SEMPER: A Stateless Traffic Engineering Solution for WAN based on MP-TCP

Gines Garcia-Aviles*†, Marco Gramaglia†, Pablo Serrano†, Marc Portoles‡, Albert Banchs*†, Fabio Maino‡

* Institute IMDEA Networks, † Universidad Carlos III de Madrid, ‡ Cisco Systems

***Abstract*—Enterprise Networking has a strong set of requirements in terms of resiliency, reliability and resources usage. With current approaches being based on monolithic and expensive infrastructures using dedicated overlay links, providers are moving to more economical hybrid solutions that encompass private dedicated links with public/regular Internet connections. However, these usually rely on complex, hardware-dependent and/or proprietary Traffic Engineering (TE) solutions, which are computationally costly, in particular for the forwarding nodes. In this paper, we propose SEMPER: a lightweight TE solution based on MP-TCP that, in contrast to other TE solutions, moves the complexity to the endpoints of the connection, and relieves the forwarding elements from complex operations or even maintaining state. As our evaluation shows, SEMPER efficiently makes use of all available paths between the endpoints while maintaining fairness, and properly adapts to variations on the available capacity.**

I. Introduction

The enterprise wide area networks (WAN) paradigm enables the creation of a virtual private area network (VPN), linking different branches of an enterprise with their headquarters. Resiliency and Fault Tolerance are two of the most important requirements for enterprise WANs, and are becoming even more important nowadays with the availability of multiple links between nodes. This increase of different (physical) paths among different offices motivates the proliferation of Traffic Engineering (TE) solutions. Indeed, there is a lot of research effort focused on the improvement of these algorithms, with e.g. SD-WAN [1] being one example of a "hands on" product that manages enterprise WAN traffic over multiple links.

The most common enterprise WAN deployment consists of edge routers located in different campuses, called branches, and different links that interconnect them with their headquarters. Typically, those solutions are based on private overlay models [1], [2]. However, to achieve increased resiliency, reliability and an optimal usage of the resources for computing assets distributed across several locations, these solutions rely on monitoring systems to assess the traffic level of each link at any time. This operation, which is essential to forward/reroute flows through the best path, is usually costly in terms of computation time and resources, and prone to errors as it is based on traffic probes. In this paper, we propose a novel TE technique, that overcomes the disadvantages of stateful approaches to provide efficient connectivity between edge and central headquarters.

Our solution is based on moving the complexity to the network endpoints (in this case, the end hosts), by using the multipath version of TCP (MP-TCP) [3]. We leverage on its congestion control capabilities to simplify the operation of edge routers in both branches and headquarters.

The rest of this paper is organised as follows. In Section II we discuss the use of MP-TCP as a TE solution, including its potential benefits and challenges. In Section III we detail the design of our TE solution (SEMPER), a light-weight solution to efficiently and fairly distribute flows across existing paths between branches and headquarters. Section IV describes the setup and methodology to perform the evaluation, which is provided in Section V, showing the benefits of SEMPER. Finally, Section VI concludes the paper.

II. Using MP-TCP for Traffic Engineering

As discussed above, efficiently exploiting the availability of multiple (and possibly heterogeneous) paths is still an open research problem. Since the introduction of Equal Cost Multi-Path routing (ECMP) [4], the research community has been looking for a mechanism to efficiently deal with the availability of multiple paths. By "efficiently" we refer to solving a number of challenges entailed by the multi-path approach, such as, e.g., packet reordering, load balancing or fast re-routing in case of failures. In general, the main challenge is that the solutions to these problems do not have a straightforward implementation, and therefore multi-path is not a fully exploited paradigm in nowadays networks.

However, the introduction of MP-TCP [3] could completely change this situation. By coupling the congestion control over different sub-flows [5], MP-TCP can efficiently handle the shortcomings that the use of multiple paths introduces, dynamically balancing the transmission windows over different sub-flows. However, despite the clear advantages of an extensive deployment of MP-TCP, its far from widely adopted, relegated to a few remarkable use-cases such as e.g. Apple Siri [6] or Proxy enhancements [7]. However, even taking into account these examples, most applications and studies still focus on using a small set (in most cases: just a couple) of heterogeneous interfaces, and not in the design of a generic solution that can scale up to many paths.

