The Performance of Multi-Path TCP with Overlapping Paths

Lajos Zongor, Zalán Heszberger, Alija Pašić, János Tapolcai
MTA-BME Future Internet Research Group and MTA-BME Information Systems Research Group
Budapest University of Technology and Economics (BME), Hungary
{zongor,heszi,pasic,tapolcai}@tmit.bme.hu

ABSTRACT

Turning the Internet into a multi-path environment could solve many difficulties network operators are facing today. There are already solutions to configure an IP network to offer multiple partially disjoint paths towards the destination. In this demo, we focus on the performance of how MPTCP can distribute the traffic along these paths. When the routes are not fully disjoint their throughput could be limited by some bottleneck links. Because of this dependency finding the maximal throughput in MPTCP may call for solving a complex maximization problem. Through constructing an example network as an illustrative model of real network conditions, we show, how complicated the underlying optimization problem MPTCP may face, and through measurements, demonstrate how the various congestion mechanisms deal with finding a solution.

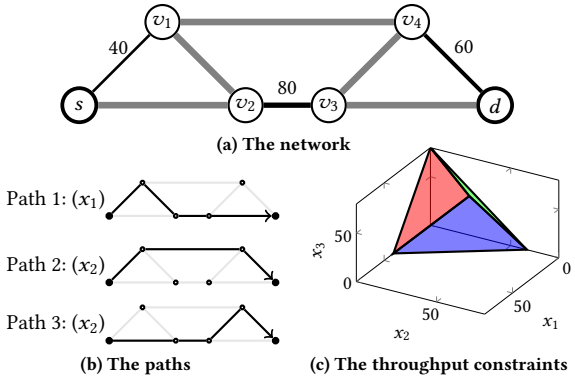


Figure 1: Illustration of the network topology

1 INTRODUCTION

Offering a selection of forwarding paths to the end-users of the Internet, who in turn monitor, control, and optimize their sending rate along these paths in an end-to-end fashion, would improve end-to-end reliability, security, and latency by allowing users to avoid congested links, and even provide some control to applications to meet their performance requirements [1, 4, 5]. The promising technique to allow end-systems to influence path selection securely is tagging [3]. Tags are some short identifiers that are part of the header and can be used to control the route of the packet. The identifier has no global meaning, and the routing is deterministic, meaning that packets with the same tag always routed along the same path towards the destinations. There are many proposals to implement tagging: either as a shim protocol header or overloading specific bits in the IP header field [3], or through the hashing used in equal-cost multi-path routing (ECMP) [4].

Another mechanism that is readily available for multi-path Internet is Multi-Path TCP (MPTCP) [1], which could end-to-end monitor, control, and optimize the sending rate along these paths. MPTCP extends TCP so that a single connection can be striped across multiple sub-flows, each being a TCP session along a unique path. MPTCP became a part of Linux, iOS, and OSx operating systems since 2013, and presents the same socket interface as TCP. The primary use case of MPTCP is when the host is connected to the internet through multiple wireless networks; such as both Wi-Fi and cellular networks. However, in this use case the delay and bandwidth characteristics of the paths used by MPTCP are mostly independent. In this paper, we are mainly focusing on how MPTCP behaves if there are common bottleneck links along the paths MPTCP is running on. We set up a carefully constructed model of networking environments, where the paths are competing for the same resources, and thus sending more packets to one path degrades the throughput on the other. *In the demo, we attempt to compile stirring networking scenarios where finding the optimal performance is a challenging optimization task.* In the rest of the paper, we explain one use-case of the demo in detail, where the efficiency of some MPTCP congestion control mechanisms is demonstrated in searching for the optimal throughput by balancing the transmission rate between paths whose characteristics depend on each other.

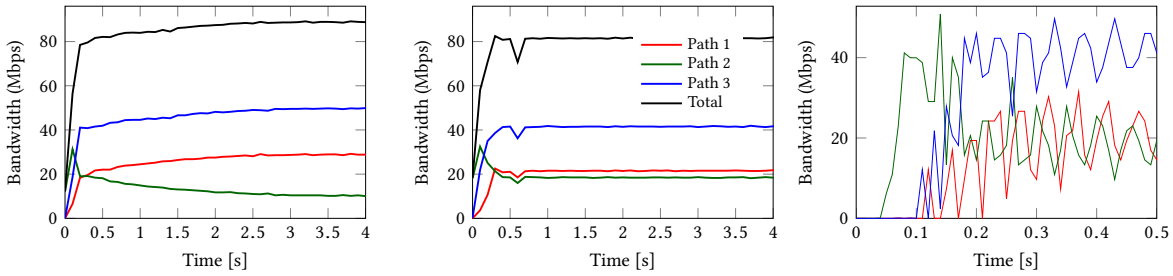
2 METHODOLOGY

2.1 Setup

Fig. 1a shows the network we have constructed to measure the MPTCP performance. Fig. 1b shows the three paths the MPTCP at node s can select to communicate with node d. The capacities are written next to the links unless they are the default 100. Let x_i denote the throughput of the i^{th} path for $i = 1, 2, 3$. In this case Path 1 and Path 2 have a common link $s - v_1$ with capacity 40, which means we have the following inequality $x_1 + x_2 \leq 40$. Similarly we have $x_2 + x_3 \leq 60$, $x_1 + x_3 \leq 80$. Informally speaking we have three bottleneck links corresponding to each pair of paths. It means the MPTCP load balancer is facing a multidimensional optimization problem with the following objective function max $x_1 + x_2 + x_3$. Solving the above linear program we get the following optimal solution $x_1 = 10$, $x_2 = 30$, and $x_3 = 50$, which is 90 altogether. In the

