Tracing Multipath TCP Connections

B. Hesmans, O. Bonaventure
ICTEAM, Université Catholique de Louvain
Louvain-La-Neuve – Belgium
firstname.name@uclouvain.be

1. INTRODUCTION

Multipath TCP [2] is a major extension to the TCP protocol. Multipath TCP enables a host to send packets belonging to one connection over different interfaces and IP addresses. This is a major change to TCP where one connection is always bound to one four-tuple (source and destination addresses and ports). To understand the basics of Multipath TCP, let us consider the simple and important use case of a smartphone[1] using WiFi and 3G to connect to a server. To start a Multipath TCP connection, the smartphone initiates a three-way handshake with the server over the 3G interface. This handshake includes specific options to negotiate the utilisation of Multipath TCP. Once this TCP connection has been established, it becomes the first subflow of the Multipath TCP connection. To use the WiFi interface, the smartphone starts another three-way handshake over this interfaces and indicates that TCP connection is the second subflow of the Multipath TCP connection. The number of subflows associated to one Multipath TCP connection can vary over time.

Once the subflows have been established, data can be transmitted over anyone of them. If there are losses, data transmitted initially on one subflow can be retransmitted over another one. For this, Multipath TCP uses two levels of sequence numbers. The regular TCP sequence number ensures that the data sent over one subflow are in sequence. The Multipath TCP sequence number enables the receiver to reorder the data received over subflows having different delays.

During the last years, various researchers have analyzed the performance of Multipath TCP [1, 4] and proposed various improvements [5]. Unfortunately, analyzing Multipath TCP performance remains difficult given the lack of flexible tools. We fill this gap with `mptcptrace`[3] that converts a Multipath TCP packet trace in various graphs and metrics that can be used by researchers. We illustrate a few of the capabilities of `mptcptrace` in this paper.

[1]As of this writing, there are already hundreds of millions of Apple smartphones that use Multipath TCP.

SIGCOMM'14, August 17–22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2836-4/14/08
http://dx.doi.org/10.1145/2619239.2631453.

2. ANALYZING MULTIPATH TCP TRACES

`tcptrace` [2], written by S. Ostermann, is a flexible packet trace analysis software. It extracts a wide range of metrics from a TCP packet trace and has been used by many researchers who analyze TCP performance. When confronted with performance problems, the first reaction of a Multipath TCP researcher will probably be to explore a packet trace with `tcptrace` to understand the problem. `tcptrace` is very convenient for such analysis since it provides both an interactive visualisation and summary statistics about each TCP connection.

As an example, we captured two long Multipath TCP connections between two virtual machines in distant Amazon EC2 datacenters. Each connection was configured to use four TCP subflows and a window of 1.2 MB. The goodput of these two connections was very different. The first one reached 21 Mbps while the second managed to achieve 40 Mbps.

`tcptrace` cannot be used directly on these traces because it cannot link the subflows together. This link can be extracted from the Multipath TCP options used in the three-way handshake. If we do this manually, we can compare the round-trip-times of the four subflows of each Multipath TCP connection. On the worse performing Multipath TCP connection, we observe a large deviation of the round-trip-time values. On the other hand, the best connection has stable round-trip-times, but the subflows achieve different throughputs. On this connection, the subflow with the lowest average round-trip-time reaches the highest throughput.

What is the reason for large deviation in the round-trip times of the low performing connection ? Figure 1 provides a CDF of the round-trip times measured over each subflow. The right part of the figure confirms the stable round-trip-times of high goodput connection. The left part shows that for the low-performing connection, three subflows have two clusters of round-trip times. One cluster arounds 200 msec and another around 400 msec. This indicates a rerouting even during the lifetime of this connection that forced those three subflows to follow a longer path. This rerouting did not affect the other subflow.

[2]`http://www.tcptrace.org/`

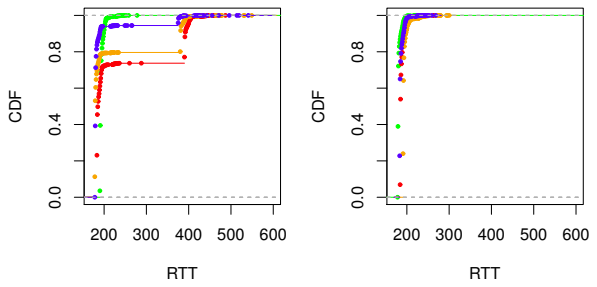

Figure 1: Per flow RTT CDF

2.1 Measuring Reordering

A regular TCP receiver reorders the data received out-of-sequence before delivering them to the application. In today's networks, reordering mainly happens after a packet loss. With Multipath TCP, reordering becomes normal. Since Multipath TCP uses subflows with different round-trip times, data is not always received in sequence at the receiver. These out-of-sequence data consume both memory and CPU time and furthermore increase the end-to-end delay perceived by the application.

