# SDN based Testbeds for Evaluating and Promoting Multipath TCP

Balázs Sonkoly[†], Felicián Németh[‡], Levente Csikor[†], László Gulyás[*] and András Gulyás[‡]
[†] MTA-BME Future Internet Research Group
[‡] MTA-BME Information Systems Research Group
[†‡]Budapest University of Technology and Economics
[*]College of Szolnok
Email: [†‡]{sonkoly,nemethf,csikor,gulyas}@tmit.bme.hu, [*]gulyasl@szolf.hu

***Abstract*—Multipath TCP is an experimental transport protocol with remarkable recent past and non-negligible future potential. It has been standardized recently, however the evaluation studies focus only on a limited set of isolated use-cases and a comprehensive analysis or a feasible path of Internet-wide adoption is still missing. This is mostly because in the current networking practice it is unusual to configure multiple paths between the endpoints of a connection. Therefore, conducting and precisely controlling multipath experiments over the real "internet" is a challenging task for some experimenters and impossible for others. In this paper, we invoke SDN technology to make this control possible and exploit large-scale internet testbeds to conduct end-to-end MPTCP experiments. More specifically, we establish a special purpose control and measurement framework on top of two distinct internet testbeds. First, using the OpenFlow support of GÉANT, we build a testbed enabling measurements with real traffic. Second, we design and establish a publicly available large-scale multipath capable measurement framework on top of PlanetLab Europe and show the challenges of such a system. Furthermore, we present measurements results with MPTCP in both testbeds to get insight into its behavior in such not well explored environment.**

***Index Terms*—GÉANT, PlanetLab, OpenFlow, Multipath TCP**

## I. Introduction

The success of the internet was closely coupled with the evolution of TCP (Transmission Control Protocol) over the past decades. Different versions and enhancements of the protocol have been proposed to emerging network environments in order to ameliorate user experience from several aspects. Some TCP variants can achieve better end-to-end throughput, others mitigate fairness problems, while other approaches improve responsiveness to varying network conditions. A particularly interesting and promising candidate is Multipath TCP, which has been recently standardized by the IETF (RFC 6182 [1]). Multipath TCP enables the simultaneous use of several network interfaces of a single host presenting a regular TCP interface to applications, while in fact spreading data across several subflows and controlling sending rates of subflows in a coupled manner. By means of these subflows traversing different paths at the same time, better resource utilization, enhanced throughput and smoother reaction to failures can be achieved.

In spite of its great future potential, MPTCP is well investigated only for limited use-cases [2], [3]. But a comprehensive evaluation and measurements on today's internet infrastructure with fixed lines and single-homed users are not available. Why? The reason is quite clear: ISPs do not provide multiple connection services to end users[1]. The currently used networking paradigm and practice is settled to provide single path service between the endpoints of a TCP connection. Among such circumstances, it is hard to obtain multiple paths in an inter-domain setting, which is clearly an obstacle to experimenting with and adopting the technology.

In this paper, *we invoke the concept of SDN to eliminate this obstacle and establish a control and measurement framework where multiple paths can be configured dynamically, and MPTCP experiments can be orchestrated in a flexible way to evaluate the protocol.* As a platform, we can benefit from the abundance of free-access real internet testbeds, such as GENI, Internet2, OFELIA, GÉANT, PlanetLab, PlanetLab Europe, which have been established with different purposes, features and services. A key component which we require here is OpenFlow support. PlanetLab Europe and GÉANT have been recently extended by this new feature, so we have the chance to work as early birds with these experimental tools. Thus, *we build our test framework on top of GÉANT and PlanetLab Europe, respectively, in order to catalyze both the evaluation and, as we hope, the diffusion of multipath TCP.* Our publicly available test tool [4] can be used either by experimenters and researchers to test and verify their multipath-related ideas (e.g., enhancing congestion control, fairness or multipath routing) and also by early adopters to enhance their internet connection even if they are single-homed.

The rest of the paper is organized as follows. In Section II, the related works on multipath transport mechanisms and evaluation of the protocols are summarized briefly. Section III is dedicated to present our control and measurement frameworks, and Section IV highlights our main measurements results. Finally, Section V concludes the paper.

## II. Related Work

There were significant efforts on designing and evaluating different multipath transport protocols during the last decade

[1]An ISP might use load-balancers, however (i) the experimenter does not have control over the data paths, (ii) the transport on disjoint links usually occurs in that ISP's domain only.

978-1-4799-2003-7/14/$31.00 ©2014 IEEE

[5], [6], [7], [8], [9], however none of them has reached the standardization phase. In contrast to these proposals, Multipath TCP (MPTCP) has already been standardized, and it got into the focus of the research community becoming a de-facto reference work by an available Linux kernel implementation of the protocol [10].

