Monitoring the I2P network

Juan Pablo Timpanaro, Isabelle Chrisment, Olivier Festor

Monitoring the I2P network

Juan Pablo Timpanaro, Isabelle Chrisment*, Olivier Festor

INRIA Nancy-Grand Est, France
*LORIA - ESIAL, Henri Poincaré University, Nancy 1, France

Abstract. We present the first monitoring study aiming to characterize the usage of the I2P network, a low-latency anonymous network based on garlic routing.
We design a distributed monitoring architecture for the I2P network and we show through a one week long experiment that we are able to identify 32% of all running applications, among web servers and file-sharing clients. Additionally, we identify 37% of published I2P applications, which turn out to be unreachable after its publication on the I2P distributed database.

1 Introduction

Over the last decade, anonymous communications have gained more and more interest. These networks offer an excellent support for preserving users anonymity and privacy against eavesdroppers or third-party sniffing, as well as support for illegal activities, i.e. copyright file-sharing.

As with the normal Internet, in an anonymous network a message is router through a set of intermediate nodes, before it reaches its destination. However the goal of most anonymous networks are to hide the originator of the message, so the final destination will only see the last intermediate node as originator and not the real initiator of the message. Most of the current anonynous networks use a variant of onion routing[4] and layered encryption for achieving their goal.

There are mainly two types of anonymous networks which run on top of the Internet: *high-latency* and *low-latency* networks.

Most of nowadays high-latency networks are based in the concept of *mix*[2], which can be defined as a process which accepts messages, groups these messages into a batch and forwards them later on, based on a batching algorithm. Since these algorithm introduce delays in the routing of a message, these kind of networks are only suitable for non-interactive applications, like e-mail clients. The *Mixmaster*[12] network is one of nowadays available networks.

On the other hand, most of nowadays applications require a fully interactive connection among users, such as web-browsing or chatting. Low-latency anonymous networks reduce delays and bandwidth usage to improve the speed of the network, although these approaches weak the anonymity provided by the network itself.

The I2P[7] network, a low-latency message-oriented anonymous network was mainly designed to allow a fully anonymous conversation between two parties

within the network limits. Since out-proxying traffic is not the goal of the network, the number of out-proxies in I2P is reduced, on the contrary to the Tor network[14], for example.

I2P contains a full range of available applications: anonymous web-browsing, chatting, file-sharing, web-hosting, e-mail and blogging among others. Except for anonymous web-browsing that necessarily requires an out-proxy to the normal Internet, all other listed applications interact between each other within the network boundaries.

We make the following contributions: First, we present the first application-layer monitoring of I2P. Second, we propose a fully operational architecture for monitoring *destinations* on I2P. Finally, over a a one-week measurement of the I2P network, we show which kind of applications are the most used.

Statistics web sites already exist[1], but we take our analysis a step further and provide high-level results of the network.

This paper is organized as follows. Section 5 points out previous and current work on low-latency anonymous networks, mainly I2P and Tor, ranging from attacks development to network monitoring. Section 2 introduces the I2P network and its main components and features, including peer profiling and its distributed database. Section 3 describes our monitoring architecture. We detail how we use I2P's distributed database to collect data and how we process those data to characterize the network. In Section 4 an initial experiment is presented, covering a full week of monitoring. Section 6 concludes the paper.

2 The I2P network

The I2P network allows anonymous communications between two parties through an abstraction layer, which uncouples the association between the user and its identity. Basically, an application using I2P will not longer be reachable through an IP, but through an location independent identifier. The following sub-sections address the I2P network and its features.

2.1 Overview

The network is formed by a group of *routers*, which is the software that allows any application to communicate through I2P. Applications running on top of it will have a *destination* associated, which receives incoming connections from third parties. The secret lies in which destination is associated to which router and not in the fact that an user is running an instance of the router. This uncoupling between the router and the destination provides a certain degree of anonymity.

I2P uses an extension of the well-known onion routing approach [4], in which a message is routed from its originator to the final endpoint through several intermediate nodes using layered encryption. The originator adds to the message

[1] Such as `http://stats.i2p.to/`

to be sent an encryption layer per every node in the path, each intermediate node *peels off* one of these layers, exposing routing instructions along with still-encrypted payload data, and finally the last node removes the final layer of encryption, exposing the original message to the endpoint.

