Performance Analysis of Anonymous Communication Channels Provided by Tor

Andriy Panchenko, Lexi Pimenidis, Johannes Renner
Department of Computer Science – Informatik IV
RWTH Aachen University
Ahornstr. 55, D-52074 Aachen, Germany
Email: {panchenko,pimenidis,renner}@cs.rwth-aachen.de

Abstract— **Providing anonymity for end-users on the Internet is a very challenging and difficult task. There are currently only a few systems that are of practical relevance for the provision of low-latency anonymity. One of the most important to mention is the Tor network that is based on onion routing. Practical usage of the system often leads to delays which are not tolerated by the average end-user. This, in return, discourages many of them from the use of such systems and hence indirectly lowers the protection of remaining users due to a smaller user base. In this paper we show to which extend overloaded nodes and links, as well as geographical diversity of nodes have an influence on the general performance of Tor communication channels. After that, we propose new methods of path selection for performance-improved onion routing which are based on actively measured latencies and estimated available capacities using passive observations of link-wise throughput.**

I. Introduction

With the growth of the digitized world privacy issues get more and more importance. Anonymous communication is a basic fundamental building block for privacy-friendly web browsing, any viable identity management system, privacy-aware eGovernment, eCommerce and eHealth technologies. It is also necessary for providing freedom of speech, mind, and the achievement of democratic principles even in those countries that try to filter and censor access to information. Thus, strengthening privacy-enabled communication can be seen as a major project goal from a social point of view.

Anonymous communication deals with hiding relationships between communicating parties. Without this protection an attacker is able to deduce information about the network addresses of involved senders and recipients. This is often enough to uniquely identify a person. Time, duration, and volume of communications can be used by attackers to infer further information, like e.g. a social relation between two communicating parties.

Many approaches have been proposed in order to provide protection on the network layer. Still, only some of them have been implemented in praxis, e.g. [1], [2]. The most popular and widespread system today is Tor [1]. The Tor network is a circuit switching, low-latency anonymizing network to provide privacy-protection on the network layer. It is an implementation of the so-called *onion routing* technology, that is based on routing TCP streams through randomly chosen paths in a network of routers using layered encryption and decryption of the content. The number of servers in the network is currently about one thousand whereas the number of users is estimated to be hundreds of thousands [**?**], [3].

The Tor overlay network itself is very dynamic. Everybody can join the network and offer available resources for the other users. Today, usage of Tor often leads to significant additional delays caused by the network layers. These delays are often perceived as unnecessary and unacceptable by the end-users, who then choose to continue surfing without Tor. One reason for this is that many nodes have limited bandwidth at their disposal, are overloaded or fail temporarily (e.g. disconnected, switched off). This leads to a small user base, since many users are not willing to sacrifice much of the usability in order to achieve anonymity. One study indicates the existence of an acceptable tolerated latency of about 4 seconds for requesting a website [4]. In another study [5], the authors show differences between polychronic cultures (e.g. Saudi Arabia) and monochronic ones (e.g. Germany) in terms of delay-acceptance during web browsing. Users from polychronic cultures were eager to accept longer delays than those from monochronic ones. For the use of anonymizing systems, another reason for a higher delay tolerance of the users from Saudi Arabia might be the following: due to censorship they have higher incentives to wait longer in order to browse anonymously. The subsequent research [6] shows no considerable difference between tolerated waiting times in different cultures and a linear relationship between increased delays and the drop-out-rate of users in JAP [2].

Since the degree of anonymity provided by such a system is usually linked to the number of active users, the protection for the remainders is reduced, if a significant number of users leave the network. Therefore, the long-term objective of our work is to improve the quality of service of anonymous communication channels, while sacrificing as little as possible of the user's protection. In order to achieve this, we need to study the reasons for the performance downfall first.

Our contribution in this area is the following:

1) we study reasons for performance degradation in the Tor network in terms of:
   - overloaded Tor nodes and links;
   - geographical diversity of routers on a path;
2) we make proposals for improvements in path selection.

II. Fundamentals

This section covers the basic principles of the onion routing technology as implemented in Tor, as well as a description of how the selection of nodes for creating circuits is handled in currently known implementations of the Tor protocol.

A. Onion Routing

The Tor network is an overlay network consisting of single servers that are called *onion routers* (ORs). Currently there are about 1000 ORs in the Tor network that are running more or less permanently. Each OR runs on an Internet end-host and maintains TLS connections to many other ORs at every time. To anonymize Internet communications, end-users run an onion proxy (OP) that is listening locally for incoming TCP-connections to redirect them as *streams* through the Tor network. To achieve this, the OP constructs *circuits* of encrypted connections through paths of randomly chosen onion routers. A Tor circuit, per default, consists of three individual hops, while each hop knows only who has sent the data (predecessor) and to whom it is relaying to (successor). The default circuit length of three hops states a reasonable trade-off between security and performance. To avoid that the last node of a path (*exit node*) learns the first (*entry node*), an additional third node (*middle node*) is used.

