reste des utilisateurs. Contrairement au programme statistique Tino [9], nous prenons en compte chaque serveur web anonyme actif dans la netDB et fournissons ainsi un cliché précis et en temps réel du réseau, y compris pour des *eepsites* non-référencés.

2.3.5 Synthèse

Nous avons focalisé notre analyse sur I2P et son environnement anonyme, le caractérisant de manière détaillée au travers d'une surveillance distribuée. Avec de l'exploitation de sa base de données distribuée, appelée netDB, nous avons été capables de déployer un ensemble de sondes distribuées passives et de collecter ainsi une large quantité de métadonnées réseau. En analysant avec précision ces métadonnées, nous avons pu déterminer diverses caractéristiques des utilisateurs d'I2P, mais aussi des applications I2P. Nous avons établi que la Russie représentait

9. Accessible à l'adresse `tino.i2p.in`.

quasi 40% de tous les utilisateurs localisés d'I2P et avons pu identifier 75% des villes russes que nous avons détectées. Nous avons également montré que le client I2PSnark, tel un client BitTorrent, manifestait une activité importante sur le réseau, avec une moyenne de 450 clients. Nous avons ensuite évalues le nombre de serveurs web anonymes hébergeant des *eepsites*, qui atteint une moyenne de 510, confrontant ainsi nos résultats au programme statistique Tino, qui n'annonce que 350 sites web anonymes accessibles, pour ainsi montrer qu'un certain nombre de sites web anonymes n'étaient pas référencés dans le réseau I2P. Le facteur de réplication de la netDB d'I2P a évolué suivant le niveau d'exigence de notre surveillance, dans le but d'augmenter le nombre de nœuds nécessaires à une couverture totale du réseau. Initialement de 8 nœuds, il a été réduit à 5. En diminuant le facteur de réplication dans la netDB, les concepteurs I2P ont certes rendu plus difficile la couverture totale du réseau, mais ils ont en contrepartie, affaibli la netDB vis à vis de la résilience au *churn*.

2.4 Caractérisation fondée sur les groupes dans l'environnement anonyme I2P

Nous avons décrit précédemment une architecture de surveillance distribuée de l'environnement anonyme I2P, qui nous a permis de caractériser à la fois des utilisateurs d'I2P mais aussi des applications anonymes au sein du système. Nous présentons la première étape vers une caractérisation de groupe, pour laquelle nous ciblons le réseau anonyme I2P tout entier. Pour ce faire, nous menons une analyse de corrélation basée sur des données collectées par notre architecture de surveillance distribuée, prenant en compte les utilisateurs des villes les plus actives et l'application I2PSnark. Notre objectif est d'évaluer dans quelle mesure ces utilisateurs contribuent à l'activité globale de partage de fichiers sur I2P en nous basant sur le coefficient de corrélation de Pearson.

Comme déjà mentionné, les utilisateurs russes sont très largement représentés parmi les utilisateurs d'I2P et nous pourrions ainsi les considérer comme exclusivement à l'origine de l'activité réseau engendrée par le partage de fichiers anonyme. Cependant, nous ne considérons ici qu'une quantité d'utilisateurs, et non leur comportement au sein du système, et ne pouvons donc pas tirer de conclusion valable de ce simple facteur. En revanche, en tenant compte des comportements respectifs spécifiques des utilisateurs et des applications anonymes, nous pourrons déterminer si les deux données sont liées ou pas et, le cas échéant, évaluer la portée de cette relation.

2.4.1 Cadre de l'expérimentation

Nous avons considéré que l'analyse de la corrélation sur une seule journée ne représenterait pas l'utilisation réelle du réseau. En effet, des particularités présentes un jour peuvent biaiser les échantillons, les cœfficients de corrélation seraient donc sous- ou sur-estimés. Nous avons donc choisi une période de quinze jours incluant trois week-ends pour avoir une bonne fenêtre de temps et détecter une corrélation à long terme entre une ville et les applications I2PSnark. Notre analyse de corrélation utilisent des données provenant du réseau réel I2P du `15/03/2013` au `30/03/2013`.