This extension is called garlic routing, and allows to include several messages in a single *onion*. I2P currently uses this approach to include the return destination for a given message as well as status messages. Future releases of the network will use this garlic approach to effectively sends several messages into a single garlic message.

The path through a selected list of nodes is called a *tunnel* and represents a key concept in I2P.

2.2 Tunnels

Every tunnel is unidirectional, and is formed by the gateway (entry point), a set of participants (intermediate nodes) and an endpoint. Additionally, two types of tunnels exist. *Inbound* tunnels allow an user to receive data, and *outbound* tunnels to send data. A fully bidirectional communication between two users will involve four tunnels, one inbound and one outbound for each user.

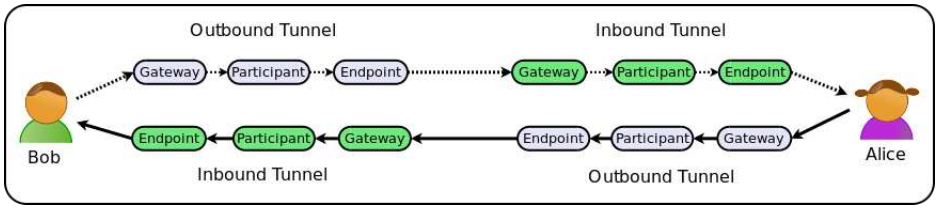


Fig. 1. Simple tunnel-oriented communication in I2P

Figure 1 illustrates a communication between Alice and Bob. First, lets assume that Alice and Bob know every other destinations. Alice will send a message through her outbound tunnel, targeting one of Bob's inbound tunnel gateway. Once the message reaches the *gateway*, which is the entry point for Bob's tunnel, it will be forwarded all the way through Bob's router. Alice does not have a knowledge about Bob's inbound tunnel, but only about the entry point: Bob might have as well a tunnel composed with 1 or 100 intermediate nodes, but Alice ignores this. This property is the earlier mentioned *decoupling*. The same procedure is used by Bob when sending data to Alice, he will send a message targeting Alice's inbound gateway.

2.3 Profiling algorithm

Tunnels are created every 10 minutes, and then dropped. This feature complicates a traffic analysis attack, since a 10 minutes time window is rather small

to acquire any knowledge of a network[1]. Therefore, every 10 minutes, an user selects new nodes for its tunnels, which are selected among all the routers in the network.

The easiest way would be to choose random routers among the fastest ones in the network, and place them in a random order through the tunnel. However, I2P's anonymity and performance depend on not being random. I2P leans on a constantly local profiling of all seen routers, so as to characterize every peer regarding its performance, latency, and availability. A four-tier scheme is used to classify routers: fast and high capacity, high capacity, not failing, and failing.

Tunnel participants are randomly chosen from the fast and high capacity tier, and ordered through the tunnel according to the *XOR* distance from a random value. This *XOR* ordering prevents information leakage in predecessor[15]. Keeping the gateway and the endpoint fixed, but yet randomize the rest of the tunnel will prevent further attacks; however this counter measurement is not yet implemented.

Peers are profiled based on their interactions with the profiling peers. Among the attributes used for profiling we can find success rate of lookups in the network database, success rate of message through a tunnel containing the profiled peers, the number of peers a profiled peer can introduce to us. All kind of indirected behaviour is recorded and used for profiling, to the contrary of claimed performance from routers. No published performance information is used in local profiling, so as to avoid simple attacks.

2.4 I2P netDB

The *netDB* is one key concept in the I2P network. It is a distributed hash table based on the Kademlia [10] protocol, used to store and share network metadata. However, contrary to the Kademlia protocol, not every peer in the I2P network form part of the DHT, but only those fast I2P users, the so called *floodfill* peers. Any user with high bandwidth can appoint itself as floodfill peer, if the number of floodfill peers in the network decreases for some reason.

There are two types of network metadata, *leasesets* and *router infos*. A leaseset provides information about a specific destination, like a web server, a BitTorrent client, an e-mail server, etc. A router info provides information about a specific router and how to contact it, including the router identity (keys and a certificate), the address where to contact it (ip and port), several text options and a signature.

A leaseset provides a number of entry points for a client destination. When Alice creates a destination (for an e-mail server, for example), a set of *leases* are created and grouped into a leaseset. Each lease contains the tunnel gateway router (where Bob should send the messages), when the tunnel expires (deadline for using this gateway) and a tunnel ID.