Measuring and understanding when reordering occurs is major issue when analyzing Multipath TCP performance. `mptcptrace` can help such analysis by parsing all the Multipath TCP options and understanding their semantics.

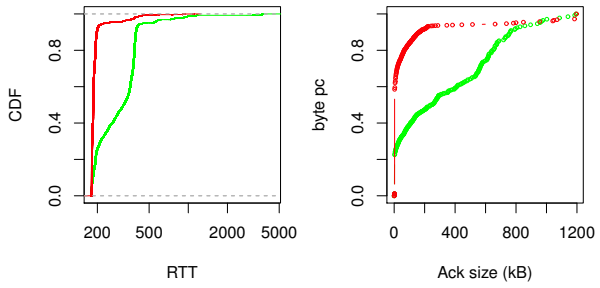

Figure 2: MPTCP rtt CDF and Ack size

The left part of figure 2 analyzes the round-trip times at the Multipath TCP level on each connection. For this, `mptcptrace` computes a round-trip time for each data successfully transmitted by analyzing the Multipath TCP sequence numbers and acknowledgements. The high performing connection is characterized by small end-to-end round-trip times while they are much larger with the low performing connection. Note that compared with figure 1, for this connection, the Multipath TCP round-trip times are much larger than the per-subflow round-trip times. This indicates additional delays at the receiver side.

Another angle to analyze this problem are the Multipath TCP acks. With regular TCP, when data is received in sequence, it is immediately acknowledged and each ack acknowledges one packet's worth of data. With Multipath TCP, data is sent over different subflows. Even if there are no losses, data can be out-of-order at the Multipath TCP level when subflows have different delays. The right part of figure 2 shows that the cumulative percentage of the bytes that are acknowledged by acks of size x. On the high performing connection, 70 % of the traffic is acknowledged by acks that acknowledge less than 10 KBytes. 90 % of the traffic is acknowledged by acks of size less than 200 KBytes. This indicates a small reordering queue. For the low performing subflow, the situation is different. We observe that a large fraction of the traffic is acknowledged in bursts. This indicates a large occupancy of the reordering queue.

Figure 3 highlights the impact of this problem on the performance of the Multipath TCP connection. Three curves are plotted. The green curve is the receive window advertised by the receiver. The read lines show the sum the amount of unacknowledged data over each of the four subflows. The orange lines show the data that has been sent by Multipath TCP (over any subflow) and not yet acknowledged by a Multipath TCP acknowledgement. The gap between the red and the orange curves corresponds to the size of the reordering queue at the Multipath TCP level.

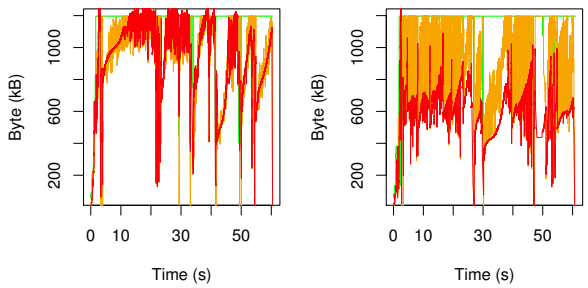

Figure 3: TCP and MPTCP unacknowledged data

The goodput of a Multipath TCP connection is not simply the sum of the goodput of the underlying subflows because the same data can be transmitted over several subflows. This can happen when losses occur, subflows terminate or when a subflow is underperforming [5]. By parsing all Multipath TCP options, `mptcptrace` detects all these retransmissions. In our example traces, 0.53% of the bytes were transmitted on more than one subflow for the low performing connection. For the high performing connection, 0.11% of the bytes were transmitted on more than one subflow. `mptcptrace` detected that 0.01% of the bytes were transmitted over three of the four subflows. This was likely due to the reinjection algorithm explained in [5].

Acknowledgements

This work was supported by FNRS under project T.0128.13.

3. REFERENCES

[1] Y.-C. Chen et al. A Measurement-based Study of Multipath TCP Performance over Wireless Networks. In *ACM IMC*, 2013.

[2] A. Ford et al. TCP Extensions for Multipath Operation with Multiple Addresses. RFC6824, Jan. 2013.

[3] B. Hesmans. MPTCP traces analysis tool. `http://mptcptrace.multipath-tcp.org/`.

[4] 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.

[5] C. Raiciu et al. How hard can it be? Designing and Implementing a Deployable Multipath TCP. In *USENIX NSDI*, 2012.