Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya

Performance evaluation of TOR Project

Authors: **Alejandro Montero Rivero**

Chetan KC

Decentralized Systems

Master in Innovation and Research in Informatics

Spring 2017

This page intentionally left blank

Abstract

Tor is currently one of the most used system for providing anonymity when accessing both deep and surface web, using a 3-way proxy system in which intermediate nodes only know how to reach the previous and next hop. Even though this architecture provides high level of anonymity to the user, content can only be reached after jumping from 3 different hops that have heterogeneous network characteristics. This network heterogeneity and increased number of hops make Tor users experience variable delays and slow throughput on connecting to servers.

In this document we propose a Bash+Python framework capable of automatically measure Tor network latencies as well as throughput speeds, used to estimate users potential QoS. To test Tor latency we measure the Round-Trip-Time (RTT) since a circuit is requested, until the initial HTML file is downloaded. Throughput tests are conducted in a similar fashion; a file of fixed size is downloaded and Bandwidth is calculated as a combination of size versus time. As part of our framework the user is capable of selecting which pages to evaluate and which files to download, as well as the number of iterations to perform.

As part of our study, we conducted several experiments using Tor network to analyze the latencies to access 24 very geographically distinct web-pages, as well as bandwidth when downloading a fixed 37MB file situated in a publicly accessible UPC repository. Experiments are conducted for both the Tor network and without using any proxy. Initial results show that Tor suffers heavy latency and bandwidth variability between circuits and very small performance compared to regular Internet connections.

Index

1. Introduction

With increasing use of the Internet, people are becoming more concerned about security and privacy issues. Till now many researchers have developed and proposed solutions which can securely communicate without revealing potentially identifying information such as IP addresses. Not only that, this architecture makes possible to communicate with countries with heavy Internet restrictions and in consequence, promoting freedom of speech, liberty and overcoming censorship.

Tor is a low latency anonymity system that is loosely based on onion routing principle. It gets it popularity in 2004 and by now there are around 2 million daily users from more than 120 different countries. Like in other anonymity designs Tor seeks to hide the relationship between source and destination from the network analyzer, and anonymization infrastructure itself. Tor protects the user by bouncing Internet requests and data through a distributed network of relays that are run by volunteers all around the world preventing man-in-the-middle attacks,

Since it bounces the internet requests and data all over the world several times, end to end latency is very high comparing to the normal internet connections. Although both the latency and throughput of the Tor has been improved in these last years still the quality of service is unpredictable, jumping from very fast to awful connections depending on the circuit created at every instance. This undoubtedly harms users trying to browse the web, but also makes Tor a no-option to those applications that require low-latency and/or high bandwidth with assured QoS.

2. Background

As TOR is a non-profit organization it consists a volunteers-operated node also called relays(servers) which forward traffic on behalf of users running Tor clients [1]. The main idea of the tor is to establish an anonymous channel between clients to the destination. To do so at first a client selects a path through the Tor network by choosing suitable nodes available at that moment. A path usually consists of three nodes, the entry, middle and the exit one. The nodes that were selected to create a path only knows its predecessor and successor, not the identities of both communicating parties which make Tor network anonymity [2].

There also exists a hidden service in tor network. The hidden services are servers that accept the inbound connections only via the Tor network. It is therefore possible to provide anonymity to those servers. The IP address of those servers remains unknown, but the service can be access through the onion address. Tor relays are able to use those address to route the traffic to and from the servers that provide hidden services while providing anonymity to both sides.

2.1 Path selection

The first step in Tor network is selecting a path. To select a path, Tor network uses path selection algorithm which selects a path depending upon some certain constraints, such as it doesn't select two nodes which are in the same network range. Eg: if the first node is in /18 network then it won't select the second node which is in the same range. To improve the performance, it also selects the nodes which has the high bandwidth comparable to the rest of the available nodes.

2.2. Circuit Building

As soon as the path is selected clients start to negotiate a session keys with all the involved nodes in a selected path. The final encrypted path in a Tor network is called circuit. Now the client will use this circuit to pass it internet requests or the data. Circuit build is done one hop at a time as you can see in the fig1. Clients uses Diffie-Hellman key exchange, which need high computationally expensive public-key cryptography and multiple packets round trip time which is the main reason it takes lots of time to build a circuit. The time taken to build a circuit gives you the idea how the created circuit will work for you in the future. Client can take one time out value and discard the circuit creating process which takes longer time than the timeout value.

