Traffic Optimization in Anonymous Networks

Patrik Kristel
Faculty of informatics and information technologies
Slovak University of Technology in Bratislava
Bratislava, Slovakia
patrik.kristel@onedata.sk

Ján Lučanský
Faculty of informatics and information technologies
Slovak University of Technology in Bratislava
Bratislava, Slovakia
jan.lucansky@stuba.sk

Ivan Kotuliak
Faculty of informatics and information technologies
Slovak University of Technology in Bratislava
Bratislava, Slovakia
ivan.kotuliak@stuba.sk

***Abstract*—Anonymous communication networks, such as Tor, are facing big challenge how to deliver content to users in low latency and with no interruption. The latency issues were caused by increased amount of transferred data and low-bandwidth nodes in Tor network. Those are limiting overall circuit capacity providing to users. Conflux, the Tor plugin, is improving effort and decreasing latency time by creating multipath within Tor circuits. Conflux is doing dynamic traffic-splitting and load-balancing through multipath to improve throughput and avoid bottlenecks in Tor circuits. Our solution is focusing to analyze and modification network flows and sessions in Tor network. As an output of problem's analysis we're proposing possibilities to improve Conflux's performance by its modification and deploy to the Tor network. The paper describes solution implementation, setup and configuration Tor client and Tor exit node. We're explaining necessary modifications that need to be provided on Tor components. Our solution achieved average improvement versus Conflux more than 20% decrease of download time in various file size.**

Keywords— Anonymity Networks, Tor network, Bottlenecks in Networks, Network Traffic load-balancing

I. INTRODUCTION

Tor network is hiding source or destination address and help to protect end user and other communication instances like devices or equipment's to against their identity in the network. Low-latency anonymous systems, such as Tor, are designed for applications which are operating in real-time. As a real-time applications increased their popularity in last couple of years by increased amount of users and amount of transferred data, it caused performance issues in Tor network's nodes bandwidth.

Currently, Tor network has over two million active users [2] and over 7,000 active nodes as of April 2017 [3]. Tor is very popular mainly in countries where access to Internet is restrict or where is strong censorship and users cannot web-browsing freely. Countries as China, Russia or countries of middle-east. By using Tor, users can avoid the censorship in effective way. In another cases we just don't want uncover our real Internet identity, location and avoid collecting data about ourselves by various web services.

Amount of users are increasing as well as amount of servers but nowadays communication, such as real-time or streaming need to their functionalities definitely more bandwidth capacity then network applications couple years ago. Regarding to statistics we can see while in year 2014 (January) was transferred about 50 Gbit/s so in the year 2017 (January) it was a bit more than 200 Gbit/s. It means about fourfold transferred data increased [4]. Obviously, it could cause an issue in Tor performance near future.

Tor nodes are running on volunteer-based platform and most of the nodes are not connected with high speed Internet link with enough bandwidth capacity. The overall circuit bandwidth is limited by the low-bandwidth node which is participating on circuit. Means, if user creates a circuit through the Tor network and for some of them use same low-bandwidth node it will be limited by the low-bandwidth node even if the exit node will be super-fast. [1]

Conflux is Tor network tool which provides better stability and performance for Tor users. Conflux is creating multipath inside Tor network to divide and load-balance amount of data transferred from client side to server side or vice-versa. Amount of the data (traffic) is divided by dynamic traffic-splitting algorithm. It is ongoing client side and exit node, those are shared within multipath. During all the time when data are transferred conflux is load-balancing data flow base on latency measurement each of the created path.

Measurements and evaluation prove approximately 30% improvement in expected download time [1]. Based on prove facts we can state it significantly improves the experience of streaming data's users.

II. TOR NETWORK OVERVIEW

A. How Tor Network works

Tor network consist of three types of nodes: onion routers (nodes), onion proxy (clients) and directory servers. Directory servers consists of a few machines which stored *network consensus* file, a database of all active nodes with their parameters. Nodes are volunteer-based servers across the world which are backbone of Tor network. They're responsible for all traffic passed by or to users through the Tor. Nodes are only instance which using to communicate with 'other' world, outside of Tor. Clients are running on end-users machines and they're 'consumers' of data and data streams transferred in Tor.

