A first look at real Multipath TCP traffic

B. Hesmans, H. Tran-Viet, R. Sadre, O. Bonaventure

ICTEAM, Université catholique de Louvain, Louvain-la-Neuve, Belgium

Abstract. Multipath TCP is a new TCP extension that attracts a growing interest from both researchers and industry. It enables hosts to send data over several interfaces or paths and has use cases on smartphones, datacenters or dual-stack hosts. We provide the first analysis of the operation of Multipath TCP on a public Internet server based on a one-week long packet trace. We analyse the main new features of Multipath TCP, namely the utilisation of subflows, the address advertisement mechanism, the data transfers and the reinjections and the connection release mechanisms. Our results confirm that Multipath TCP operates correctly over the real Internet, despite the presence of middleboxes and that it is used over very heterogeneous paths.

1 Introduction

The Transmission Control Protocol (TCP) [23] was designed when hosts were equipped with a single interface. When two hosts exchange data through a TCP connection, all packets generated by the client (resp. server) must be sent from the same IP address. This remains true even if the communicating hosts have several interfaces and thus IP addresses that could be used to improve performance or resilience. In today's networks, this limitation is becoming a major drawback. Cellular and WiFi networks are available in most cities and smartphone users would like to be able to start a TCP connection in a WiFi hotspot and continue it later via their 3G interface. Reality with TCP is different [20]. Datacenters provide multiple paths between servers, but all packets from a given connection always follow the same path [24]. Dual stack hosts would like to exploit their IPv6 and IPv4 paths simultaneously but with regular TCP they can only rely on Happy Eyeballs [29].

Multipath TCP (MPTCP) is a recent TCP extension that has been standardised by the Internet Engineering Task Force [10] to solve this problem. In a nutshell, thanks to Multipath TCP, a multihomed host can use several interfaces (and thus IP addresses) to support a single TCP connection. Multipath TCP can pool all the resources available to improve the performance and the resilience of the service provided to the applications [30]. Several use cases for Multipath TCP have already been studied by the research community including datacenters [24] and WiFi/3G offload [20, 4, 6]. Implementers and industry are adopting Multipath TCP quickly. As of this writing, Multipath TCP implementations exist on Linux [18], Apple iOS and MacOS [1], FreeBSD [28] and Solaris [7]. Apple has enabled Multipath TCP by default for its voice recognition

SIRI application running on all recent iPhones and iPads. Today, there are thus hundreds of millions of devices that use Multipath TCP on the Internet.

Despite of this large scale deployment, little is known about how Multipath TCP really behaves in the global Internet. In this paper, we provide a first analysis of the Multipath TCP packets received and sent by the server that hosts the reference implementation in the Linux kernel [1]. Besides Apple's servers that support the SIRI application, this is probably the most widely used public Multipath TCP server. By observing how real users use this new protocol, we complement the existing measurement that relied on simulations, emulations or active measurements.

This paper is organised as follows. In Section 2, we describe our dataset, how the packets have been collected and the software that we used to analyse them. Section 3 describes the main features of the Multipath TCP protocol and analyses their impact based on the collected packets. Section 4 compares our work with related work. We conclude the paper in Section 5.

2 Dataset

The dataset[2] used in this paper is a one-week long packet trace collected in November 2014 at Université catholique de Louvain (UCL). It has been collected using `tcpdump` and contains the headers of all TCP packets received and sent by the server hosting the Multipath TCP Linux kernel implementation. Apart from a web server, the machine also hosts an FTP server and an `iperf` server. It has one physical network interface with two IP addresses (IPv4 and IPv6) and runs the stable version 0.89 of the Multipath TCP implementation in the Linux kernel [18].

To analyse the Multipath TCP connections in the dataset, we have extended the `mptcptrace` software [11] developed by the same authors. `mptcptrace` handles all the main features of the Multipath TCP protocol and can extract various statistics from a packet trace. Our extensions to `mptcptrace` have been included in the last release. Furthermore, we have combined it with `tcptrace` [17] and its output has been further processed by custom `python` scripts.