Clients choose paths for creating circuits by selecting three suitable servers from a list of all currently active routers, the so-called *directory*. Certain trusted nodes therefore act as *directory servers* and provide signed documents that are downloaded by users periodically via HTTP. Such a network status document contains *router descriptors* of all currently known ORs, including several flags that are used to describe their current states.

During circuit creation, Diffie-Hellman key exchange is used to establish shared symmetric session keys with each of the routers in a path. A proxy encrypts all traffic that is to be sent over a circuit using these keys in corresponding order. While relaying the data, every hop on the path removes one layer of encryption, so that only the exit node knows the actual destination of a stream. Application data is generally transferred unencrypted on the link from the exit node to the destined Internet end-host, unless an encrypted connection is used, e.g. when using TLS/SSL. The operators of exit nodes are in the first instance responsible for any abuse that is done using their nodes, which can be a legal risk in some countries.

Once a circuit is established, the proxy can use it as a tunnel for arbitrary TCP connections through the Tor network, while multiple TCP-streams can share a single circuit. Proxies stop using a specific circuit after a configured amount of time (or data volume), which prevents the users from certain profiling attacks. On the application layer, the SOCKS protocol is used to tunnel arbitrary TCP-traffic through the Tor network. For web-browsing it is further recommended to point a web browser to Privoxy [7], which can be configured to use SOCKS for sending HTTP-traffic over Tor while performing basic application layer filtering.

B. State of the Art in Path Selection

Ideally, all clients would select nodes to be used in virtual circuits in a completely uniform way from the set of all currently active routers. Since the probability to be chosen by clients is the same for all routers, this would offer the maximum achievable anonymity, but at the cost of performance. Without involving any routing metrics, there would be a security-related advantage: attackers could in no way influence the path selection of clients. The disadvantage is, that routers with a weak performance, having very limited capabilities or bandwidth (e.g. modem users) are chosen with the same probability as very powerful nodes having abundant resources.

Beside the legacy Tor software[1], two additional independent client implementations of the Tor protocol exist. One of them is OnionCoffee[2], where the primary goal was to provide network layer anonymity to be used within the EU Project PRIME[3]. Further, the developers of JAP[4] integrated Tor functionality into their anonymizer. For practical reasons, this paper considers default Tor and OnionCoffee only.

Both implementations use a similar bootstrapping process on startup: directory services are contacted to request the list of signed router descriptors. These descriptors include, among other things, current bandwidth information and especially an observed (max-)throughput value for each node. However, the information contained in the descriptors is created by the nodes themselves and therefore can not be considered trustworthy. As soon as enough directory information is gathered, clients begin creating circuits. A client should maintain at least one general purpose circuit in a preemptive way (before it is actually required by any application) in order to save circuit build up time.

As already described in Section II-A, Tor clients distinguish between entry, middle and exit nodes [8]. Node operators need to specify a so called *exit policy* to narrow or prohibit connections to hosts outside of the Tor network. Because of the legal risks of operating an exit node and the high responsibility for any outbound traffic, there are currently less nodes that allow outgoing connections to any Internet end-hosts than non-exit nodes, that can only be chosen by clients on entry or middle positions of paths.

Selecting the first node of a path also puts a major responsibility on it, since as the network entry, it directly learns the IP of the initiator of all traffic. Therefore, the standard Tor client makes use of so-called *guard* nodes. This means that clients keep n (the default is $n = 3$) routers ready that have a high uptime and are known to be fast and stable. One of these n long-term guards is then picked as the entry node for all of this client's circuits. This way it is avoided that clients will eventually end up with a corrupted entry node, when choosing different entries for every new circuit.

[1] http://tor.eff.org/
[2] http://onioncoffee.sourceforge.net/
[3] https://www.prime-project.eu/
[4] http://anon.inf.tu-dresden.de/

The actual selection of the middle and exit nodes is performed in a probabilistic manner where the probability of a router to be chosen for a path is proportional to its advertised bandwidth. To avoid that a router claims to have infinite bandwidth, an upper bound was introduced, that was recently raised from 1.5 MBps to 5 MBps. Exit nodes are considered for entry and middle positions only if the total available bandwidth of exit nodes is at least one third of the overall available bandwidth of all routers. In this case, their bandwidth is lowered in a weighted way in order to not choose possible exit nodes on other positions too often to contribute towards load balancing among the nodes.

OnionCoffee is the first Tor client that introduced a so called *ranking index* for single routers. This ranking index is a number between 0 and 1 that is calculated from the uptime of a node, as well as the average and current bandwidth values parsed from the router's descriptor. Nodes are selected in a weighted probabilistic manner regarding the ranking indices of the routers. Furthermore, ranking indices may be reduced in case of a failing router, to decrease the probability of choosing it again. The end-user is able to specify how big the influence of this ranking index shall be versus a total uniform selection, thus giving the user control over the trade-off between security and performance.