MPTCP in its current form, however, is not without shortcomings. It is based on TCP Reno [11], which is unable to efficiently utilize available bandwidth when the connection's bandwidth-delay product (BDP) is large. Therefore, MPCubic [12] transplants the multipath part of MPTCP from Reno to Cubic to get rid of the traditional conservative congestion control. Instead of the lost-based Reno, wVegas [13] uses packet queuing as congestion signals, and achieves finer grained load-balancing among the multiple paths. Lossy wireless links and sudden changes in path characteristics have negative effects on MPTCP's performance as well. A novel congestion window adaptation algorithm [14] has been proposed to reduce the large number of out-of-order packets caused by wireless links. Fountain code-based Multipath TCP [15] mitigates the negative impact of the heterogeneity of different paths by specially encoding the data and distributing it among multiple paths. Unfortunately, all the above improvements over MPTCP have been examined analytically or by simulations only. Nonetheless, there is an exception. The opportunistic linked increases algorithm (OLIA) [16] providing better responsiveness has been studied in a very simple testbed. As a consequence, the OLIA implementation is now part of the official Linux MPTCP distribution, which shows an important side-effect of choosing measurement based evaluation over simulations.

In addition to the OLIA paper, there are only a few works available that evaluate MPTCP by measurements. Implementers of MPTCP showed its behaviour in a data center setting, where multiple edge disjoint paths between server nodes were provided by careful topology design [2]. They addressed challenging issues of the mobile user-case design, and measured overall throughput gain [10]. The benefits of the transport mechanism over parallel 3G and WiFi accesses are clearly shown in several additional studies [3], [17], [18]. Moreover, it has been demonstrated recently that MPTCP can achieve ultra high throughput under special circumstances [19].

TCP versions can be compared based on many performance metrics [20]; but only a portion of those can be evaluated in numerous simulated and emulated environments[2]. For instance, link bandwidths can be easily managed by a flexible tool called *dummynet*. *Mininet* emulates a whole virtual network with real kernel, switch and application codes running on a single machine. *Emulab* allows to specify an arbitrary network topology by means of PC nodes deployed in a facility. However, these tools are unsuitable for internet-scale MPTCP experiments. These experiments are necessary, e.g., to determine the most appropriate retransmission strategy [1]. On the other hand, the OpenFlow protocol, or more generally Software Defined Networking (SDN) enables programmatically creating complex topologies and dynamically configuring their nodes. By now, several OpenFlow capable real internet testbeds with free access are in operation. These systems including GENI, Internet2, OFELIA, GÉANT, PlanetLab, PlanetLab Europe[3], can be a basis of a common measurement platform.

[2]info.iet.unipi.it/~luigi/dummynet, mininet.org, www.emulab.net

## III. Control and measurement framework

This section is devoted to summarize our SDN based control and measurement framework designed and established on top of GÉANT and PlanetLab Europe, respectively.

### A. On top of GÉANT OpenFlow facility

GÉANT is the pan-European research and education network interconnecting Europe's National Research and Education Networks. The OpenFlow facility of this backbone network is launched recently and gives great opportunity for researchers as provides geographically distributed sites connected by high speed, point-to-point optical links. Currently, OpenFlow testbed consists of five nodes connected in full-mesh (see Fig. 1). The OpenFlow switches are implemented by Open vSwitches[4] running on Debian Linux machines. Connections between virtual ports of Open vSwitches are realized by Layer2 MPLS tunnels. The host components of the testbed run as virtual machines on top of dedicated hypervisors applying XEN paravirtualization technique and connect to nodes via virtual links. In order to share the facility among experimenters, a FlowVisor based slicing mechanism is also applied[5].

The architecture of the GÉANT OpenFlow facility provides precise and flexible control of networking components (e.g., provisioning disjoint paths) via dedicated APIs, while lower layer physical links between geographically distributed hosts enable real-life scenarios. Moreover, this facility makes possible to investigate a novel approach to network resiliency based on multipath transport. Currently, each networking layer redundantly implements its own recovery mechanism resulting in more expensive networking equipment and higher operational costs. The question naturally comes up. Can we get rid of all these mechanisms below the transport layer and use multipath transport protocol, e.g., MPTCP, to provide the required resiliency? Our framework makes it possible to address this question.

**Testbed objective:** Our main goal is therefore to implement a control framework for network resiliency measurements, which runs on a shared OpenFlow facility (GÉANT) used by several experimenters and also supports creating edge disjoint transport paths, the emulation of different recovery mechanisms, and the use of various transport protocols.

**Framework:** Our extensible control framework provides automated orchestration for measuring the characteristics of

[3]www.geni.net, www.internet2.edu, www.fp7-ofelia.eu, www.geant.net, svn.planet-lab.org/wiki/Openflow, www.planet-lab.eu/openflow

[4]http://openvswitch.org

[5]The presence of FlowVisor poses challenges on designing a general purpose framework (discussed later).