Table 1 summarizes the general characteristics of the dataset. In total, the server received around 136 million TCP packets carrying 134 GiBytes of data (including the TCP and IP headers) during the measurement period. As shown in table 1 (in the block "Multipath TCP"), a significant fraction of the TCP traffic was related to Multipath TCP. Unsurprisingly, IPv4 remains more popular than IPv6, but it is interesting to note that the fraction of IPv6 traffic from the hosts that are using Multipath TCP (9.8%) is larger than from the hosts using regular TCP (3.7%). On the monitored server, dual-stack hosts are already an important use case for Multipath TCP. It should be noted that the monitored server is mainly used by Multipath TCP users and developers who are likely to be

[1] `http://www.multipath-tcp.org`

[2] The anonymised packet trace is available at `http://multipath-tcp.org/data/TMA-2015.tar.gz`.

Table 1: Dataset characteristics

Collection period	Nov. 17 – 24, 2014		
All TCP	Total	IPv4	IPv6
# of packets [Mpkt]	136.1	128.5	7.6
# of bytes [GiByte]	134.0	129.0	5.0
Multipath TCP	Total	IPv4	IPv6
# of packets [Mpkt]	29.4	25.0	4.4
# of bytes [GiByte]	20.5	18.5	2.0

researchers or computer scientists. These users probably have better connectivity that the average Internet user.

We have also studied the application protocols used in the Multipath TCP traffic. Around 22.7% of the packets were sent or received on port 80 (HTTP). A similar percentage of packets (21.2%) was sent to port 5001 (`iperf`) by users conducting throughput measurements. The FTP server, was responsible for the majority of packets. It hosts the Debian and Ubuntu packages for the Multipath TCP kernel and is thus often used by Multipath TCP developers.

Considering the number of connections, 89.7% of them were targeted on HTTP, 6.4% for `iperf`, 1.9% for FTP control connections and the remaining 2.0% on higher ports are likely FTP data connections.

Another important figure is the number of distinct Multipath TCP clients connected to our server. While identifying this figure exactly is not trivial, information about client IP addresses may be useful to give some feeling about the variety of clients' location. Among 790 distinct client IP addresses we observed, there are 562 IPv4 addresses coming from 464 distinct class C IPv4 network prefixes and 228 different IPv6 addresses coming from 79 distinct 48-bit IPv6 prefixes.

3 Analysis

Multipath TCP is a major extension to TCP that modifies many features of the protocol. A detailed overview of Multipath TCP is outside the scope of this paper and may be found in [19, 10]. In this section, we first describe the key features of Multipath TCP. Then, each subsection focuses on a particular aspect of the protocol that is explained and analysed in more details based on the collected packet trace.

Like most TCP extensions, Multipath TCP defines a new TCP option that is used during the initial three-way handshake. This `MP_CAPABLE` option contains several important parameters [19, 10]. A prominent characteristic of Multipath TCP is that, to be able to use several paths, Multipath TCP combines several TCP connections called subflows inside a single Multipath TCP connection. These subflows are linked together by using a token extracted from the

MP_CAPABLE option and included in the MP_JOIN option that is exchanged during the handshake of the other subflows. To support mobile hosts, Multipath TCP allows each host to advertise its current list of addresses to its peer [10]. This is done thanks to the ADD_ADDR and REMOVE_ADDR options that can be sent at any time over one of the TCP subflows. It is important to note that the subflows that compose a Multipath TCP connection is not fixed. The set of subflows changes during the lifetime of the connection since each host can add or remove a subflow at any time.

3.1 Which hosts use Multipath TCP?

The first question that we asked ourselves while analysing the trace was the characteristics of the clients that contacted the monitored server. Since the packet trace was collected on the server that hosts the Multipath TCP implementation in the Linux kernel, we can expect that many Linux enthusiasts use it to download new versions of the code, visit the documentation, perform tests or verify that their configuration is correct. These users might run different versions of Multipath TCP in the Linux kernel or on other operating systems[7]. Unfortunately, as of this writing, there is not enough experience with the Multipath TCP implementations to detect which operating system was used to generate specific Multipath TCP packets. This is an interesting direction for future work. Instead, we focus our analysis on the number of addresses used by the clients.

Thanks to the ADD_ADDR option, it is possible to collect interesting data about the characteristics of the clients that contact our server. Over the 5098 observed Multipath TCP connections, 3321 of them announced at least one additional address. Surprisingly, only 21% of the collected IPv4 addresses in the ADD_ADDR option were globally routable addresses. The remaining 79% of the IPv4 addresses found in the ADD_ADDR option were private addresses and in some cases link-local addresses. The large number of private address confirms that Multipath TCP's ability to pass through NATs is an important feature of the protocol [10].

