In-Band Network Telemetry for Efficient Congestion Mitigation

Youcef Magnouche, Sébastien Martin, Jeremie Leguay, Paolo Medagliani
Huawei Technologies France, Paris Research Center, France.
firstname.lastname@huawei.com

ABSTRACT

Tactical traffic engineering solutions are a must to adapt traffic steering when unexpected congestions occur. While centralized solutions are already available to solve congestion issues, they can be too slow and not suitable for some deployment scenarios. To address this issue, a distributed congestion mitigation mechanism that leverages Segment Routing (SR) to offload traffic away from congested links over alternative paths has been proposed. However, to accurately re-route traffic while not inducing other congestions, it requires fresh information about link loads inside alternative paths. In this paper, we propose to rely on the computation of additional paths, called "monitoring paths", that can be used to collect link loads efficiently. We investigate the associated optimization problem to decrease the number of paths used for monitoring. Also, thanks to Bilevel optimization, we show that the load information of several links can be recovered without monitoring. Results show that the overhead can be drastically reduced.

KEYWORDS

OSPF, Shortest-Path, Monitoring, Optimization, Segment-Routing, Path-Tracing, Network Telemetry, Bilevel optimization.

1 INTRODUCTION

As IP networks are continuously increasing in traffic, scale and complexity, service providers must carefully plan and design their networks to anticipate network evolution, e.g. traffic or failure scenarios, and meet custom requirements, e.g. in terms of Quality of Service (QoS) or routing requirements. In order to keep the network management as simple as possible, most of the traffic is routed across the network following the shortest path given by the Open Shortest Path First protocol (OSPF) [8], a routing protocol that works by flooding Link State Advertisement (LSA) information throughout the network. This information includes the cost of each link, its capacity, as well as some performance metrics about the link utilization. Routers use these metrics to compute and update their routing strategies.

However, in the case of unexpected link failures, the original planning may no longer hold, as traffic may be redirected to the post-convergence paths, leading to excessive use of some links, i.e., to congestions. This calls for fast reaction mechanisms to mitigate these congestions in less than 50 ms. As the interaction with the centralized controller is not suitable, due to the slow communication with devices, it is preferable to take decisions locally at node level. In [1], the authors propose a distributed congestion mitigation (CM) mechanism using Segment Routing (SR) to load balance traffic from a congested link to different alternative paths, as soon as a router detects congestion over an outgoing interface. However, if the load balancing weights are not properly tuned, the rerouted traffic can introduce new congestion in other parts of the network. Therefore, routers need to get accurate loads of the links in each alternative path, to decide how much traffic to reroute over them.

A solution to get accurate information can be provided by in-network telemetry [18], which aims to collect data from devices at high speed and in real-time. In particular, In-band Network-wide Telemetry (INT), or In-situ Operations, Administration, and Maintenance (IOAM) [2, 12, 19] embeds the telemetry information in the header of user packets or probe packets [11] to perform end-to-end or hop-by-hop measurements. To collect link loads over alternative paths, we can use Path Tracing (SR-PT) [9, 14] or iFit [15], i.e., two candidate solutions to provide a record of end-to-end delay, per-hop delay, and load on each egress interface along the packet delivery path.

Unfortunately, the number of alternative paths may be very high as routers maintain k alternative paths per destination and outgoing link (potentially congested). As explained in [5], path-based measurements are prohibitive due to the massive number of existing paths inside a network. First, probing packets have a maximum size (e.g. 1500 B), and therefore, if the path to be monitored is too long, it is not possible to collect telemetry data within a single packet. Second, as paths often overlap, a significant amount of collected information is redundant. This calls for the design of a new mechanism to improve the efficiency of alternative paths monitoring so that a more accurate reaction can be taken, avoiding introducing cascade congestions in remote links in the network.

In this paper, we investigate the computation of a small additional subset of paths, referred to as *monitoring paths*. These paths, which are deployed by each node, are used to collect remote link load information over the links used by its alternative paths (i.e., for all the destinations). The number of monitoring paths must be (i) much smaller than the number of alternative paths, and (ii) allow monitoring of all links in the alternative paths. In addition, our solution allows using some alternative paths for monitoring. As they are already deployed inside the routing table of a node, they can be immediately used at congestion time for faster reaction. In order to further reduce the monitoring overhead, we also show that it is possible to skip the monitoring of some links, without any loss of information.