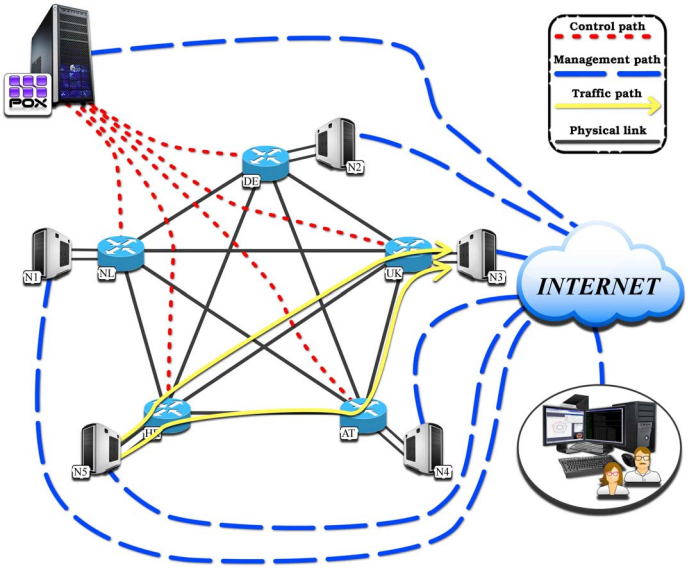



Fig. 1. Control and measurement framework on GÉANT OpenFlow testbed

different transport protocols under several error models and recovery strategies. It is able to emulate pre-scheduled link down events at different time scales (from 50ms to several seconds). The components of our framework are shown in Fig. 1. On the one hand, OpenFlow switches are connected by physical, point-to-point optical links and controlled by our dedicated POX-based[6] controller via an out-of-band control channel. On the other hand, we use a separated management channel in order to orchestrate the measurements, which includes the configuration of controller and end-hosts, starting logging functions, collecting measured parameters and providing results in a user-friendly way. The orchestrator of the measurements can connect to the network from the public internet. The framework is capable of running measurements with configurable traffic mixes between different source-destination pairs and pre-configured background traffic. Several parameters of TCPs can also be tuned (buffer sizes, initial behavior, etc.) and during an experiment, individual and overall throughput performances can be shown on-the-fly. Furthermore, the extended POXDesk component visualizes network topology with current link loads, plots per-flow throughput performance and provides network configuration options.

**Challenges:** GÉANT separates concurrent testbed users with FlowVisor; as a side effect, FlowVisor filters out port configuration commands. Hence, we need to emulate link outages by temporarily installing high priority drop rules into the flow tables of the corresponding switches. Moreover, the minimal hard timeout of flow entries is one second, therefore the controller needs to send flow remove messages after the flow install messages with precise timing. However, intersending time at the controller differs from the interarrival time at the switch. Our controller sends barrier messages to ease the estimation of the actual length of the emulated outage [21], which makes it possible to determine the accuracy of the measurement.

The following restoration scenarios are considered and integrated with our framework, and their impact on the end-to-end performance can be evaluated. At the *optical layer*, 50ms failure restoration is the well-accepted time limit while the network has to respond to any optical failure and restore the disrupted connection before any upper layer protocol would detect and react to these failures. TCP segments which are lost during this outage are retransmitted after the service is restored, however this can lead to temporary performance degradation. As we cannot access the optical devices from OpenFlow we emulate this behavior with priority drop rules. The *IP recovery* component makes it possible to measure the effects of an OpenFlow-based IP error recovery mechanism on the performance of transport protocols. The mechanism contains three steps: (i) detecting the failure at the OpenFlow switch, (ii) setting up a new route flow entry from the controller (or using another pre-installed one) and (iii) forwarding the packets using the new flow entry. The reaction time of the mechanism is inline with the range of the available MPLS/IP fast re-route and routing re-convergence times.

### B. On top of PlanetLab Europe

To take the next step toward real scale scenarios, PlanetLab Europe (PLE) is a reasonable choice. It has more than 300 nodes at more than 150 sites, and supports the development and testing of novel network mechanisms and applications through its additional services, such as the experimental OpenFlow support[7]. However, arranging slices, installing, configuring and integrating the currently available software components (e.g., MPTCP implementation, network settings, routing configuration) are not evident even for a qualified researcher. Furthermore, PLE gives a dedicated large-scale network, which could be used by end-users to get additional paths beside their regular internet paths. This raises the idea of an interesting experimental service for enhancing connections of simple end-users.

**Testbed objective:** Our main goal is to build a multipath playground on top of PLE both for experimenters and educated end-users. Our experimental framework is to enable experimenters to easily set up multipath capable overlay networks on PLE nodes with conveniently configurable routing, e.g., by managing simple configuration files. Additionally, a novel PLE service for early adopter users is also addressed, which can improve their internet connection in terms of throughput and resiliency.