The IPv6 addresses collected in the ADD_ADDR option had more diversity. We first observed 72% of globally routable IPv6 addresses. The other types of addresses that we observed are shown in Table 2. The IPv4-compatible and the 6to4 IPv6 addresses were expected, but the link local and documentation addresses should have been filtered by the client and never be announced over Multipath TCP connections. The Multipath TCP specification [10] does not currently specify the types of addresses that can be advertised over a Multipath TCP connection. It should probably be updated to specify which types of IPv4 and IPv6 addresses can be announced with the the ADD_ADDR option.

Another interesting point to note is that although the ADD_ADDR option defined in [10] can also be used to advertise transport protocol port numbers together with IP addresses, we didn't detect any utilisation of this feature in our trace.

Table 2: Special addresses advertised by clients

Address type	Count
Link-local (IPv4)	51
Link-local (IPv6)	241
Documentation only (IPv6)	21
IPv4-compatible IPv6	13
6to4	206

3.2 How quickly are subflows established?

The previous section has shown that a large number of (multihomed) hosts exchange Multipath TCP packets with our server. Our server has a single interface but two IP addresses (IPv4 and IPv6). Like the other Multipath TCP implementations [7], it never creates subflows [18] because, as confirmed by the measurements in the previous section, the client is likely to reside behind a NAT or a firewall that would block these subflow establishments.

The current implementation of Multipath TCP in the Linux kernel [18] uses two strategies to create subflows. The first strategy, supported by the default path manager on multihomed hosts, is to create a full-mesh of subflows as soon as possible. Figure 1 shows the delay between the `SYN` segment sent on the initial subflow and the `SYN` segment of the first subflow. For 88.9% of the Multipath TCP connections composed of two or more subflows, the first subflow is established within less than one second. For 46.5% of the connections, this delay is shorter than 100 msec. The longest delay between the establishment of a Multipath TCP connection and the first subflow is 360 seconds.

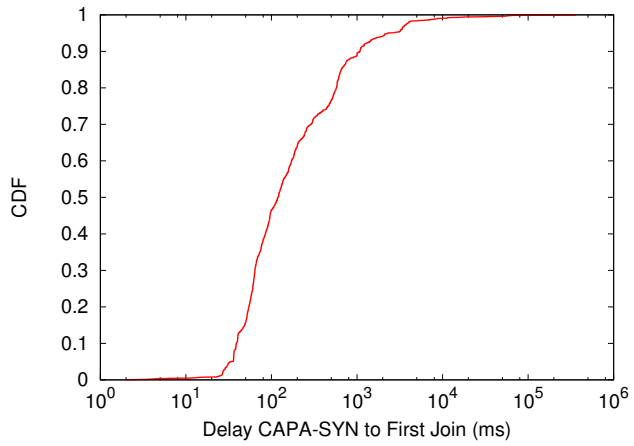

Fig. 1: Delay between the SYN of initial flow and the first `MP_JOIN`

For test purposes and to support local balancing over multiple equal cost paths [24], another path manager can be used on single-homed hosts. This path manager creates n subflows from the single IP address of the client to the server. An analysis of all the observed Multipath TCP connections indicates that this path manager was used by about one third of the connections that we observed.

Multipath TCP can also support backup subflows [20]. In this case, the client indicates that a subflow is a backup subflow at subflow establishment time or by sending the `MP_PRIO` option on an existing subflow. In the analysed trace, we did not observe any utilisation of this option.

When Multipath TCP is used over heterogeneous paths, its performance can decrease if the paths have different round-trip-times or bandwidth [22]. Figure 2 plots, for all Multipath TCP connections using two or more subflows, the CDF of the difference between the minimum and the maximum average round-trip-times over all the subflows that compose each Multipath TCP connection. These average round-trip-times were computed by using `tcptrace` [17]. Less than 10% of the Multipath TCP connections have subflows having the same average round-trip-times. Almost 54.3% of the Multipath TCP connections combine subflows with a spread of up to 10 msec. 13.7% of the connections have a spread larger than 50 msec. This is an important indication about the heterogeneity of the paths over which Multipath TCP is used today.

