TCP Fast Open: initial measurements

Anna Maria Mandalari
University Carlos III of Madrid, Spain

Marcelo Bagnulo
University Carlos III of Madrid, Spain

Andra Lutu
Simula Research Laboratory, Norway

ABSTRACT

In mid-2011 a new mechanism that enables data exchange during TCP's initial handshake has been proposed by Google. The technique is called TCP Fast Open (TFO). TFO decreases application network latency by one full round-trip time. In this paper, we evaluate to which degree the current Internet paths support TFO and how often we encounter cases of interferences from eventual middleboxes along those paths. By expanding the traditional crowdsourcing focus from the human element, we develop a methodology to use a numerous group of end-user devices as measurement vantage points. We find that the 41,3% of the paths we test allowed for a successful TFO communication.

1. INTRODUCTION

Latency nowadays plays a key role in producing a successful Internet product and making content easily reachable. It is critically important for trading in financial markets [6], telemedicine, teleconferencing and cloud services.

To ensure low latency, several TCP mechanisms have been revisited in the last years. For example, lower speed links can be treated increasing the initial window of 10 segments (IW10) [4] or techniques for reducing the delays in communications has been started [2].

Google proposed a protocol to allow clients to send data in the initial SYN request, without waiting for a full TCP handshake, saving an entire Round-trip time (RTT). This mechanism is called TCP Fast Open (TFO) [9].

Tweaking TCP to reduce latency in the Internet seems the right direction, but the real question is how the Internet reacts to all these tweaks. Internet has become extremely difficult to evolve in terms of protocols and performance [5]. This is due to the presence of middleboxes (i.e., switches, routers, firewalls, NATs and proxies). This made the community to react and put in motion different mechanism to find solutions to the identified problems, e.g., an IETF research group was proposed to study this phenomenon [1].

In this paper we measure in the wild whether TFO is supported by the Internet paths. To this end we recruit TFO users from Microworkers crowdsourcing platform. We create a tool called *ExploreTFO* that allows users to connect to our TFO server.

Leveraging the access to these active users and their equipment we find that the 41,3% of the paths we test allowed for a successful TFO communication. We present a first step analysis of the deployability of TFO and we demonstrate that early results are not promising, reducing the possibility to decrease the latency due the TCP handshake.

CoNEXT Student Workshop'15, December 01 2015, Heidelberg, Germany

 ISBN 978-1-4503-4066-3/15/12…$15.00
DOI: http://dx.doi.org/10.1145/2842665.2843561

2. MEASUREMENT SETUP

In standard TCP connections, the Three-way Handshake (3WHS) process employs one RTT before the server's application receives the data and two RTTs for the server response. TFO should speed up the exchange of information between client and server and reduce of one RTT the 3WHS. Sending the initial SYN with data, TFO could reduce the loading of pages up to 40% [9].

According with the TFO standard [3], at the begging of a TCP connection, the client creates a SYN packet with the TFO option enabled. This option corresponds to a *TFO cookie request*. In this first connection, no data are sent into the SYN packet. Once the server receives the SYN with the TFO cookie request and it supports TFO, it generates a message authentication code (MAC) for the client, using the IP address of the client and its secret key. Then, it sends the cookie to the client in the SYN-ACK packet, using the same TFO option and it establishes a standard TCP connection. The client, receiving the TFO cookie, caches it and the correspondence to the server's IP that sent it. The client will use the cookie to establish a TFO connection to the same server in the next connection. When the client tries to connect to the server again, it sends a SYN

packet containing the TFO option enabled, the cookie previous cached that corresponds at the server's IP, and the application data. When the server receives the SYN packet with the TFO option, the cookie and the data, it generates the cookie and compares it to the cookie sent by the client. If the cookie matches, the server sends a SYN-ACK, acknowledging the SYN and the data, consecutively it sends the data to the client.

In this work, we apply the methodology model described in [8] to collect a unique dataset capturing the manner in which the Internet ecosystem reacts to the deployment of TFO. To this purpose we create a tool namely *TFOExplorer* and we recruit users from the Microworkers crowdsourcing platform [7]. *TFOExplorer* is composed by a client (from now on, *TFO Client (TC)*), a target server that supports TFO protocol (from now on, *TFO Server (TS)*) and a script needed to be run in the TC machine (from now on, *TFO Script (TSC)*). Using a crowdsourcing platform to recruit vantage points we are not able to guarantee the repeatability of experiment, for this reason we perform a single measurement per port.

We store and analyze in detail the server side packet exchanges to 68 different ports, namely 10 well-known ports, 56 registered ports and 2 ephemeral ports. If an error occurs in some ports, we analyze the behavior of the TCP with respect to the middleboxes active along the path. The *TFOExplorer* allows us to control the client, capturing automatically the packets in the TC's machine and sending the files containing the results of the experiment directly to our server to be analyzed.