SIGCOMM Posters and Demos '19, August 19–23, 2019, Beijing, China

ACM ISBN 978-1-4503-6886-5/19/08…$15.00
https://doi.org/10.1145/3342280.3342328



(a) The rate of each flow with CUBIC (100ms) (b) The rate of each flow with OLIA (100ms) (c) The rate of each flow (10ms)

Figure 2: MPTCP throughput with CUBIC and OLIA congestion control algorithms measured with tshark.

measurement setup *we are interested in whether the MPTCP congestion mechanism can find the optimal throughput or not.* Note that the above problem is a convex optimization, therefore, there is a single (global) maximum. Convex optimization is often solved with some type of gradient descent method, which is an iterative approach always stepping towards the gradient (also called steepest descent). On the other hand, MPTCP controls the rate of each TCP path together to find the largest total throughput. In the above settings the simplest greedy approach to increase the rates independently would give a suboptimal solution.

2.2 Measurement

We used version 0.94 of the Multipath TCP linux kernel, and Mininet for the network simulation. To make MPTCP use preselected paths in the model network a tagging mechanism is used by applying 3 different tags on the packets of the subflows corresponding to the 3 different paths. We modified the ndiffports [4] path-manager to tag each subflow. The exact tags and the number of subflows is given as an argument for our path-manager module. We used the default MPTCP scheduler, and run the measurements with three congestion control algorithms: CUBIC (the default in Linux), and LIA (Linked Increase algorithm) [6] and OLIA (Opportunistic Linked Increase Algorithm) [2].For traffic generation we used iperf, and captured the data stream by tshark at the destination node. Then we filtered the captured packets based on the tags, to determine how did the MPTCP protocol split them among the subflows.

3 RESULTS

Fig. 2 shows our measurement results, where an MPTCP connection was established with Path 2 as default shortest path (with the shortest round trip time) and Path 1 and Path 3 as additional routes to the destination node. Note that the default shortest path has a maximal capacity of 40 Mbps. The figure shows the throughput of each flow sampled with 10 or 100ms by tshark at the receiver side. Fig. 2(a) shows that MPTCP-CUBIC first increases the transmission rate on the default shortest path (Path 2) reaching the capacity of the bottleneck link (s, v_1), and subsequently increases the rate along Path 1 and 3 up to the corresponding bottleneck capacities: 60 Mbps on Path 1 at 0.05s and 80 Mbps on Path 3 at 0.15s. At this point, we have a Pareto optimal solution as none of the TCP rates can be increased independently. On the other hand, decreasing the rate of Path 2 by x would increase the rate for both Path 1 and 3 by $2x$ altogether. In the next 3 seconds, using its default congestion control algorithm the MPTCP is capable of finding the optimal throughput by rearranging the bandwidth on the different paths. Note that in case of CUBIC there is no interaction between the individual TCP congestion control actions. The success of MPTCP is likely due to the sawtooth nature of the TCP: when there is a packet drop along Path 2, the rate is reduced, meanwhile there is a chance to increase the rate on Path 1 or 3, see Fig. 2(c). At that point, the rate cannot be increased on Path 2, but can be on either of the paths. Intuitively speaking MPTCP does not stick in a deadlock but can "shake down" into rates giving the optimal throughput.

In our experiments, the default (CUBIC) congestion control algorithm always reached the optimum; however, later, the throughput was unstable for short periods. The more stable LIA never could reach the optimum, while OLIA was able to reach the optimum in many measurements, but only if Path 2 was the default shortest path among the three. See Fig. 2(b) for an example where it did not find the optimum. Note that OLIA had the slowest convergence time: it took 20 sec in the above example network to reach the optimum, but after that the throughput was stable.

4 CONCLUSIONS

In this demo, we set up a network topology and multiple routes between a source-destination pair, where MPTCP must perform a sophisticated optimization process to achieve the optimal throughput. We demonstrate that certain congestion mechanisms of MPTCP are capable of converging to the optimal solution while others cannot. The success of CUBIC is most likely stemming from the asynchronous controlling actions at the individual TCP paths inherently eventuating the required gradient optimization over the flows.

ACKNOWLEDGEMENTS

The research leading to these results was partially supported by the High Speed Networks Laboratory (HSNLab). Project no. 123957, 129589, 124171, 128062 and 124171 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the FK_17, KH_18, K_17, K_18 and K_17 funding schemes respectively. The research report in this paper was also supported by the BME-Artificial Intelligence FIKP grant of EMMI (BME FIKP-MI/SC). Z. Heszberger was supported by the János Bolyai Fellowship of the Hungarian Academy of Sciences.

REFERENCES

[1] O. Bonaventure, M. Handley, and C. Raiciu. 2012. An Overview of Multipath TCP. *Usenix ;login: magazine* 37, 5 (Oct. 2012).

[2] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec. 2013. MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution. *IEEE/ACM Transactions on Networking* 21, 5 (Oct 2013), 1651–1665.

[3] Murtaza Motiwala, Megan Elmore, Nick Feamster, and Santosh Vempala. 2008. Path Splicing. In *ACM SIGCOMM*. 27–38.

[4] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. 2011. Improving datacenter performance and robustness with multipath TCP. In *ACM SIGCOMM CCR*, Vol. 41. 266–277.

[5] Ashish Vulimiri, P. Brighten Godfrey, Radhika Mittal, Justine Sherry, Sylvia Ratnasamy, and Scott Shenker. 2013. Low Latency via Redundancy. In *CoNEXT*.

[6] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *Proc. USENIX NSDI*. Berkeley, CA, USA, 99–112.