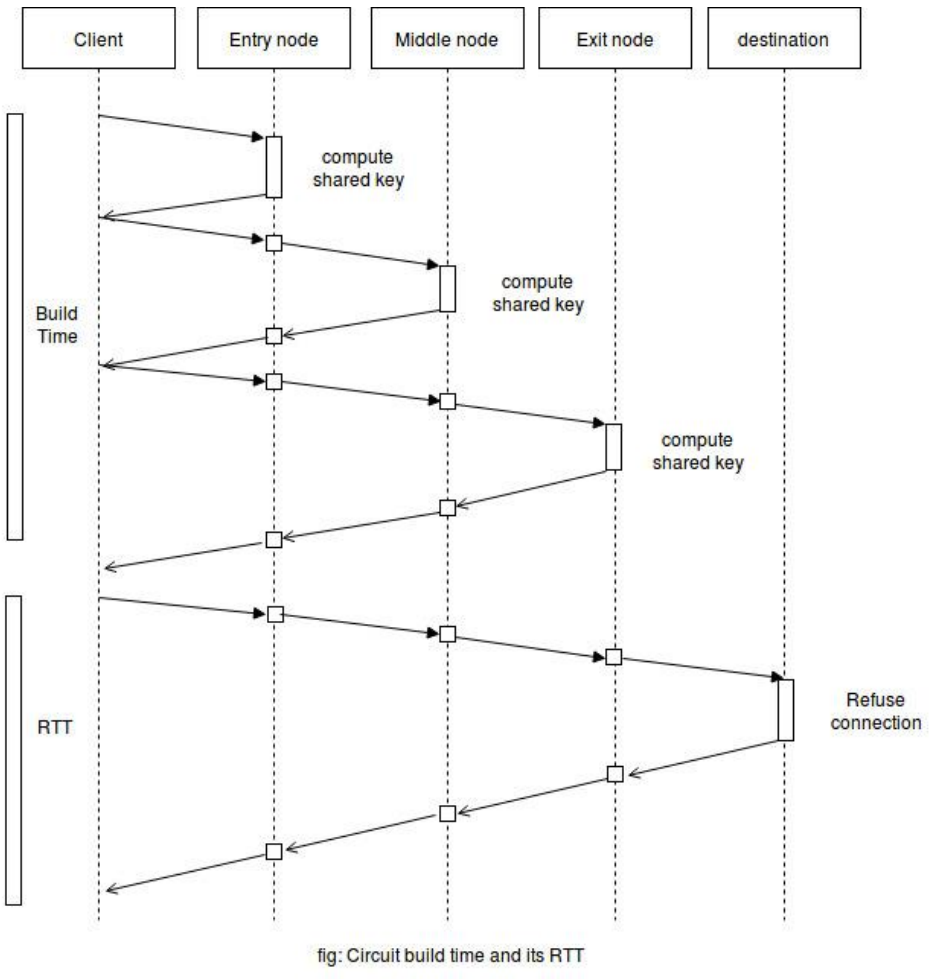


Figure 1: Circuit building and querying

When circuit built successfully a client uses a circuit-RTT method to measure the RTT of that particular circuit by asking an exit node to open a TCP connection to the destination. The RTT includes the latency of the individual nodes involved in the circuit and also the queuing and processing delays of individual nodes.

3. Framework

The objective of our project is to test the latency and bandwidth using Tor as a proxy and without using any proxy. After doing a research how we can test automatically the latency and bandwidth using Tor, we found that there exist an API called stem which can be used to automatize our test. Stem is a python controller library that can be used to interact with the Tor. So, we decided to use a python to calculate the latency and bandwidth using Tor as a proxy and without using any proxy. We also decided to use a bash script to automatized the download and unpack the Tor package. So far by using a python and bash we automatized our whole framework. We have created on file with list of websites which will be used by our framework to calculate the latency. For the latency test we have decided to download only the html files and for the bandwidth test we have decided to download a 37MB file which is hosted in a public server from the UPC.

Algorithm 1 gives a basic idea about how the bash script is working in our framework. It checks the TOR files in local folder from where we are running the script, if it doesn't find the file then, it will download it from the repository of Tor. When download completes it will unpack the file and checks for the python and import all the required packages. If it found the Tor file, then it directly jumps to check the python in the system. After finishing the importing, it starts the Tor and calls the python script where we have implemented our test function.