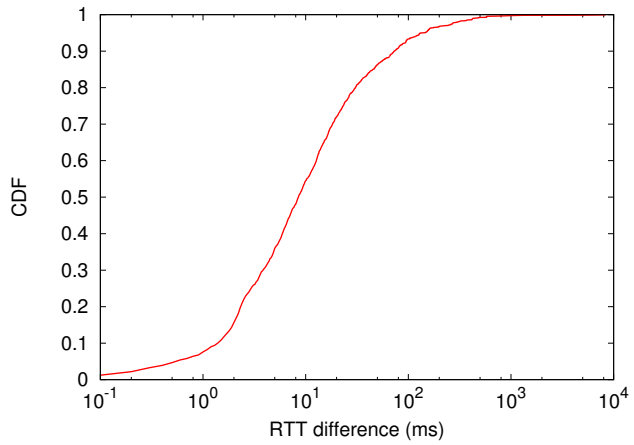

Fig. 2: Difference between the maximum RTT and minimum RTT among subflows belonging to one MPTCP connection

3.3 How many subflows are used?

In theory, a Multipath TCP connection can gather an unlimited number of subflows. In practice, implementations [7] limit the number of concurrent subflows.

The Linux implementation [18] used on the monitored server can support up to 32 different subflows. We analyse here the number of subflows that are established for each Multipath TCP connection. Since our server never establishes subflows, this number is an indication of the capabilities of the clients that interact with it.

Figure 3 provides the distribution of the number of subflows per Multipath TCP connection. We show the distribution for the number of successfully established subflows, i.e., subflows that complete the handshake, as well as for all attempted ones. As can be seen, several connection attempts either fail completely or establish less subflows than intended. In total, we observe 5098 successful connections with 8701 subflows. The majority of the observed connections (57%) only establish one subflow. Around 27% of them use two subflows. Only 10 connections use more than 8 subflows, which are omitted from the figure.

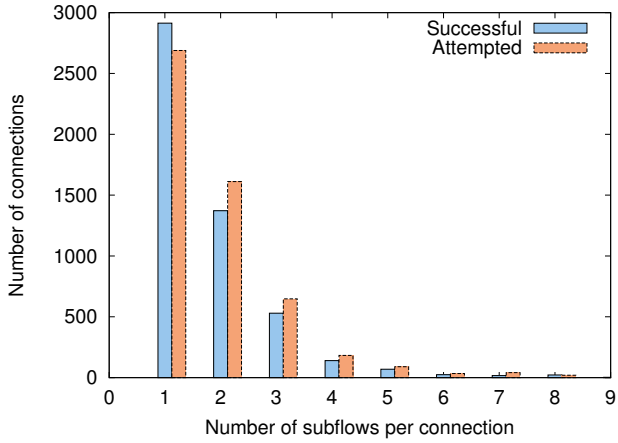

Fig. 3: Distribution of subflows per connection

The fact that many Multipath TCP connections establish more than one subflow affects the connection sizes as perceived on TCP level. In Figure 4 we show the cumulative distribution of the number of payload bytes exchanged on the Multipath TCP connections and compare it with the size distribution of the individual TCP connections. As expected, we see a larger number of small TCP subflow connections.

It should be noted that establishing multiple subflows does not necessarily mean that the payload of a Multipath TCP connection is evenly distributed over them. This will be further discussed in Section 3.6. While the `MP_PRIO` option has been introduced to Multipath TCP providing the capability to change the priority of a subflow, we do not observe any subflow using this option.