**Architecture:** Our architecture shown in Fig. 2 follows an SDN-based approach at different levels. The control logic of the system is implemented in the POX controller platform as modules dedicated to distinct tasks. The data plane is actually an overlay network over PLE nodes consisting of (i)

[6]http://www.noxrepo.org/pox/about-pox

[7]By this time, PlanetLab Central also supports OpenFlow but in a lower level than PLE and they are not compatible with each other.

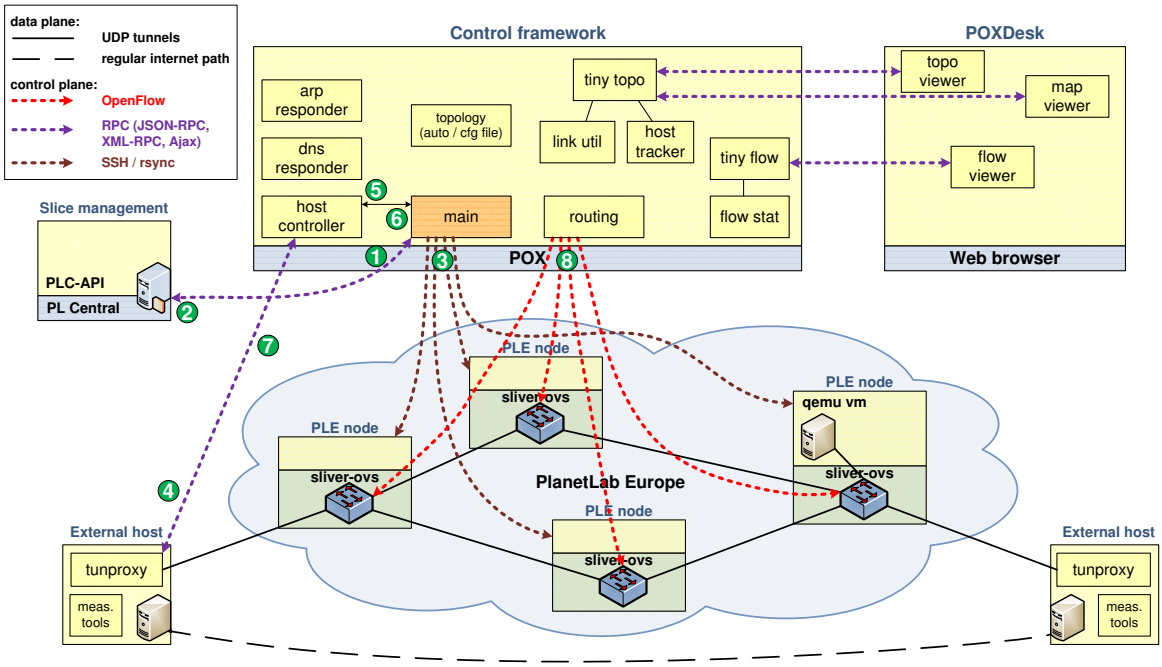



Fig. 2. PLE Architecture

`sliver-ovs` instances[8] controlled by our special purpose OpenFlow controller, and (ii) end hosts, which can be run on PLE sites as `qemu` virtual machines or can be external machines outside from the PlanetLab domain[9]. Data path elements are connected by UDP tunnels configured by the control framework according to given topology information. For all kinds of control traffic (OpenFlow, JSON-RPC, XML-RPC, SSH, rsync), we have a dedicated out-of-band channel.

The main steps of experimenting with a typical scenario are also given in Fig. 2. As an initial step, the experimenter should create a PLE slice[10] comprising the intended nodes via the PlanetLab Central API (PLC-API) or the management GUI. After starting our control framework, the main module exchanges information on the current slice via PLC-API. These steps (1, 2) are indicated by numbers beside the lines corresponding to control messages. In case of two-way communication (arrows at both line ends), the number is placed closer to the originator of the message. Based on current information, properly operating PLE sites can be chosen. At step 3, `sliver-ovs` instances are started on the nodes and connected by dynamically configured UDP tunnels. The topology information can be read from configuration file or the topology can be constructed automatically. After setting up switches, a host can initiate a connection to the overlay network. In our example, an external host requests information on the overlay network (step 4) from the host controller module via JSON-RPC channel. Host controller can provide a list of currently available overlay nodes after an internal communication with the main module (step 5-7). Then the host runs RTT measurements on the obtained list of PLE nodes in order to choose the closest (or $n$ closest) one. Based on the result, host sends a connection request to host controller and a similar communication pattern is realized than it is shown by step 4-7. At the end, host controller sends back the connection parameters, which can be used by tunproxy module to establish the UDP tunnel with the given nodes. As a last step (8), the routing module pushes flow entries into switch flow tables via OpenFlow messages according to the selected routing policy and establishes, for example the disjoint paths.