Algorithm 1 bash script

```
1.  install dependencies
2.  check <- tor in a local folder
3.  if (! tor)
4.      download tor;
5.      unpack tor;
6.  check <- python
7.  import required packages;
8.  start tor;
9.  call -> python script;
10. uninstall dependencies;
11. End
```

Algorithm 2 main function of Python script

```
1.  main ()
2.     do (number of website)
3.       configure TOR
4.       call queries (latency, Tor/without Tor)
5.       post process results and save in .csv
6.       call queries (bandwidth, Tor/without Tor)
7.       post process results and save in .csv
8.     end do;
9.  end main;
10. return to Bash;
```

A bash script calls a python script, the main function in the python script configures Tor and call another function (query) in the same script until the number of websites. While calling a query function it mentioned a parameter for what it is calling a function. For example, if it calls query function like "query (latency, tor)", then it wants that the query function calculates the latency of a website using a Tor as a proxy. The main function will call the query function for 2 times per site to calculate a latency, first to calculate latency by using a Tor as a proxy and second to calculate without using any proxy. It also calls query function to calculate the throughput by using Tor as a proxy and without any. Inside the results folder the main function of python will create four different files. One of them contain the average results calculated for the latency by the query function for each website using a Tor as a proxy. Another one contains latency test results without using any proxy, third one contains the average result calculated by the query function for the bandwidth test using Tor as a proxy and the last one contains the result of bandwidth test without using any proxy.

Query function is explained in algorithm 3 when the main function calls it, the query function first checks either it is calling to test the bandwidth or for the latency test. Then it takes the website selected by the main function and it run for 10 iterations. We take a 10 iteration to be sure that it is giving us a good result but number of iteration could be changed. For each iteration, the function selects a path, build a circuit and measures circuit building time, query time, circuit creation failures and query failures, Round Trip Time (RTT) and Throughput if the main function is calling to measure the throughput. As mentioned above the query function will iterate 10 time per website it will calculate CCT, CF, QT, RTT if it was called to measure the latency, and it will also calculate bandwidth (BW) if it is called to calculate the throughput after finishing the iteration it will make an average of all calculated parameters and send a result to the main function.

After getting a result from a query function, main function will create a folder named results and save the result in that folder in a csv format and take another website from the text file where we have listed number of websites and call a query function again. It will iterate for the number of sites mentioned in a text file. When it calls query function for the last time and gets reply from it then it returns back to the bash script from where it was called. The bash script then uninstalls everything that it had installed in the beginning and terminate the program.

Algorithm 3 testing latency/bandwidth

```
1.  queries
2.  for (every site)
3.      do(iterations)
4.          circuit <- build circuit(path)
5.          CCT <- circuit creation time ()
6.          CF <-circuit failure ()
7.          QT <- query times ()
8.          QF <- query failure ()
9.          RTT <- round trip time ()
10.         BW <- measure throughput ()
11. calculate average (CCT, CF, QT,
                       QF, RTT, BW)
12. Return average ()
13. end for
```

4. Experiments

Using the developed framework, we have done two different experiments one for latency test and another for bandwidth. To become sure that our framework is working well and we get the correct results, we did both the experiments in three different scenarios.

4.1. Latency

To do the latency test, we have chosen 24 different sites from around the globe. Among those we have taken 18 from a paper [1] and the rest 6 are chosen by us. We have selected 6 popular sites like Facebook.com, Microsoft.com, and Apple.com etc. To confirm that we get the accurate results from our experiment we did 10 iterations for each site, that means each site was requested 10 times from three different networks which have different characteristics using proxy (Tor) and without using any proxy. Another purpose to test each site for 10 times is to check either it uses the same circuit for the 10 iterations or not, and from our experiment we convinced to the conclusion that, for every iteration it starts from the zero.

4.2 Bandwidth

To test the bandwidth, we have downloaded a 37MB file which is stored in a public repository from UPC. As in the latency, for the bandwidth also we have downloaded the same file 10 times by using proxy (TOR) and without any proxy. This one is also tested in three different scenarios having different characteristics. For the bandwidth test also the framework does the same thing like in latency test, it calculates the average from all the iteration and saves it in csv format.

4.3. Scenarios

We did our experiment in three different scenarios net1, net2 and net3 having different characteristics. We have tried

- **Net1**: 200 MB symmetric optical Ethernet connection which has indirect connection to the public server.

- **Net3**: 45 MB WIFI connection which has also indirect connection to the public server.

- **Net3**: 100MB symmetric optical Ethernet connection which is also directly connected to the public server at 12MB/s.