In this paper, we provide the following contribution:

- We introduce a new set of paths, referred to as *monitoring paths*, used for hop-by-hop measurements.
- We formulate the monitoring paths computation problem using an Integer Linear Programming (ILP) model.
- We show that the monitoring paths computation problem is NP-Hard.

- We exploit Bilevel optimization to design partial monitoring paths, still guaranteeing full measurement.
- We perform extensive computational experiments to evaluate the performance of our algorithm compared to a "Naive approach" that consists in monitoring all alternative paths. We show that our solution decreases, the average number of paths for monitoring by up to 68% and the average number of monitored links by up to 71%.

The paper is organized as follows. In Sec. 2, we introduce the state of the art, in Sec. 3, we detail the considered use case. In Sec. 4, we provide a mathematical model for the computation of the monitoring paths. In Sec. 5 we show how a partial monitoring is enough to recover the load of all links in the alternative paths. In Sec. 6, we show the efficiency of our algorithm. Finally, Sec. 7 concludes this paper.

2 RELATED WORK

Several works in the literature have proposed routing optimization to mitigate congestion. In very recent works, such as [3, 4], the authors suggest leveraging mid-point SR optimization to mitigate congestion in the network. This approach is based on a centralized controller to compute alternative SR policies for congestion mitigation, which may not be desirable for scalability, fault tolerance, or commercial reasons. In [1], authors proposed a distributed 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 congestion elsewhere in the network. Our paper provides a distributed solution to optimize the collection of link loads, in real-time, over alternative paths for this type of mechanism.

Several works in the literature have proposed solutions for the computation of monitoring paths computations. In [16], authors develop a heuristic called "Graph Partitioned INT" so that a centralized controller can organize path measurements to cover all the nodes in the network, guarantee the freshness of telemetry information, and minimize redundancy. In [13], authors developed an algorithm to generate, at the controller, non-overlapped INT paths that cover the entire network with a minimum path number. In [5], authors investigate the computations of probing cycles that collect telemetry information over time employing a MILP model and a mathematical-based heuristic.
Our paper proposes a distributed mechanism to organize the collection of link data from a given router. In addition, it leverages as many as possible alternative paths for a smooth transition when congestions happen and further reduces the overhead with the partial collection.

3 USE CASE: MONITORING FOR CONGESTION MITIGATION

Typical service provider networks are configured to support reliability (up to 1-link failure) and QoS satisfaction (low MLU). As shown in Fig. 1, the backbone network is composed of Provider (P) nodes, i.e., nodes belonging to the same Internet Service Provider (ISP). The backbone network receives traffic from different sites, which

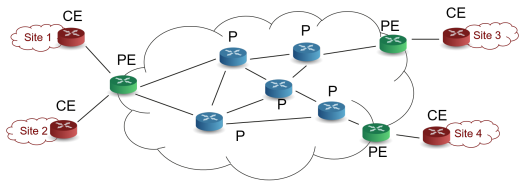

Figure 1: Example of a network with "CE" nodes attached to sites, "P" nodes in the ISP backbone, and "PE" nodes interconnecting "P" and "CE" nodes.

are interconnected to the service provider network via Customer Edge (CE) routers. The "PE" nodes interconnecting "P" and "CE" nodes are referred to as Provider Edge (PE) routers. In this backbone network, the "PE" nodes act as both sources and destinations of traffic aggregates. In order to keep the routing plane as simple as possible, flows follow the OSPF path, i.e., the shortest path between each pair of "PE" nodes. We point out that, in this paper, we only consider Segment Routing (SR) Best Effort (BE) traffic following the OSPF shortest path.

In the case of a link failure, the node detecting the failure broadcasts an LSA message to all the other links in the network to notify them of the network change. Each node independently computes a new OSPF shortest path tree to route the traffic avoiding the failed link. However, after that the network has reconverged, it may no longer guarantee low MLU and congestion, i.e. links whose load exceeds a given threshold, may appear. For this reason, it is necessary to implement efficient congestion mitigation mechanisms that locally react to congestions in less than 50ms, i.e. without requiring any interaction with an external controller.

A reference solution, that we will consider throughout this paper, is presented in [1], where the authors present a congestion mitigation mechanism that load balances traffic away from the link whose load is above 70%. The traffic is rerouted over k alternative paths that allow reaching the original destination. In this preliminary work, the authors only consider the link load of the congested interface. However, efficient congestion mitigation requires the knowledge of the load of all the links in the alternative paths, in order to avoid introducing remote congestions due to the unawareness of remote link loads.

