Distributed Tactical TE with Segment Routing

Paolo Medagliani*, Sébastien Martin*, Youcef Magnouche*, Jérémie Leguay*, and Bruno Decraene†
*Huawei Technologies Ltd., Paris Research Center, France
†Orange Innovation, France

***Abstract*—Tactical Traffic Engineering (TE) solutions are a must to adapt traffic steering when unexpected congestions occur. While already available, centralized solutions to locally optimizes congested tunnels and links suffer from a slow reaction time of several minutes. To address this issue, we propose a distributed Congestion Mitigation (CM) mechanism that leverages Segment Routing (SR) to offload traffic away from congested links using Unequal Cost Multi Paths (UCMP) over alternative paths.**

In this paper, we introduce an efficient algorithm for alternative paths' computation, and two methods to compute UCMP weights, depending on whether remote link loads are available or not. We show that the proposed path computation method is faster than a modified K-shortest path algorithm. For traffic splitting, we show that the knowledge of remote link loads and per-destination traffic is a key to mitigate congestions in loaded scenarios, approaching the results obtained with an optimal solution. However, when not available, a local solution can already mitigate congestions in lightly loaded scenarios.

***Index Terms*—Segment Routing, Congestion Mitigation, Traffic Engineering, Distributed routing.**

I. Introduction

Routing is key to control how bandwidth is shared among different applications in computer networks. Traffic Engineering (TE) policies [1] can be optimized by a centralized controller and applied by routers to steer traffic with data plane mechanisms such as Segment Routing (SR) or MPLS (Multi- Protocol Label Switching) [2], [3]. While most of the traffic follows shortest path and hop-by-hop routing from Interior Gateway Protocols (IGP), a smaller portion is generally "engineered" to improve the bandwidth utilization or meet specific requirements (e.g, latency, security). Traffic engineering is typically applied at two time scales. Proactively, operators "strategically" plan and design policies to face plausible network evolution, e.g. traffic or failure scenarios, and meet custom requirements, e.g. Quality of Service (QoS). Furthermore, in real-time, they can "tactically" steer traffic by quickly re-engineering it as soon as an unexpected event occurs. These operations are often referred to as Strategic TE and Tactical TE, respectively.

Typical tools for Strategic TE leverage the presence of a centralized Path Computation Element (PCE) that implements routing algorithms, which are solving constrained shortest path or multi-commodity-flow problems to derive efficient policies [1]. PCE controllers can proactively re-optimize and reconfigure the network whenever traffic differs too much from the planned one [4]. Decisions can materialize by the configuration of SR policies or MPLS tunnels, or the re-optimization of IGP metrics. While Strategic TE operates at a slow pace and optimizes network utilization for the long-term, unexpected issues may occur. In this case, faster reactions are required with Tactical TE solutions. Despite PCE controllers can still be used to automatically react and adapt traffic steering, reconfiguration operations are slow, as they require the interaction between devices and the controller. In addition, the presence of a centralized network controller may not be desirable for scalability, fault-tolerance, or commercial reasons.

Congestion mitigation with Tactical TE calls for fast reactive mechanisms that are able to take control over forwarding operations, as soon as a problem in the network is detected by a router. It is then preferable to rely on distributed approaches directly implemented in network devices, as some applications, such as video streaming or VoIP, require quick reactions against performance degradations [5]. Despite that centralized solutions to locally optimizes congested tunnels and links exist, such as LCM (Local Congestion Mitigation) [6] feature from Cisco's Crosswork controller or "Network Path Navigation" feature in iMaster NCE-IP controller [7] from Huawei, they check for congestions typically at a cadence of 10 minutes [8], moving down to 1 minute if responsiveness is important. Therefore, centralized tactical TE solutions fail at providing quick reactions to sudden network variations due to failures or unexpected traffic bursts.

One efficient way to implement Tactical TE at device level is using SR, as it allows steering traffic over specific paths by adding an explicit list of network elements (i.e., segments) to be traversed. Solutions such as Topology Independent Loop Free Alternate (TI-LFA), which are already deployed in real networks to provide Fast ReRoute (FRR) functionalities, are based on SR [9]. Alternative paths are already installed in the routing table, so that they can be quickly activated as soon as a failure is detected. In the case of link failures, these mechanisms quickly protect the traffic flowing through a failed link and reroute it over the post-convergence shortest path towards each destination. The effectiveness of TI-LFA comes from the fact that the paths for rerouting are precomputed by the devices and activated when needed, without any interaction with a central controller, making it possible to achieve traffic protection in less than 50ms [5]. Another local reactive mechanism that benefits from SR is micro-loop avoidance [10], where potential loops due to routing IGP convergence issues are managed by explicitly listing the segments to be traversed. However, while distributed mechanisms for FRR and micro-loop protection are widely adopted in current networks, equivalent solutions for congestion mitigation do not exist.

Taking those issues into consideration, we propose a distributed Congestion Mitigation (CM) mechanism based on SR. When a router detects congestion on a link, a portion of the traffic can be automatically offloaded and load balanced over a set of alternative paths using UCMP (Unequal Cost

Multi Paths). The goal is to select lightly loaded paths to reroute a maximum of traffic, mitigate the congestion, and avoid creating new congestions elsewhere in the network. As congestions need to be quickly handled, we propose to carefully pre-compute paths, and select, on-the-fly, the weights (or split ratios) to load balance traffic. As for TI-LFA, alternative paths are proactively pre-computed and stored in the Routing Information Base (RIB). They are activated by updating the Forwarding Information Base (FIB) as soon as a congestion happens. Depending on the available information about local and remote link loads and the per-destination traffic over the congested link, different strategies can be adopted by routers.

In this paper, we introduce an alternative paths' computation algorithm and two methods to compute UCMP weights. We also discuss system aspects around the mechanisms for link load collection, traffic estimation, timer settings, etc., and propose practical solutions for the implementation in devices. We benchmark the performance of our solution using a packet-based simulator we developed. We show that the proposed Neighbor Deviation Algorithm (NDA) for alternative paths' pre-computation is more efficient in terms of running time, even if at the price of a lower path diversity than a K-shortest path algorithm. We also show that in lightly loaded scenarios, a load balancing strategy based on link capacities and local link loads, referred to as Local, can mitigate congestions. In high load scenarios, instead, only the Remote load balancing solution, using remote link load information and traffic sampling to estimate per-destination traffic over the congested link, can achieve near-optimal congestion mitigations, i.e. when compared to an ideal mechanism, called Optimal. We also show how the congestion mitigation mechanism can help in reducing packet losses and queue latency in the case of high link utilization. Finally, we show via realistic simulations how the freshness of collected statistics can impact the congestion mitigation mechanism. To the best of our knowledge, this is the first paper approaching the congestion mitigation problem from a completely distributed perspective with SR, making it suitable for implementation in existing devices for fast congestion mitigation.

This paper is structured as follows. Sec. II presents an overview of the state of the art and Sec. III introduces the proposed mechanism. Sec. IV describes algorithms for alternative paths computation and traffic splitting. Sec. V presents numerical results and Sec. VI concludes this paper.

II. Related Work

Several mechanisms have been proposed for tactical uses of SR. TI-LFA has been proposed to handle single failures (nodes, link or SRLG [9], [11]). For multiple failures, [12] proposed to compute several backup paths to consider different failure scenarios. SR extensions have also been proposed for micro-loop avoidance [10], [13].

Several works in the literature have introduced routing optimization to mitigate congestions. In very recent works, such as [14], [15], the authors suggest to leverage midpoint SR optimization to mitigate congestion in the network. This approach is based on a PCE to compute alternative SR policies for congestion mitigation. At the same time, as the PCE has the complete view of the network, the authors aim at keeping the number of deployed policies as low as possible. However, this centralized approach fails to guarantee fast reaction times due to required interactions with a central element.

Centralized load balancing solutions have been proposed for Unequal Cost Multi-Pathing (UCMP) [16] or Weighted Cost Multi-Pathing (WCMP) [17]. Decentralized load balancing solutions have also been developed to dynamically adjust routing in case of congestion or link failure. For instance, CONGA [18], and HULA [19] can dynamically adjust at ingress nodes load balancing policies based on path-based measurements to reduce network congestion. However, they assume that paths are given, they are not designed to offload traffic from a congested link so as to not create new congestions elsewhere.

Also of interest and based on SR, a semi-distributed load balancing solution has been proposed for application-awareness [20]. In this case, SR is used to direct data packets from each incoming flow through a chain of candidate servers, until one decides to accept the connection, based on its local state. A controller takes care of installing the mapping between a virtual IP (VIP), identifying a given application, and the set of addresses of servers capable of serving that VIP. The main benefit is that the controller does not need anymore to care of healthchecking backend servers, and removing them from the pool of available servers if unresponsive.

III. Distributed Tactical TE

This section introduces from a use case and system perspective the distributed tactical TE solution, referred to as CM.

A. Use Case

Service providers typically plan their transport networks to provide customer satisfaction and resilience. They typically size them to operate at low load, e.g., with a Maximum Link Utilization (MLU) of 30-50%, to sustain at least all possible 1-link failures. However, as real traffic can differ from the expected one, and multiple failures (e.g., of an SRLG) may occur simultaneously, link congestions can still happen. One possible approach to mitigate congestions is to use a centralized PCE, but this solution can be ineffective for large backbone transport networks, where the geographical distance can impact the reaction speed to handle local congestions. The need for quick congestion mitigation calls for fast and local reaction mechanisms that can steer traffic away from the congested resources.