Also the experiment was done in different timing to see that either Tor network will be effected in the timing or not. The motivation of doing the experiment in three different networks was to get overall view of does it matters for the Tor network from where and when the request comes from.

5. Results

Web-Site	**Net 1 (seconds)**	**Net 2 (seconds)**	**Net 3 (seconds)**
	Tor / Standard	**Tor / Standard**	**Tor / Standard**
https://raco.fib.upc.edu/	24,498$_s$ / 0,570$_s$	12,348$_s$ / 0,243$_s$	2,756$_s$ / 0,264$_s$
https://www.iscte-iul.pt/	14,036$_s$ / 0,321$_s$	17,405$_s$ / 0,129$_s$	4,487$_s$ / 0,041$_s$
https://stackoverflow.com/	10,640$_s$ / 0,540$_s$	17,957$_s$ / 0,261$_s$	4,849$_s$ / 0,046$_s$
https://www.cs.vu.nl/	12,711$_s$ / 0,971$_s$	31,377$_s$ / 0,466$_s$	10,915$_s$ / 0,294$_s$
https://www.mn.uio.no/ifi/	17,658$_s$ / 1,480$_s$	24,264$_s$ / 0,509$_s$	16,425$_s$ / 0,551$_s$
https://www.robtex.com/	6,763$_s$ / 0,431$_s$	23,394$_s$ / 1,345$_s$	8,687$_s$ / 1,232$_s$
http://www.ece.upatras.gr/en/	13,691$_s$ / 0,409$_s$	24,249$_s$ / 0,527$_s$	14,549$_s$ / 0,560$_s$
https://check.torproject.org/	6,396$_s$ / 0,238$_s$	45,130$_s$ / 1,256$_s$	26,358$_s$ / 0,463$_s$
https://www.mta.ac.il/he-il/	14,414$_s$ / 0,398$_s$	23,345$_s$ / 0,861$_s$	12,412$_s$ / 0,453$_s$
http://www.csg.uzh.ch/csg/	5,096$_s$ / 1,877$_s$	20,800$_s$ / 0,128$_s$	12,461$_s$ / 0,097$_s$
https://www.ait.ie/	9,888$_s$ / 3,317$_s$	17,475$_s$ / 1,893$_s$	21,510$_s$ / 1,422$_s$
https://www.upf.edu/	13,308$_s$ / 0,243$_s$	17,689$_s$ / 0,439$_s$	10,449$_s$ / 0,274$_s$
https://www.lkn.ei.tum.de/	13,596$_s$ / 3,220$_s$	25,836$_s$ / 0,222$_s$	6,573$_s$ / 0,099$_s$
https://www.microsoft.com/	8,193$_s$ / 1,052$_s$	18,557$_s$ / 0,338$_s$	8,806$_s$ / 0,046$_s$
https://www.cs.uit.no/	6,194$_s$ / 0,105$_s$	13,025$_s$ / 0,301$_s$	15,403$_s$ / 0,356$_s$
http://www.eecs.qmul.ac.uk/	7,129$_s$ / 0,768$_s$	24,248$_s$ / 0,928$_s$	9,515$_s$ / 1,041$_s$
https://www.upc.edu/	12,108$_s$ / 0,315$_s$	38,179$_s$ / 0,599$_s$	10,387$_s$ / 0,589$_s$
https://www.apple.com/	8,127$_s$ / 0,271$_s$	21,617$_s$ / 0,264$_s$	13,636$_s$ / 0,032$_s$
http://www.jcp-consult.net/	6,891$_s$ / 0,145$_s$	12,432$_s$ / 0,452$_s$	14,862$_s$ / 0,403$_s$
http://www.diku.dk/	12,208$_s$ / 0,235$_s$	21,526$_s$ / 1,029$_s$	12,149$_s$ / 1,041$_s$
http://www.nagariknews.com/	24,156$_s$ / 1.015$_s$	39,542$_s$ / 5,132$_s$	21,358$_s$ / 9,502$_s$
https://www.facebook.com/	5,895$_s$ / 0,378$_s$	12,915$_s$ / 0,217$_s$	7,575$_s$ / 0,641$_s$
https://www.kth.se/ees/	19,011$_s$ / 0,687$_s$	29,635$_s$ / 2,009$_s$	13,394$_s$ / 2,018$_s$
https://www.ait.ac.th/	5,281$_s$ / 0,114$_s$	23,894s / 0,154$_s$	9,932$_s$ / 0,082$_s$
Average	**11,579$_s$ / 0,796$_s$**	**23,202$_s$ / 0,821$_s$**	**12,060$_s$ / 0,898$_s$**

Table 1: Latency measurements for Tor vs Standard connections

As part of our study we observed and analyzed the influence of circuits in the overall performance of the Tor architecture. Table 1 shows latency measurements from Tor vs Standard connections gathered for the 24 selected web-pages, all data point being the average of 10 iterations.