Clients are building circuits across of Tor network to reach their destination (server or other side of communication). First of all, clients connect to the directory server, then load the network consensus file and start to establish new connections – circuits. Circuits are Tor network paths through which the client's traffic is transferred and routed. Traffic data are relayed in fixed-sized (512 byte) units called cells [5].

The onion proxy can also multiplex several TCP streams over one circuit, which generally has a lifetime of ten minutes. To ensure flow control of data in flight, Tor employs an end-to-end window-based flow control mechanism in which every time the OP (or exit OR) receives 100 cells, it acknowledges windows of data cells using SENDME (or acknowledgement) cells. [6]

B. Improving Tor's quality of service

If we take a look to various video streaming or file sharing services, we will see much more data flooding across Internet nowadays. For example, in last four years average size of web page increased from 1000KB in 2012 to 1620KB in 2016 [7]. Which means 60% increase over just four years. In a future we can expect web page's size increasing as well and Tor need to be adapt to the changing web.

Currently, Tor project not recommends to use Tor network to any video streaming, file sharing or torrent using [8]. Although it hard to separate streaming from web browsing. Therefore, in order to survive and continue to attract new users, it is crucial for Tor to meet the demands of its users and the changing web by improving the experience for streaming users.

III. CONFLUX AND ITS MODIFICATION

A. Multipath creating and Dynamic Traffic splitting

As we mentioned, Conflux is creating parallel path to already exist primary path in Tor network. Primary path is constructed by Tor's client according to the bandwidth-weighted router selection[1] [9]. There is only one constrain while Conflux is creating multipath that exit node must be same as exit node linked with primary circuit. Entry and middle nodes are different each other and they're selected by bandwidth-weighted router selection algorithm. After both paths are established, client sends a "multipath cell" through both circuits (paths) to the common exit node. This process enables the exit router to associate the OP's TCP streams with its linked circuit [10]. After that client use both circuits to send and receive data through Tor. Closing a multipath is no different from closing circuits in Tor. If a circuit in a multipath exceeds its lifetime or if it is idle, the circuit is turned down. Closing one circuit it not affect other, as client builds a few circuit more to spare. In fact, it also not required any higher load on client side, as Conflux using circuit build in spare [10].

After multipath is created, there is a mechanism to dynamically split traffic to two sub-streams. The splitting end points assign different amount of traffic to each circuit. Algorithm is design to load-balancing traffic between circuits based on latency inside them. It is observing current throughput relative to other circuits [10]. The algorithm works as first splitting point (Tor client or Exit node) is frequently measuring latency on all circuits. This can be done by storing time every 100^{th} cell is sent down a particular circuit, noting the time that the corresponding circuit-level *SENDME* arrives. Splitting end points are allow to compute current round-trip time (RTT) based on this. The list, where the RTTs are stored is periodically updated regarding to measurements assigned to each linked circuit [10].

Based on mechanism describe above, every time when data need to be transmitted through multipath circuit, algorithm choose one with best current RTT.

B. Conflux vs. pure Tor

Conflux was developed primary for clients using low-bandwidth nodes such as bridges. Bridges are specific types of Tor nodes. They're not distributed by standard way, through directory servers, but just on demand. Their mainly use in countries with a strong censorship [11], where access to Tor network nodes is blocked by default. Bridges are typically lower-bandwidth as normal Tor nodes. This is one of the reason to build tools such as Conflux.

Conflux has significantly better improvement in performance than pure Tor. Based on present results it reaching an improvement of approximately 30% in expected download time for web browsers who use Tor bridges and for streaming application users [10]. The results were taken while Conflux was ran with one primary and one secondary circuit, while it divided traffic to two parts based on dynamically splitting traffic algorithm, what we mentioned above.