The OnionCoffee Tor client is additionally equipped with GeoIP data which allows to determine the country and continent a router is located in, while selecting the nodes. This makes it possible to put additional geographical constraints on the circuits. It is currently already possible to e.g. exclude nodes in specific countries from being used in circuits, allow only at most one node from the same country, etc.

III. Related Work

Rollyson [9] proposed a method to improve the client latency in Tor by an enhanced path selection algorithm using measurements of latencies between the routers. The proposal requires the Tor directory servers to provide a list of router-to-router latencies that can be consulted by clients when choosing the nodes. The proposed algorithm is limited to picking only the middle node of circuits in an efficient way, because of issues concerning trustworthiness of entry and requirements for exit nodes.

Since Tor directory servers do not provide such a list of link-wise latencies for the clients, the author proposes to use an approximation technique that is based on measuring latencies between responsible DNS servers as proposed in [10]. The quality of the latter is very questionable, because of several reasons: first of all, DNS servers are often located far away from the actual hosts, sometimes even on another continent, considering global ISPs which are common nowadays. Second, Tor nodes may be overloaded, located behind channels with narrow bandwidth, which is not reflected in the latencies between DNS servers.

TorFlow [11] is a multi-purpose framework with the general aim to improve the performance and security in the Tor network. It contains an extended implementation of the Tor Control Protocol [12], a text-based protocol that allows to implement *controllers* that are able to control a running Tor process by listening to events and sending commands. TorFlow-specific extensions include additional features to support path building while preserving arbitrary restrictions on the properties of complete paths as well as single nodes.

Further, the framework contains several controllers that provide different functionalities. These are e.g. capable to scan the Tor network for misconfigured or overloaded nodes. On the long run, TorFlow aims to build an automated, distributed reputation system that should feed into the directory servers to provide them with information on the reliability of nodes.

IV. Performance Analysis

The main goal of this performance study, and thus our main contribution, is to find out what are the bottlenecks in the Tor network, as well as to learn about the overall situational behavior of the network and limits of nodes regarding various performance metrics, especially latency and throughput. Such a study can further serve as an input for the design of methods that generally improve the performance of onion routing[5] by the means of new path selection algorithms.

This section is structured as follows: at first, results from a performance analysis that was done in a private Tor network will be presented. This will give the possibility to study the limits of various performance metrics, as well as the overall situational behavior of onion routers in an optimal environment. Second, we will study the performance of Tor "in the field" – the real public Tor network. To this end we measured the time required to establish single circuits, as well as average RTTs and throughput when using different configurations and path selection methods. Furthermore, interdependencies between the different performance metrics will be studied.

Our experimental setup of a private Tor network used Intel Pentium III Dual Core machines with 1Ghz CPU and 2 GB RAM on the nodes, while two of the existing Tor client implementations were evaluated: default Tor and OnionCoffee. The local backbone is 100Mbps, while using a 10Gbps connection to the Internet. It should be noted, that the processing in Section IV-A is completely CPU bound, whereas those in Section IV-B can be either CPU or network bound.

A. Load Implications on the Performance of Tor

In order to study the implications of load on the behavior of Tor nodes, we performed measurements in a private Tor network. Hence we were able to determine important values like e.g. the maximum possible throughput of a single node, influence of circuit setup durations on the throughput, etc. Having knowledge about such values is of great importance for further refinements of the used path selection methods. So, for example, knowing the maximum possible throughput and the average time that is needed to establish a circuit, it is possible to make conclusions about the load of nodes and/or the condition of the links between them. Further, it does not make

[5] Onion routing and Tor are used as synonyms here.