5.1. Latency Evaluation

Tor latencies of queries are divided in two main steps, first, the circuit creation in which a path is created between the user and three relay nodes, one of which accesses the Internet. Creation times of Tor circuits tends to be consistently fast regardless of the selected web-page, averaging a creation time of just 0.6 seconds. Despite the very fast circuit creation the geographic location of the web-page is not considered when the circuit is created, the result is a circuit that may be in a clear disadvantage compared to other circuits to access certain web-sites.

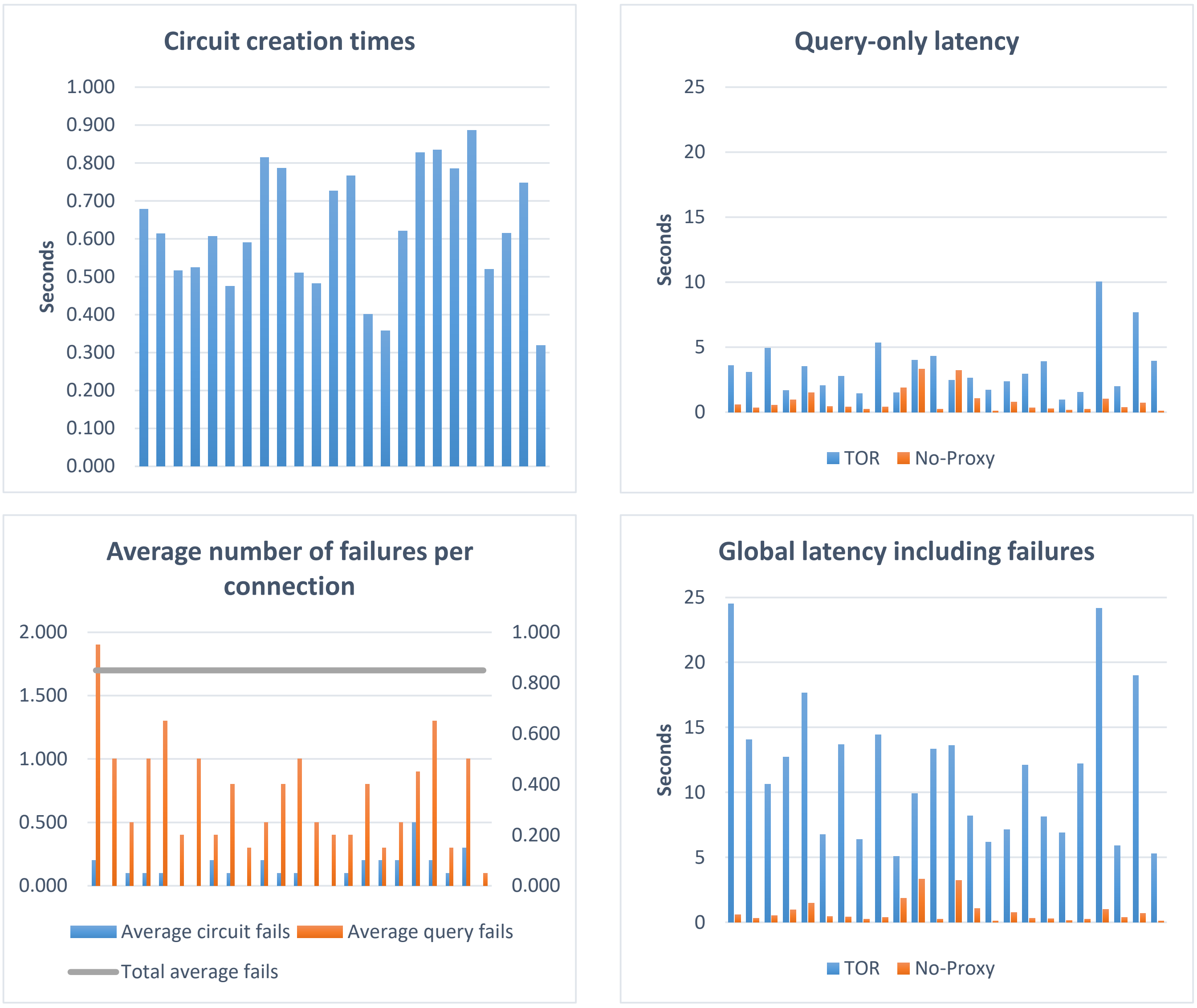


Figure 2 Circuit creation times, Query latencies, Query latencies with failures and number of failures per connection