C. Our approach-ConfluxM

Our approach is based on question: How to do traffic-splitting even more effective? If we take a look to the Conflux's design, we can find scheme how traffic is divided. We can assume if we extend number of circuits in multipath and we will be able to divide traffic to three independent sub-streams in the first end point (and collect in the other end point), we can reach more improvement in same scenarios.

[1] Bandwidth-weighted router selection algorithm is intending to improve the overall system performance in Tor network. It is trying to select best path in created Tor circuit [9]

Figure 1 below shows our re-design of Conflux. We're adding one more parallel circuit in to the multipath. In this scheme Tor client will start to build a circuit from primary path (through *Entry1*, *Middle1* and *Exit* nodes) by ordinary Tor's way, after that client will build secondary path by Conflux's way (*Entry2*, *Middle2* and *Exit* nodes), and finally we will build one more path from client stand point (*Entry3*, *Middle3* and *Exit* nodes). All traffic will be handled by dynamic traffic-splitting algorithm to load-balancing data flow through all three paths. On the other side of communication will be buffer proceeding [10] running by Conflux to collect traffic back together in same order as it was split.

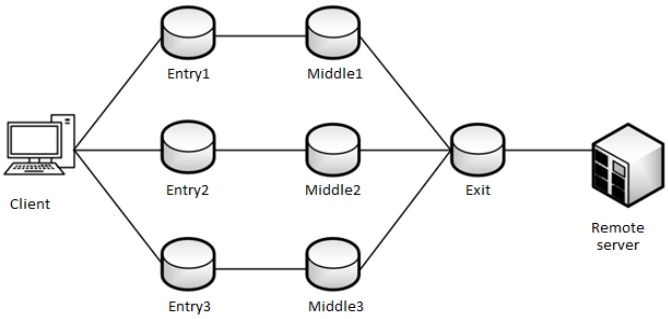

Fig. 1. Live performance comparison between Tor, Conflux and ConfluxM for 400 KB file

Our approach assuming that our Conflux's re-design will affect mainly Tor client and *Exit* node. Both will divide and split traffic on their side. *Exit* node will also connected with remote side of communication, same as in Tor.

The dynamically traffic splitting process will be same as Conflux implemented. Once we get all three paths up and ready we will send multipath cell to test their availability [10].

Our approach is based on the recommendation as a future work in publication: The Path Less Travelled: Overcoming Tor's Bottlenecks [1] and motivation to find relevant difference and relation between Conflux and our approach.

D. Implementation details

We have run this implementation in private Tor network builds with a several Tor nodes, as two of them were run as Tor directory servers and a Tor client. All nodes have communicate to each other through TCP connections build just after private Tor network's convergence was done [12] [13]. Tor client and all nodes were running on separate Linux machines and on same Tor version (0.2.9.5-alpha) with the Conflux code. We did modification Tor and Conflux source code in order to reach design requirements presented in part of our approach.

IV. PERFORMANCE EVALUATION

Conflux's paper presented improvement of download time average at least at 30%. There are also presented partial improvement results for small-size files (300–400KB) it was 42% and for larger-size files (1–10MB) improvement was about 25% of decreased download time.

In our approach we're focusing to compare our result with presented ones [10], but off course we cannot simulate and compare results on same topology, with same scheme. We are conscious there will variance caused by different implementations setup what we used. Accordingly, we will do first at all same measurement as was presented by Conflux's paper, in order to get as much accrued results, as we can.

In generally, we've setup local remote server outside of private Tor network's nodes, but in directly connected network. On remote server we've run file server, which we kept running and serving files "forever". On the other side, we have setup file application which was connected through ORPort [14] on Tor client machine.

In our measurements we have choose samples of test files of 400KB, 5MB and 10MB. File sizes were taken to maximal our effort to approximate web browsing and download (or streaming) users [7] [15].

All the values in measurements, what we're presenting on Figures 1 and 2 are averages of series of measurement. Each value represent average of 10 measurements, according to implementation setup.