The above cases are typically based on one of the two possible modes of operation supported by MP-TCP, namely, the `full-mesh` mode, which supports exploiting path diversity by generating one flow for each distinct source destination IP addresses pair. Another mode of operation is the `ndiffports` mode, which generates multiple flows at the

DOI: 10.1109/ICC.2018.8422991 https://ieeexplore.ieee.org/document/8422991/

source irrespective of the number of interfaces available, and relies on subsequent hops to exploit path diversity.

Indeed, MP-TCP `ndiffports` has already been proven as a potential effective solution for single-homed hosts in datacenter networks [8], being able to compete with alternatives specifically designed to take advantage of the regularity of such networks. However, WAN deployments will likely present a notable path heterogeneity, which is costly to exploit with *ad-hoc* solutions. This is where MP-TCP becomes even more valuable: by activating MP-TCP on all the end hosts within the branch campus, the outgoing traffic from the branch is already "multipath-ready," so an edge router can effectively exploit multiple paths at a relative low cost by forwarding sub-flows to the different paths.

One of the main benefits of this approach (that we will detail in the next section) is that all the complexity typically required by TE solutions, such as the complex monitoring operations (e.g., actively assessing the link capacity) and the corresponding load balancing schemes is pushed towards the edges of the network, as MP-TCP runs on the end hosts. With this approach, the design of the forwarding strategy (i.e., the path assignment) to be implemented at the edge routers becomes of paramount importance. We remark that it is a different problem from the one studied for ECMP forwarding, as we are not considering path assignment of single path flows, but rather path assignment of sub-flows belonging to same "parent" multipath flow.

In the next section, we design a solution for Traffic Engineering, which we refer to as SEMPER (StatEless Multi Path forwarding for Edge Routers). Two outstanding features of SEMPER are:

- It does not require any knowledge of the topology, this including key variables such as the number of disjoint paths between any two sites. This is in contrast with other solutions that might require the knowledge of this variable, to fix the number of sub-flows that have to be generated by MP-TCP accordingly.
- It does not require a complex scheme for the distribution of sub-flows between paths. Again, this is in contrast with other solutions that, knowing the "optimal" number of sub-flows, introduce the added complexity of a scheme to properly balance these among the set of available paths.

III. SEMPER: A TE SOLUTION BASED ON MP-TCP

We next describe the design of SEMPER, a TE solution that does not require the knowledge of the topology nor the use of complex sub-flow balancing schemes on the forwarding elements. The enabling technology is MP-TCP, that is enabled by default in all the hosts running in the enterprise branches, to deal with the congestion on the available paths. Therefore, we first describe the MP-TCP configuration used in the end hosts, and then the simple forwarding functionality that the edge routers connecting branches and headquarters need to implement.

To describe the changes to the end hosts, we start by describing MP-TCP by means of its main components, which

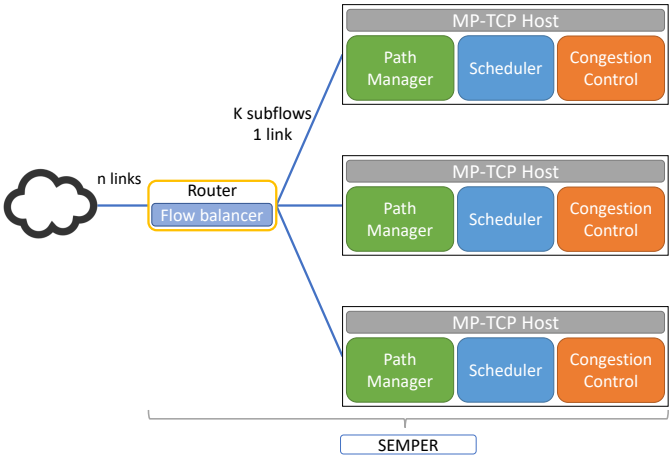


Fig. 1: SEMPER reference architecture

are depicted in Fig. 1. The first component is the *path-manager*, which is responsible for the TCP sub-flows that will conform the MP-TCP connection. As described before, this component supports different policies: the `full-mesh` policy creates a sub-flow for each available (source IP, destination IP) pair, while with the `ndiffports` policy the host will create a concrete number of sub-flows for the same pair of IP addresses, changing the source port.[1]

The second component is the *scheduler*. This module assigns higher-layer TCP segments over the existing sub-flows, taking into account the different characteristics of each sub-flow (e.g., the RTT), trying to optimise the overall performance. Here, among the different policies available, there are three worth mentioning: `default`, where paths with lowest RTT are preferred; `roundrobin`, which implements such policy across sub-flows, and `redundant`, where the same information is transmitted over all existing interfaces (for instance, this is the one used by Siri in Apple's iPhones, to maximise reliability). Finally, the *congestion control module* implements the congestion control algorithm, that computes the congestion window to be used on each existing sub-flow. During the SEMPER design phase, we decided to use the `default` scheduler and the OLIA [9] congestion control algorithm.