Second step consists in a pure query to the web-page. Compared to a Standard connection Tor fist has to route the request through the three relay nodes that provide the anonymity, and then a normal petition is requested through the Internet, crossing as many Autonomous Systems as needed, depending on the endpoint relay. As with the petition, the response HTML needs, again, to travel to the endpoint relay and go through the two remaining relays before the user receives it. This architectural overhead makes the RTT of Tor to be way lengthier compared to a Standard query (Figure 2). Despite this overhead the RTT times remain in an acceptable level with latencies consistently under 5 seconds.

Unfortunately Tor is due to failures both when creating a circuit and when querying a web-page. For both kind of failures a new circuit must be created and a whole new query must be performed. Circuit failures are not specially common as our measurements show that for every request a circuit fails only at around 0.2 times. The major issue is the failures occurring when requesting the HTML of a web-page, this failures can be of very different kind: The endpoint relay does not have access to the web-page, timeouts, packet losses, censorship, etc. Unfortunately the number of query failures is very high for our tests averaging around two 1 failure in every iteration (remember we performed all tests 10 times), with a maximum of 2 query failures per iteration in specific web-pages. For all our tests we averaged a 0.8 failures per query, regardless of the kind of failures, this means, that users are going to perceive most of the times at least one failure (circuit or query) every time they request a web-page.

This very high failure rate undoubtedly harms the overall performance of the architecture. Our real-world latency measurements show that at max, Tor query latencies are 5 times slower when errors occur compared to the performance when not a single error is perceived.