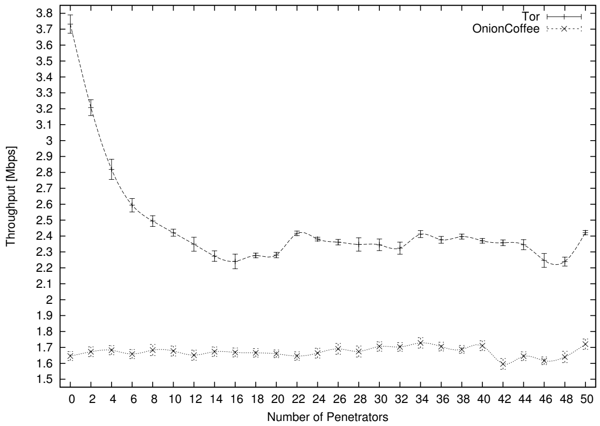

Fig. 1. Influence of Penetrators on a Node's Throughput

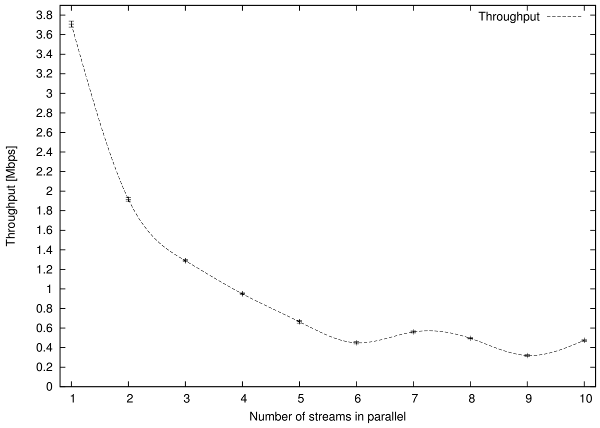

Fig. 2. Stream Throughput

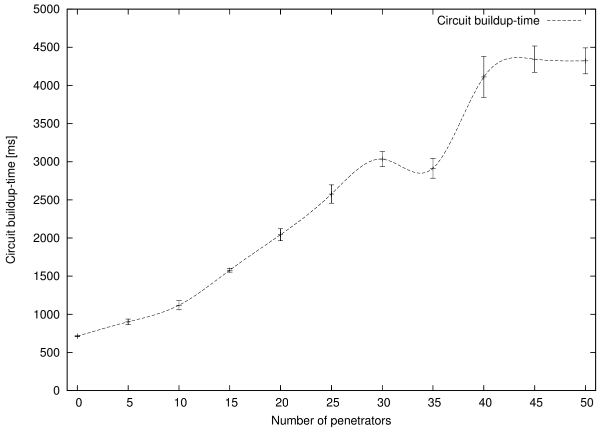

Fig. 3. Circuit Setup Duration

sense to believe nodes that are advertising a higher bandwidth value than the maximum in an optimal environment, which is represented by our scenario. Similarly, the default Tor implementation currently clips advertised bandwidth values at 5 MBps.

1) Throughput under Load: The first experiment that was performed, shows the throughput of a Tor node in dependence on the number of penetrators that are using this specific node for creating circuits (see Figure 1). A client continuously downloads a stream, while other nodes are steadily establishing new circuits involving the node.

The achieved throughput when using the standard Tor client decreases with the growth of the number of penetrators until their number reaches about 14. After that, no further implications on the throughput of a node can be seen. This results from the performance of the circuit establishment operations (which includes expensive public key cryptography) in a separate thread. The throughput reached by OnionCoffee is below the throughput of the standard Tor client[6]. This is due to the fact that the efficiency of cryptographic operations implementations in the programming language C is much higher than those written in Java. This was confirmed by running OnionCoffee in a profiler, where it showed up that most CPU time was spent in the cryptography library. Due to the performance issues in the client and not the server, the throughput of the stream using OnionCoffee was not affected by the employed number of penetrators.

The results from the next experiment that was conducted in the private Tor network, show the throughput reached by a client while a number of other clients perform a download as well, using the same set of nodes. In Figure 2 it is interesting to see that the throughput of the measured stream drops down to 0.5 Mbps when there are 6 streams in parallel, but it remains nearly constant with the further growth of up to 10 parallel streams.

2) Circuit Establishment: In the last experiment we were interested in the time that is required to establish a circuit in an optimal environment (using the same private Tor network). We measured the time required to establish a circuit consisting of three nodes, while having penetrators doing the same task. This means clients establishing circuits using the same nodes and closing them right away after the establishment was successful. Figure 3 shows the time needed for establishing a circuit consisting of three nodes. In order to calculate the mean values together with 95% confidence intervals each experiment was repeated 50 times.

[6]Please note that the Y-axis in Figure 1 does not start at zero.

B. Experiments in the Real Tor Network

The following experiments were conducted in the real Tor network in order to measure the performance of Tor "in the field". To this end, we performed measurements of throughput, latency, and setup durations of circuits. We further paid also attention to the correlations between different performance metrics. A strong correlation might possibly enable us to make conclusions about a circuit's latency or throughput from its setup duration only.

1) Circuit Establishment: Table I shows general statistics on circuit establishments using nodes that were chosen uniformly. The row containing *create* lists the median, mean times, standard deviation and min/max values that were needed for the initial creation of a circuit involving the first hop only. *extend* 1 lists the average time needed to extend a circuit to its second hop, and *extend* 2 the extension to the third hop. *total* provides the overall time needed to create a circuit consisting of three hops. The time is summed up since Tor builds circuits in a consecutive way, extending them from hop to hop. The latter is due to the need of establishing symmetric keys during the circuit setup between the initiator and the onion routers on the path.

	Median(s)	Mean(s)	Stddev(s)	Min/Max(s)
create	0.58	1.18	2.65	0.11/32.89
extend 1	1.23	2.24	3.63	0.10/45.43
extend 2	2.23	4.35	6.54	0.10/48.80
total	4.36	7.78	9.20	0.46/60.72

TABLE I
STATISTICS ON SINGLE STEPS OF CIRCUIT ESTABLISHMENT

Table II provides results of an experiment where setup durations of circuits using paths consisting of two to four hops were measured. Besides the actual durations, also the fraction of failed attempts is depicted. About 30 to 40% of the overall circuit establishment attempts failed in our experiment, using uniform selection of the nodes. The reason for this is that nodes are overloaded or already left the network while their descriptors are still available.