This distributed database contains an extra security feature, to harden a localized *Sybil attack*. The key used to index a record in the netDB is computed as `HASH(ID + date)`, in which the *ID* is the record ID, and *date* is the current

date. A record ID remains fix as long as the record is conserved, however the record indexing key will change every day.

It means that at midnight, all indexing keys will be changed and therefore re-published in other locations in the DHT. Even though some queries might fail around midnight, this approach avoids an attacker to launch a simply localized Sybil attack, since the attacker will have to re-compute its Sybils IDs using a key-to-key dictionary. A deeper view of this network database is out of the scope of this document, however a further insight of the netDB and the flooding mechanism can be found in the official I2P website [7].

2.5 Differences with the Tor network

We can dedicated an entire section for specifying differences between these two network, however we will focus in high-level differences. Both Tor and I2P are low-latency networks, although Tor is intended for out-proxying traffic to the regular Internet, whilst I2P hardly has out-proxies. However, one of the main difference is how both networks manage the participants. Tor has a central server directory, which provides an overall view of the network and eases statistics retrieval.

On the other hand, I2P is based on a distributed Kademlia-based database, along with a peer profiling algorithm for peer selection. This distributed component hardens the network and makes it more resilient to shut-down attempts, on the contrary to the Tor central-based directory.

3 I2P monitoring architecture

We aim to monitor the I2P network, and determinate which are the most used applications running on top of it, so as to characterize the usage of the network. Is I2P mostly used for file-sharing? Or is it mostly used for anonymous web hosting? Can we even characterize the usage of the network?.

We focus on two main applications, web servers (anonymous web hosting) and file-sharing clients. For file-sharing clients, we will consider I2PSnark, an built-in BitTorrent client in I2P.

So as to achieve this goal, we could inquire every single user in the network for the running application(s) at a given moment. However, as earlier mentioned, I2P bases its anonymity on de-coupling the identity of an user (provided by its *router info*) from the application it is running (provided by its *leaseset*). Therefore, the challenge is to determine which application is running in a given *leaseset*, even if we can not link an user with its running applications.

3.1 I2P and its netDB

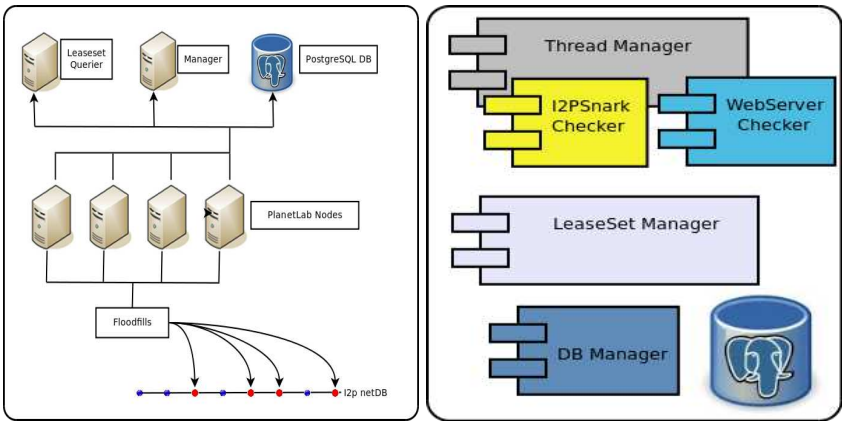
Since the netDB contains, normally, all the *router infos* for every router in the network, and all the published *leasesets*, it becomes a key component in our monitoring architecture.

We try to retrieve from the netDB as many *leasesets* as possible, and query them. To collect *router infos* and *leasesets*, we place a small number of *floodfills* in the distributed hash table. As mentioned before, floodfill nodes have the capacity to store and index this information, and they are a sub-set of all the nodes in the network.

We take advantage of the mechanism to become a floodfill in the I2P network, in which any router can postulate itself as a floodfill router. Once our routers announce themselves as floodfills, they will start to receive store requests and provide to others with this information.

3.2 Monitoring Architecture

Our monitoring architecture is divided in two main parts.