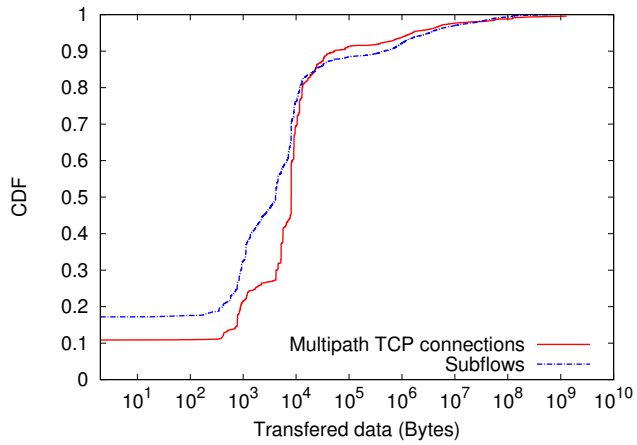

Fig. 4: Distribution of data on Multipath TCP connections and the subflows

3.4 Do middleboxes interfere with Multipath TCP?

The interference caused by various types of middleboxes has significantly affected the design of the Multipath TCP protocol [10, 25, 13]. Multipath TCP can cope with middleboxes that modify the source/destination IP addresses and ports, but also change the TCP sequence numbers or split/reassemble TCP segments [12]. If a middlebox strips TCP options, the Multipath TCP implementation in the Linux kernel performs a fallback to regular TCP [10]. The worst middlebox interference for Multipath TCP is when a middlebox such as a NAT using a Application Level Gateway changes "transparently" the payload of TCP segments, e.g. to translate the ASCII representation of an IP address. Multipath TCP can detect this type of interference by using the DSS checksum which is computed over the data and the Multipath TCP option [25, 10]. Given that the TCP segments are already protected by the TCP checksum, an error in the DSS checksum is always a sign of middlebox interference.

If a host receives a TCP segment with a valid TCP checksum and an invalid DSS checksum, it sends an `MP_FAIL` option to inform the peer of the interference and performs a fallback to regular TCP. Among more than 5000 Multipath TCP connections that we have analysed, we observe three transmissions of the `MP_FAIL` option. In all three cases, `MP_FAIL`s are sent on port 21 (FTP). This is a relatively small number knowing that the monitored server provides FTP services that are subject to middlebox interference [12] due to Application Level Gateways included in NAT devices. We did not observe the `MP_FAIL` option on port 80 despite the fact that transparent proxies are often reported [26]. Surprisingly, we also observe this `MP_FAIL` option inside IPv6 packets.

As explained earlier, the utilisation of the DSS Checksum [10] ensures that middlebox interference is detected and that the data is transported correctly. However, computing the DSS Checksum consumes CPU ressources [25] and the

DSS checksum can be disabled on a per connection basis. During the three-way handshake on the initial subflow, the client and the server can opt out the DSS Checksum. The DSS Checksum is only disabled if both propose to disable it. If either the client or the server is configured to use the DSS Checksum, then it is used in both directions. Since the monitored server was configured to always us the DSS Checksum, it was active for all Multipath TCP connections. This is the default configuration which is recommended in the Internet [10]. The DSS Checksum should only be disabled in controlled environments such as datacenters that are known to be immune of middlebox interference. We were surprised to measure that about 5% of the Multipath TCP connections established with our server requested to disable the DSS Checksum. This configuration was probably chosen for performance reasons, but it puts the data transfert at risk of undetected middlebox interference.

3.5 How do Multipath TCP connections terminate?

TCP uses two different mechanisms to terminate a connection. Most connections should terminate by exchanging `FIN` segments in both directions [23]. Some connections terminate abruptly with the transmission of a `RST` segment due to problems or because the server does not want to wait in the `CLOSE_WAIT` state [2]. Each of the TCP subflows that compose a Multipath TCP connection can be terminated using one of these mechanisms. Above the subflows, Multipath TCP also includes similar mechanisms to terminate the Multipath TCP connection [10]. If all the data has been transferred correctly, a Multipath TCP connection should terminate with the exchange of `DATA_FIN` options in both directions. We observe that 89% of the 5098 monitored Multipath TCP connections are terminated by exchanging `DATA_FIN`s. Multipath TCP also includes a fast close mechanism that allows a host to terminate a Multipath TCP connection by sending a `FAST_CLOSE` option inside a `RST` segment. This feature was included in [10] to enable a server to quickly terminate a Multipath TCP connection. It is used by roughly 10% of the observed Multipath TCP connections.

3.6 How is data distributed?

Since a Multipath TCP connection combines several TCP subflows, data can be transmitted over any of these subflows. The Linux implementation of Multipath TCP uses a packet scheduler [21] to select the subflow over which each data segment is transmitted. The monitored server uses the default scheduler that tries to send data on the subflow having the smallest round-trip-time among the subflows whose congestion window is open. The client could use another scheduler such as the round-robin scheduler or any custom scheduler [21].

The utilisation of the default scheduler implies that the data is not evenly distributed among the different subflows. Figure 5 shows, for connections composed of more than one subflow, how the data is distributed among the initial subflow and the other subflows of the connection. We observe that around 52% of the connections send less than half of their data over the initial subflow.