A. Small-size files evaluation

Firstly, we took sample of 400KB test file to evaluate time for Tor, Conflux and ConfluxM setup, each of them separately. Figure 2 below shows average improvements what we get once we did all measurements. From the graph we can see Conflux and ConfluxM time decrease improvement against Tor. We have reach average improvement of 23% (Conflux), 12% (ConfluxM) and 33% (total) decrease of downloaded time.

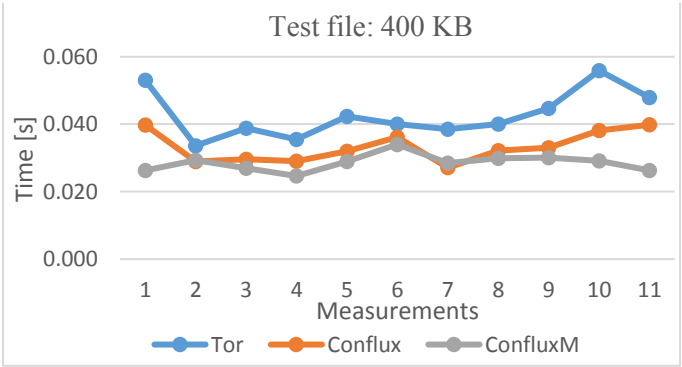

Fig. 2. Live performance comparing between Tor, Conflux and ConfluxM for 400 KB file

From the graph we can also see that time decreasing, if we compare Conflux and ConfluxM, the improvement is not that significant as in Tor vs. Conflux case. From this statement we can assume, if we add even one or few more parallel path, we cannot reach so much improvement as in Conflux case.

B. *Lagrer-size files evaluation*

Next, our attention will shift to larger-size files. We took two files of size 5MB and 10MB. These two file samples should good approximate a video stream on most wide streaming service, as YouTube or Netflix [16]. Figure 3 below shows comparing downloaded times for both samples. The main improvement we have reach for files size of 5MB. We reach up to 69% (total) decrease of downloaded time against Tor setup. For the partials results, we have reach average improvement of 57% (Conflux) and 28% (ConfluxM).

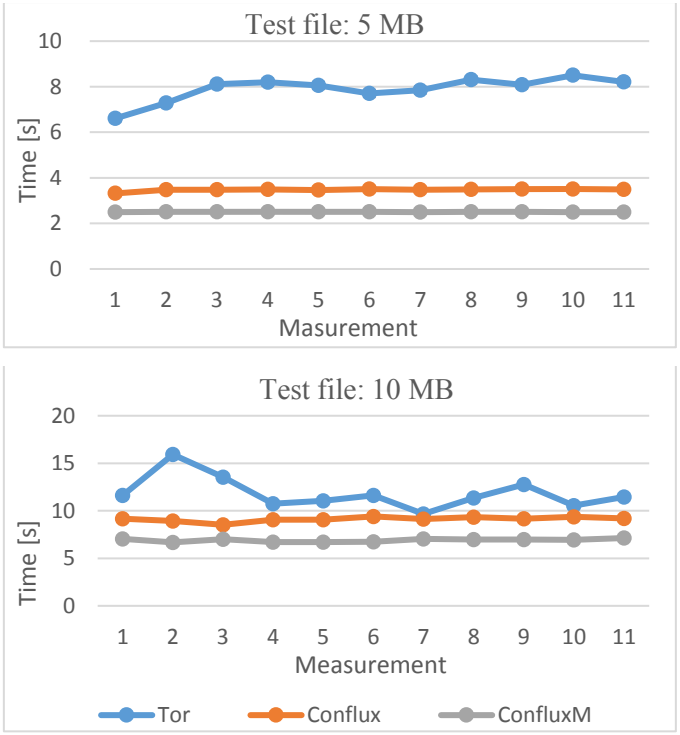

Fig. 3. Live performance comparing between Tor, Conflux and ConfluxM for 5MB and 10MB file

As we can see, average improvement was much better for the files sizes of 5MB then 10MB. On the other side, both Conflux and ConfluxM setup times of download were much stable without any unexpected peaks. We achieved that by dynamically traffic slitting to two or three paths in whole multipath. Basically, we have split all the traffic load by ongoing measurements during Conflux or ConfluxM was ran. Another advantage, what we observed by Conflux, if one of path has an issue with latency increase, we can cover that by others running paths.