Fig. 1 depicts a typical service provider backbone network where CM can be useful. Several Customer Edges (CEs) are connected to the backbone via Provider Edge (PE) nodes, i.e., the routers that are at the boundary of the backbone network. The PE nodes are responsible for forwarding the different types of traffic to the Provider (P) nodes, which route traffic inside the network. In case of failure or traffic variations, one or several links may become congested, and P nodes need to quickly react locally. However, to keep network operations safe and avoid any impact on critical traffic, part of the traffic must not be re-routed. Indeed, a subset of the traffic may be

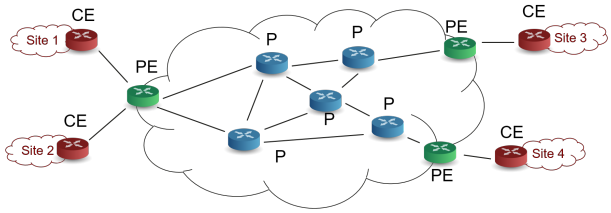


Figure 1: Example of a backbone network.

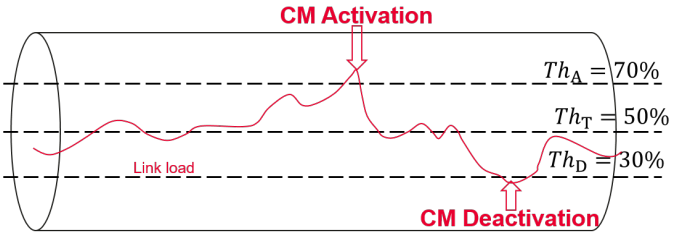


Figure 2: An example of the CM mechanism within a link. When the link load exceeds $Th_{\mathrm{H}} = 70\%$, the CM mechanism reroutes traffic over alternative paths to reach $Th_{\mathrm{T}} = 50\%$.

already routed with specific TE policies. In case of congestion, "engineered" traffic should not change path as TE policies have been selected based on specific requirements, such as QoS assurance, which are unknown to P nodes. In addition, a wide variety of control plane traffic is exchanged between backbone routers and should not be touched. For instance, STAMP [21], BFD [22], traffic to monitor the performance of the network, OSPF LSA messages to broadcast the network status, etc. Also of interest, some routers may have specific functions, e.g. BGP Router Reflectors (RR), and may not be used to detour traffic.

B. Overall principle

Whenever the load on a link exceeds a given congestion threshold, the Congestion Mitigation (CM) mechanism can be immediately activated by the P node detecting the congestion, in order to quickly react locally, without interacting with an external controller. As soon as a P node detects a congestion on a link, it activates the CM mechanism and steers a portion of the traffic traversing the congested link over a set of paths, referred to as alternative paths. In order to avoid creating congestions elsewhere in the network, the CM mechanism must carefully choose which traffic and what amount should be rerouted, and the split ratios that should be applied.

As mentioned above, only the low priority traffic should be steered by the CM mechanism, while high priority or engineered traffic must remain on their original path. Once the traffic to be moved has been quantified, the CM mechanism load balances the traffic over available alternative paths following the UCMP mechanism. The details on alternative paths computation and weights (split ratios) assignment will be provided in Sec. IV.

In Fig. 2, we show an example to illustrate how CM operates on a link. When the link load exceeds a congestion Activation threshold Th_{A}, e.g., 70%, part of the traffic is rerouted on alternative paths. The CM mechanism defines a second threshold, referred to as Target threshold (Th_{T}), which indicates the target link load (e.g., 50%). More precisely, this second threshold is used in combination with the current link load to define the amount of traffic that should be rerouted over alternative paths when congestion happens. We point out that if routers only have local information, Th_{T} is set to a static value, while if they also have remote information on link loads and per-destination traffic, this threshold can be dynamically set to maximize the distance from Th_{A} for each link on the alternative paths. Link load statistics are updated according to a given frequency that depends on the hardware implementation, meaning that the CM mechanism may not be activated immediately when the link load exceeds Th_{A} and the amount of traffic to be steered away can be larger than $Th_{\mathrm{A}} - Th_{\mathrm{T}}$. A third threshold, referred to as Deactivation (Th_{D}), is defined to determine when the CM mechanism must be switched off, i.e. when the traffic on the link becomes sufficiently low to avoid any further congestions. In case a significant amount of traffic is routed over alternative paths, this threshold needs to be set to make sure traffic stays below Th_{T} after congestion mitigation is deactivated. In a more advanced implementation, the deactivation threshold could be set dynamically, but we leave this improvement for future work and consider a low threshold, as the portion of rerouted traffic is expected to be much lower than that on the original path.

C. System Considerations

In this paper, we focus on a totally distributed CM mechanism, as we want to provide a solution that is agnostic to the presence of a PCE for reliability and scalability concerns. We also point out that the CM mechanism can either compute alternative paths towards each destination PE or only towards the other endpoint of the congested link. In this paper, we will focus only on the first case.

Basic requirements for alternative paths are as follows: (i) they should allow reaching all destination nodes (towards which the offloaded traffic is directed), (ii) they should avoid any failed link, (iii) they cannot use forbidden nodes and links (such as PE nodes), and (iv) they should bypass the congested link. On top of that, alternative paths should be designed and utilized to solve the original congestion problem and avoid creating any additional congestions elsewhere in the network.

In order to ensure a fast response to congestion events, alternative paths need to be pre-computed. Therefore, as the actual link loads at congestion time cannot be known a priori, we propose to consider, as path computation metric, the link costs, which are typically designed to be inversely proportional to the capacity, to capture the willingness of the operator to attract traffic over large capacity links. In this way, it is possible to prioritize the use of alternative paths with large capacity and minimize the use of links that are prone to be quickly congested. More complex alternative paths computation procedures can be developed to account for future evolutions of traffic or failure scenarios at the cost of some complexity. In this case, a centralized controller may assist routers to pre-compute paths. In the rest of this paper, we stick to fully distributed solutions.

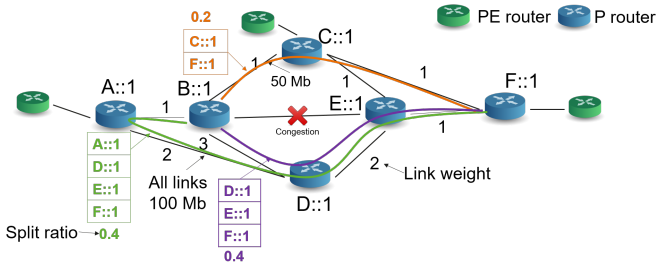


Figure 3: An example of the CM mechanism with 3 alternative paths towards the PE node connected to F::1.

On top of path computation, the UCMP weights assigned to each set of alternative paths for every destination need to be decided at congestion time, depending on the available information. Ideally, the knowledge about all link loads in the network and about the per-destination traffic demand over the congested link is required to take an optimal decision. In backbone networks, LSA messages may be configured to collect remote link loads, but it is not a desirable solution in practice. Typically, the rate, at which LSAs are issued, is kept very low to avoid congestions to happen in the control plane and to process topology changes in case of failures more quickly. Two main alternatives can be used by routers to collect link load statistics: (i) via centralized approaches, such as BGP with Link State (BGP-LS) [23], and (ii) via distributed telemetry methods for path tracing, such as SR-PT [24] or iFit [25]. In the remaining of the paper, we will assume, without loss of generality, that BGP-LS can be deployed to collect and obtain link load statistics, so each node can immediately retrieve the full network view. However, in one specific simulation scenario, we will leverage the broadcast of LSA messages to highlight the need for consistent operations between monitoring and congestion mitigation. However, for the sake of completeness, we will also consider the case where remote link loads cannot be obtained.

D. Congestion Mitigation (CM) Mechanism

Fig. 3 presents an example to highlight how CM operates inside routers. As soon as the link B::1 - E::1 becomes congested, i.e., its load exceeds Th_{A}, part of the traffic is rerouted over the three alternative paths. For each P node, up to K alternative paths are pre-computed for each destination PE node. In the example in Fig. 3, $K = 3$ alternative paths towards the node E::1 are depicted in green, purple, and orange.

Forwarding operations. The CM mechanism steers traffic over alternative paths which are encoded using Segment Routing IDs (SIDs). This mechanism is applied only to Best Effort (BE) traffic. Traffic cannot be rerouted more than once when multiple routers activate CM as traffic is recognized as "engineered" after a first CM is applied. To recognize TE traffic, we can classify packets based on whether SIDs from P nodes or from their adjacent links are used. Simpler solutions based on CoS or flow label fields can also be used.

When the CM mechanism is not active, the packet forwarding is based on standard RIB/FIB (Routing Information Base/Forwarding Information Base) mechanism. As shown in Fig. 4, on top of the standard entries populated by protocols such as OSPF, the RIB, which stores control plane information, also contains the alternative paths, the weights, and the SID lists as pre-computed by the devices. When a congestion is detected, the forwarding information for the alternative paths is written down into the FIB. As the activation of alternative paths may impact TCP flows, due to jitter and re-ordering issues, additional constraints on the end-to-end delay or length of alternative paths should be considered. In the rest of the paper, we consider a maximum elongation in number of hops.