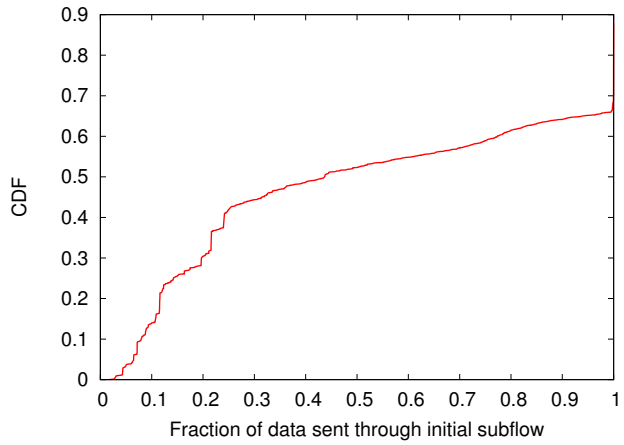

Fig. 5: Fraction of data on initial subflow over total data for connections with more than one subflow

In contrast, around 30% of the connections transfer their data nearly entirely over the initial subflow. There are different possible explanations for this behaviour. One reason could be that those connections are simply too short or too small to use the additional subflows. However, we have studied their length and size distribution and could not find any prevalence of very short or small connections among them in comparison to other connections.

Like regular TCP connections, Multipath TCP connections can be impacted by packet losses. The Multipath TCP implementation in the Linux kernel includes three main strategies to cope with losses. If an isolated packet is lost, Multipath TCP will use the fast retransmit mechanism to retransmit it over the subflow where it was initially sent. This is the normal behaviour of a TCP connection. If a retransmission timer expires, this usually indicates a more severe loss. In this case, Multipath TCP evaluates whether the data should be retransmitted over the same subflow as the original transmission or over another subflow. In the latter case (*reinjection*), data will be retransmitted over both subflows to ensure that any middlebox over one of the paths observes in-sequence data [10, 12]. In some cases, such retransmissions can also occur when one of the subflows is too slow compared to the other ones [25]. If a subflow dies, e.g. due to the failure of a WiFi interface, then all unacknowledged data sent over this subflow is retransmitted over the remaining subflows. These reinjections are an indication of the inefficiency of Multipath TCP. Since reinjected data is sent over two or more subflows, a large reinjection rate will result in a badly performing Multipath TCP connection.

Figure 6 shows the fractions of reinjected data (bytes) for the Multipath TCP connections composed of at least two subflows. We provide the CDF for transmissions from the client to the server resp. from the server to the client. We observe that more than 90% of the connections do not have any reinjection.

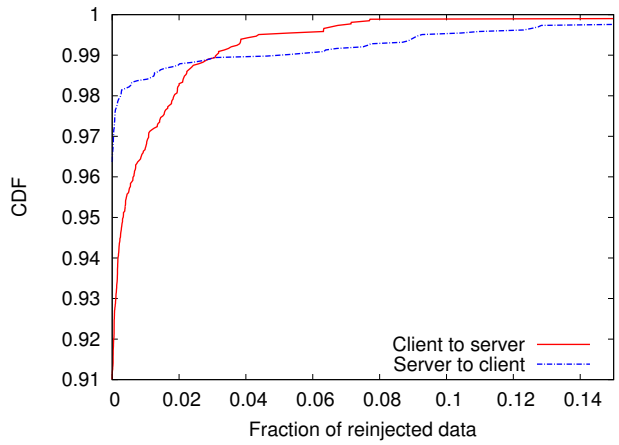

Fig. 6: Fraction of reinjected data over total data for connections

More than 98% of the connections have fewer than 2% of reinjections. This shows that reinjections are rare and that the overhead caused by them is very low in comparison to the overall TCP retransmission rate of 0.8–2.4% reported in [5]. We also observe that the fraction of reinjections is larger in the client-to-server direction than in the opposite one. The reasons for this will be the topic of future work. Finally, it should be noted that the results shown in the figure should be taken with caution for fractions above 30%. Due to the small population size and the presence of very short connections, extreme cases, such as a single-packet connection with reinjection, do have a visible impact on the tail of the empirical CDF.

4 Related work

Various researchers have studied the performance of Multipath TCP on cellular networks and/or the global Internet. As of this writing, most of these studies have focussed on performing active measurements among a relatively small set of connected devices. We are not aware of published studies where large Multipath TCP packet traces have been analysed in details. In this section, we thus compare our findings with the main results obtained with active measurements. Although several implementations of Multipath TCP exist [7], we are not aware of published results that compare their performance. In fact, most of the published papers use the Linux implementation.