Number of sub-flows per host to generate

The first challenge when designing SEMPER is to select the proper number of sub-flows that each MP-TCP host has to generate, which we denote as K. Given an unknown number of available paths P between the branch and the headquarters, the actual value of K that should be used has an impact on performance, depending on the number of end hosts in each branch N:

- If $N \times K < P$, then not all capacity is used, as there are not enough sub-flows to occupy all the available paths.

[1]There is a also a `default` policy, where no flows are created but only accepted.

- if $N \times K \geq P$, there is at least one sub-flow per available path. Still, there are two challenges to address: (i) how to assign sub-flows to paths, and (ii) analyse if there is any penalty when using more sub-flows than the number of paths available [10].

The maximum number of sub-flows K that can be generated by a MP-TCP host for the `ndiffport` operational mode is 32. We remark that any adaptive solutions that tune the parameter K to the estimated topology, will require to use OS-dependent configuration interfaces. Therefore, finding a K value that behave well under all circumstances is the first challenge that we solved in SEMPER: we will discuss it in the next section. As for the second challenge, in our experiments we demonstrate that there is no major penalty in using an overly large value of K, which results in the following key building block for SEMPER:

SEMPER design principle #1: Each MP-TCP end host generates $K = 32$ sub-flows.

Mapping of sub-flows to paths

The other functionality to be designed, to run in the edge router, is the *flow balancer*. This module is responsible for assigning each sub-flow to one of the available paths. As mentioned, while this functionality falls outside MP-TCP, it is one of the main components of SEMPER. This module is placed in the edge router, as illustrated in the architecture depicted in Fig. 1.

One of the key objectives of the designed solution is a reduced complexity, leveraging on the end hosts running MP-TCP. To continue with this reduced complexity, we discard the use of passive or active monitoring, and build on MP-TCP's congestion control to properly adapt to the available capacity. More specifically, we rely on MP-TCP to efficiently and fairly distribute resources among the end hosts flows, and design a very simple flow balancer. The only requirement for this flow balancer is that TCP segments and acknowledgements from the same sub-flow have to be always assigned to the same path, so the congestion control can properly react.

The SEMPER flow balancer is based on performing a hash function on the 4-tuple composed by the source and destination IP addresses and port numbers, and map this hash to one of the available paths (numbered from 1 to n). For simplicity, we use a simple hash function based on the modulo operation, which will then map a sub-flow to a path with a probability $1/n$. We implement the corresponding (reverse) mapping function at the router at the other side, to support the required path symmetry.

SEMPER design principle #2: Use a hash function to randomly assign a TCP sub-flow to a path.

While this approach results in an extremely light-weight solution, there are two key issues that might result on non-optimal performance, that we discuss in the performance evaluation:

1) Due to the random assignment of sub-flows to paths, the complete utilization of all the available paths is not guaranteed (even when there are more sub-flows than paths).
2) All the paths used by all the sub-flows belonging to the same MP-TCP connection may not be disjoint.

Given the use of the hash, the last point above is influenced by source port generation, which we discuss next. Source port generation is an OS dependent procedure that is regulated only by a recommendation [11]. However, despite some remarkable exceptions such as e.g. certain Windows flavours [12], almost all the state of the art solutions are using random hash values to generate the source port, either at a global level (i.e., different processes share the same random seed) or at local level (one random seed per process). Therefore, we cannot use any predictive technique to guess the "next source port" for a given flow, and we have to assume a purely random process to avoid keeping any state. This further motivates the use of a hash function such as the one used by SEMPER.

Benchmark: a stateful solution

To have a proper performance comparison during our experiments, we designed an alternative approach to SEMPER, which will serve as a reference benchmark for the use of MP-TCP for traffic engineering. This approach consists on the following configuration:

- Each MP-TCP host is aware of the number of existing paths between the end points P, and generates one sub-flow per path, i.e., $K = P$.
- The *flow balancer* module distributes à la round-robin each of these sub-flows across the available paths.