	Median(s)	Mean(s)	Min/Max(s)	Failed(%)
2-Hop	2.13	4.22	0.22/36.15	35
3-Hop	4.99	6.83	0.25/49.21	32
4-Hop	7.18	9.65	0.73/38.37	38

TABLE II
STATS ON CIRCUIT SETUPS USING DIFFERENT PATH LENGTHS

2) RTT and Throughput Comparison: The average RTTs of circuits using different path lengths were also measured and are depicted in Figure 4. It is possible to see, how the average RTT is growing with the increasing hop count. For the standard Tor path length, involving three single nodes, the mean RTT measured in our experiment is about 1.6 seconds, whereas for 2-hop paths it is 1.1 *s*, and for 4-hop paths 1.9 *s* seconds. Note that uniform node selection was used for creating all of the circuits in this experiment.

For achieving the maximal degree of anonymity, routers would have to be chosen in a completely uniform way, as already described in Section II-B. For being able to provide an improved performance, both considered onion routing implementations – Tor and OnionCoffee – perform path selection in a probabilistic way regarding bandwidth information provided by the nodes themselves through their descriptors. Therefore, nodes that are advertising a higher bandwidth are chosen by clients with a higher probability respectively.

Note that if path selection is done depending on information advertised by the routers themselves (e.g. in a probabilistic way regarding advertised bandwidth), attackers are able to

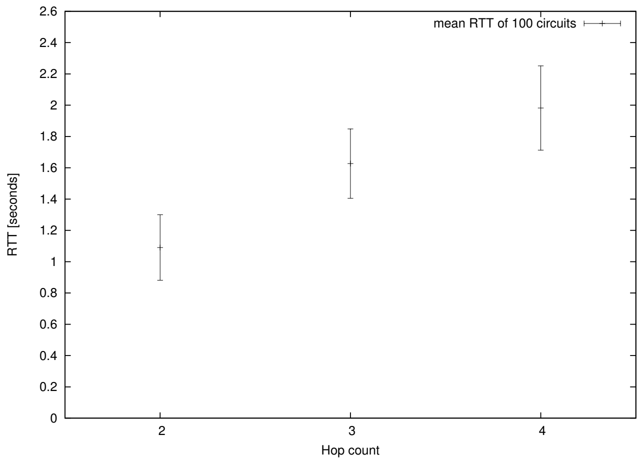

Fig. 4. RTT Comparison of Circuits Using Paths Consisting of 2-4 Hops

influence the selection of clients by advertising a very high value. Ideally, any newly proposed methods would offer no possibilities for attackers to influence the path selection of clients at all. Otherwise, additional mechanisms will be necessary to somehow minimize the influence of attacker nodes on the path selection mechanisms of clients. Any such implications of the used methods have to be studied and a reasonable tradeoff between the quality of protection and quality of service has to be found, before any improved methods will become mature for real life usage.

Table III compares results that were gained by creating at least 600 circuits with each of the methods. The table shows mean measured setup durations, RTTs and throughput of all circuits together with standard deviations (*mean/stddev*). It can be seen that choosing paths weighted by advertised bandwidth values generally leads to significantly better results regarding all metrics.

	Setup(s) *[mean/stddev]*	RTT(s) *[mean/stddev]*	Throughput (KB/s) *[mean/stddev]*
UNIFORM	6.31/6.66	1.61/1.16	11.04/14.55
WEIGHTED	3.33/3.91	1.09/0.98	45.33/69.67

TABLE III
COMPARISON OF UNIFORM AND WEIGHTED PATH SELECTION

3) Correlations of Performance Metrics: Further, it is of great interest to find out whether there is a correspondence between the time needed to establish a circuit and other performance metrics (like a circuit's measured bandwidth or latency). If significant correlations could be found, it might be possible to e.g. conclude from a circuit's long setup duration, that its latency will not be acceptable in order to satisfy a user's current requirements and the user might wish to directly omit using this circuit in the favor of another one.

Figures 5 and 6 show results from an experiment where 1530 circuits were created, partly using uniformly and partly weighted probabilistically chosen paths. The performance of every created circuit in terms of the setup duration, RTT and throughput, was actively measured directly after the creation of a circuit. The results from this experiments show a medium positive correlation of circuit setup durations and the averaged latencies of the circuits. The sample correlation coefficient (Pearson product-moment correlation coefficient) of the considered values is 0.47. Apparently circuits that were established very quickly, seem to generally deliver lower latencies than those having experienced long setup durations.

There is, however, only a small negative correlation between the throughput and RTTs of circuits. This means that one can generally not deduce that a circuit's low latency indicates high throughput rates and vice versa. The computed sample correlation coefficient is -0.34, while the correlation coefficient of setup durations and measured throughput is -0.16, which can only be interpreted as a very small negative correlation.