Several researchers have analysed the performance of Multipath TCP in cellular networks. Paasch et al. propose, implement and evaluate path management strategies to efficiently support 3G and WiFi interfaces with Multipath TCP [20]. The Linux implementation [18] is used by the monitored server and likely most of the clients that sent packets. Chen et al. [4] and Deng et al. [6] also

perform measurements with the Multipath TCP implementation in the Linux kernel. Chen et al. focus on bulk transfers over WiFi and 3G and conclude that Multipath TCP provides performance benefits compared to regular TCP. Deng et al. analyse different scenarios. Their measurements show that the benefits of using Multipath TCP increase with the volume of data transferred. Lim et al. present other measurements with cellular and WiFi networks in [16] and focus on tuning the Multipath TCP implementation to reduce the energy consumption. Williams et al. use experiments to investigate the use of Multipath TCP (MPTCP) to augment cellular 3G connections with roadside infrastructure [27]. Ferlin et al. use the NorNet infrastructure [15] to analyse the performance of Multipath TCP in heterogeneous networks (WiFi and 2G or 3G) [9]. Their measurements show that when the difference in bandwidth between the two interfaces is large (e.g. 2G and WiFi), then the performance of Multipath TCP may suffer. They propose an algorithm that monitors the subflow performance and disables the under-performing ones.

Multipath TCP is not the only transport protocol that is capable of supporting multiple paths. The Stream Control Transmission Protocol [8] (SCTP) has been implemented in most operating systems. It was designed with multihoming in mind and support for concurrent multipath was added to SCTP [14] before the deployment of Multipath TCP. However, we have not found in the scientific literature detailed measurements of the performance of SCTP based on passive measurements. As shown in [3] only a small fraction of the published SCTP research is based on real measurements on the Internet.

5 Conclusion

In this paper, we have presented a first analysis of the behaviour of Multipath TCP based on a one-week long packet trace collected on a popular Multipath TCP server. Since the server hosts the Multipath TCP implementation in the Linux kernel, it is mainly used by developpers and researchers. Furthermore, we expect that most clients use the same Linux implementation as the one running on the server. We have analysed the real utilisation of the new features introduced in this TCP extension, namely the establishment of the subflows, the advertisement of addresses, the reinjection of data, the detection of middlebox interference and the termination of the Multipath TCP connections.

Some of our results confirm that the protocol operates correctly over the Internet. Others were less expected and provide an insight on the operation of this protocol over the Internet. Firstly, Multipath TCP hosts can use many interfaces. Some hosts announced up to 14 different IP addresses over a single Multipath TCP connection. Secondly, the subflows that compose a Multipath TCP connection are usually established very quickly. This corresponds to the default strategy of the current Linux implementation and confirms that clients mainly use this implementation. Thirdly, when two or more subflows are used, their average round-trip-times can differ by 10-100 msec for 40% of the Multipath TCP connections. This is a large delay difference that indicates that Multipath

TCP is used in heterogeneous environments. This delay difference must be taken into account by Multipath TCP implementors who are tuning their implementations. Fourthly, Multipath TCP spreads the data over the different subflows and rarely needs to reinject some data over another subflow. Finally, there are middleboxes that interfere with Multipath TCP, even in IPv6 networks.

For our future work, we will collect a longer trace to study in more details other aspects of the protocol such as identifying the congestion control scheme used on the subflows, the packet scheduler used by the client, or measuring the short term dynamics of the data transmission on the different subflows or middlebox interference.

Acknowledgements

This work was partially supported by the ITN METRICS and the FP7 TRILOGY 2 projects funded by the European Commission and by the BESTCOM IAP.