Our architecture also supports running internal hosts on PLE sites as `qemu` virtual machines. In this case, connecting to the overlay is managed by similar methods. Deploying own virtual machines enables experimenting with customized versions of MPTCP in a convenient way. Moreover, we provide pre-configured images, which can be deployed automatically by rsync. Beside the main components presented above, we have several additional building blocks. For example, on top of POXDesk, we have implemented different topology and link information viewer services, and a special module for presenting throughput of running flows on-the-fly. The communication between our POX and POXDesk components is based on Ajax. Other useful functions, such as ARP and DNS responders, can make the experimenter's life much easier.

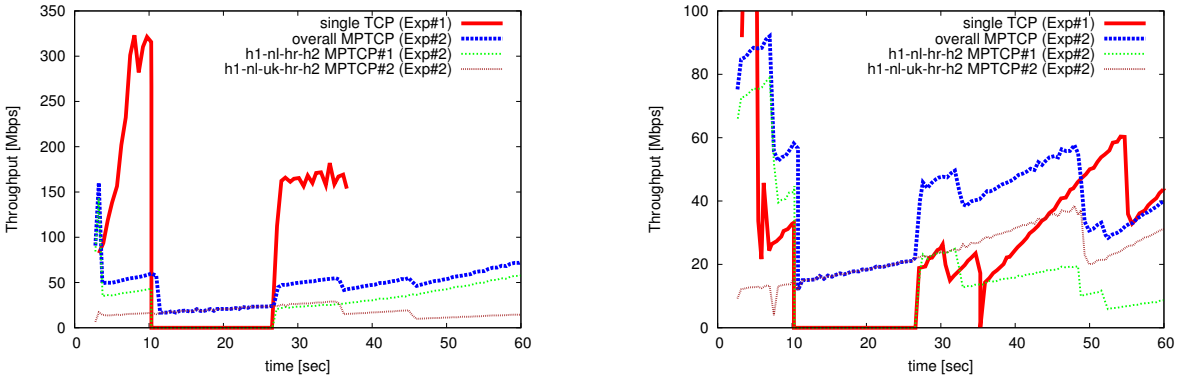
## IV. Experiments and Results

In this section, we present the results of the measurements we carried over our testbeds. Our results reinforce some

[8]OpenFlow support in PLE (sliver-ovs) is built around a modified version of the Open vSwitch software switch.

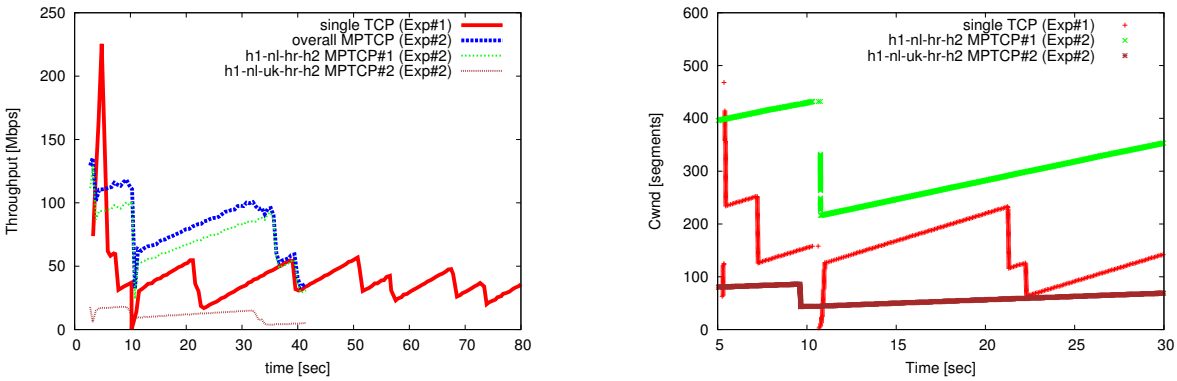
[9]Instead of OpenFlow, one could use NEPI[22] to customize routing on overlays. We think that an OpenFlow-based approach gives a more universal framework.

[10]Or she/he can use a pre-configured one, e.g., our slice.




(a) Without background traffic (b) With 200Mbps UDP background

Fig. 3. Comparing TCP and MPTCP performance in single-flow scenarios with link failure from 10s to 20s




(a) Throughput (b) Congestion windows

Fig. 4. Comparing TCP and MPTCP behavior in case of 100ms link failure at 10s

already known facts about MPTCP, but unforeseen and counter intuitive features are also discovered.

During the experiments, we use the default TCP version of current Linux kernel, namely TCP Cubic, and the "coupled" congestion control for multipath scenarios which is the default setting of MPTCP. TCP traffic and UDP background traffic are both generated by iperf.