On the other side, running both Conflux and ConfluxM have additional node's performance (*Exit* nodes mainly), which are involved to construct multipath. As we know, the exit node is usually participating also on others Tor network connections as any node's role. We can expect, deploy Conflux to Tor network globally could cause additional performance requirements to these nodes (machines). But this is another stand point, what we cannot evaluate currently, as we are doing our setup on laboratory based environment in Private Tor network.

C. *Overall improvement evaluation*

In our approach evaluation we got several interesting results. If we take all measurements what has been done, we can see that ConfluxM setup has better results than any other. In the Table 1 below we can see comparing data for all three implementation. All measurements were done in same Private Tor network. We have changed only configuration of Tor client and Tor Exit node, while we have shift the implementation setup.

TABLE I. OVERALL IMPROVEMENT EVALUATION

	Average improvement [%]		
File size [MB]	ConfluxM	Conflux	Overall from Tor
0.4	12.12	23.26	32.56
5	28.25	56.74	68.96
10	24.43	25.36	43.60
Average (overall)	20,43	35,22	47,88

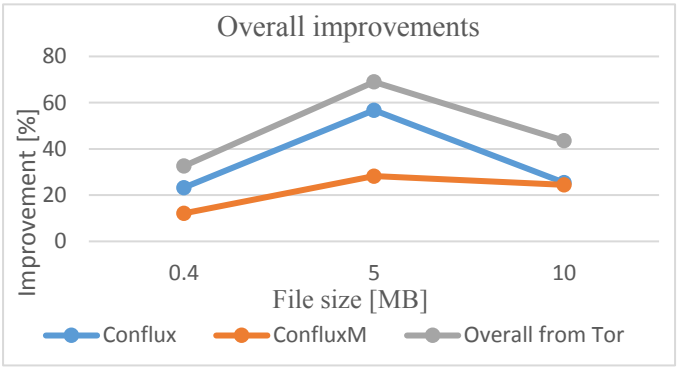

Fig. 4. Overall improvement evaluation comparing between Tor, Conflux and ConfluxM

Figure 4 above shows us an interesting point of view to our results. From the graph we can see that in every case Conflux does better average improvement than ConfluxM (in percentage). These findings proving that adding more parallel paths are no more needed. Also, we can assume that if we would add one more path, efficiency will decreased directly meager to number of paths. Next, we can see that in case of 10MB files improvements goes down in all of the solutions. It is caused mainly by longer time that is needed to transfer larger size data through Tor network because paths in Tor network used to failed time-to-time and then need to be re-establish.

V. Conclusion and related work

In the paper we're presenting another approach to the dynamic traffic splitting. We presented analyses, re-design and implementation of Conflux Tor's improvement tool. We did overall more than 1,400 measurements to evaluate our approach. We evaluated our approach named ConfluxM, using a small-scale experiments using a private Tor network. In particular, we have demonstrated the improvement against Tor and Conflux with using ConfluxM. Our results, as well as Conflux, indicate performance benefits with downloaded time decrease for Tor users. In our approach we achieved average download time decrease more than 20% versus Conflux.

As we mentioned in overall results, for every setup case, Conflux does better average improvement then ConfluxM. From that point, we can assume that there is no space no needs to additional time improvement, as extending Tor multipath. Therefore, we are recommend as future work focus to privacy and improving anonymity in Tor [17] [18] [19].

VI. Acknowledgement

This work is a partial result of the Research and Development Operational Program for the projects Support of Center of Excellence for Smart Technologies, Systems and Services II, ITMS 26240120029, co-funded by ERDF. It is also a part of APVV-15-0731 project and VEGA 1/0836/16, KEGA KEGA 011STU-4/2017.