As the reception of an LSA denotes that a network change has occurred, each node that receives an LSA message knows that congestion may happen. For this reason, after that it has updated its routing tables and converged to the new OSPF tree, it can deploy some extra paths, referred to as *monitoring* paths. As for SR-PT paths, the node sends probe packets through the monitoring paths to collect statistics about the load of all the links that belong to its alternative paths towards each destination. In this way, the node can collect all the information that it needs to make better load-balancing decisions in the case of congestion. In order to reduce the number of additional monitoring paths, some alternative paths can be chosen for monitoring purposes. As the failing link is not known a priori, the monitoring paths for each failure can be pre-computed offline, stored in the devices, and activated only when needed.

Once congestion mitigation is initiated, the node calculates the split ratio for each alternative path by considering both the local and the remote link loads. We neglect this computation as out of scope for this paper. In order to reroute traffic, an explicit Segment Routing

ID (SID) list is encapsulated in the header of the packets forwarded over alternative paths. This traffic, which is called ”engineered”, is no longer following the OSPF shortest path.

As soon as the congestion issue is resolved (i.e., the utilization of the egress link falls below a given threshold, i.e. 30%), the node stops the mitigation process and uninstalls the monitoring paths.

4 MONITORING PATHS COMPUTATION

The network can be modelled as a graph $G = (V, A)$, where V is the set of nodes and A is the set of arcs (links). Nodes of $R \subset V$ represent the set of "PE" nodes that generate the traffic in the backbone network, as they receive traffic from the "CE" nodes. The graph of "P" nodes, i.e., restricted to nodes of $V \setminus R$, corresponds to the *backbone network*.

The traffic, between two nodes $u, v \in V$ in the network, follows the shortest path with respect to OSPF weights. This path is called, *OSPF path*, denoted by p_u^v. Let S be the set of all shortest paths between every pair of "PE" nodes in R. Let $\bar{A}$ be the set of links belonging to OSPF paths, i.e., $\bar{A} = \bigcup_{u,v \in V} p_u^v$. In the following, we denote by $\delta^+(v) \subseteq A$ (resp. $\delta^-(v)$) the outgoing (resp. ingress) links of $v \in V$. Let $\delta(v) = \delta^+(v) \cup \delta^-(v)$. In this paper, we investigate the monitoring paths computation problem from the point of view of one arbitrary node $u^* \in V$. For a "PE" node $r \in R \setminus \{u^*\}$ and every arc $\bar{a} \in \delta(u^*)$, let $P_{\bar{a}}^r$ be the set of $k \in \mathbb{N}$ alternative paths between u^* and r avoiding $\bar{a}$. These paths are designed by the network operator (not necessarily shortest-paths) to reroute traffic in case of congestion. Let $P^r = \bigcup_{\bar{a} \in \delta(u^*)} P_{\bar{a}}^r$ be the set of all alternative paths to destination r and $A' = \bigcup_{r \in R} \bigcup_{p \in P^r} p$ be the set of all links in the alternative paths.

4.1 Problem definition

The monitoring paths computation problem consists in computing at most $q \in \mathbb{N}$ paths between u^* and "PE" nodes R such that:

- each link in the alternative paths, nonadjacent to u^* (links of $A' \setminus \delta(u^*)$), is monitored,
- the length (number of hops) of each monitoring path is at most $L_{\max} \in \mathbb{N}$,

under the following multi-objective function:

1) minimize the number of monitoring paths,
2) maximize the number of alternative paths used for monitoring,
3) minimize the total monitoring paths cost. The link costs may be used to prioritize some links for monitoring or to minimize the number of hops in the paths.

We consider a weighted sum of objective functions, by assigning a weight $w^1 \in \mathbb{R}^+$ to objective 1), $w_p^r \in \mathbb{R}^+$ to objective 2) and w_a in objective 3) for every link $a \in A$. Let $Q = \{1, \ldots, q\}$. Note that "Paths used for monitoring" represent the union of monitoring paths and a subset of alternative paths used for monitoring.