References

1. Apple. ios: Multipath tcp support in ios 7. `http://support.apple.com/en-us/HT201373`.
2. M. Arlitt and C. Williamson. An Analysis of TCP Reset Behaviour on the Internet. *SIGCOMM Comput. Commun. Rev.*, 35(1):37–44, January 2005.
3. L. Budzisz, J. Garcia, A. Brunstrom, and R. Ferrús. A Taxonomy and Survey of SCTP Research. *ACM Comput. Surv.*, 44(4):18:1–18:36, September 2012.
4. Y-C Chen, Y-S Lim, R. Gibbens, E. Nahum, R. Khalili, and D. Towsley. A Measurement-based Study of Multipath TCP Performance over Wireless Networks. In *ACM SIGCOMM IMC*, 2013.
5. H.K. Jerry Chu. Tuning TCP Parameters. In *Proceedings of The Seventy-Fifth Internet Engineering Task Force*. IETF, 2009.
6. S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan. WiFi, LTE, or Both?: Measuring Multi-Homed Wireless Internet Performance. In *IMC '14*, pages 181–194, New York, NY, USA, 2014. ACM.
7. P. Eardley. Survey of MPTCP Implementations. Internet-Draft draft-eardley-mptcp-implementations-survey-02, IETF Secretariat, July 2013.
8. R. Stewart (Ed.). Stream Control Transmission Protocol. IETF RFC 4960, September 2007.
9. S. Ferlin, T. Dreibholz, and O. Alay. Multi-Path Transport over Heterogeneous Wireless Networks: Does it really pay off? In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Austin, Texas/U.S.A., December 2014.
10. A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. IETF RFC 6824, January 2013.
11. B. Hesmans and O. Bonaventure. Tracing Multipath TCP Connections. In *SIGCOMM '14 (poster)*, pages 361–362, 2014.
12. B. Hesmans, F. Duchene, C. Paasch, G. Detal, and O. Bonaventure. Are TCP Extensions Middlebox-proof? In *CoNEXT Workshop HotMiddlebox*, 2013.

13. Michio Honda, Yoshifumi Nishida, Costin Raiciu, Adam Greenhalgh, Mark Handley, and Hideyuki Tokuda. Is It Still Possible to Extend TCP? In *2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 181–194, 2011.
14. J. Iyengar, P. Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *Networking, IEEE/ACM Transactions on*, 14(5):951–964, 2006.
15. A. Kvalbein, D. Baltrūnas, K. Evensen, J. Xiang, A. Elmokashfi, and S. Ferlin. The NorNet Edge Platform for Mobile Broadband Measurements. *Computer Networks, Special Issue on Future Internet Testbeds*, 61:88–101, March 2014. ISSN 1389-1286.
16. Y. Lim, Y. Chen, E. Nahum, D., and R. Gibbens. Improving energy efficiency of MPTCP for mobile devices. *CoRR*, abs/1406.4463, 2014.
17. S. Ostermann. `tcptrace`. `http://www.tcptrace.org`.
18. C. Paasch, S. Barre, et al. Multipath TCP implementation in the Linux kernel. available from `http://www.multipath-tcp.org`, 2014.
19. C. Paasch and O. Bonaventure. Multipath TCP. *ACM Queue*, 12(2):40:40–40:51, 2014.
20. C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. Exploring Mobile/WiFi Handover with Multipath TCP. In *ACM SIGCOMM workshop CellNet*, pages 31–36, 2012.
21. C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure. Experimental Evaluation of Multipath TCP Schedulers. In *2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop*, CSWS '14, pages 27–32, 2014.
22. C. Paasch, R. Khalili, and O. Bonaventure. On the Benefits of Applying Experimental Design to Improve Multipath TCP. In *Proceedings of CoNEXT '13*, pages 393–398, New York, NY, USA, 2013. ACM.
23. J. Postel. Transmission Control Protocol. IETF RFC 793, September 1981.
24. C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving Datacenter Performance and Robustness with Multipath TCP. In *ACM SIGCOMM 2011*, 2011.
25. C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *USENIX NSDI*, 2012.
26. Nicholas Weaver, Christian Kreibich, Martin Dam, and Vern Paxson. Here be web proxies. In *Passive and Active Measurement*, volume 8362 of *Lecture Notes in Computer Science*, pages 183–192. Springer, 2014.
27. N. Williams, P. Abeysekera, N. Dyer, H. Vu, and G. Armitage. Multipath TCP in Vehicular to Infrastructure Communications. Technical Report Centre for Advanced Internet Architectures, Technical Report 140828A, Swinburne University of Technology, 2014.
28. N. Williams, L. Stewart, and G. Armitage. FreeBSD kernel patch for Multipath TCP. available from `http://caia.swin.edu.au/urp/newtcp/mptcp/tools.html`, July 2014.
29. D. Wing and A. Yourtchenko. Happy Eyeballs: Success with Dual-Stack Hosts. RFC 6555, April 2012.
30. D. Wischik, M. Handley, and M. Braun. The Resource Pooling Principle. *ACM SIGCOMM Computer Communication Review*, 38(5):47–52, 2008.