(a) Leasesets retrieval (b) Leaseset querier

Fig. 2. Complete monitoring architecture

Figure 2(a) presents the approach used to collect *leasesets*. Each floodfill is running in one PlanetLab node, and logs every received request to a PostgreSQL server, located in our high security lab.

Figure 2(b) shows a component architecture of the *querier*, which runs in parallel with the architecture showed in figure 2(a). The *Leaseset Manager* periodically retrieves new leasesets from the database, while the *Thread Manager* creates different threads to test a given destination for a particular application (a Web Server or an I2PSnark client). Once a leaseset is tested, the result is kept in the PostgreSQL server. This architecture needs a running I2P router, so as to join the I2P network and communicate with different leasesets.

Posterior analysis is made off-line, so as to generate the statistics of the network for a particular period of time. A real-time analysis is possible with a few modifications to the architecture shown in figure 2(b).

3.3 Analysis of I2P applications

At the moment, we analyse two kinds of applications running on top of an I2P router. A web server and an I2PSnark client, which is a modified bittorrent client designed to run on top of I2P.

On the one hand, to tag a given leaseset as a web server, we open a socket through it and send a `GET` message. If the response contains well-known http keywords, then that leaseset corresponds to a web server.

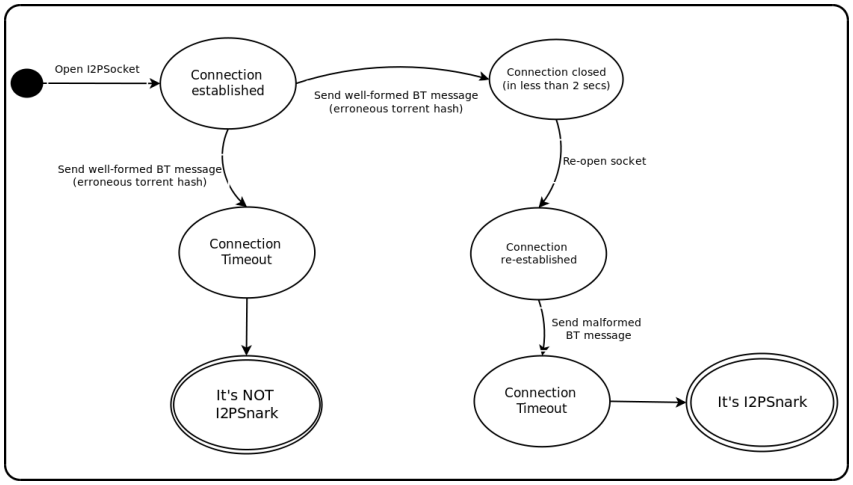


Fig. 3. State machine for testing an I2PSnark client

On the other hand, the approach used to test an I2PSnark client is shown in figure 3, which we obtained by analysing the I2PSnark code, along with basic black-box testing.

Firstly we open a socket through a given leaseset using the underlying I2P Router. Once the connection is established, we send the first message, a well-formed BitTorrent message, requesting a random torrent. If that given leaseset is actually running an I2PSnark client and not sharing the torrent, it will close the socket immediately. Secondly, we re-open the connection and send a malformed BitTorrent message. If the response timeouts, then we decided that given leaseset is running an I2PSnark client. If we do not receive any answer for the first message, we can not assume anything, and we will keep trying that leaseset with different applications.

4 Experiments

In this section, we first present an experiment on monitoring the I2P network with the previously presented architecture, and second we detail a set of results.

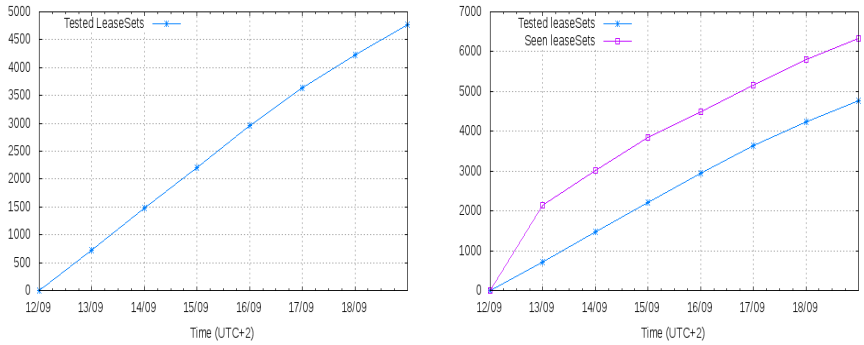
4.1 Set-up

We randomly placed 15 floodfill routers in the netDB, each of them running in one planet-lab node, and logging every `store leaseset` request. Our leaseset querier ran in parallel, analysing new leasesets and logging the results as well. We ran 3 different one week experiments, starting on September 12th, September 27th and October 4th.