3. RESULTS

We collect and analyze the results from 46 users from 18 different countries and 22 different ISPs. Each user tries to establish TFO connections in a large number of ports to our TS.

Considering that each TC performs 68 connections to our TS three times (twice to perform a complete TFO connection and one to send data in the initial SYN packet), we build a dataset for a total of 9,568 connections [1].

Table 1 shows the results from workers that attempts to connect to our TS, using TFO. Analyzing the data, we check for the following behaviors: TC is able to perform a TFO connection (label *Successful*); middleboxes drop packets with unknown TCP options and we receive the SYN without option (label *No option SYN*); middleboxes drop packets with unknown TCP options and we do not receive the SYN without option (label *No option no SYN*); middleboxes drop packets with data in the SYN packet and we receive the SYN without data (label *No data SYN*); middleboxes drop packets with data in the SYN packet and we do not receive the SYN without data (label *No data no SYN*).

[1]The data set and the code are freely available on http://www.it.uc3m.es/amandala/tfocampaign.html

Table 1: TFO connection

TFO behavior	Number of workers	Number of workers (%)
Successful	19	41,3
No option SYN	18	39,13
No option no SYN	0	0
No data SYN	8	17,39
No data no SYN	1	2,18

Table 2: Data in the SYN

TFO behavior	Number of workers	Number of workers (%)
Successful	23	50
No data SYN	23	50
No data no SYN	0	0

Results show that only the 41,3% of the packets with the TFO experimental option are able to arrive to our TS. The 39,13% of the SYN packets arrive with the option removed.

TFO allows for payloads to be carried in SYN frame. Once the client has the cookie, it tries to send the data in the SYN. In this case, only the 67,86% of the packets are received by our TS. The remainder of the packets arrive with no data in the SYN. In this case, middleboxes block the connection completely after processing the SYN packet with data, causing a connection timeout and the TC switches to a standard TCP connection, retransmitting the SYN packet with no option and no data.

If, for example TFO connection fails in port 80, then it fails in all other ports. We detect only three TCs that succeed to perform a TFO connection in the Well-known and the Ephemeral ports, but that fail in some Registered ports.

Table 2 shows the results considering that case when TCs send data in the initial SYN with no TFO option set. In this case, the 50% of the SYN packets containing data is received by our TS, the rest of the packets are dropped by middleboxes, forcing a SYN retransmission client side.

4. CONCLUSIONS AND FUTURE WORK

In this paper we evaluate the interaction of TFO with the elements of the path. Lessons learned from this study can help in designing robust TCP protocol extensions in the presence of middleboxes. Only the 41,3% of the paths we test allowed for a successful TFO communication. Once the client is able to receive the cookie, the 32,14% of the packets are received with no data in the SYN. In the case in which we force data in the SYN without the TFO option, the percentage of success is 50%.

For future work, we are planning to expand our measurement campaign and recruit more measurement agents. TFO requires client and server specific Linux kernel support (i.e. 3.7+). Thus, using the current measurement methodology we propose it is challenging to support TFO by default. We are overcoming this issue by building crafted packets and running the code in devices that do not have necessarily TFO mechanism enabled by default.

5. REFERENCES

[1] https://www.ietf.org/mailman/listinfo/hops. *IETF*, 2015.

[2] B. Briscoe, A. Brunstrom, D. Ros, D. Hayes, A. Petlund, I. Tsang, S. Gjessing, and G. Fairhurst. A Survey of Latency Reducing Techniques and their Merits. In *ISOC Workshop on Reducing Internet Latency, Sep*, 2013.

[3] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain. TCP Fast Open. Technical report, RFC 7413 , December, 2014.

[4] J. Chu, Y. Cheng, N. Dukkipati, and M. Mathis. Increasing TCP's initial window. 2013.

[5] M. Handley. Why the internet only just works. *BT Technology Journal*, 2006.

[6] J. Hasbrouck and G. Saar. Low-latency trading. *Journal of Financial Markets*, 16(4):646–679, 2013.

[7] M. Hirth, T. Hoßfeld, and P. Tran-Gia. Anatomy of a crowdsourcing platform-using the example of microworkers.com. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, pages 322–329. IEEE, 2011.

[8] A. M. Mandalari, M. Bagnulo, and A. Lutu. Informing protocol design through crowdsourcing: the case of pervasive encryption. ACM SIGCOMM Workshop on Crowdsourcing and crowdsharing of Big (Internet) Data (C2B(I) D), Aug. 2015.

[9] S. Radhakrishnan, Y. Cheng, J. Chu, A. Jain, and B. Raghavan. TCP Fast Open. In *Proceedings of the Seventh Conference on emerging Networking EXperiments and Technologies*, page 21. ACM, 2011.