For each incoming packet to be forwarded over a congested link, the FIB table is queried. If the packet must not be rerouted (e.g., TE or control plane traffic), it follows the standard forwarding (e.g., packets to F::1 continue to be sent via C::1). Otherwise, the FIB indicates that a group table must be used for UCMP. As presented in Fig. 4, the FIB states to apply the group table 1 for the destination E::1 and each entry for the destination E::1 is associated with a hash value, computed over the standard 5 tuple of the IP header (i.e., source port, source address, destination port, destination address, and protocol). Once a specific entry of the group table has been selected, the packet is forwarded and the specific SID list is injected into the Segment Routing over IPv6 (SRv6) header of the packet.

UCMP strategies. While link capacities and IGP link costs are learned from LSA messages, the information about link load is not always be available, as discussed above. For this reason, we propose to pre-compute alternative paths by prioritizing paths with the largest capacity, and we study three approaches for the computation of UCMP weights (see Sec. IV for more details):

- Local: where only the load of adjacent links is used to compute split ratios.
- Optimal: where all link loads and per-destination traffic information is available. In this ideal target, which is hard to put in practice and considered as a benchmark, routers can get all data to perfectly mitigate congestions.
- Remote: where local and remote link loads are used to compute split ratios, but the per-destination traffic over the congested link is unknown. In this case, we propose a sampling method derived from the *Local* strategy to estimate it and accurately compute split ratios according to the *optimal* strategy.

When remote link loads are used, their freshness has an impact how accurate split ratios can be initialized and updated. If information is too old, it may lead to inconsistent decisions, while if the frequency update is too high, it may introduce overhead. We analyze the impact of monitoring later in the evaluation section.

We assume that the congestion threshold Th_{A} is the same for all routers and given by the management system. In case different per-link thresholds should be used, an additional mechanism to distribute them to routers should be developed so that UCMP split computation algorithms can take them into account. However, we believe that static thresholds are fine for a first implementation.

Handling multiple congestions. Multiple congestions can be locally observed at the same time. In our current solution, CM is applied in sequence over congested links. Based on

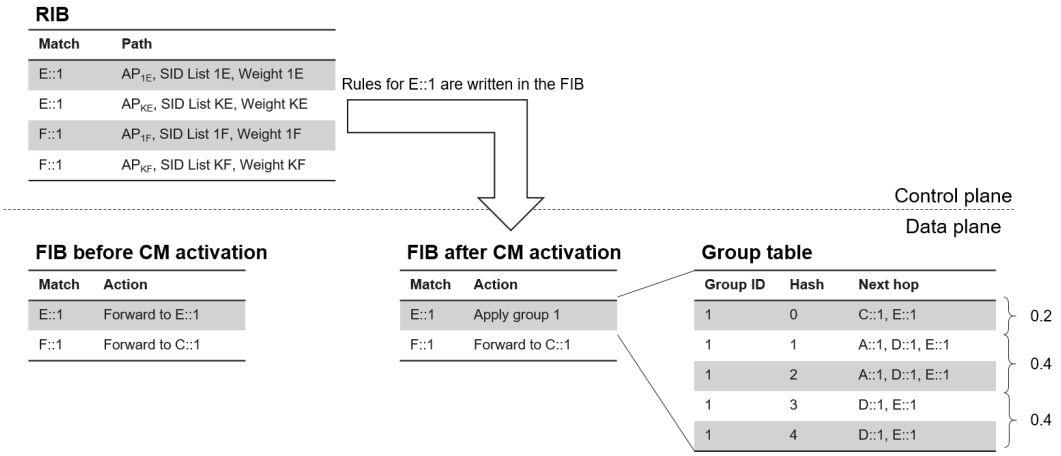


Figure 4: The CM mechanism implementation in RIB/FIB.

available link load, if CM expects that a new congestion will be induced, it simply steps back and does not activate. CM estimates the congestion it may induce based on the information it disposes. The Local strategy, where remote link loads are unknown, can only estimate locally induced congestions. A more advanced policy could be to solve multiple congestions at the same time. In addition, even when remote link loads are known, distant routers may experience a congestion at the same time and decide to offload traffic to alternative paths sharing the same bottleneck link. To mitigate synchronization effects, specific timers need to be designed. We let these extensions for future work.

In the next section, we will describe how to compute both alternative paths and UCMP weights for the load balancing of detoured traffic.

IV. Core Algorithms

The proposed CM solution consists, in a first phase, of alternative paths computation, which is carried out offline, and in a second phase, of UCMP weight adjustment, carried out online, i.e. when congestion happens. This section presents algorithms for the two steps.

For the computation of alternative paths, we consider two algorithms: a first one based on the K-Shortest Path (KSP) algorithm, considered as a benchmark, and a second one based on an extension of the Dijkstra algorithm to provide local diversity. For the assignment of UCMP weights, we present three approaches: local that considers only local link loads, optimal that provides optimal split ratios as it disposes of remote link loads and per-destination traffic, and remote, our practical proposal in which remote link loads are available to adapt split ratios in two steps: a first one to sample per-destination traffic and a second one to compute the best split ratios according to the input data from the first step.

In the remaining of this section, we will focus on a congested link (u, v) and we will denote by D the set of destinations that are reached by the node u on the "original" path traversing the congested link. We also denote the "original" path as the OSPF path used before congestion.

A. Path Computation

The alternative paths' computation problem consists in finding at most K paths from node u to each destination node $d \in D$. Remark that the number of pre-computed paths for a router is equal to $K \times |V|$, where V is the number of nodes. As OSPF is used for BE traffic, only one egress link of u is used before congestion to reach a given destination. The following requirements must be considered for alternative paths: (i) they must avoid the congested link as well as any other critical / failed resources, (ii) they must provide a high minimum remaining capacity, and (iii) they should have a length similar to the one of the original path. This latter constraint, referred to as max hop elongation, is introduced to ensure that the alternative paths have similar characteristics compared to the original path. We point out that we refer to the minimum capacity of a path as the smallest capacity over all the links of this path. As alternative paths are pre-computed and load information is not available, we consider the capacity instead of the remaining capacity in the rest of this paper. Also, as we must provide a solution in a short amount of time, we first neglect any constraint on path disjointness and we let this aspect for future improvements.

We consider two different algorithms for the alternative paths' computation problem. The first one, used as baseline algorithm, is an adaptation of KSP algorithm [26]. The second algorithm, referred to as Neighbor Deviation Algorithm (NDA), is an extension of the Dijkstra algorithm to enforce locally-disjoint paths (over the egress links of u), in order to improve the local diversity. In addition to improving local diversity over KSP, NDA is also much faster as its polynomial complexity is even better than Dijkstra algorithm.

Let us consider the following inputs:

- K: the maximum number of alternative paths,
- $G = (V, A)$: a network graph, where V is the set of nodes and A the set of links,
- $c_a \in \mathbb{R}^+$: the capacity of the link $a \in A$,
- d: a considered destination in the set D,
- (u, v): the congested link,
- p_d: the original path from u to d,

- $\tilde{P}_d$: the set of alternative paths from u to d.
- $P = \{p_d : d \in D\}$ the set of "original" paths from u.
- md: the max hop elongation between the path p_d and any alternative path.
- mh: the max hop of alternative paths.

For both algorithms, in order to prioritize links with large capacity, we consider IGP costs, which are normally used to let BE traffic follow the preferred (shortest) path in the network. They are typically set to the inverse of link capacity, i.e., $10^8/c_a$ for each link $a \in A$ [27] or they follow a step function [28].

1) KSP-based algorithm: The KSP alternative paths' computation algorithm consists, for every destination node $d \in D$, in generating K paths from u to d, using a modified version of Yen's algorithm [26], after removing the congested link from the graph. Yen's algorithm computes shortest paths successively by forbidding some links or by merging paths. To ensure that the maximum hop elongation constraint is satisfied, we replaced, in Yen's algorithm, the subroutine for computing the shortest path with a constrained shortest path algorithm. This latter is based on a bounded-depth dynamic programming algorithm. The complexity of the maximum hop constraint shortest path algorithm is in $O(mh|V| \max_{v \in V} |N_v|)$. Fig. 5a shows an example of $K = 3$ paths provided by the KSP algorithm. Each path is given by a color.

2) Neighbor Deviation Algorithm (NDA): Let us consider the set of neighbors of u denoted N_u. The algorithm consists in solving a Dijkstra algorithm from the destination d in the reverse graph to reach each node of $N_u \setminus \{v\}$, where u is filtered out. This allows generating one path per egress link of u, different from (u, v), to the destination d.

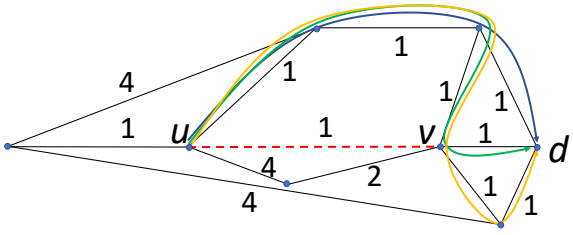
Moreover, the Dijkstra algorithm has been modified to take into account the maximum hop elongation constraint. The depth of the Dijkstra tree is kept less than or equal to $|p_d| + md - 1$. We use the constrained shortest path algorithm presented previously to solve the maximum hop constraint shortest path problem.