We log four possible outcomes for a given leaseset.

- WebServer
- I2PSnark client
- Unknown: *The given leaseset is not running a web server nor an I2PSnark client*
- Destination not found: *The given leaseset is unreachable*

4.2 One week experiment



(a) Number of leasesets tested (b) Total number of leasesets retrieved

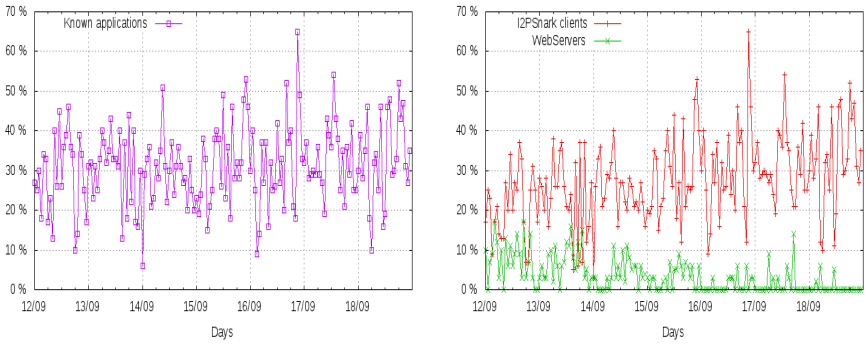
Fig. 4. Leasesets results

We show the results for the experiment ran on September 12th, although the three of them present the same results. Figure 4(a) shows the accumulative value of already tested leasesets at every hour . We tested, in average, 676 new leasesets per day, at a rate of 28 leaseset per hour.

We queried and tagged a total of 4774 leasesets and our 15 floodfills received a total of 6331 leasesets *store requests* during the experiment. Our analysis only considers new leassets, therefore 6331 *different* leasesets were seen during one week, although our floodfills had been receiving, during the experimentation, *store requests* for the already seen leasesets.

During the first day of the experiment, we received a high number of `store leaseset` requests, however the following days the rate of new leasesets got

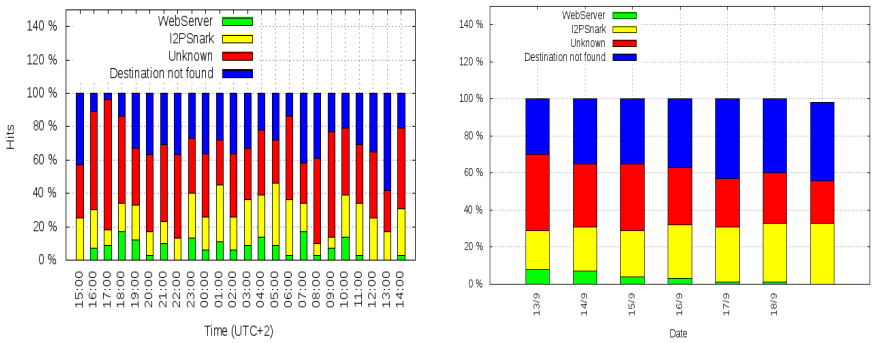
stable. We tested 75.4% of all seen leasesets (4774 out of 6331). Figure 4(b) shows that we tested every new leaseset that arrived after the first day of the experiment, once the rate of incoming new leasesets got stable.



(a) Both applications (b) Separate values

Fig. 5. Identifiable leasesets

Figure 5(a) presents the percentage of leasesets that we were able to identify and tagged. In average, we were able to identify 32.06% of the 4774 leasesets queried, which most of them were I2PSnark client, as shown in figure 5(b).