With this approach, we guarantee that all sub-flows are evenly distributed across all the available paths. To distribute the load of the TCP handshakes across paths, the first sub-flow from a given host is hashed as in SEMPER, while the next sub-flows are assigned sequentially (modulo n, the total number of paths). Note that the price to pay for this approach, which evenly distributes flows across paths, is to keep state at both routers.

IV. Testbed and Methodology

Our aim is to design a solution readily deployable in a real-life environment. However, given the number of hosts and paths considered during our experiments, the cost of deploying an actual testbed of the required size would have been prohibitive. Because of this, we decide to use virtualisation techniques to carry out the performance evaluation, which supports using the *real* software implementation of the modules while considering large deployments.

We evaluate SEMPER against its stateful counterpart along three dimensions: throughput performance, fairness in throughput distribution, and fault tolerance. In the following, we describe the hardware and software configuration, and the methodology to compute the evaluation figures.

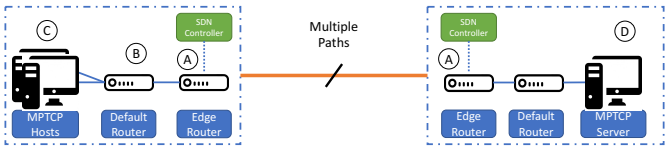


Fig. 2: Emulated topology throughout our experiments

Hardware and software configuration

We built our virtualisation environment using the `Mininet` network emulator [13], creating a network consisting of virtual hosts, switches, controllers, and links, where hosts run standard Linux network software. All the end hosts run a kernel implementation of MP-TCP, while the switches run Open Virtual Switch, supporting OpenFlow.[2] The software framework is depicted in Figure 2, which illustrates that we focus on the classical *dumbbell* topology, where two sites are connected via multiple disjoint paths. On the left, the edge routers (marked with an A) are connected to "default routers," (B) that serve a number of hosts (C), each running MP-TCP. On the right, there is an `Iperf` server (marked with a D), running MP-TCP. All routers run also the lightweight `RYU` SDN controller, which simplifies the implementation of the different sub-flow forwarding techniques discussed above.

Setting up a experiment

An experiment is defined by a number of configuration parameters that particularises the general topology depicted in Fig. 2, such as: the number of end hosts connected to the server N, the number of MP-TCP sub-flows generated by each host K, and the total duration of the experiment T. The capacity of each link is set to 50 Mb/s, and the edge routers run either the SEMPER configuration or the stateful forwarding algorithm.

As our objective is to measure performance under steady-state conditions, we cannot rely on the statistics provided by `Iperf` to assess the performance of each solution for different configurations (e.g. there is a non-negligible delay between the first and the last sub-flow connection). In fact, given the relatively large number of sub-flows that we consider, we found out that depending on the schedule of the sub-flow starts, the bandwidth of the links and the experiment duration, it could happen that the last sub-flow to start may send its first SYN way after the first sub-flow has already finished. Therefore, we next describe how we compute the performance for a given experiment.

Computing the throughput figures

We measure the throughput achieved during an experiment by parsing the `tcpdump` traces of the TCP segment sent over the various links. To consider only the steady-state conditions, we run an experiment for T seconds but only consider a window of T_M seconds, in which we confirmed that all the MP-TCP sub-flows are connected. After a thorough sampling of the configuration parameter space (omitted due to space reasons), we set $T = 400$ s and $T_M = 200$ s. We denote as R_i the throughput obtained by host H_i, which is obtained as the sum of the total number of bytes acknowledged for all its sub-flows divided by T_M.

[2]We use Mininet v2.3.0d1, the Linux kernel 3.18.20-90-mptcp, OVS v2.0.2, RYU v4.9 and Openflow v1.3.

V. Performance evaluation

Building on the methodology described above, we next evaluate the performance achieved by SEMPER, and compare it vs. the stateful solution. We start by validating the chosen configuration for the number of sub-flows per host of the stateful TE solution. After this validation, we first compare the performance in *static scenarios*, analysing the impact of the number of hosts N and the number of paths connecting the routers, and then in *dynamic scenarios*, where one of the links becomes unavailable at some point in time.