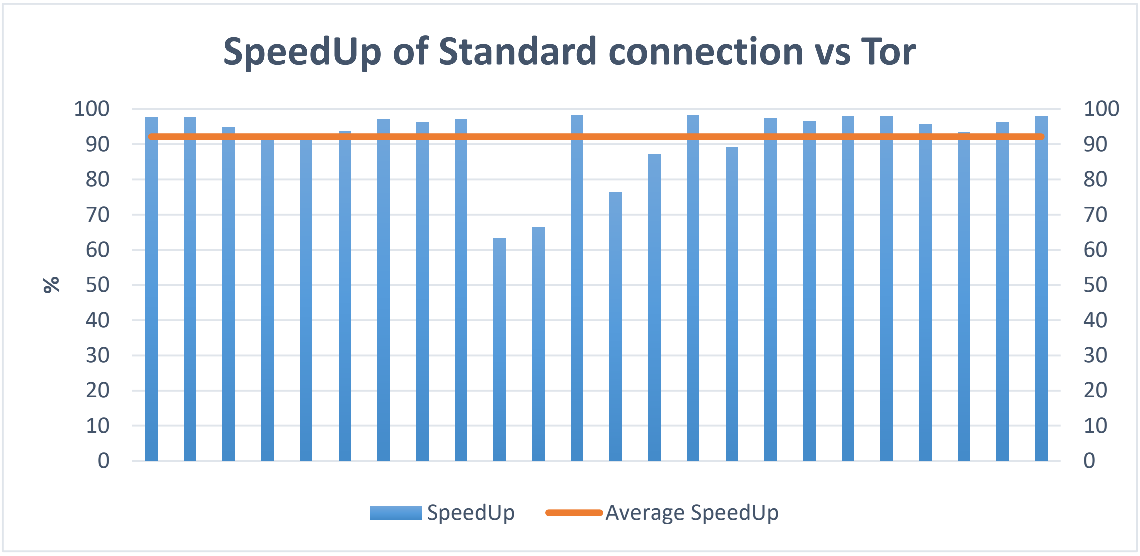


Figure 3: Speedup of Standard connection vs Tor

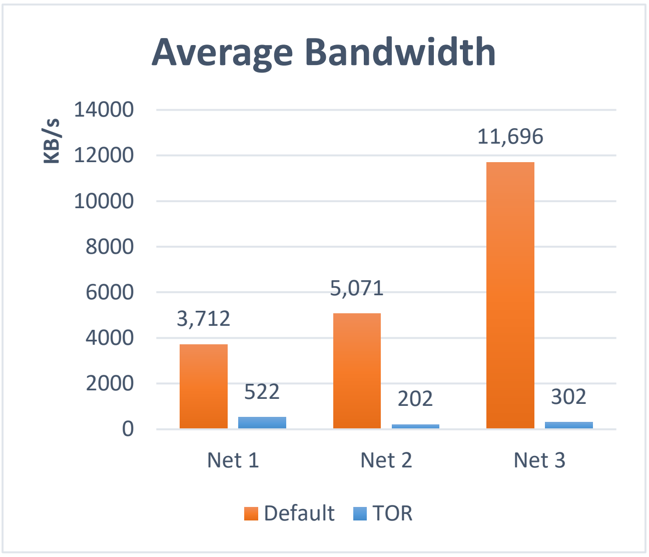
Compared to any Standard connection without proxies Tor performs much worse than expected (Figure 3). In average Tor is a 92% slower when connecting to any web-page regardless of the content type or geographic location

5.2. Bandwidth Measurements

Iteration	**Net 1 (Kbps)** **Tor / Standard**	**Net 2 (Kbps)** **Tor / Standard**	**Net 3 (Kbps)** **Tor / Standard**
Iteration 1	330kbps / 3,684kbps	158kbps / 4,485kbps	257kbps / 11,695kbps
Iteration 2	518kbps / 3,248kbps	31kbps / 4,451kbps	120kbps / 11,693kbps
Iteration 3	833kbps / 3,121kbps	404kbps / 4,586kbps	335kbps / 11,694kbps
Iteration 4	417kbps / 3,266kbps	187kbps / 5,120kbps	728kbps / 11,699kbps
Iteration 5	222kbps / 3,479kbps	118kbps / 5,973kbps	528kbps / 11,696kbps
Iteration 6	303kbps / 4,234kbps	98kbps / 5,286kbps	215kbps / 11,691kbps
Iteration 7	289kbps / 5,554kbps	476kbps / 4,452kbps	110kbps / 11,697kbps
Iteration 8	663kbps / 2,673kbps	167kbps / 5,990kbps	477kbps / 11,697kbps
Iteration 9	1,140kbps / 3,603kbps	110kbps / 4,446kbps	117kbps / 11,700kbps
Iteration 10	507kbps / 4,265kbps	269kbps / 5,923kbps	135kbps / 11,696kbps
Average	**522**kbps **/ 3,713**kbps	**202**kbps **/ 5,071**kbps	**302**kbps **/ 11,696**kbps

Table 2: Bandwidth measurements for Tor vs Standard connections

As well as latencies our study compares how well Tor performs in Bandwidth tests. Table 2 summarizes the throughput speed in every one of the 10 iterations we performed of downloading a fixed 37MB file. In this test we do not take into account circuit creation times, RTT of the initial request or possible failures, we only take into account the time spent downloading once the connection is successfully stablished.



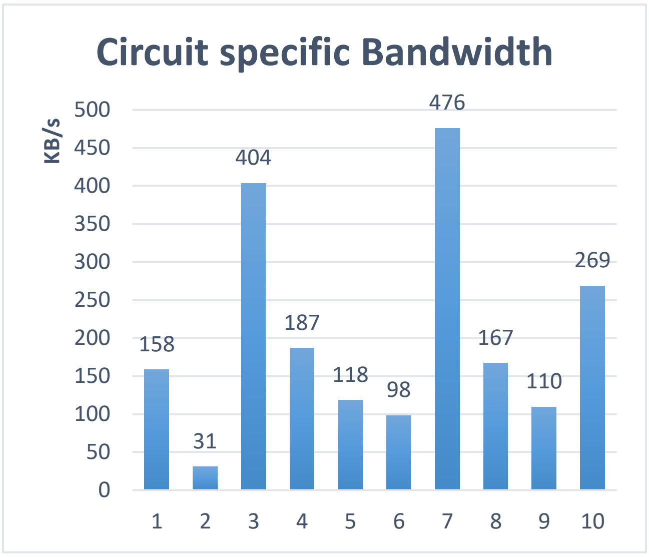


Figure 4: Average Bandwidth and circuit specific Bandwidth

Network Bandwidth is highly dependent on the of connection the host has (Ethernet, Wi-Fi…), the ISP, or even the state of the network in that moment. Tor has also to deal with three more hops with completely heterogeneous conditions which results in consistently slower download speeds (Figure 4). For all our measurements Tor in all iterations has lower speed ranging from 85% to 97% fewer bandwidth compared to a Standard download. On top of that the network topology of the user is not considered, Network 3 has direct access to the requested data but unfortunately the petition still goes under the onion network and the endpoint relay still has to perform a Standard petition to the server.