4.2 Complexity

Theorem 4.2.1. *The monitoring paths computation problem is NP-hard.*

Proof. We propose a polynomial reduction from the Hamiltonian path problem, known to be NP-complete [10], that consists, given a graph $G = (V, A)$, in computing a path, between a source s and a destination t, in G crossing all vertices in $V \setminus \{s, t\}$ exactly once. We construct a graph $G' = (V', A')$ from G, as follows:

- replace each node $v \in V \setminus \{s\}$ by two nodes v' and v'' connected by a link (v', v'') of weight 0,
- for each link $(u, v) \in A$, such that $u \in V \setminus \{s\}$ and $v \in V \setminus \{s\}$ add a link (u'', v') with $w_{(u'',v')} = 1$,
- for each $v \in V \setminus \{s\}$ add a link (s, v') of weight $w_{(s,v')} = |A| + |V|$, if $(s, v) \notin A$ and $w_{(s,v')} = 0$, otherwise.

Let $R = \{v'' \mid \forall v \in V \setminus \{s\}\}$ be the set of "PE" nodes. For all $r \in R$, let $p^r = \{(s, r'), (r', r'')\}$ be an alternative path. Consider weight $w_p^r = |A| + |V|$ and $w^1 = 0$. See Fig. 2. For $q = 1$, solving the monitoring paths computation problem in G' allows us to solve the Hamiltonian path problem in G. Indeed, since $\delta^+(t) = \emptyset$, the monitoring path crosses every link (r', r'') $\forall r \in V \setminus \{s\}$, and the result follows. □

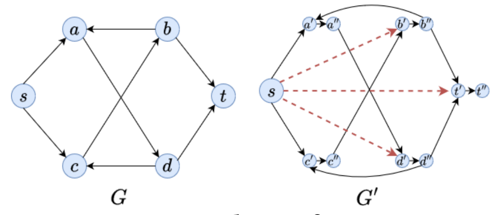

Figure 2: Graph transformation

4.3 Mathematical model

Let $z_p^r \in \{0, 1\}$ be a binary variable that equals 1 if alternative path $p \in P^r$ between $u^* \in V$ and $r \in R$ is used for monitoring and 0 otherwise. Let $y_i \in \{0, 1\}$ be a binary variable that equals 1 if monitoring path $i \in Q$ is considered in the solution, and 0 otherwise. Let x_a^i be a binary variable that equals 1 if link $a \in A$ belongs to monitoring path $i \in Q$ and 0 otherwise.

The monitoring paths computation problem is equivalent to the following Integer Linear Program (MPCP):

$$\min \quad \sum_{i \in Q} \Big(w^1 y_i + \sum_{a \in A} w_a x_a^i \Big) - \sum_{r \in R} \sum_{p \in P^r} w_p^r z_p^r \tag{1}$$

$$\sum_{a \in \delta^+(v)} x_a^i - \sum_{a \in \delta^-(v)} x_a^i = \begin{cases} y_i & \text{if } v = u^*, \\ -y_i & \text{if } v = r^*, \\ 0 & \text{otherwise.} \end{cases} \quad \forall i \in Q, v \in V, \tag{2}$$

$$\sum_{a \in A[W]} x_a^i \leq |W| - 1 \qquad \forall W \subseteq V, \forall i \in Q, \tag{3}$$

$$\sum_{i \in Q} x_a^i + \sum_{r \in R} \sum_{p \in P^r | a \in p} z_p^r \geq 1 \qquad \forall a \in A' \setminus \delta(u^*), \tag{4}$$

$$\sum_{a \in A} x_a^i \leq L_{\max} \qquad \forall i \in Q. \tag{5}$$

where r^* is a dummy node connected to each "PE" node $r \in R$ by the following dummy link (r, r^*). Constraints (2)-(3) represent the flow conservation equalities and sub-tour elimination inequalities. They allow computing the monitoring paths. Constraints (4) ensure monitoring every link in the alternative path by at least one path. Note that, links adjacent to u^* can be unmonitored. Finally, Constraints (5) bound the number of hops in the monitoring paths.

5 PARTIAL LINKS MONITORING

In this section, we show that it is possible to get the measurements of all links in the alternative paths without monitoring all of them. This allows us to decrease the number of paths used for monitoring and the number of links to be monitored. For example, consider the graph in Fig. 3 where the weights on the links represent the link cost. The graph contains 8 nodes including 3 "PE" nodes (in blue).