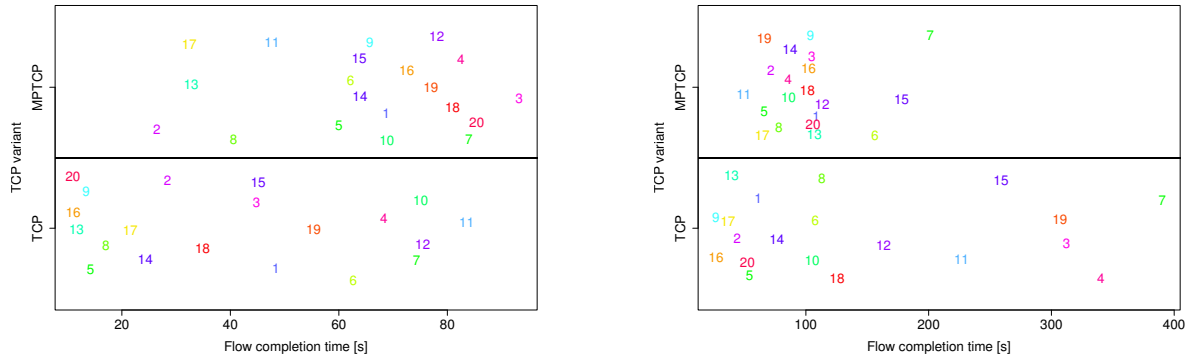
### A. Resilience measurements with simple traffic patterns

Since almost nothing is known about the behavior of the protocol in the presence of link failures we start with our resiliency measurements. First, we show the results of scenarios having a single flow between a single source-destination pair. This is a simple setup but brings us closer to understand the fundamental behavior of the protocol in a tractable fashion. In Fig. 3, the throughput performance of single TCP and MPTCP are compared in case of a link failure from 10s to 20s in the GÉANT testbed. The two plots correspond to scenarios with and without background traffic, respectively. In case of large BDP (a), MPTCP is not responsive enough due to its conservative additive increase mechanism and shows longer flow completion times. The poor performance and underutilization of link capacity is rooted in the Reno-based congestion control of MPTCP, while single flow TCP is controlled by the more effective Cubic mechanisms. When lower bottleneck capacity is available (b), MPTCP outperforms single TCP in the same environment. (Exp#1 and Exp#2 are different experiments in both cases.)

Similar experiments are presented in Fig. 4, where the BDP is reduced by heavy UDP background traffic and a 100ms link failure is emulated at 10s. Throughput plots (a) indicate the much better performance of MPTCP with smaller flow completion time, while the congestion windows (b) help to explain the phenomenon around the link failure event. Here, MPTCP is able to recover from losses with fast recovery mechanism whereas single-flow TCP is forced back to slow start phase.

In our second set of scenarios, an all-to-all communication pattern is used, i.e., traffic is generated from all hosts to all other hosts in the GÉANT testbed. The comparison of flow completion times (FCT) measured in this case is depicted in Fig. 5. 20 flows are transferred by single TCP and MPTCP in two respective experiments. Corresponding flows (between the same source and destination) are identified by the same numbers [11] (and colors). (a) Large BDP has significant impact on MPTCP performance and FCT results show poor performance in contrast to single TCP. (b) In case of heavy background

[11] Y values are slightly randomized to improve visibility.




(a) Without background traffic

(b) With 125Mbps UDP, all-to-all background

Fig. 5. Comparing flow completion times (FCT) in case of all-to-all communication

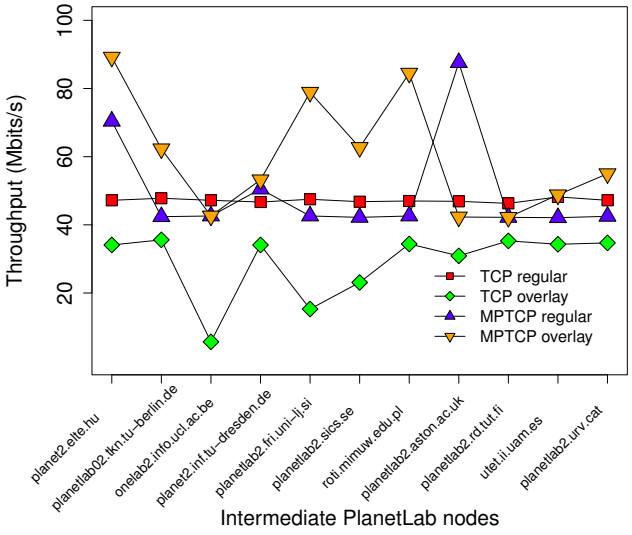
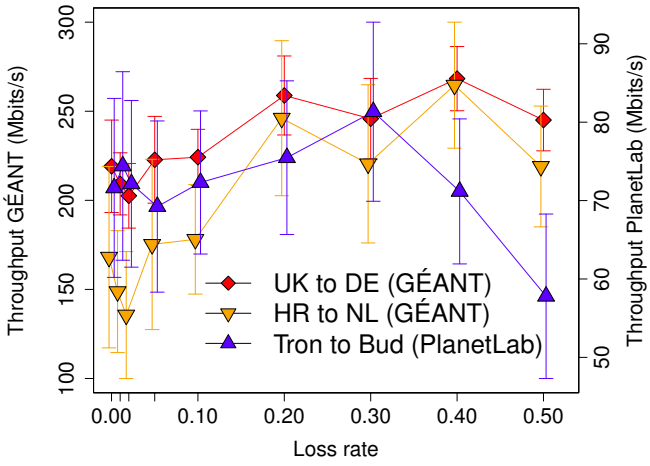