If the number of generated paths is higher than K, i.e. because of the number of neighbors, we select the best K paths (those maximizing the smallest link capacity in the path). On the other hand, if the number of paths is smaller than K, we keep them as it is. Fig. 5b shows an example with $K = 3$ paths provided by the NDA algorithm.

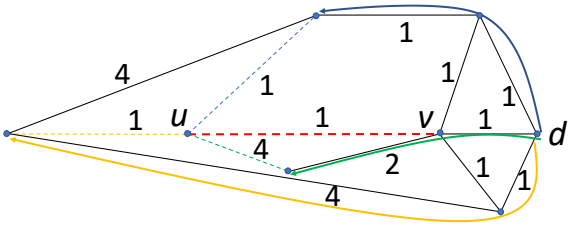
B. Split Computation for UCMP

We now provide three methods for determining how much traffic to reroute over each alternative path. We point out that the three presented strategies apply no matter the path computation strategy used for generating the alternative paths.

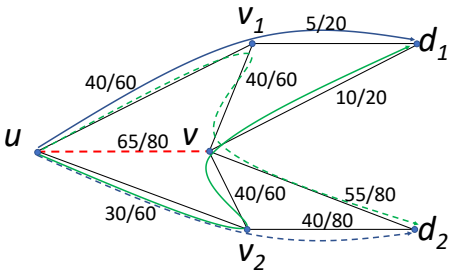
Motivating example. In Fig. (5c), we show two alternative paths, associated with the congested link (u, v), from u to two different destinations d_1 and d_2. Based on this example, we analyze the impact of the amount of information available in router u on congestion mitigation. The values on links represent the link load over the link capacity. The original path from u to d_1 crosses v with associated traffic of 10 and the original path from u to d_2 crosses v with associated traffic of 55. We consider $Th_{\mathrm{A}} = 70\%$ and $Th_{\mathrm{T}} = 50\%$, as congestion and target thresholds, respectively.



(a) Three paths given by the KSP algorithm to reach d (blue, green, yellow) from u. Link labels indicate the link weights. Congested link is in red,



(b) The three paths given by the NDA algorithm to reach d (blue, green, yellow) from u. Link labels indicate the link weights. Congested link is in red,



(c) Example of four alternative paths to reach two different destinations d_1 and d_2 from u. Congested link is in red, The first link label indicates the link load before the congestion mitigation and the second one the link capacity. The alternative paths (1 blue and 1 green) to d_1 are depicted with solid lines, while those for d_2 are depicted with dashed lines.

Figure 5: Examples on (a) alternative paths computed by KSP, (b) alternative paths computed for NDA, and (c) alternative paths for 2 different destinations considering different information available at node u.

We consider three levels of available information (in all cases we also have information about link capacities):

a) *Local link loads:* while local link loads are available to detect congestions, only link capacities and the load of the congestion link are useful to derive split ratios. In the example of Fig. (5c), we can remark that the two alternative paths, in green and blue, to reach d_1 (resp. d_2) have the same minimum capacity 20 (resp. 60). We remark that this value corresponds to the minimum capacity over all the links of a path. For instance, both alternative paths with solid lines, i.e., those to d_1, have a minimum capacity of 20, while the two depicted with dashed lines, i.e. to d_2, have a minimum capacity of 60.

The best solution is to send the same quantity of traffic on each alternative path for each destination. Clearly, without information about remote link loads, it can induce new congested links. To reduce the congestion of the link (u, v) to 50% of its capacity (i.e. to 40), CM needs to reroute $25/65 = 38.46\%$ of the traffic of (u, v). Thus, 3.85 (resp. 21.15) units of traffic for destination d_1 (resp. d_2) with a 50/50 split on alternative paths. This implies two new congestions on links (u, v_1) and (u, v_2) with, respectively, a load of $40 + 21.15/2 + 3.85/2 = 52.5$ for 60 of capacity, i.e., a link load of 87.5%, and $30 + 21.15/2 + 3.85/2 = 42.5$ out of 60, i.e., a link load of 70.8%.

b) *Local and remote link loads:* we consider that link loads ℓ_a of all links a in the original and alternative paths, i.e., $a \in A(P \cup \bigcup_{d \in D} P_d)$, are available. However, by considering only the link capacity and the link load of each link, we cannot avoid new congestion on alternative paths. In our example, if we replace the capacity with the remaining capacity to compute the split for alternative paths, we obtain the following splits: 15/25=60% (resp. 30/50=60%) for the blue path to d_1 (resp.d_2) and 10/25=40% (resp. 20/50=40%) for the green path to d_1 (resp. d_2). Thus, the link (u, v_1) becomes congested with $40 + 21.15 \times 0.40 + 3.85 \times 0.6 = 50.7$ of bandwidth for 60 of capacity, that represents 84.5% of utilization. To take a better decision we need to know the quantity of traffic sent towards each destination to compute the right quantity of traffic to offload for each destination.

c) *Traffic per destination:* we now consider that the amount of traffic sent over the original (congested) path bw_d towards each destination $d \in D$ is also available. If we know link capacities, link loads and the quantity of traffic per destination, then we can find the optimal solution that minimizes the worst link load on the network. By solving the "optimal" split ratio, we can get the following:
   - for destination d_1, 83.5% stays on the "original" path and 16.5% units of traffic is rerouted on the blue path.
   - for destination d_2, 82% stays on the "original" path and 18% of traffic is rerouted on the blue path.

   This solution induces a maximum link utilization of 69% and, therefore, no link is congested, i.e. above Th_{A}.

Algorithm solutions. The split ratio computation problem consists in finding, for each destination $d \in D$, the ratios for all alternative paths. The total split ratios over all alternative paths and original must be equal to 100%. The primary objective is to decrease congested traffic as close as possible to Th_{T} while avoiding creating new congestions, i.e. that the load of remote links do not increase above Th_{A}. We now present three approaches to tackle the split ratio computation.

The first approach we describe here is called *Local*. The splits are computed based on link capacities and the quantity of traffic to offload from the congested link when the congestion happens. The second approach, called *Optimal*, requires perfect link load knowledge over the network, as well as the "traffic per destination" that flows through the congested link. The last approach, referred to as *Remote*, combines the local mechanism, to estimate per-destination traffic, and the optimal approach This method allows providing a good-quality solution when remote link loads are also available.

1) Local: we first consider that we only know the capacity of links and the load of local links, in particular the load of the congested link, denoted as $\ell_{(u,v)}$.

The computation of split ratios is the combination of two levels. The first level, which is static and depends only on link capacities, is to derive the split ratios SR_d^p over alternative paths $p \in P_d$ to reach a destination $d \in D$ as follows:

$$SR_d^p = \frac{\min_{a \in p} c_a}{\sum_{p' \in P_d} \min_{a \in p'} c_a}$$

The second level, computed on-the-fly when the congestion is detected, consists in determining the fraction of traffic, referred to as RT (Removed Traffic), to be offloaded from the link.

$$RT = \frac{\ell_{(u,v)} - Th_{\mathrm{T}}}{\ell_{(u,v)}}$$

We can then derive the split ratio for each alternative path p as $SR_d^p.RT$.

2) Optimal: we propose a strategy for the UCMP weight computation to optimize split ratios for all destinations.

Let us define the following variables

- x_d: is the quantity of traffic sent to d on the original path.
- $y_{p'}^{a,d}$: is the quantity of traffic sent on the alternative path p' when the link a is congested to reach the destination d, for all $d \in D$, $p' \in P_d$.
- z: fraction of the available capacity over all links that remains unused after CM.

Let us define the maximum available capacity ac_a on a link $a \in A$.

$$ac_a = \max(c_a.Th_{\mathrm{A}} - \bar{\ell}_a, \ell_a - \bar{\ell}_a)$$

where $\bar{\ell}_a$ is the quantity of traffic on the link a not associated with the paths P and $P_d, d \in D$ (e.g., TE or monitoring traffic). The first term of the maximum function ensures that the link will not become congested by our method, whereas the second term allows considering remote congested links.

As the traffic on links of alternative paths varies over time, we introduce the variable z to maximize the minimum normalized distance, in terms of load, from Th_{A} on all links in alternative paths, as well as those in the original path. This allows us to ensure fairness between links to manage future traffic fluctuation.

The following linear program solves the split ratio computation optimally.

$$\max \ z$$

$$x_d + \sum_{p' \in P_{(u,v),d}} y_{p'}^{(u,v),d} = bw_d \qquad \forall d \in D \tag{1}$$

$$\sum_{d \in D} x_d + \sum_{d \in D} \sum_{p' \in P_d : a' \in p'} y_{p'}^{(u,v),d} + z.ac_{a'} \leq ac_{a'} \ \ \forall a' \in A \tag{2}$$

$$x_d \geq 0 \qquad \forall d \in D$$

$$y_{p'}^{(u,v),d} \geq 0 \qquad \forall d \in D, \forall p' \in P_d$$

where the objective function is similar to the one for the minimization of the maximum link utilization. Inequalities (1) ensure that the quantity of traffic for each destination is respected. Constraints (2) are the capacity constraints.

3) Remote: for this algorithm, we consider that we know the utilization of links belonging to the original path and the alternative paths. Let us introduce $\bar{A} \subset A$ the set of links covered by at least one alternative path or the original path.