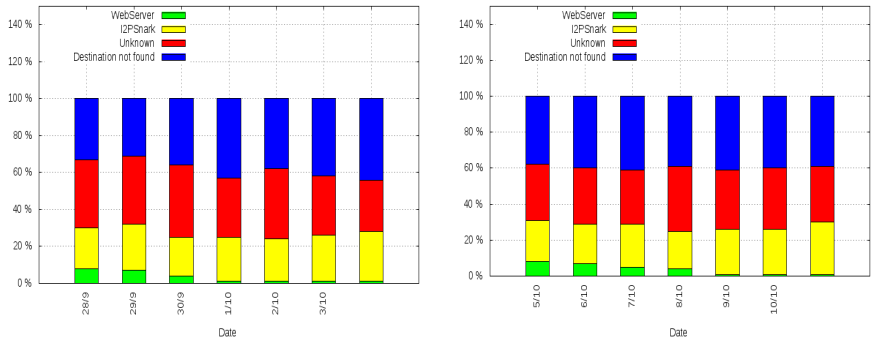


(a) One day results (September 12th) (b) One week results (September 12-19th)

Fig. 6. One day/week results

On the other hand, figure 6 presents the fully results of the experiments, in which the *unknown* destinations along with the *destinations not found* values are shown. Figure 6(a) presents a daily view of the network on September 12th,

and figure 6(b) presents our weekly view of the network starting on September 12th.



(a) One week results (27/09 - 04/10) (b) One week results (04/10 - 11/10)

Fig. 7. One week results

Figure 7 shows the remaining two one-week experiments, which present the same distribution of results with the first week of experimentation.

4.3 Analysis of results

Synthesis of results We consider new leasesets, and for this reason the number of anonymous web servers or eppsites decreases every day. By the end of the experiment we hardly see new anonymous web servers been published. This is a particular case, since anonymous web servers maintain its published destination over time so as to be known and reachable every time. On the other hand, an I2PSnark client creates a new destination every time it starts, and therefore we have a stable rate of new I2PSnark clients in our results.

Among the 4774 leasesets we analysed during our first experiment, we got 184 anonymous online web servers running on top of I2P, 1350 I2PSnark clients, 1440 unknown applications and 1800 unreachable leasesets.

Most of I2PSnark clients were identify during European night time, since after 06:00 the number of I2PSnark decreases. Figure 8 shows an increase of I2PSnark clients starting at 20:00 and a decrease around 07:00 AM. Moreover, during the week days, we have an average of 25.5% I2PSnark clients, and during the weekend this value climbed to 32.5%, indicating that I2P users dedicate more time for file-sharing during weekends, which is not surprising. Additionally, there are not time ranges during the weekend, since we had different pikes in our analysis during day time as well.

Finally, we had, in average, 30.16% of *unknown* applications, which means that we successfully opened a socket through these leasesets, but we failed at

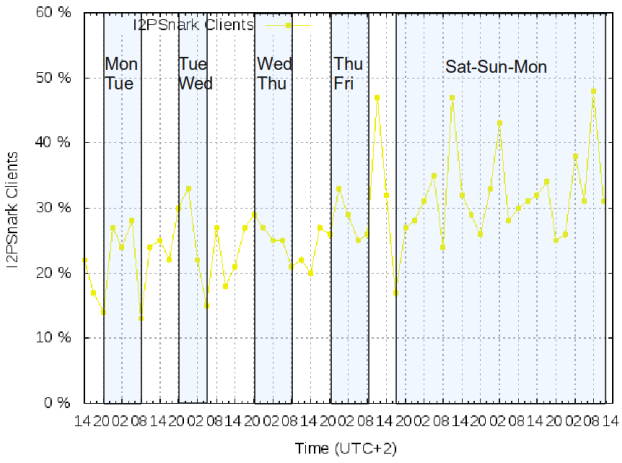


Fig. 8. I2PSnark clients during one week (September 12th - 19th)

tagging them. Future work will be focus on developing new methods for testing further applications.

***Destination not found* analysis** During our experiment, we had an amount of leasesets that were not available after being store in the DHT by our floodfills, and as a result these were tagged as *Destination not found*. Despite every leaseset was double-check after the first *destination not found* result, we kept on getting these status messages from our local router. In average, we got 37.70% of leasesets in which no answer was obtained while trying to open a socket through them.

We ran the same experiment 3 times, and we got 37.70%, 38.14% and 39.71% for the first, second and third experiment respectively, with an overall average of 38.42%.

We consider a set of possible problems for this situation. Firstly, a particular application might be publishing a set of destinations and only using a sub-set of destination, discarding the rest. Since there are a considerable set of available applications, it is non-viable to test every single application.