Our acknowledgment belongs also to Mashael AlSabah and Ian Goldberg, University of Waterloo, CA. They gave us a lot of information and technical details about Conflux implementation. Also, they gave us a many helpful advices how to proceed with an issues, what we observed.

VII. References

[1] R. Dingledine, N. Mathewson and P. Syverson, "Tor: The Second-Generation Onion Router," August 2004. [Online]. Available: https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf.

[2] The Tor Project, "Tor Metrics - Users," 20 April 2017. [Online]. Available: https://metrics.torproject.org/userstats-relay-country.html.

[3] The Tor Project, "Tor Metrics - Servers," 20 April 2017. [Online]. Available: https://metrics.torproject.org/networksize.html.

[4] The Tor Project, "Tor metrics - Traffic," April 2017. [Online]. Available: https://metrics.torproject.org/bandwidth.html?start=2013-01-23&end=2017-04-23.

[5] S. Chakravarty, M. V. Barbera, G. Portokalidis, M. Polychronakis a A. D. Keromytis, „On the Effectiveness of Traffic Analysis Against," August 2013. [Online]. Available: https://mice.cs.columbia.edu/getTechreport.php?techreportID=1545.

[6] M. AlSabah1, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage and G. Voelker, "DefenestraTor: Throwing out Windows in Tor," July 2011. [Online]. Available: https://www.freehaven.net/anonbib/papers/pets2013/paper_65.pdf.

[7] A. King, „Average Web Page Size Septuples Since," 2016. [Online]. Available: http://www.websiteoptimization.com/speed/tweak/average-web-page/. [Cit. December 2016].

[8] The Tor project, "Bittorrent over Tor isn't a good idea," 30 April 2010. [Online]. Available: https://blog.torproject.org/blog/bittorrent-over-tor-isnt-good-idea.

[9] R. Pries, W. Yu, S. Graham and X. Fu, "On performance bottleneck of anonymous communication networks," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, 2008.

[10] M. AlSabah, K. Bauer, T. Elahi and I. Goldberg, "The Path Less Travelled: Overcoming Tor's Bottlenecks," November 2013. [Online]. Available: https://www.freehaven.net/anonbib/papers/pets2013/paper_65.pdf.

[11] A. Lewman, "China Blocking Tor: Round Two," March 2010. [Online]. Available: https://blog.torproject.org/blog/china-blocking-tor-round-two.

[12] F. Liu, "Setup Private Tor Network," January 2015. [Online]. Available: http://fengy.me/prog/2015/01/09/private-tor-network/.

[13] A. Smith, "Private tor network on kubernetes," March 2017. [Online]. Available: https://andrewmichaelsmith.com/2017/03/private-tor-network-on-kubernetes/.

[14] R. Dingledine a N. Mathewson, „Tor Protocol Specification," Apríl 2015. [Online]. Available: https://gitweb.torproject.org/torspec.git/plain/dir-spec.txt?id=77d040439b787556aab4979789eddc66c6964abd.

[15] S. Ramachandran, "Web Metrics: Size and Number of Resources," August 2011. [Online]. Available: https://code.google.com/speed/articles/web-metrics.html. [Accessed April 2017].

[16] C. Moldovan, C. Sieber, P. Heegaard, W. Kellerer and T. Hoßfeld, "YouTube Can Do Better: Getting the Most Out of Video Adaptation," in *2016 28th International Teletraffic Congress (ITC 28)*, September 2016.

[17] D. Gopal and N. Heninger, "Torchestra: Reducing Interactive Traffic Delays over Tor," in *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, New York, NY, USA, 2012.

[18] J. Reardon and I. Goldberg, "Improving Tor Using a TCP-over-DTLS Tunnel," in *Proceedings of the 18th USENIX Security Symposium*, August 2009.

[19] C. Tang and I. Goldberg, "An Improved Algorithm for Tor Circuit Scheduling," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2010.

[20] The Tor Project, "Tor Network Status," April 2017. [Online]. Available: https://torstatus.blutmagie.de/router_detail.php?FP=986d375159e7dcf9ef8e225fb9a687ffd09f9a85.