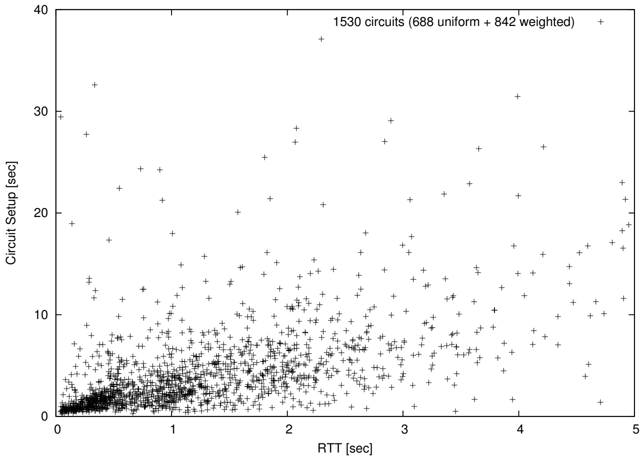

Fig. 5. Correlation of RTTs and Setup Durations

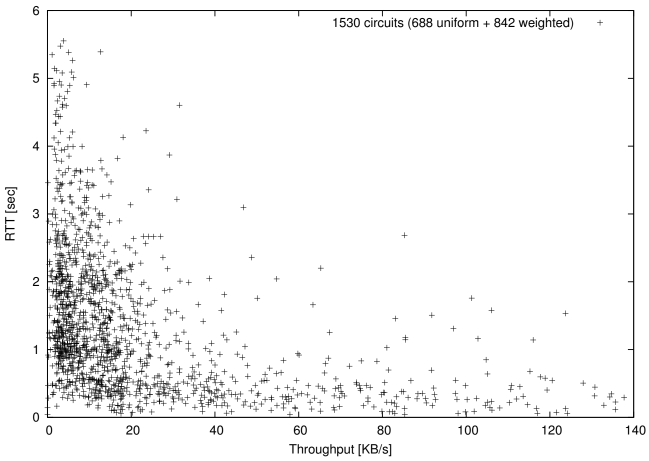

Fig. 6. Correlation of Throughput and RTT

4) Influence of Geographical Diversity: It is also of interest to study the influence of geographical diversity of routers on the path on the performance. Nodes that are located in geographical vicinity could be chosen in order to improve the experienced network latency. A reasonable application of location-based path selection would be to let the user choose the specific countries where entry and exit nodes of circuits should be located in. With performance in mind, it could also make sense to choose an entry node in the sender's country, while choosing an exit node that is located in the stream destination's country.

We have evaluated algorithms that are based on the location of the used routers in terms of their countries or continents. For comparing the performance of the different algorithms, 300 three-hop circuits were created, employing each of the single restrictions once, while the latency of every circuit was measured five times. From these five results, a mean value was computed, while the average value of all the single means finally represents the latency of circuits created with a method. The different tested restrictions are in fact:

- Use no geographical information at all (UNIFORM)
- All routers from different continents (*UniqueContinent*)
- Routers from a single continent (*SingleContinent-EU*)
- Routers from the same country (*SingleCountry-DE/US*)

Germany and the United States, as well as the continent Europe, were specifically chosen as examples since it can be assumed that globally the largest amounts of Tor nodes are located in Europe and North America, specifically in Germany and the United States. All of the circuits created with SingleCountry-EU contain european nodes only, starting with an entry node in Germany while the nodes for the other positions were chosen randomly from all european nodes, without ever using two nodes from the same country in a path. For SingleCountry-DE and -US, only nodes in Germany were chosen, respectively in the United States.

Figure 7 shows the results of the latency tests using different path selection methods. As expected, the UniqueContinent restriction delivered the overall worst latencies while SingleCountry-DE resulted in very fast circuits. All of the measurements were done from within Germany during the day (between 10am and 4pm in the afternoon).

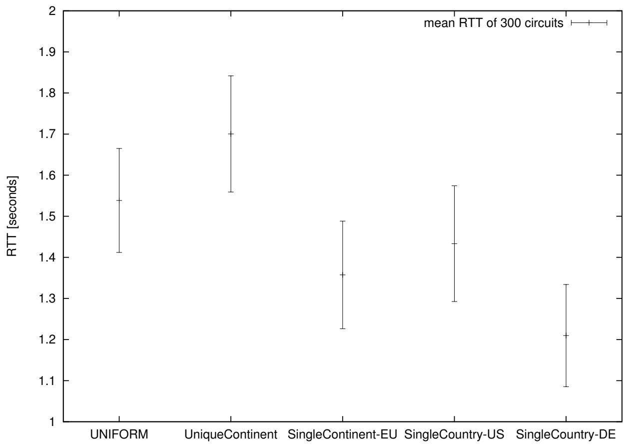

Fig. 7. Comparison of GeoIP-Based Path Selection Methods

The additional Table IV shows global statistics about circuit creations with each of the methods: the median, average and min values are related to the time needed to establish a circuit. Note that 60 seconds is the hard coded default timeout on the circuit building process in the default implementation and therefore all successfully created circuits needed less than a minute to be created. The max durations for setups are of minor interest here, since all of them are shortly below one minute. The last column lists the number of circuits in percent, that timed out during their setup.