The first step consists in estimating the traffic sent to each destination of D over the congested link, as this information is not available. Let ℓ_a^b be the link load on each link $a \in \bar{A}$ before congestion is detected. We first apply the *Local* strategy to offload a small amount of traffic, denoted as RT, from the congested link (e.g., RT =5%). Then, after a short amount of time, we observe the new link loads ℓ_a^c induced by our congestion mitigation for each link $a \in \bar{A}$. Thanks to the variation of the link loads (i.e. ℓ_a^b and ℓ_a^c values), the removed traffic RT, and the split ratio applied for the *Local* strategy, we can estimate the quantity of traffic sent to each destination, referred to as *local traffic matrix*, by solving the following linear programming model.

Inputs:

- ℓ_a^b: the link load of a before the CM.
- ℓ_a^c: the link load of a induced by CM (*Local* strategy).
- sr_p: the split ratio given by the *Local* strategy for path p (sum of sr_p on alternative paths and the original path for one destination is equal to 1).

Variables:

- x_d: estimated quantity of traffic sent to d before CM over the original path.
- xo_d: estimated quantity of traffic sent to d on the original path after CM is applied (*Local* strategy).
- z_a^p: the positive slack variables for each link $a \in \bar{A}$.
- z_a^m: the negative slack variables for each link $a \in \bar{A}$.

$$\min \quad \sum_{a \in A} (z_a^p + z_a^m)$$

$$xo_d + \sum_{p \in P_d} (x_d - xo_d) sr_p = x_d \qquad \forall d \in D \quad (3)$$

$$\sum_{d \in D} \sum_{p \in P_d : a \in p} (x_d - xo_d) sr_p - \sum_{d \in D} xo_d = \ell_a^c - \ell_a^b + z_a^p - z_a^m \qquad \forall a \in \bar{A} \quad (4)$$

$$xo_d \leq \ell_a^b - RT \quad (5)$$

$$\sum_{d \in D} \sum_{p \in P_d : a \in p} (x_d - xo_d) sr_p \leq RT \quad (6)$$

$$0 \leq x_d, 0 \leq xo_d \qquad \forall d \in D$$

$$0 \leq z_a^p, 0 \leq z_a^m \qquad \forall a \in \bar{A}$$

where inequalities (3) are the traffic conservation constraints, ensuring that the sum of the traffic sent on the original path and alternative paths corresponds to the traffic sent to the destination. Constraints (4) ensure that link load variations before and after the sampling phase, applying congestion mitigation with the Local strategy, are taken into account. We need to consider slack variables to ensure that we find a solution. Inequalities (5) and (6) are bounded constraints. As it tries to maximize the distance from Th_{A} of all the links on the original and alternative paths, the objective function minimizes the impact of the traffic fluctuation. Remark that this model optimizes the amount of traffic sent to each destination so that it corresponds to the observed link load variation on $\bar{A}$. However, it is possible that no local traffic matrix fits with the observed link load variation as ℓ_a^c values can account for fluctuations of background traffic and traffic from other congestion mitigations. For this reason, we consider the z variables that minimize the violation of the local traffic matrix generated by our model. Our goal is to find the most probable local traffic matrix that minimizes the violation with regards to the observed link load variation. Note that a possible drawback of this model is that equivalent solutions generally exist. Indeed, the problem is related to traffic matrix estimation [29] where an under-determined inverse problem must be solved. To obtain a unique solution, routing information and some assumptions about the traffic are typically used to constrain the problem (e.g., gravity model [30]). In the current implementation, we arbitrarily select a solution. Similarly to traffic estimation, we expect that a better estimation can be realized if additional information (e.g., traffic models) can constraint the solution. To solve this linear program, we used CPLEX 12.6.3 as for the Optimal scenario. With this estimation, we can run the *Optimal* algorithm. The whole process is repeated periodically.

V. Performance Evaluation

This section presents numerical results for the evaluation of our CM solution.

A. Packet Based Simulator

To test the behavior of the different congestion mitigation approaches, we have developed a packet based simulator on top of the Python Simpy Discrete Event Simulator (DES) [31]. We leveraged a dedicated simulator to implement only the relevant SR and IGP features for the evaluation of our solution. In this way, we managed to scale out our simulation framework to tens of nodes and hundreds of links, while evaluating the network dynamics in terms of link utilization, queue latency, and packet losses.

We developed three main modules: (i) an SR-enabled source, (ii) an SR-enabled sink, and (iii) an SR-enabled router. We point out that SR sources and sinks correspond to PE nodes, while SR routers correspond to P nodes. As PE nodes are not considered in our network, we directly connect them to the SR routers as sources/sinks of traffic. We considered that the outgoing buffer of each port can be modeled as a FIFO queue. In order to avoid any synchronization effect on the results, the starting time of each node is randomly chosen.

The source module is in charge of generating packets with a given rate and according to a given traffic distribution (e.g., constant rate). It inserts into packet headers either the SID list (for TE traffic) or the destination node (for BE traffic). A timestamp and a sequence number is inserted for the sake of statistics collection.

Link failures are triggered in simulations, and instantaneous IGP convergence is considered to re-route traffic on post-

convergence paths. We made this assumption as the convergence time of the IGP is between hundreds of milliseconds and a few minutes for large scale network with more than 100 nodes [32]. As the traffic variations at the scale of a link are slowlier, and the statistic collection process periodically checks each interface and updates the link utilization statistics by averaging the sent data over a given time window, the detection of a congestion is not instantaneous and may take a few seconds. In addition, as the traffic is highly dynamic during the reconvergence phase, it may happen that the CM mechanism is activated only for short time intervals. For these reasons, we assume that, during the reconvergence phase of the IGP, no CM can be carried out, meaning that this phase can be neglected in simulation. In real implementations, we will need to set and configure specific timers to avoid CM during IGP reconfiguration. The SR router has three main functionalities: (i) forwarding packets, (ii) queuing and transmitting packets, and (iii) computing statistics. They are implemented as different processes that coexist inside a router. The forwarding process periodically checks if there is a packet to be processed, in this case, it inserts it into the egress interface queue determined by the FIB. This process is also responsible for forwarding packets over alternative paths if the CM mechanism has been activated. If the load exceeds the Th_{A}, it activates the CM mechanism. Finally, a dequeuing process is associated with each interface: if a packet is available, it sends it over the link.

In the simulations we consider either an ideal monitoring mechanism or a realistic monitoring mechanism based on Link State Advertisement (LSA). In the former case, each node is instantaneously aware of the load of other links. In the latter case, instead, each node periodically broadcasts an LSA message containing the link loads of each interface. The other nodes, which keep a local representation of the network, update the local statistics with the received information and broadcast the message over the other interfaces. The nodes are fully asynchronous and their starting time is randomly tossed, meaning that the generation and the propagation of the LSAs does not happen at the same time in the network, introducing an offset between the statistics broadcasted in the network.

We also considered a perfect load balancing where all packets towards a destination PE are randomly assigned to alternative paths according to weights. In a real implementation, more advanced solutions should be applied to deal with imbalance issues or packet disordering. However, our goal is primarily evaluate the overall benefit of the solution.

B. Considered Scenarios

We consider two networks: a real backbone network from a regional operator in China and a topology from SND Lib [33]. The chinese regional operator network is composed of 30 core P routers interconnected by 348 links. The second network is germany50 from SND Lib and is composed of 50 P routers and 176 links. To keep simulation times reasonable, we considered that link capacities are distributed between 100Mb and 1Gb. For the sake of simplicity, we assume that we only have link aggregates, so there are no parallel links between two nodes (for instance, parallel links are aggregated and managed by a Link Aggregation Group). The PE nodes are filtered out from the graph to avoid using them for load balancing of traffic.

In our setting, all the P nodes can be associated with SR sources or sinks (i.e. with PE nodes). For each pair of nodes, we consider an Origin-Destination (OD) flow, but we neglect pairs that are too close (less than 2 hops on the shortest path). We consider a packet size of 1250 Bytes. The packet arrival distribution can either be constant or follow a Poisson distribution.

The alternative paths are computed offline and loaded into the routers at network startup. Up to $K = 5$ alternative paths can be computed for each pair congested link-destination. We consider $Th_{\mathrm{A}} = 70\%$ as congestion threshold activation, $Th_{\mathrm{T}} = 50\%$ as static target threshold (for *Local*), and $Th_{\mathrm{D}} = 40\%$ as congestion deactivation threshold. The statistic collection frequency is set to 250ms for each node. We point out that the traffic "engineered" on alternative paths is encoded as SR strict, i.e., as a SID list corresponding to the adjacent interfaces traversed by the packet, to avoid being rerouted at other nodes. During the forwarding operation, if the next SID corresponds to one of the outgoing interfaces, the congestion mitigation is not applied. In order to keep it into account in the computation of the splits, the bandwidth of the engineered flows is removed from the link capacity, so that split decisions are taken only the flows that can be rerouted.

For the Chinese operator, we consider four scenarios: (i) low load, where the rate of the OD pairs have been designed so that the MLU is 30% before failure and 80% after reconvergence, (ii) high load, where the MLU is 40% before failure and 80% after reconvergence, (iii) high load with LSA propagation, and (iv) extreme load, where the MLU is 40% before failure and above 100% after reconvergence. The packet arrival distribution of the first two scenarios is constant, while for the last two is Poisson. We consider a total of 136 different link failures that introduce at least a congestion in the network. Each simulation lasts for 10s (of simulated time). After 4s, we introduce a random failure of a link into the network. As we neglect the reconvergence time, the traffic immediately reconvergences to a new set of IGP shortest paths.