On the other hand, we focused on the file-sharing applications. We retrieved 2200 active torrents from the postman tracker[2] of I2P. For each torrent we got a list of destinations (which represents a torrent client) sharing that particular torrent, and we repeated this procedure every hour, so as to get as many new destinations for a given torrent.

We cross checked all the retrieved destinations from the tracker along with the destination not found of our previously analysis. We got that 15% of the destination not found were from file-sharers using the postman tracker.

[2] `tracker2.postmam.i2p`

Ongoing work is focus on querying alternative torrent trackers on I2P, such as `tracker.welterde.i2p` and `tracker.rus.i2p`. However, alternative trackers do not seem to be online most of the times, and therefore we focused on the postman tracker.

Anonymous web servers analysis Since destinations for anonymous web servers or eppsites do not change over time, we enquire them every hour, to check their availability through time. Figure 9 shows the accumulative value of anonymous web servers online for a two-week period (28/09 - 11/10).

We analysed 139 eppsites, in which we can observed that more than 50% of them were online 70% of the total time and only two of them were only during the whole two week experiment.

These statistical data can be as well obtained in `perv.i2p` or `inproxy.tino.i2p`. Both anonymous websites scrape the list of known anonymous hosts from the I2P hosts file and present the uptime of every eppsite, which is the same results we achieved but from a different approach.

We showed that we can have the same results of scraping the hosts file, using the netDB as source of destinations. Additionally we can detect new anonymous web servers even though they are not publish in the hosts file, since they need, in most cases, an netDB entry to be reachable.

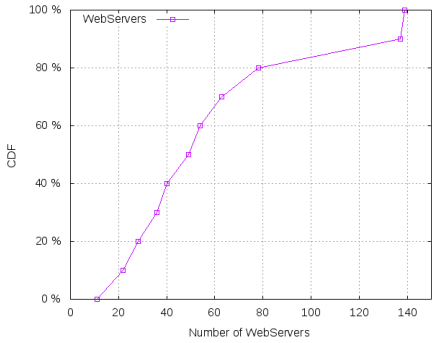


Fig. 9. CDF of online anonymous web serves

5 Related work

Anonymous low-latency networks are gaining more and more interest. The *Tor* network is probably the one that is receiving most of the academic attention. There are a series of studies regarding attacks on the Tor network, from timing attacks [6] to discovering *hidden services* from their clock skew [13].

Regarding monitoring the Tor network, McCoy et al.[11] describe their monitoring approach, which takes advantage of being a Tor *exit node*, and analyse the application layer of outgoing traffic to determinate the protocol distribution in the network. They discover that most of the connections through Tor are interactive http traffic, while a few of them are for BitTorrent traffic. However, these few connections (3.33% according to their study) consume a disproportionally amount of bandwidth (40.20% of the total measured).

The study in [11] resembles what we want to achieve for I2P, however, the data collection methodology applied by McCoy et al. can not be used in I2P. Services in I2P interact within the network limits, and there is not out-going traffic for file-sharing or e-mail, for example.

More recently, Loesing et al. [9] present a measurement of sensitive data on the Tor network, such as country of connections and exiting traffic by port. Additionally, Loesing [8] measured the *trends* of the Tor network from directory information. However, this network has a central component, a *directory server* and hence the monitoring approaches can not be applied in the I2P network, which does not include any kind of central component. The Tor metrics website, `http://metrics.torproject.org`, provides further technical reports on measurement of network components, such as relays and bridges.

In [16] an analysis of the peer profiling algorithm in I2P is presented, discussing its strengths and weaknesses together with future improvements. They conclude that I2P's profiling algorithms behave accurately when finding faster peers, increasing its performance when the I2P router requires more bandwidth. They also conclude that these algorithms do not provide an easy way of *tuning* the trade-off between anonymity and performance, although an user can tweak the length of a tunnel to make its connection faster or more anonymous.

Adrian Crenshaw[3] studied how to identify I2P's hidden services. He successfully links an anonymous website, known as *eepsite*, to its real Internet address. He takes advantage of the lack of control in the application layer, errors and mistakes that system administrators as well as developers made during the configuration of a web server. Even if it is not an I2P problem, he shows how it is important to properly set-up any application running on top of an anonymous layer.