In our performance evaluation, we follow the methodology described in the previous section to compute set of the throughputs obtained by each host, $\{R_i\}$. Based on this set of values, we evaluate the performance via two variables:

- Total throughput: defined as the sum of the throughput obtained by each of the hosts, which will serve as a measurement of efficiency (the closer to the maximum capacity, the better).

$$\text{Total throughput} \triangleq \sum_i R_i \tag{1}$$

- Fairness: which serves to quantify the evenness in the distribution of the resources, and is computed following Jain's definition [14], i.e.,

$$\text{Fairness} \triangleq \frac{(\sum_i R_i)^2}{N \sum_i R_i^2} \tag{2}$$

  which ranges from one (perfect fairness) to $1/N$.

Validation of SEMPER configuration

We first evaluate the performance of SEMPER by using a varying number of generated sub-flows per host (K), to gather insight on the impact of this parameter and to confirm that our $K = 32$ setting is appropriate. We set-up a dumbbell topology with 8 links of 50 Mb/s connecting the two routers. We consider two different configurations for the number of hosts connecting to the MP-TCP `Iperf` server, namely, $N = \{2, 32\}$, and the following set of sub-flows per host $K = \{2, 4, 8, 16, 32\}$. We also compute the performance of scenario in which 8 hosts use a single TCP flow connected to the `Iperf` server, where each flow uses a different path, which will serve for reference purposes (i.e., a baseline).

For each scenario, we repeat each measurement 10 times, and compute the median of the results. We depict these in Fig. 3, illustrating the total throughput (left) and fairness (right) for an increasing number of sub-flows per host K.

Concerning the *total throughput*, the results show that when the number of hosts is small ($N = 2$) and SEMPER does not use enough sub-flows ($K < 8$), the performance is notably worse than the baseline. This is caused, of course, by the

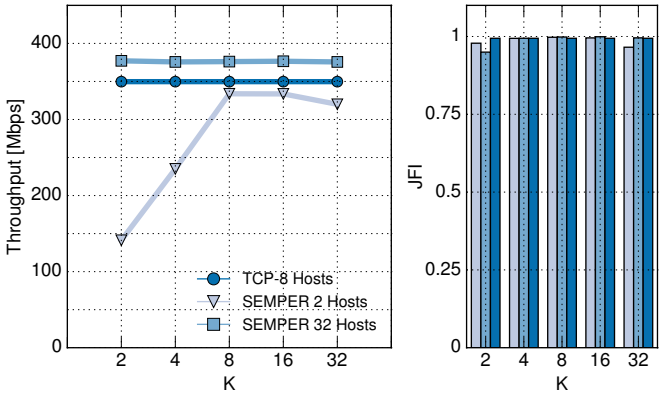


Fig. 3: Validation of SEMPER configuration

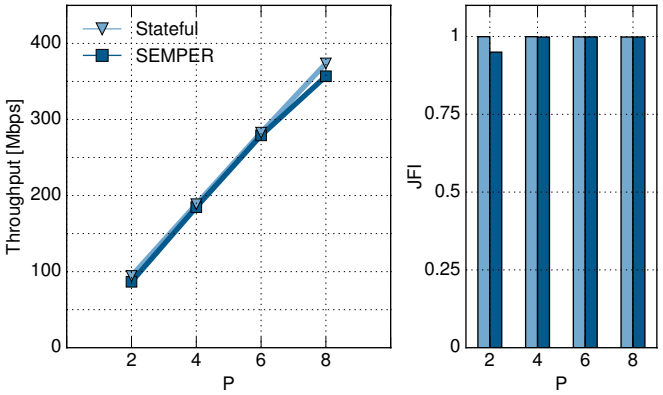


Fig. 4: Performance evaluation in static scenarios, $N = 4$ hosts

random assignment of sub-flows, which fails to occupy all the available paths (and therefore, the capacity). In contrast, when the number of hosts is large ($N = 32$) or the number of sub-flows is large ($K \geq 8$), the performance is closer to the baseline. We also note that the median values for $N = 2$ fall below those for $N = 32$. This is also caused by the random assignment of sub-flows to paths, which for the case of only 2 hosts, fails to occupy all the capacity in some experiments.

Considering *fairness* results, the performance is remarkably fair for all the considered scenarios, as for all configurations of the experiments the median is well above 0.9. There is a small drop for the cases of $N = 32$ hosts and only 2 sub-flows per host, which is caused by the very few cases in which a host has its two sub-flows sent over the same path (and therefore a relative lower throughput than others). Furthermore, this drop is "corresponded" for similar reasons by another small drop for $N = 2$ hosts and 32 sub-flows per host.