Fig. 6. Throughput of different MPTCP and regular TCP scenarios




| | UK to DE GÉANT | HR to NL GÉANT | Tron to Bud PL |
|---|---|---|---|
| Path 1 | direct<br>320.8 Mbits/s<br>18.6 ms | direct<br>258.5 Mbits/s<br>44.8 ms | regular internet<br>66.35 Mbits/s<br>48.2 ms |
| Path 2 | through HR<br>255.8 Mbits/s<br>75.4 ms | through UK<br>257.9 Mbits/s<br>47.8 ms | thr. ple2.ipv6.lip6.fr<br>17.53 Mbits/s<br>49.5ms |

Fig. 7. MPTCP throughput in case of packet loss

traffic (125Mbps UDP, all-to-all), MPTCP shows much better performance than regular single-path TCP.

### *B. Measurements with nodes having geographical diversity*

Our second set of measurements incorporate massive geographical diversity. Through this experiment, we can have a picture about how MPTCP behaves among real-life conditions bridging distant users. In this scenario, we use our PlanetLab testbed (also shown in Fig. 2). We have two end-hosts residing outside of the PLE, one is located at the UK GÉANT site, while the other is operated in our campus at BME. These hosts are connected through the regular internet path, but additionally, we configure them to have a second path through a variable intermediate PLE node. Then, we examine the throughput of MPTCP in comparison to regular TCP. Results are depicted in Fig. 6, where the different PLE nodes on $x$ axis are sorted in ascending order regarding to their latency. One can easily observe that the throughput of the single-path TCP using the regular internet path provides an approximately constant 47 Mbps (`TCP_regular`). If the single-path TCP uses the PLE path (`TCP_overlay`), on average its throughput is still above 25 Mbps. This is an acceptable performance knowing that traffic goes through a research network containing ordinary PCs. `MPTCP_regular` and `MPTCP_overlay` results stand for the cases when MPTCP is enabled and the primary path[12] is the regular and the overlay path, respectively. Interestingly, the performance can depend on the choice of the primary path. Nevertheless, in both cases, the performance of MPTCP is comparable and what is more, in several cases it outperforms regular TCP.

In our second scenario, we set up different loss rates on the primary path connecting our hosts. We conducted this experiment on both testbeds. In case of the PlanetLab testbed,

[12]The primary path is the path where the connection setup happens.

the primary path is the regular internet path connecting our host machine at NTNU Trondheim and in our campus at BME. Since the GÉANT testbed is fully meshed, the primary path is the direct path between the GÉANT nodes. The measured throughput is shown in Fig. 7 with respect to the loss rates. The outcome is rather counter intuitive, as common sense would expect that increasing loss rate implies lower throughput. Even so, after the loss rate reaches $2-5\%$, MPTCP starts to make better use of the secondary path indicating an improvement in the overall throughput. Putting it differently, one can have better throughput by simulating traffic loss on its primary path. On the table in Fig. 7 the properties of our paths are listed. From the values measured in the GÉANT testbed, we can see that this phenomenon appears either when the properties of the paths are different (UK to DE) or similar (HR to NL). Additionally, we observed that as the loss rate exceeds $50\%$ on the primary path, then the connection is not even established. Extreme loss rate is fatal for regular TCP as well, but MPTCP could try to establish connection through its secondary interface, which could be an easy enhancement to the current codebase.

## V. Conclusion and Future Work

In this paper, we have shown how large-scale SDN based testbeds can enable, speed up, and simplify the testing and evaluating emerging networking technologies. More specifically, we created SDN-based frameworks for evaluating and promoting the adoption of multipath TCP. The measurements over these testbeds pointed out clearly that, albeit the current MPTCP implementation can be used considerably well, it has also some major issues. Our findings can be the first steps of a comprehensive performance study on MPTCP. We close our paper by highlighting these issues to catalyze future improvements of the protocol.