Au cours de notre période d'analyse, nous établissons un classement des villes suivant le nombre d'utilisateurs détectés, prenant également en compte leur activité journalière durant les 15 jours de notre expérience. Nous avons ainsi, au fil de notre analyse, détecté 16 085 villes au final, dont la plupart ne comptaient finalement pas plus de 10 utilisateurs sur toute la période de 15 jours. Moscou et Saint-Pétersbourg, en revanche, ont présenté un nombre élevé d'utilisateurs

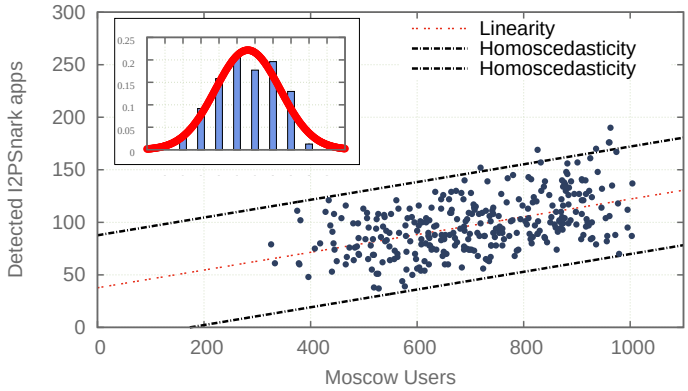


Figure 2.5 – Analyse de Pearson pour Moscow/I2PSnark.

détectés, ceux de Moscou représentant près de 8% du nombre total d'utilisateurs détectés. Ces deux villes contribuant le plus largement à l'activité sur le réseau I2P, nous les avons considérées tout d'abord comme cas d'étude pour notre analyse de corrélation. Nous avons ensuite examiné une ville de moindre importance, pour établir la pertinence et le poids de sa participation, ou non, à l'activité de partage de fichiers détectée.

2.4.2 Analyse de corrélation : Moscou

Le graphique 2.5 représente la distribution des données analysées. Les résultats montrent que les données suivent une distribution normale et décrivent, en outre, une ligne droite, répondant ainsi au critère de linéarité requis pour l'application du cœfficient de Pearson. Nous constatons enfin que la dispersion des données est constante et que l'on peut donc parler d'homoscedasticité, dernier paramètre nécessaire à l'application du cœfficient de Pearson.

Le cœfficient de Pearson a ici une valeur de $r = 0.4901$, indiquant une **forte corrélation**. Ceci nous montre que les utilisateurs d'IP2 moscovites ont hautement contribué à l'activité de partage de fichiers sur le réseau durant la période analysée. Le cœfficient de détermination de $r^2 = 0.2401$, nous indique qu'un quart de la variabilité des applications I2PSnark détectées (c'est-à-dire de l'activité détectée) sur cette période de 15 jours est expliqué par la relation avec les utilisateurs de Moscou.

2.4.3 Analyse de corrélation : Saint-Pétersbourg

Le graphique 2.6 représente la distribution des données analysées. Les résultats montrent que les données suivent une distribution se rapprochant de la normale. Linéarité et homoscedasticité caractérisent l'interprétation des résultats obtenus, avec des mécanismes similaires à ceux observées pour la ville de Moscou.

Pour le cas de Saint-Pétersbourg, nous pouvons établir une **corrélation modérée**, avec un cœfficient de Pearson d'une valeur de $r = 0.3952$. Le coefficient de détermination indique que les utilisateurs de Saint-Pétersbourg expliquent à hauteur de 15,61% l'activité de partage de fichiers (variation des applications I2PSnark) durant notre période d'analyse.

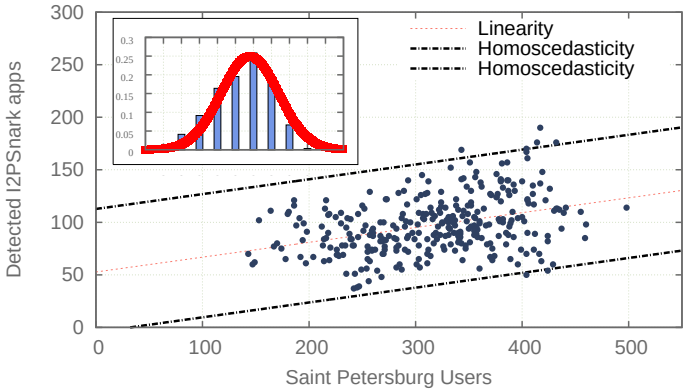


Figure 2.6 – Analyse de Pearson pour Saint Petersburg/I2PSnark.