Given that the number of sub-flows has an impact on performance, but only if there are too few (and not too many), we conclude that our SEMPER configuration with $K = 32$ sub-flows per host is a suitable candidate to implement a TE solution, given its good performance in terms of efficiency and fairness. In the following section, we evaluate its performance in a number of scenarios, comparing the results obtained vs. the stateful solution.

Comparison in static scenarios

To compare the performance of the two TE solutions, we consider different dumbbell topologies in terms of the number of 50 Mb/s paths between the two routers, and different number of hosts. For each configuration, we compute the total throughput and fairness (as defined before) of the stateless and the stateful solutions. It is worth remarking here that the stateless solution does not require any knowledge of the topology, while the stateful is based on generating as many sub-flows per host as disjoint paths are available. We repeat each experiment 10 times, and compute the efficiency and fairness of each configuration, providing the median across experiments. We perform our evaluation for the case of $N = 4$ hosts and an increasing number of paths P between the routers, from 2 to 8 paths. The results are depicted in Fig. 4 for the total throughput (left) and fairness (right).

Concerning the *total throughput* figures, the results show that both TE solutions perform very efficiently, as in all cases the achieved results are very close to the total capacity between the links. Only for the case of 8 paths between the routers there are some differences between this maximum capacity (namely, 400 Mb/s) and the achieved capacity of both TE solutions, with the stateless version slightly underperforming the stateful. The reason for this small drop in performance for the proposed solution is, like in the previous section, the randomness of the assignment of sub-flows to paths, that in very few cases fails to occupy all available links.

For the case of *fairness*, the figure illustrates that both solutions perform very close to one (note that for the stateful solution, this is guaranteed by design). There is only one small drop in fairness for the case of two paths between hosts, which is caused when the total number of sub-flows is way larger than the total number of paths (note that we had a similar behaviour in the previous section).

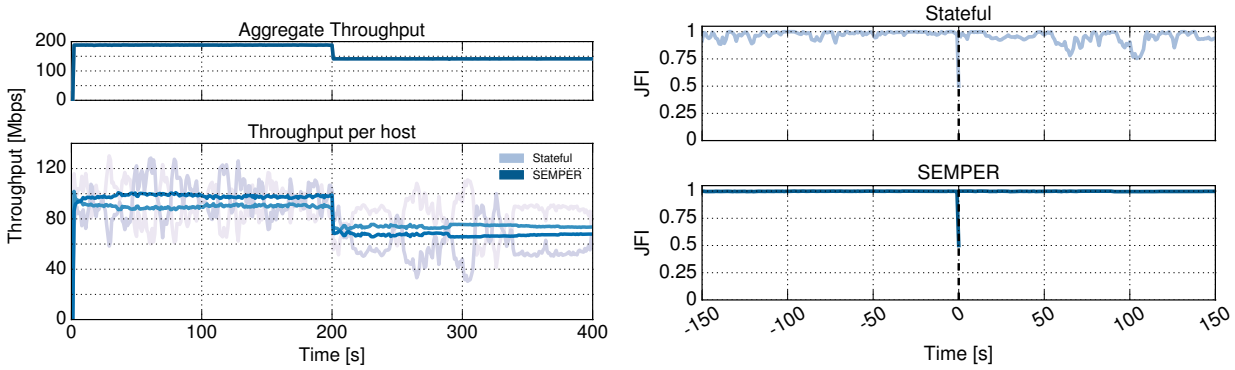
Following these results,[3] we conclude that the performance of both TE solutions is almost optimal. Furthermore, given that our TE proposal does not require a prior knowledge of the topology (to configure the number of sub-flows per host to generate), nor the use of state in the routers, it results a more deployable solution to efficiently use all resources available.

Comparison in dynamic scenarios

In the previous section, we consider static scenarios in terms of the available resources, i.e., a fixed number of paths between the hosts. As the results show, the designed TE solution makes an efficient and fair use of the available resources. Next, we address an scenario in which the amount of available resources are reduced at a given point in time, i.e., a link is down, to assess the performance in these circumstances.

For ease of visualisation, now we consider the same dumbbell topology as before, but with only $N = 2$ hosts, and $P = 4$ paths between the routers. We run the experiments

[3]Note that in this paper, for space reasons we only present the results corresponding to $N = 4$, but the performance is very similar for other N values.



(a) Instantaneous throughput over time.

(b) Instantaneous fairness over time.