For the germany50 topology, we consider the following setting: before failure, the MLU is 40%. After 3s, a link fails and the network reconverges immediately. The MLU after reconvergence is 80%. At t=7.5s, a new traffic matrix with MLU=40% is loaded. For this scenario, we consider a simulation duration of 12s, a Poisson packet arrival distribution, and a congestion deactivation treshold $Th_{\mathrm{D}} = 0.4$.

We evaluate the network performance, in terms of MLU improvement, time to solve the congestion, link utilization impact, packet losses, and queue latency.

Throughout this section, some results are presented in the form of box plots that account for the points between the 1st (Q_1) and the 3rd quartile (Q_3), while in the bar in the middle of the box plot represents the median (Q_2). The whiskers represent $Q_1 - 1.5IQR$ and $Q_3 + 1.5IQR$, where $IQR = Q_3 - Q - 1$. The points represent the outliers.

C. Numerical results

We first provide an analytical comparison of alternative paths' computation algorithms in terms of path capacity, path cost, computational time, and number of paths found. For all our tests, we have used an Ubuntu 22.04 server with 104 Intel(R) Xeon(R) Platinum 8164 CPU @ 2.00GHz cores and 1.5Tb of RAM.

1) Experimental results on Split Computation for UCMP: For the numerical results we used CPLEX 12.6 [34] to solve the two linear programs presented in the algorithmic section to compute weights for UCMP. In the case of Optimal, the maximum computation time is 10.5 ms and 1.149 ms on average. To solve the linear program associated with the Remote solution, the maximum computation time is 2.6 ms and 1.195 ms on average. Solving the Optimal model (1)-(2) in a Huawei Router using a basic internal linear solver requires an average of 56 ms on the China topology. On average, the computational time slowdown between a Huawei Router with an internal solver and CPLEX on a powerful server is around 50 times.

2) Comparison between KSP and NDA: In this section, we compare the quality of alternative paths computed by KSP and NDA. We consider the following KPIs to compare the two algorithms:

- Total time (s): the total computational time for all congested links scenarios.
- #paths: the total number of paths for all congested links scenarios.
- Max #paths: the maximum of paths computed by each algorithm.
- Sum Path Cap: the total sum of the minimum capacity of each alternative path towards a destination in Mbps.
- Total Cost: the total cost of the alternative paths.
- Avg Path Cap: the average minimum capacity of alternative paths in Mbps.
- Avg Cost: the average cost of the alternative paths.

We compare NDA and KSP algorithms with max hops elongation $md \in \{2, 3\}$. Let us denote by NDA (X) and KSP (X) the algorithms with a max hops elongation $md = X$.

Tables I and II show the value of each KPI on China and germany50 topologies, respectively. For each line and setting, we put in bold the best value. Note that the two algorithms find the same alternative paths for germany50 when $md = 2$. For all other instances, the KSP algorithm generates more paths than the NDA algorithm, less than 1% when $md = 2$ and less than 14% when $md = 3$. For $md = 2$, the two algorithms cannot find more than 3 different alternative paths. This is due to the too-tight md value. When $md = 3$ then the two algorithms diverge more in terms of the number of alternative paths found. KSP algorithm finds a better value for the Sum Path Cap criterion since it generates more paths. In comparison, the average path capacity is better for the NDA algorithm. For the cost, it is not always the case. KSP finds a slightly better average cost for the $md = 2$ on the China topology. Otherwise, NDA provides better cost than KSP because this latter can find more paths which are, however, less interesting in terms of cost and capacity. For the computational time, NDA obtains better performances in comparison with the KSP algorithm and the computational time gain is between 3 and 5 times. The experiments have been done on a powerful server. However, on a router, the algorithms will need more time to find alternative paths. For the simulation, we consider the setting $md = 2$ which is typically recommended by traffic engineering experts. As the performances of NDA and KSP are comparable for $md = 2$ for cost and capacity, we will consider NDA in the simulation section.

	NDA (2)	KSP (2)	NDA (3)	KSP (3)
Total time (s)	**0.016488**	0.050005	**0.022320**	0.104395
#paths	1368	**1380**	3232	**3718**
Max #paths	**3**	**3**	4	**5**
Sum Path Cap	34500000	**34800000**	140800000	**149600000**
Total Cost	**956000**	964000	**3141900**	3829100
Avg Path Cap	**25219.2**	25217.3	**43564.3**	40236.6
Avg Cost	698.8	**698.5**	**972.1**	1029.8

Table I: Performance comparison between KSP and NDA algorithms for China instance. NDA (X) and KSP (X), X represents the max hops elongation $md = X$.

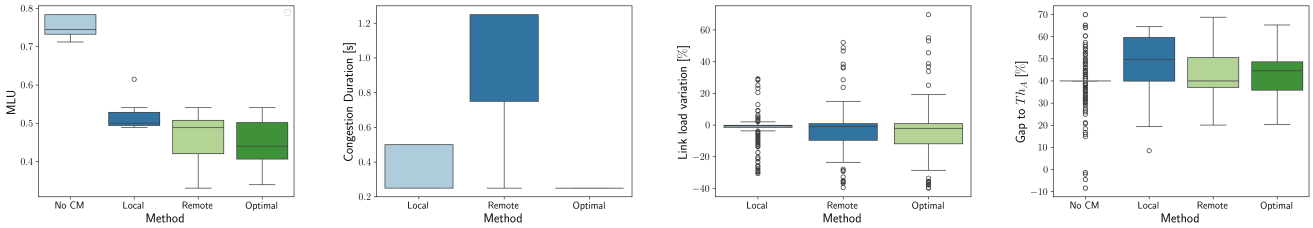
	NDA (2)	KSP (2)	NDA (3)	KSP (3)
Total time (s)	**0.048506**	0.135189	**0.064532**	0.245721
#paths	**1924**	**1924**	4729	**5731**
Max #paths	**3**	**3**	4	**5**
Sum Path Cap	**1298010**	**1298010**	3638030	**4198140**
Total Cost	**4107.3**	**4107.3**	**14308**	18139.4
Avg Path Cap	**674.6**	**674.6**	**769.3**	732.5
Avg Cost	**2.1**	**2.1**	**3**	3.1

Table II: Performance comparison between KSP and NDA algorithms for germany50 instance.

Remark that if alternative path computations can be done in a centralized way, more advanced algorithms, such as the one described in [35], could be used to provide a better capacity and cost. These algorithms need more resources and cannot be run in each router.

3) Packet simulations: In this subsection, we present simulation results in terms of MLU, congestion duration, gap of link utilization to Th_{A} after CM, and link load variations, considering different 1-link failure scenarios that induce congestion over some post-convergence paths. We assume that link load updates can be available every 250ms at each node.

a) Low load scenario: In Fig. 6, we present the results for the low load scenario, i.e., with MLU=30% before failure. We compare against the case without the congestion mitigation mechanism (No CM). The MLU is measured at the end of the simulation, i.e., after that the congestion mitigation has converged to a new stable MLU. Without congestion mitigation, we can observe that post-convergence MLU is indeed above $Th_{\mathrm{A}} = 70\%$. In terms of MLU reduction, both the Local and the Remote methods manage to solve all the congestions in the network. In particular, the Remote mechanism provides almost the same performance as the Optimal one. In Fig. 6b, we show the time required to mitigate the congestion. The Remote mechanism requires a longer time to reduce accurately the load of the congested link. Indeed, as explained in Sec. IV-B, the Remote mechanism needs to sample link loads, over a given time interval, in order to compute the final split ratios. Instead, as the Local method does not take into account for the per-destination traffic, it can apply directly the splits to reach Th_{T}, achieving a shorter congestion mitigation phase. In Fig. 6c, we show the distribution of link load variations



(a) Maximum Link Utilization. (b) Congestion duration. (c) Link load variations. (d) Gap to Th_{A} after CM.

Figure 6: Low load scenario: (a) MLU distribution with and without CM, (b) congestion mitigation duration, (c) link load variations after CM, and (d) gap from Th_{A} for each link after CM.

(in [%]), i.e. difference link loads after and before CM. We point out that a negative variation means that the traffic has been moved away from the link, while a positive one means that a part of the traffic has been rerouted on the link. We have filtered out the links where the variation is smaller than 0.1%. As the Local mechanism only focuses on the reduction of the congested link load to $Th_{\mathrm{T}} = 50\%$, only a few links change their load. However, in the Remote case, which targets the maximization of the gap from Th_{A} for all the links on the alternative paths and the original path, CM reduces the load on several links, at the price of a larger variation for a few of them, which receive much more traffic, allowing a better load balancing. The performance of the Remote mechanism is close to the Optimal one, which achieves a slightly better link load variation to improve the MLU. In Fig. 6d, we present the gap (in [%]) from Th_{A} after congestion mitigation for all the links in the network. In order to restrain the number of observed links, we have filtered out the links whose variation is smaller than 0.1% after CM. The Local mechanism, as it solves the congestion locally, keeps the gap from Th_{A} for almost all the links, while the Remote mechanism plots smaller gaps, as several links in alternative paths receive more traffic to balance the load among them. For the case without CM, we can observe that the majority of links lie around 40% gap to Th_{A} (as the MLU was 30% for the low load scenario before congestion). The points below 0 mean that a congestion is happening for the different instances.

b) High load scenario: In Fig. 7, we show the performance achieved for the high load scenario, i.e., when the MLU before link failure is 40%. In terms of MLU, as shown in Fig. 7, the median of the MLU is moved from 77% to 60%. In particular, while the MLU distribution lies below 70% for the Remote and the Optimal method, meaning that the congestion have been mitigated in all the instances, the Local mechanism fails in a few instances, remaining above 70%, as shown by the top whisker.