2.4.4 Analyse de corrélation : Munich

De l'analyse des deux villes les plus actives basée sur le cœfficient de Pearson, nous avons montré que l'on pouvait en extraire, d'une part, une corrélation forte avec les applications de partage de fichiers IP2Snark pour la ville de Moscou et, d'autre part, une corrélation modérée pour la ville de Saint-Pétersbourg. Considérons maintenant une ville telle que Munich : cette ville présente, en terme d'activité, une participation journalière non-négligeable sur le réseau, mais ne représente pourtant que 0,2% du nombre total d'utilisateurs détectés.

Le graphique 2.7 met en avant les propriétés statistiques des données analysées, qui présentent une distribution anormale. De plus, les données présentent un caractère hétéroscedastique : plus la valeur représentant le nombre d'utilisateurs est importante, plus la variance est faible, se rapprochant ainsi de la représentation d'une droite de tendance. Nous ne pouvons, dans ce cas, mettre en application le cœfficient de Pearson, car les caractéristiques de nos données ne répondent pas aux critères requis. Nous pouvons visuellement constater que la majorité des points se concentre entre 14 et 24 utilisateurs, alors que le nombre d'applications détectées varie de 50 à 150. Cette tendance est observable sur notre histogramme, sur lequel apparaissent deux pics pour des intervalles correspondant à un nombre d'utilisateurs compris entre 14 et 18, et 18 et 22 seulement, mais représentant 65% des points de données recueillis, nous indiquant qu'un changement dans le nombre d'applications IP2Snark détectées n'était pas représentatif de, ni lié à, un changement dans le nombre d'utilisateurs basés à Munich. Sur la base de cette analyse visuelle, nous en concluons ainsi que ces utilisateurs ne constituent pas une part significative du partage de fichiers et de l'activité sur le réseau.

2.4.5 Synthèse

Nous avons démontré qu'il était possible d'analyser simultanément le comportement spécifique des utilisateurs d'I2P et de ses applications, tout en caractérisant le réseau I2P. Cette caractérisation de groupe nous a permis d'établir un profil plus précis des utilisateurs d'I2P et de déterminer les groupes qui contribuaient de la manière la plus significative au partage de fichiers et, par-là même, à l'activité sur le réseau I2P.

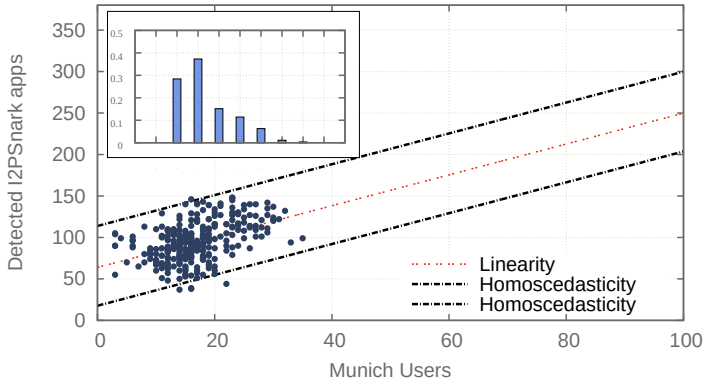


Figure 2.7 – Distribution des données pour Munich/I2PSnark

En appliquant en parallèle le cœfficient de corrélation de Pearson aux données recueillies par notre architecture distribuée, nous avons pu établir une forte corrélation entre la ville la plus représentée sur le réseau I2P, à savoir Moscou, et l'application phare de partage de fichiers de ce même réseau, *i.e.* I2PSnark. Nous avons également mis en avant une corrélation modérée entre les utilisateurs de Saint-Pétersbourg et les clients d'I2P, pour finalement être en mesure de constater que plus de 30% de l'activité relative au partage de fichiers sur le réseau I2P que nous avons détectée lors de la période étudiée est liée aux utilisateurs de ces deux villes.