Fig. 5: Performance evaluation in dynamic scenarios, $N = 2$ hosts

during T=400 s, and we remove one of the links at t=200 s. The resulting instantaneous throughput figures, averaged over 1 s, are depicted in Fig. 5, including the case of the total throughput (Fig. 5a, top) and the per-host throughput (Fig. 5a, bottom), for the two TE solutions considered.

According to the figure, both schemes adapt to the new circumstances with practically no loss of efficiency due to some "transient" conditions when adapting from a 4 paths to a 3 paths scenario.[4] We acknowledge that this result is somehow expected, as all involved transmission windows are operating at around the required values, and therefore the sudden unavailability of a link does not harm their operation in terms of total throughput performance. However, when considering the per-host performance, it is worth remarking the differences between the two TE solutions: while SEMPER provides a very smooth operation over time, the stateful solution solution exhibits the "usual" variability of TCP throughput, which is caused by the fast reaction of one flow to losses experienced by the other flow. Indeed, it can be seen the "symmetry" across the average throughput per flow between the two lines. We argue that this is an additional benefit of the proposed TE algorithm, namely, better fairness properties over time.

To look further into the fairness properties of SEMPER, we compute the fairness figures over the same 1 s time windows, and represent them in Fig. 5b. As expected, the smooth behaviour from the use of our TE solution results in very stable behaviour of fairness over time, while the use of the stateful approach results in less predictable figures, due to the throughput variability described above.

VI. Conclusions

Current Enterprise networks are moving from monolithic infrastructures based on dedicated links and costly components to newer approaches that reduce costs, as well as improving resources utilization, reliability and resiliency. Costly dedicated links are replaced by multiple public/shared internet connections, increasing the availability of multiple paths. Traffic engineering solutions are very used in practise to exploit multiple paths, but they exhaust the available resource at the edge routers because of the computational time and memory they require. In this paper we proposed SEMPER, a stateless solution for traffic engineering, that natively exploits path redundancy by efficiently matching MP-TCP subflows to paths, avoiding thus monitoring and fault avoidance techniques. Moreover, SEMPER moves the complexity to the end hosts by relying load balancing on the MP-TCP congestion control. Our results show that SEMPER is as effective as a stateful solution in terms of aggregating throughput and fairness, but with a practically negligible implementation cost.

[4]Like in the previous case, we do not report results corresponding to more paths between the routers as the observed behaviour is very similar in all cases. Furthermore, we omit the results corresponding to the re-activation of the link, as recovery is practically immediate.

References

[1] O. Michel and E. Keller, "SDN in wide-area networks: A survey," *IEEE Conference In Software Defined Systems*, 2017.

[2] B. S. Davie and Y. Rekhter "MPLS: technology and applications," *Morgan Kaufmann Publishers Inc.*, 2000.

[3] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "RFC 6824: TCP extensions for multipath operation with multiple addresses," *IETF*, 2013.

[4] C. Hopps, "RFC 2992: Analysis of an equal-cost multi-path algorithm," *IETF*, 2000.

[5] C. Raiciu, M. Handley, and D. Wischik, "RFC 6536: Coupled congestion control for multipath transport protocols," *IETF*, 2011

[6] O. Bonaventure and S. Seo, "Multipath TCP deployments," *IETF Journal 12*, 2016.

[7] X. Wei, "MPTCP proxy mechanisms, draft-wei-mptcp-proxy-mechanism-00", *IETF draft*, 2014.

[8] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene, "Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks," *ACM CoNEXT*, 2012.

[9] R. Khalili, N. Gast, M. Popovic, and J. Y. Le Boudec,"Mptcp is not pareto-optimal: Performance issues and a possible solution," *IEEE/ACM Transactions on Networking (ToN)*, 2013

[10] M. Sandri, A. Silva, L. A. Rocha, and F. L. Verdi. "On the benefits of using multipath tcp and openflow in shared bottlenecks," *IEEE AINA*, 2015.

[11] M. Larsen and F. Gont, "RFC 6056: Recommendations for Transport-Protocol Port Randomization," *IETF*, 2011.

[12] J. Kristoff, "Ephemeral Source Port Selection Strategies," Available Online https://www.cymru.com/jtk/misc/ephemeralports.html

[13] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible Network Experiments using Container-Based Emulation," *CoNEXT*, 2012.

[14] R. Jain, D. Chiu, and W. R. Hawe., "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," *Eastern Research Laboratory, Digital Equipment Corporation*, 1984.