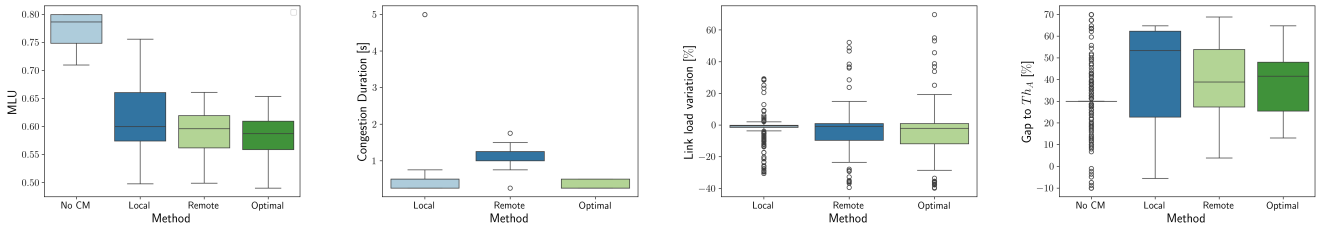
Due to the different nature of the split mechanisms, the average reduction of the Remote mechanism is larger than the Local one, and close to the Optimal one. This is because the Local split mechanism tries to reach $Th_{\mathrm{T}} = 50\%$ locally, while the Remote one tries to better distribute traffic over alternative paths' links.

In Fig. 7b, we show the duration of the congestion mitigation phase. By design, the Local mechanism solves the congestion quickly, due to its policy of targeting an MLU of 50%. Similarly, the Optimal strategy manages to solve the congestion quickly. However, the Remote mechanism takes a longer time to solve the congestion, as it first needs for 1s to sample traffic before applying the final CM strategy.

In Fig. 7c, we present the link load variations. As for the low load scenario, variation of link loads induced by the Local mechanism is smaller than the one of the Remote mechanism. In Fig. 7d, instead, we show the gap (difference) between link utilization and Th_{A} after CM. While the Remote mechanism has a behavior similar to the one experienced for the low load scenario, the Local mechanism has some value below 0, meaning that it fails to mitigate congestion on some instances. As for the low load scenario, the points below 0 for the case without CM means that a congestion is happening.

c) Impact of monitoring frequency: In Fig. 8, we show the impact of the monitoring mechansim on the CM decisions. We consider the same network load as for the high load scenario, a Poisson packet arrival distribution, and we compare two transmission frequencies for the LSA messages for monitoring: (a) 0.5s, i.e., below the duration of the sampling phase of the Remote mechanism, and (b) 4s, i.e., above the sampling duration. We point out that the Optimal method disposes of ideal statistics over the link load, while the Local method only has local statistics as, by construction, does not need for remote information. For the case with t=0.5s frequency, the Remote method well approaches the Optimal one, while for the case with t=4s, the Remote method fails to solve the congestion, as it takes decisions blindly. This highlights the importance of a statistic collection mechanism with a transmission frequency coherent with the sampling phase duration. More specifically, if no statistic update happens during the sampling phase, the split decisions are not reliable as taken on outdated statistics. Vice versa, the higher the frequency, the better the accuracy of decisions.

d) Extreme load scenario: In the extreme load scenario, we consider an MLU after failure that may exceed the link capacity. In this scenario, a Poisson packet arrival distribution is considered. In Fig. 9a, we show the packet loss performance of the three methods compared against the scenario without congestion mitigation. In the absence of the CM mechanism, losses may rise up to 4% of the traffic transmitted over the congested link. In the case of CM, instead, as traffic is rerouted over alternative paths, the packet loss remains almost 0, as



(a) Maximum Link Utilization. (b) Congestion duration. (c) Link load variations. (d) Gap to Th_{A} after CM.

Figure 7: High load scenario: (a) MLU distribution with and without CM, (b) congestion mitigation duration, (c) link load variations after CM, and (d) gap to Th_{A} for each link after CM .

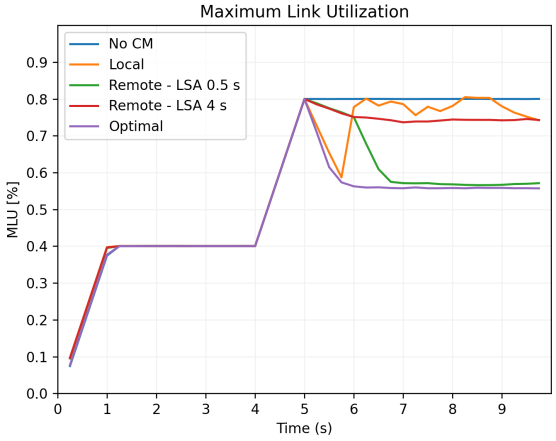
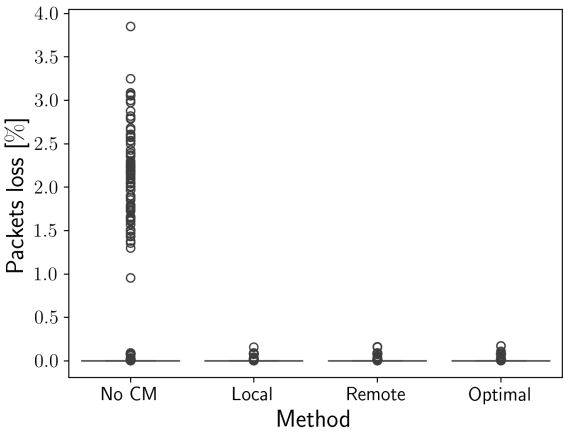


Figure 8: MLU evolution over time.

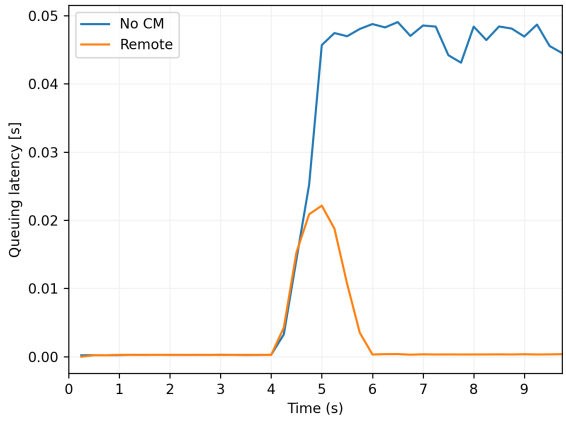
it only impacts a few packets when the congestion peak is experienced.

In Fig. 9b, we present the queuing delay evolution for the congested link without the CM mechanism and with the Remote CM mechanism. When the congestion happens at t=4s, the queuing delay without CM explodes up to 50 ms and then stabilizes as packet losses are starting to appear. Vice versa, the Remote mechanism manages to keep the queuing delay smaller, up to 20ms, as traffic is successfully rerouted over alternative paths.

e) Germany50 network: In Fig. 10, we show the performance on the germany50 instance. In terms of MLU, as shown in Fig. 10a and Fig. 10b, the three CM approaches manage to efficiently solve the congestion and lower the MLU below 70%., except for a few cases where the Local method fails to mitigate the congestion due to lack of information. In a few cases, the Optimal method fails as well as the decision of the split is taken only at the CM activation phase, while it should be updated periodically to keep track of traffic evolution. In Fig. 10c we show the link load evolution of the congested link for the cases without the CM and with the Remote method. While in the former case, the congestion remains till a new traffic matrix is installed at t=5.5, in the latter case the congestion is correctly mitigated, and when a new traffic matrix arrives, the CM mechanism is deactivated, as shown in Fig. 10d, which refers to the CM activation over the same link for the Remote method. When the load goes



(a) Packet loss.



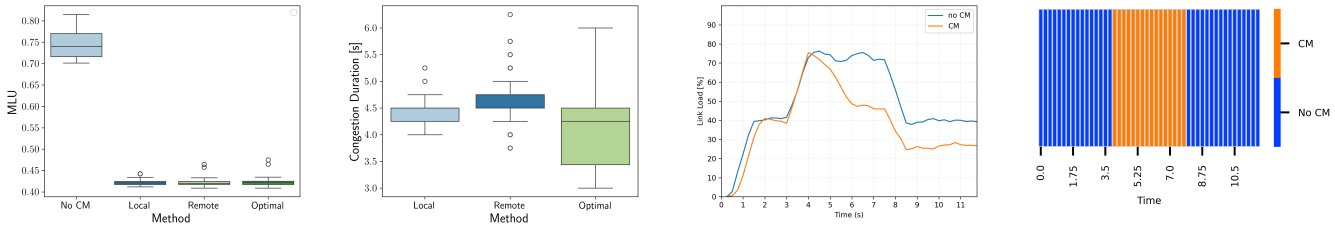
(b) Congested-link queue latency with the CM Remote mechanism and without CM.