- The throughput and resiliency of MPTCP highly depends on the choice of the primary path.
- The connection is not established at all if the primary path is down or heavily congested.
- In some cases, losses on the primary path results in much higher overall throughput.
- Sudden changes in the quality of the paths are not handled appropriately. In case of a link failure, the traffic from the failed path is redirected very slowly to the other available paths, which results in poor resiliency.
- The congestion control mechanism performs poorly in case of paths having large BDP.

## Acknowledgements

This work was partially performed in the High Speed Networks Laboratory at BME-TMIT. The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°287581 - OpenLab, and from the Hungarian National Development Agency through the Economic Development Operational Programme GOP-1.1.1-11-2012-0340. Levente Csikor was supported by the hungarian Sándor Csibi Research Grant.

## References

[1] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," IETF RFC 6182, Mar. 2011.

[2] C. Raiciu *et al.*, "Improving datacenter performance and robustness with multipath tcp," in *ACM SIGCOMM 2011*, 2011.

[3] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure, "Exploring Mobile/WiFi handover with multipath TCP," in *ACM SIGCOMM workshop on Cellular Networks (Cellnet'12)*, 2012.

[4] F. Németh, B. Sonkoly, L. Csikor, and A. Gulyás, "A large-scale multipath playground for experimenters and early adopters," in *ACM SIGCOMM (DEMO)*, Hong Hong, China, Aug. 2013, pp. 482–483. http://sb.tmit.bme.hu/mediawiki/index.php/MptcpPlayground.

[5] B. Chihani and D. Collange, "A survey on multipath transport protocols," in *arXiv:1112.4742.*, 2011.

[6] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using sctp multihoming over independent end-to-end paths," *Networking, IEEE/ACM Transactions on*, vol. 14, no. 5, pp. 951–964, 2006.

[7] H.-Y. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts," *Wireless Networks*, vol. 11, no. 1-2, pp. 99–114, 2005.

[8] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath congestion control for shared bottleneck," in *Proc. PFLDNeT workshop*, 2009.

[9] K. Rojviboonchai and A. Hitoshi, "An evaluation of multi-path transmission control protocol (m/tcp) with robust acknowledgement schemes," *IEICE trans. on communications*, vol. 87, no. 9, pp. 2699–2707, 2004.

[10] C. Raiciu *et al.*, "How hard can it be? designing and implementing a deployable multipath tcp," in *USENIX (NSDI'12), San Jose (CA)*, 2012.

[11] M. Allman, V. Paxson, and W. Stevens, "Tcp congestion control," IETF RFC 2581, Apr. 1999.

[12] T. A. Le, C. S. Hong, and S. Lee, "Mpcubic: An extended cubic tcp for multiple paths over high bandwidth-delay networks," in *IEEE Int. Conf. on ICT Convergence (ICTC)*, 2011, pp. 34–39.

[13] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath tcp," in *IEEE Inter. Conf. on Network Protocols (ICNP)*, 2012, pp. 1–10.

[14] D. Zhou, W. Song, and M. Shi, "Goodput improvement for multipath tcp by congestion window adaptation in multi-radio devices," in *Consumer Communications and Networking Conference (CCNC)*, 2013.

[15] Y. Cui, X. Wang, H. Wang, G. Pan, and Y. Wang, "Fmtcp: A fountain code-based multipath transmission control protocol," in *Int. Conf. on Distributed Computing Systems (ICDCS)*, 2012, pp. 366–375.

[16] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec, "Mptcp is not pareto-optimal: performance issues and a possible solution," in *Proc. of the 8th int. conf. on Emerging networking experiments and technologies*. ACM, 2012, pp. 1–12.

[17] G. Miguel and A. Singh, "Multipath TCP over WiFi and 3g links," Reproducing Network Research, blog post, Jun. 2012.

[18] 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 *Internet Measurement Conference*, Barcelona, Spain, Oct. 2013.

[19] R. van der Pol, M. Bredel, and A. Barczyk, "Experiences with MPTCP in an intercontinental multipathed OpenFlow network," in *TERENA Networking Conference (TNC)*, Maastricht, Netherlands, Jun. 2013.

[20] S. Floyd, Ed., "Metrics for the evaluation of congestion control mechanisms," IETF RFC 5166, Mar. 2008.

[21] F. Németh, B. Sonkoly, A. Gulyás, L. Csikor, J. Tapolcai, P. Babarczi, and G. Rétvári, "Improving resiliency and throughput of transport networks with openflow and multipath tcp," Budapest University of Technology and Economics, Technical Report, 2013. [Online]. Available: http://sb.tmit.bme.hu/mediawiki/index.php/GeantCompetition

[22] C. D. Freire, A. Quereilhac, T. Turletti, and W. Dabbous, "Automated deployment and customization of routing overlays on planetlab." in *TRIDENTCOM*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 44. Springer, 2012, pp. 240–255.