Chapitre 3

Conclusion

Sommaire

Nous avons présenté nos deux principales contributions. D'une part, nous avons conçu et évalué deux architectures hybrides de partage de fichiers visant à améliorer l'indexation du contenu et la disponibilité de ce dernier dans des environnements de partage de fichiers basés sur BitTorrent. D'autre part, nous avons déployé à grande échelle une architecture de contrôle distribuée qui permet de caractériser le réseau anonyme I2P.

3.1 Environnements hybrides de partage de fichiers

Nous avons étudié les environnements hybrides de partage de fichiers selon deux angles de vue. Premièrement, nous avons amélioré le système d'indexation de contenu dans l'environnement de partage de fichiers BitTorrent via une approche BitTorrent-Kad/Ed2k hybride, en considérant les points forts de chacun des deux réseaux. Dans un second temps, nous avons amélioré la disponibilité de contenu dans l'environnement de partage de fichiers I2P, en permettant aux utilisateurs anonymes d'I2P d'accéder à du contenu BitTorrent à travers une approche à la demande.

Amélioration du l'indexation de contenu dans l'environnement de partage de fichiers BitTorrent

Les applications de partage de fichiers pair-a-pair génèrent, encore, une quantité considérable du trafic Internet. Près de la moitié du trafic européen correspond à du trafic BitTorrent et Kad/Ed2k, ce qui indique que les deux réseaux sont très largement utilisés et déployés.

D'une part, BitTorrent est la plate-forme de distribution de contenu la plus populaire mais reposant sur une base de données décentralisée (DHT) peu sécurisée, qui rend l'ensemble de l'architecture très vulnérable. Le réseau Kad/Ed2k, au contraire, utilise une table de hachage distribuée avec un mécanisme de double indexation offrant plusieurs niveaux de protection. D'autre part, l'algorithme de téléchargement de BitTorrent réduit de près de 50% le temps de téléchargement par rapport à l'algorithme Ed2k, dans le cas d'un téléchargement d'un seul fichier. Par conséquent, nous soutenons que les deux réseaux peuvent tirer parti l'un de l'autre grâce à la réalisation d'un environnement hybride de partage de fichiers.

Nous avons conçu, évalué et mis en œuvre un client de partage de fichiers hybride nommé *hMule*. Ce client est capable d'indexer le contenu BitTorrent dans la DHT Kad, offrant ainsi aux nœuds du réseau BitTorrent la possibilité d'utiliser le service de double indexation de Kad où non seulement les tuples `<content,peers>` sont pris en compte, mais également les tuples `<keyword,contents>`. Cette double indexation introduit un niveau d'indexation supplémentaire dans le réseau BitTorrent, une fonctionnalité absente dans l'ensemble des clients actuels. Le client *hMule* permet également aux utilisateurs de télécharger du contenu en utilisant l'algorithme performant de téléchargement de BitTorrent, combinant ainsi (i) un système robuste d'indexation de contenu et (ii) une excellente plate-forme de distribution de contenu.

Le client *hMule* démontre comment un système d'interconnexion hybride entre deux réseaux de partage de fichiers peut conduire à un environnement amélioré. Le client *hMule* est rétro-compatible ; il a été testé à la fois dans le réseau BitTorrent et dans le réseau Kad/Ed2k. Le projet *hMule* [10] propose aux internautes d'installer et d'utiliser le client *hMule* afin de bénéficier de cette coopération hybride.

Amélioration de la disponibilité de contenu dans l'environnement de partage de fichiers I2P

Les communications anonymes se développent de plus en plus avec des utilisateurs d'Internet soucieux de préserver leur anonymat et leur vie privée. Dans ces communications anonymes, les environnements de partage de fichiers anonymes représentent un domaine important, conduisant à des réseaux dédiés. Le réseau I2P permet ainsi aux utilisateurs d'indexer et de distribuer du contenu anonymement. Toutefois, le contenu disponible via le système I2P est largement réduit, et les réseaux publics non anonymes, tels que le réseau BitTorrent, sont encore les principales sources de contenu.