Figure 9: Extreme load scenario: (a) packet loss and (b) congested-link queue latency behavior over time.

below $Th_{\mathrm{D}} = 0.4$, the CM is deactivated.

VI. Conclusion

We proposed a distributed tactical TE solution leveraging SR to efficiently mitigate congestions with quick and local reaction, without requiring any interaction with a centralized controller. When a router detects a congestion on a link, a portion of the traffic can be automatically offloaded and



(a) Maximum Link Utilization. (b) Congestion duration. (c) Congested link load. (d) CM activation.

Figure 10: Germany50 scenario: (a) MLU distribution with and without CM, (b) congestion mitigation duration, (c) congested link load variations after CM and after traffic matrix update, and (d) CM activation over time over the congested link for the Remote method.

load balanced over a set of alternative paths using UCMP. We presented the Neighbor Deviation Algorithm (NDA) for alternative paths' pre-computation and showed that it solves alternative path computation faster than KSP algorithm. We also proposed two approaches to compute split ratios on top of alternative paths, depending on whether remote link loads are available or not. We showed that a strategy based on link capacities and local link loads can already mitigate congestions in lightly loaded scenarios. Moreover, a strategy using remote link load information and traffic sampling to estimate per-destination traffic over the congested link can achieve near-optimal congestion mitigation in all scenarios. We also show how the congestion mitigation allows reducing the packet loss in the case of highly congested link as well as reducing the queuing latency. By testing the mechanism under a realistic setting, we highlight that an efficient monitoring mechanism is required to provide accurate congestion mitigation.

We believe that the work presented in this paper is a first step, and it paves the road for many additional contributions (system, algorithm, standardization) to provide a comprehensive SR-based Tactical TE solution for congestion mitigation. For future work, several extensions will be considered. To improve reliability and routing diversity, disjointness and fault tolerance can be considered for alternative paths' computation. As highlighted by the results, efficient telemetry solutions must be designed to provide efficient congestion mitigation and reduce the reaction time. Furthermore, if some predictions about the traffic can be made available, then we can consider them to provide more robust alternative paths and split ratios. To better coordinate distant routers reacting at the same time, IGP protocol extensions can be proposed. Also of interest, as some routers can measure the N most important flows over an interface, e.g. on congested links, alternative paths can be selectively activated (only for N destinations) or the per-destination traffic about top-N flows can be used to refine the estimation for all destinations.

References

[1] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," IEEE Communications Surveys Tutorials, vol. 10, no. 1, pp. 36–56, 2008.

[2] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, "Segment routing: a comprehensive survey of research activities, standardization efforts, and implementation results," IEEE Communications Surveys & Tutorials, vol. 23, no. 1, pp. 182–221, 2020.

[3] D. Wu and L. Cui, "A comprehensive survey on segment routing traffic engineering," Digital Communications and Networks, 2022.

[4] A. Destounis, S. Paris, L. Maggi, G. S. Paschos, and J. Leguay, "Minimum cost sdn routing with reconfiguration frequency constraints," IEEE/ACM Transactions on Networking, vol. 26, no. 4, 2018.

[5] K. Qiu, J. Zhao, X. Wang, X. Fu, and S. Secci, "Efficient recovery path computation for fast reroute in large-scale software-defined networks," IEEE Journal on Selected Areas in Communications, vol. 37, no. 8, pp. 1755–1768, 2019.

[6] "Local Congestion Mitigation (LCM) White Paper," https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/crosswork-network-automation/local-congestion-mitigation-wp.pdf.

[7] "Huawei iMaster NCE," https://e.huawei.com/es/products/network-analysis/imaster-nce-ip.

[8] "Cisco Crosswork Optimization Engine 4.1 User Guide - Local Congestion Mitigation (LCM)," https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/crosswork-optimization-engine/4-1/UserGuide/b_cisco-crosswork-optimization-engine-4_1_userguide/m_mitigate-congestion-locally.html.

[9] L. Roelens, O. G. d. Dios, I. d. Miguel, E. Echeverry, and R. J. D. Barroso, "Performance evaluation of ti-lfa in traffic-engineered segment routing-based networks," in 2023 19th International Conference on the Design of Reliable Communication Networks (DRCN), 2023, pp. 1–8.

[10] A. Bashandy, C. Filsfils, S. Litkowski, B. Decraene, P. Francois, and P. Psenak, "Loop avoidance using Segment Routing," Internet Engineering Task Force, Internet-Draft draft-bashandy-rtgwg-segment-routing-uloop-15, Jun. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-bashandy-rtgwg-segment-routing-uloop/15/

[11] S. Litkowski, A. Bashandy, C. Filsfils, P. Francois, B. Decraene, and D. Voyer, "Topology Independent Fast Reroute using Segment Routing," Internet Engineering Task Force, Internet-Draft draft-ietf-rtgwg-segment-routing-ti-lfa-12, Nov. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-rtgwg-segment-routing-ti-lfa/12/

[12] K.-T. Foerster, M. Parham, M. Chiesa, and S. Schmid, "Ti-mfa: keep calm and reroute segments fast," in IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2018.

[13] S. Hegde and P. Sarkar, "Micro-loop avoidance using SPRING," Internet Engineering Task Force, Internet-Draft draft-hegde-rtgwg-microloop-avoidance-using-spring-03, Jul. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-hegde-rtgwg-microloop-avoidance-using-spring/03/

[14] A. Brundiers, T. Schüller, and N. Aschenbruck, "Midpoint optimization for segment routing," in IEEE INFOCOM 2022-IEEE Conference on Computer Communications. IEEE, 2022, pp. 1579–1588.

[15] A. Brundiers, T. Schuller, and N. Aschenbruck, "Tactical traffic engineering with segment routing midpoint optimization," in IFIP Networking, 2023.

[16] P. Medagliani, J. Leguay, M. Abdullah, M. Leconte, and S. Paris, "Global optimization for hash-based splitting," in Proc. IEEE Global Communications Conference (GLOBECOM), 2016.

[17] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, "Wcmp: Weighted cost multipathing for improved fairness in data centers," in Proceedings of the Ninth European Conference on Computer Systems, 2014, pp. 1–14.

[18] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav et al., "Conga: Distributed congestion-aware load balancing for datacenters," in Proceedings of the ACM conference on SIGCOMM, 2014.

[19] C. H. Benet, A. J. Kassler, T. Benson, and G. Pongracz, "Mp-hula: Multipath transport aware load balancing using programmable data planes," in Proceedings of the 2018 Morning Workshop on In-Network Computing, 2018, pp. 7–13.
[20] Y. Desmouceaux, P. Pfister, J. Tollet, M. Townsley, and T. Clausen, "6lb: Scalable and application-aware load balancing with segment routing," IEEE/ACM Transactions on Networking, vol. 26, no. 2, pp. 819–834, 2018.
[21] G. Mirsky, G. Jun, H. Nydell, and R. F. Foote, "Simple Two-Way Active Measurement Protocol," RFC 8762, Mar. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8762
[22] D. Katz and D. Ward, "Bidirectional Forwarding Detection (BFD)," RFC 5880, Jun. 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc5880
[23] H. Gredler, J. Medved, S. Previdi, A. Farrel, and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP," RFC 7752, Mar. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7752
[24] C. Filsfils, A. Abdelsalam, P. Camarillo, M. Yufit, T. Graf, Y. Su, S. Matsushima, M. Valentine, and A. Dhamija, "Path Tracing in SRv6 networks," Internet Engineering Task Force, Internet-Draft draft-filsfils-spring-path-tracing-05, Oct. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-filsfils-spring-path-tracing/05/
[25] H. Song, F. Qin, H. Chen, J. Jin, and J. Shin, "Framework for In-situ Flow Information Telemetry," Internet Engineering Task Force, Internet-Draft draft-song-opsawg-ifit-framework-21, Oct. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-song-opsawg-ifit-framework/21/
[26] J. Y. Yen, "Finding the k shortest loopless paths in a network," management Science, vol. 17, no. 11, pp. 712–716, 1971.
[27] "Understand Open Shortest Path First (OSPF)," https://www.cisco.com/c/en/us/support/docs/ip/open-shortest-path-first-ospf/7039-1.html.
[28] "Huawei documentation: How Do I Calculate the Cost of an IGP Route?" https://support.huawei.com/enterprise/en/doc/EDOC1100112352/d33e78c5/how-do-i-calculate-the-cost-of-an-igp-route.
[29] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," Journal of the American statistical association, vol. 91, no. 433, pp. 365–377, 1996.
[30] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale IP traffic matrices from link loads," ACM SIGMETRICS Performance Evaluation Review, vol. 31, no. 1, pp. 206–217, 2003.
[31] "Simpy," https://simpy.readthedocs.io/en/latest/.
[32] N. Rybowski and O. Bonaventure, "Evaluating ospf convergence with ns-3 dce," in Proceedings of the 2022 Workshop on Ns-3, ser. WNS3 '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 120–126. [Online]. Available: https://doi.org/10.1145/3532577.3532597
[33] "SND Lib," https://sndlib.put.poznan.pl/home.action.
[34] IBM, "ILOG CPLEX Solver." [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer
[35] S. Martin, Y. Magnouche, P. Medagliani, and J. Leguay, "Alternative paths computation for congestion mitigation in segment-routing networks," in IEEE 10th International Conference on Control, Decision and Information Technologies (CoDIT) - accepted, arXiv:2404.19314., 2024.