	Median(s)	Mean(s)	Min(s)	Failed(%)
UNIFORM	4.54	6.64	0.20	25
UniqueContinent	5.15	8.30	0.79	38
SingleContinent-EU	3.20	5.73	0.20	30
SingleCountry-US	3.63	6.80	0.39	22
SingleCountry-DE	2.72	4.92	0.16	25

TABLE IV
STATS ON CIRCUIT SETUPS FOR GEOIP-BASED METHODS

From this table it can be seen, that SingleCountry-DE delivered the lowest values in median, average and min regarding the circuit-setups, as well as the measured latencies. This lets us assume on the one hand, that geographical non-diversity of the used routers eventually leads to better latencies. The worst results were gained by the UniqueContinent restriction, as it would be expected. On the other hand this shows a positive correlation between setup durations of circuits and their performance in terms of latency.

From the last column it can be seen, that a high geographical diversity of the routers (UniqueContinent) apparently leads to a higher percentage of circuits that timeout or otherwise fail during the setup process, at least when using the default timeout of 60 seconds.

We have seen that by ensuring a low diversity of the used routers it is possible to lower the latencies of circuits, but one always has to keep in mind that the geographical diversity of the nodes in a path is an important substance to the security of the system. Using nodes in different countries involves different jurisdictions, and thus, increases the protection of users. Additionally, choosing nodes in a same country increments the risk of choosing nodes belonging to the same operator. Therefore, it is possible to increase the security of created circuits by ensuring a high diversity of the nodes in a path. Improvements to the performance of communications should rather be achieved otherwise, e.g. by using measured performance values as routing metrics.

5) End-User Perspective: In order to learn the impact of Tor anonymization on the web browsing performance for end users, we measured the average time needed to fetch HTTP-headers of 100 of the most popular websites[7] on the Web. Table V shows the averaged median and mean values of the time needed to fetch a single HTTP-header, as well as the averaged standard deviation and min/max values. The first row contains the results from 100 tests using the default Tor implementation, while the second row shows results from requesting the same headers without anonymization.

	Median(s)	Mean(s)	Stddev(s)	Min/Max(s)
Tor	3.35	4.04	3.18	1.34/23.33
No Anonymization	0.26	0.39	1.25	0.01/11.37

TABLE V
AVERAGE TIME NEEDED TO FETCH HEADERS OF POPULAR WEBSITES

V. DISCUSSION AND PROPOSALS FOR IMPROVEMENTS

The previous section clarified the inhomogeneity of the Tor network. This is mostly due to the volunteer-based structure that does not treat nodes differently depending on their available resources. For being able to provide a certain performance it is therefore necessary to quantify the nodes w.r.t. different metrics. This section will use the results from Section IV to propose new methods of path selection to be used in Tor with the aim of improving the performance for end-users. The ideas are generally based on measuring different performance parameters in the Tor network and using the results as routing metrics when choosing nodes for circuits.

A. Latency-Based Path Selection

The Tor protocol does not currently provide any mechanisms to measure *round-trip times* (RTTs) of the provided anonymous channels. For sticking with a low latency, it would be very helpful to be able to actively measure latencies of circuits, as well as of the virtual Tor links between the single routers.

Our prototypical implementation measures RTTs of circuits by violating the exit-policy of the last router in the used path[8]. This is done by sending a *relay connect* cell on the circuit that is to be tested, using *127.0.0.1* as a *dummy*-destination. Since the exit policies of all routers will deny connections to *localhost*, this attempt results in an error that can be timed in the measuring client. Making use of Tor's *leaky-pipe* circuit topology, it is even possible to extend this technique for measuring RTTs of partial circuits. These can then be used to calculate link-wise RTTs between the single Tor routers in a path. By modifying the number of encryption layers when initiating a stream, one can specifically address every single hop of a circuit as the exit node. After performing measurements addressing every hop of a circuit once, link-wise RTTs can be calculated using

$$RTT_{n-1,n} = RTT_{0,n} - RTT_{0,n-1}$$

Note that even if this method delivers quite exact results, it is proof-of-concept code that should eventually be replaced by mechanisms that would need to be integrated into the actual Tor protocol and make use of timestamps to measure link-wise RTTs.

[7]The URLs include the top 50 websites from traffic rankings of Germany and the USA regarding to *http://www.alexa.com/* on 9 Jul 2007.

[8]This method was also used to measure RTTs in Section IV.