Nous avons ainsi développé un client de partage de fichiers hybride nommé *BiTIIP*, permettant aux utilisateurs d'I2P anonymes d'accéder aux contenus de BitTorrent publics sans compromettre leur anonymat. Le client *BiTIIP* propose une approche *à la demande*, où les utilisateurs d'I2P précisent le contenu de BitTorrent désiré et une interconnexion entre les deux réseaux est créée pour répondre à cette demande. Le client indexe le contenu dans la table de hachage distribuée BitTorrent de I2P, ce qui permet une indexation de contenu anonyme, alors que le contenu est distribué par le réseau anonyme I2P, offrant une distribution de contenu anonyme.

Le projet connectME [11] constitue la première étape vers l'interconnexion des réseaux anonymes et non anonymes, avec un accent particulier sur les environnements de partage de fichiers. Les internautes peuvent installer un client *BiTIIP*, en augmentant ainsi le nombre de clients *BiTIIP* disponibles.

3.2 Caractérisation des environnements anonymes

Nous avons effectué la première supervision à grande échelle du réseau anonyme I2P, caractérisant les utilisateurs et les services qui s'exécutent au-dessus du réseau. Nous avons d'abord déployé une architecture de surveillance basée sur des sondes distribuées placées dans le NetdB de l'I2P, ce qui nous permet de collecter une grande quantité de métadonnées du réseau. Ensuite, nous avons effectué une analyse de corrélation entre le comportement des utilisateurs et le comportement des clients de partage de fichiers anonymes pour parvenir à une caractérisation de

10. Accessible à `http://hmule.gforge.inria.fr/`.
11. Accesible à `http://connectme.gforge.inria.fr/`.

groupe afin de déterminer quelles villes sont liées le plus à l'activité de partage de fichiers dans le réseau I2P.

Caractérisation de l'environnement I2P anonyme

Les systèmes anonymes recueillent plus en plus d'adeptes. Le réseau I2P a ainsi doublé sa base d'utilisateurs au cours de l'année 2012, mais il n'y a toujours pas d'analyse complète de ce réseau, ni un suivi caractérisant le système.

Notre architecture de surveillance distribuée permet de déterminer l'utilisation des applications de partage anonyme de fichiers ainsi que celle des serveurs Web anonymes, ou *eepsites*, disponibles dans le réseau I2P. Nous avons établi que le client I2PSnark est l'application de partage de fichiers anonyme le plus utilisée, contrairement à d'autres solutions de partage de fichiers, tels que iMule ou I2Phex. Nous avons trouvé ainsi plusieurs *eepsites* non référencés sur le système, soit environ 30% du nombre total de eepsites détectés. Nous avons en outre déterminé la répartition géographique des utilisateurs d'I2P et observé que le réseau I2P est largement déployé ; la Russie étant le pays le plus actif.

Cette surveillance à grande échelle nous aide à comprendre comment le système évolue et quels sont ses principaux usages.

Caractérisation groupe basé dans l'environnement anonyme I2P

Avec les analyses au niveau applicatif nous avons détecté le comportement de certaines applications, notamment leur période d'activité. En prenant en compte le comportement d'un service anonyme particulier avec un ensemble particulier d'utilisateurs I2P, nous avons pu corréler cet ensemble d'utilisateurs et láctivité du service anonyme.

Nous avons appliqué le coefficient de corrélation de Pearson afin d'établir une relation entre le comportement des utilisateurs I2P des deux plus grandes villes avec le comportement du client I2PSnark pendant une période de temps donnée. Nous avons ainsi déterminé que 38% de l'activité de partage de fichiers détectée est liée aux utilisateurs de ces deux villes.

Nous avons montré que la surveillance à grande échelle peut non seulement nous permettre d'avoir une vue globale d'un système anonyme mais elle peut aussi conduire à une caractérisation plus fine des groupes d'utilisateurs. Le comportement des utilisateurs dans un réseau anonyme est aussi important que l'anonymat lui-même.