Herrmann et al.[5] conducted an attack on the I2P network, so as to determining the identity of peers hosting anonymous web sites. They proposed a three-step attack, in which an adversary with modest resources will 1) get an estimated view of the victim's network, 2) attack the victim's fastest peers tier[3], so as to replace them with the attacker's peers, and finally 3) tag a given router as the host of an eepsite by using a traffic analysis technique developed by their own, based on statical patterns of http requests induced by the adversary. They conclude that churn in the fast and high capacity tier might be the main problem, and they propose solutions to avoid such a churn as or reduce it.

Our work will give the I2P community a bird's eye view about the network usage, from an application point of view. We are taking the monitoring of this

[3] Peer tiers are explained in section 2.

network an step further with our work, not by providing well-known statistics, such as the number of routers or bandwidth usage[4], but by identifying running services on top of the I2P network.

6 Conclusion

We have developed the first high-level monitoring architecture for I2P, aiming to characterize the use of the network in term of available *destinations* and applications running on top of it.

After the first one week experiment we analyses 4774 leasesets. We were able to identify 181 web servers and 1350 I2PSnark clients, and we did determinate that 37% of the published leasesets were off-line after their publication on the netDB. On the other hand, we were not able to identify 30% of the entire set of leasesets, which means that 30% of the network is not running a web server nor an I2PSnark client.

We presented an uptime analysis of anonymous web servers in I2P derivative from our primary analysis. We could observed that more than 50% of seen anonymous web servers were online 70% of the total time of the two weeks analysis.

Ongoing work includes improving the approach to identify anonymous web servers and I2PSnark clients, as well as testing new I2P applications. Including new torrent trackers in our analysis for unreachable destination is another major goal, even though not every torrent tracker can be scraped or queried.

Future work will include a long-term measurement of the network. Even though a one week experiment might give us a glimpse about the current usage of the network, a monthly or even a longer experiment will provide us with a deeper characterization of I2P. Additionally, we will deploy our monitoring architecture permanently, as well as developing a front-end to have a real time view of the network, based on the data collected by our architecture.

References

1. K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against anonymous systems. In *Proceedings of the 2007 Workshop on Privacy in the Electronic Society (WPES)*. Citeseer, 2007.
2. David Chaum, Communications Of The Acm, R. Rivest, and David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–88, 1981.
3. Adrian Crenshaw. Darknets and hidden servers: Identifying the true ip/network identity of i2p service hosts. In *Black Hat DC 2011*, DC, 03 2011.
4. D. Goldschlag, M. Reed, and P. Syverson. Hiding routing information. In *Information Hiding*, pages 137–150. Springer, 1996.
5. M. Herrmann and C. Grothoff. Privacy-implications of performance-based peer selection by onion-routers: A real-world case study using i2p. In *Privacy Enhancing Technologies Symposium (PETS 2011)*, 2011.

[4] Which can be found in `http://stats.i2p.to/`.

6. N. Hopper, E.Y. Vasserman, and E. Chan-Tin. How much anonymity does network latency leak? *ACM Transactions on Information and System Security (TISSEC)*, 13(2):13, 2010.
7. I2P. The i2p netowrk. `http://www.i2p2.de/`.
8. K. Loesing. Measuring the tor network from public directory information. *Proc. HotPETS, Seattle, WA, August*, 2009.
9. K. Loesing, S. Murdoch, and R. Dingledine. A case study on measuring statistical data in the tor anonymity network. *Financial Cryptography and Data Security*, pages 203–215, 2010.
10. P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *Peer-to-Peer Systems*, pages 53–65, 2002.
11. Damon Mccoy, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the tor network. In *In Proceedings of the 8th Privacy Enhancing Technologies Symposium*, 2008.
12. Mixmaster. The mixmaster network. `http://mixmaster.sourceforge.net`.
13. S.J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 27–36. ACM, 2006.
14. Tor. The tor netowrk. `http://www.torproject.org/`.
15. M.K. Wright, M. Adler, B.N. Levine, and C. Shields. Passive-logging attacks against anonymous communications systems. *ACM Transactions on Information and System Security (TISSEC)*, 11(2):1–34, 2008.
16. zzz and L. Schimmer. Peer profiling and selection in the i2p anonymous network. In *PET-CON 2009.1.*, TU Dresden, Germany, 03/2009 2009.