Further tests indicate that Tor also suffers from great bandwidth variability depending on the created circuit. Some circuits make extremely slow and tedious to download a simple 37 MB file.

Unfortunately Tor cannot be used as a main proxy for general connections to the Internet as Bandwidth critical applications such as music or video steaming need higher bandwidth to provide acceptable QoS while interactive applications such as videoconference or gaming cannot cope with the variability in Bandwidth every circuit provides.

6. Conclusions

Tor is one of the best applications to improve anonymity and security when accessing the Internet and one of its main goals is to provide low-latency and high-throughput communication that supports the vast majority of applications in today’s Internet which in most cases are latency-sensitive or throughput-sensitive. Unfortunately, Tor still has a long road ahead as users still experience variability in delays and bandwidths, high latencies and multiple connection failures.

Using our custom made framework we were able to perform hundreds of experiments to test the latencies and Bandwidth of Tor in very different scenarios as well as provide users a tool to test their own networks and analyze if Tor can provide the minimum QoS for their needs.

Experimental results show that Tor is heavily affected by a great number of failures in the requests which ultimately affects the delays. Circuit creation is highly randomized and as a result many endpoint relays do not have access to the requested web-page or timeouts are reached due to very bad network condition. Bandwidth also suffers greatly from the architecture with very slow download speed in some instances. Common to both Bandwidth and latencies is the variability of the performance depending on the created circuit that directly depends on the network conditions and capabilities of the three relays involved in the circuit.

All in all, Tor is still not ready for applications that require top QoS; low-latency applications such as web-browsing can become extremely tedious with using Tor and high-bandwidth video-streaming is also an impossible task with Tor. All in all, we would only recommend to use Tor for high-confidential and secure tasks involving Internet connections that require high levels of anonymity and for very casual users that do not require high speeds and responsiveness.

As a conclusion we would like to emphasize in the importance of Tor, one of the only applications that can protect anonymity and security in an increasingly insecure Internet, as we strongly believe that and improvement in performance would make this proxy solution even more attractive to users, and as a result, a more protected, free and anonymous Internet.

References

[1] R. Annessi and M. Schmiedecker, "NavigaTor: Finding Faster Paths to Anonymity," URL: ieeexplore.ieee.org/document/7467356/

[2] Yossi Gilad and Amir Herzberg, “Yossi Gilad and Amir Herzberg," URL: freehaven.net/anonbib/cache/tcp-tor-pets12.pdf

[3] Roger Dingledine, Nick Mathewson and Nick Mathewson, “Tor: The Second-Generation Onion Router” URL: svn.torproject.org/svn/projects/design-paper/tor-design.pdf

[4] T. Girry Kale, S. Ohzahata, C. Wu and T. Kato, "Evaluation of dynamic circuit switching to reduce congestion in Tor, “ URL: ieeexplore.ieee.org/document/7366834/

[5] The Tor Project URL : https://www.torproject.org/projects/torbrowser.html.en

[6] Linda Lee*, David Fifield, Nathan Malkin, Ganesh Iyer, Serge Egelman, and David Wagner, ” A Usability Evaluation of Tor Launcher” URL : petsymposium.org/2017/papers/issue3/paper2-2017-3-source.pdf

[7] Øverlier, L., Syverson, P, “Locating Hidden Servers” URL: https://www.onion-router.net/Publications/locating-hidden-servers.pdf

[8] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, Paul Syverson, “Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries” URL: dl.acm.org/citation.cfm?doid=2508859.2516651

[9] K. T. Girry, S. Ohzahata, C. Wu and T. Kato, "Analyzing the drawbacks of node-based delays in Tor," URL: ieeexplore.ieee.org/document/7152451/

[10] Lei Yang and Fengjun Li, "mTor: A multipath Tor routing beyond bandwidth throttling," URL : ieeexplore.ieee.org/document/7346860/

[11] Yoshifumi Manabe,Tatsuaki Okamoto, “Anonymous return route information for onion based mix-nets” URL : portal.acm.org/citation.cfm?doid=1461464.1461466