For making use of the measured results it is proposed to model the explored subnet of the Tor network in a graph structure. This *network model* can contain nodes, links between these nodes and arbitrary *node-wise* or *link-wise* performance metrics. All of the supplied metrics, as well as additional information from the descriptors, can be combined to calculate *ranking indices* for either nodes, links or even complete paths, that influence the actual selection of paths. Path selection can then be done probabilistically from this model regarding the ranking indices, as it is already done in OnionCoffee (see Section II-B). It is possible to control the specific influence of a certain metric on node selection by introducing additional factors/weights.

If path selection is done in this way, it will be more difficult for attackers to cheat in a way that their nodes are chosen more often than others, as it is today. Experiments have to be done in order to study the impact of paths created in a probabilistic way regarding measured RTTs of single links to see performance regarding setup durations, latency and throughput of circuits. Given the possibility to measure latencies of complete circuits, it is also possible to optimize load-balancing on the Tor network by ensuring in the clients that user streams are always attached to circuits currently having low latencies.

B. Throughput-Based Path Selection

For performance-based routing, *throughput* should also be considered to be used as a metric. Instead of using node-wise bandwidth information taken from the router descriptors, it would be an advantage to measure throughput for being able to more precisely predict the capacities of specific nodes or links. Measuring throughput actively by transferring streams over certain nodes to probe their capacities is definitely too much overhead that would have negative impacts on the overall network performance. Therefore we propose to measure throughput *passively* from within the nodes, but consider the single TLS links to other routers on the network, instead of globally counting the total amount of Tor traffic that is passing a single node.

C. Performance Directory

Both of the techniques that are based on measurements are sensitive to even short term load variations, which is of importance in a highly dynamic network like the Tor network. A network model containing measured RTTs, as well as the currently available link-wise bandwidth capacities, could be provided by a trusted directory that can be downloaded by clients in a compressed format. Alternatively, clients could also distribute such a model between themselves while merging information into it.

VI. Conclusions

In this paper we studied the influence of several disturbing factors on the general performance of Tor circuits. Hence it was possible to find out limitations of nodes regarding various performance metrics, especially network latency and throughput.

We have compared the average latency of circuits created with the currently used method of path selection to uniformly chosen paths to measure the achieved improvements. Thus it is possible to justify the loss of anonymity introduced by the probabilistic path selection. Additionally, the influence of the length of paths on the performance of data transmissions was studied. Also we have shown that there is a medium positive correlation of circuit setup durations and the average measured latencies of the respective circuits.

Further, we have shown that by ensuring a low diversity of the routers in a path it is possible to lower the latencies of circuits, while one always has to keep in mind that the geographical diversity of the nodes in a path is an important substance to the security of the system. We therefore propose the integration of a geographical component into Tor clients that can be used in order to define lower and upper bounds of location diversity in paths, as well as any other geographical restrictions.

Finally, we have proposed new methods that are based on active measurements of circuit latencies and passive throughput estimations in order to improve the overall performance of anonymous communication channels provided by Tor. Still, the latter need to be implemented and practically evaluated. Special emphasis has to be placed on possible implications of new methods on anonymity and security of the system.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *Proceedings of the 13th USENIX Security Symposium*, 2004.

[2] O. Berthold, H. Federrath, and S. Köpsell, "Web MIXes: A system for anonymous and unobservable Internet access," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 115–129.

[3] "Tor Documentation," https://www.torproject.org/documentation.html.

[4] R. Wendolsky, D. Herrmann, and H. Federrath, "Performance Comparison of low-latency Anonymisation Services from a User Perspective," in *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, N. Borisov and P. Golle, Eds. Ottawa, Canada: Springer, June 2007.

[5] G. M. Rose, R. Evaristo, and D. Straub, "Culture and Consumer Responses to Web Download Time: A Four-Continent Study of Mono and Polychronism," in *IEEE Transactions on Engineering Management*, vol. 50, no. 1, Feb 2003, pp. 31–44.

[6] S. Köpsell, "Low Latency Anonymous Communication - How Long Are Users Willing to Wait?" in *ETRICS*, ser. Lecture Notes in Computer Science, G. Müller, Ed., vol. 3995. Springer, 2006, pp. 221–237.

[7] "Privoxy: Filtering Web Proxy," http://www.privoxy.org.

[8] R. Dingledine and N. Mathewson, "Tor Path Specification," https://www.torproject.org/svn/trunk/doc/spec/path-spec.txt.

[9] S. Rollyson, "Improving Tor Onion Routing Client Latency," Georgia Tech College of Computing, Tech. Rep., 2006.

[10] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating Latency between Arbitrary Internet End Hosts," in *Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002)*, Marseille, France, November 2002. [Online]. Available: citeseer.ist.psu.edu/gummadi02king.html

[11] M. Perry, "TorFlow," https://www.torproject.org/svn/torflow/.

[12] R. Dingledine and N. Mathewson, "Tor Control Protocol Specification," https://www.torproject.org/svn/trunk/doc/spec/control-spec.txt.