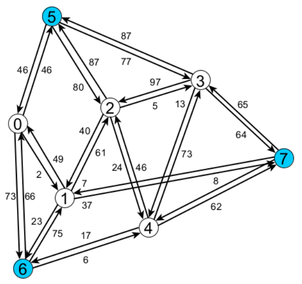

Figure 3: Network with 8 nodes. Nodes 5, 6 and 7 represent the "PE" nodes. Weights or links represent the TE Costs.

Fig. 4 represents the shortest paths between "PE" nodes. In this example, we consider node 3 for the congestion mitigation. Node 3 is aware of the link loads of all its ingress/outgoing links. Fig. 5 displays two alternative paths between node 3 for every "PE" node.

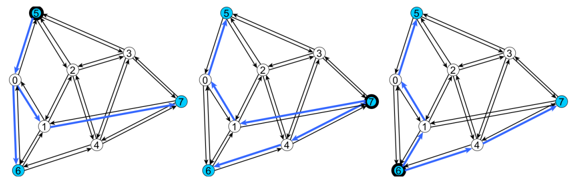

Figure 4: Shortest path trees between "PE" nodes.

Finally, Fig. 6 shows two monitoring paths, the first one is between nodes 3 and 6 and the second one is between 3 and 7.

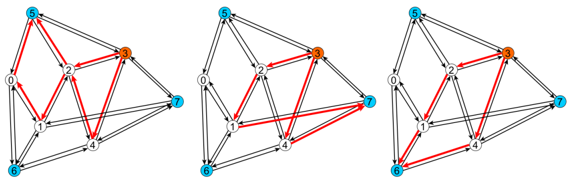

Figure 5: 2 Alternative paths between node 3 and each "PE".

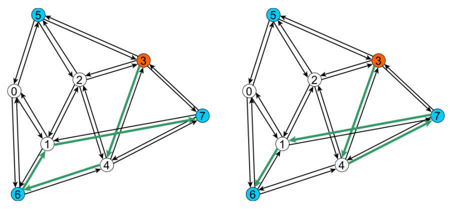

Figure 6: Monitoring paths (in green) to get loads of all links in the alternative paths given in Fig. 5 (in red).

It is easy to see that the number of monitoring paths is smaller than the number of alternative paths. Moreover, the number of links in the monitoring paths is smaller than those of the alternative paths. Although the monitoring paths do not cover all links in the alternative paths, they are enough to get the measurement of all links in the alternative paths. As links $(2, 1)$, $(2, 5)$, $(4, 2)$ do not appear in the OSPF paths, they cannot route any traffic. Hence, they are not monitored even if they belong to alternative paths. Links $(1, 0)$, $(0, 5)$ are not monitored even if they appear in the alternative and OSPF paths. The traffic over these two links can be deduced thanks to the measurements of links $(6, 1)$, $(7, 1)$. Indeed, since link $(2, 1)$ belongs to no OSPF path, and all OSPF paths crossing $(1, 0)$ do not cross $(0, 6)$ the traffic load over links $(1, 0)$ and $(0, 5)$ equals the loads sum of $(6, 1)$ and $(7, 1)$.

5.1 Mathematical model

The partial monitoring paths computation problem (PMPCP) is a variant of MPCP, described in the previous section. In contrast with MPCP, in this version, we relax Constraints (4) forcing the monitoring paths to cross over all links in the alternative paths. The monitoring on a link in the alternative path can be skipped if 1) it belongs to no OSPF path, or, 2) it is able to recover its link load only based on the monitoring of other links. We refer to the second type links as the *"Recovered links"*. For that, we need to ensure that all traffic matrices satisfying the loads on the monitored links, give the same load on every Recovered link. The PMPCP can, then, be tackled as a Bilevel optimization problem [17], where the leader selects the links to monitor and the follower tries to find two different traffic matrices giving the same load on monitored links (decided by the leader) but with different loads on at least one recovered link. From MPCP, we consider additional decision variables for the leader as follows: let $t_a \in \{0, 1\}$ be a variable that equals 1 if link $a \in A$ is monitored and 0 otherwise. Moreover, we consider further variables for the follower: let $q_a \in \mathbb{R}^+$ be the load difference between the two traffic matrices on link $a \in A$. And, let $x_p^i \in \mathbb{R}^+$ be the amount of traffic over the OSPF path $p \in S$ associated with traffic matrix $i = \overline{1, 2}$.

The partial monitoring paths computation problem is equivalent to the following Bilevel mathematical model (PMPCP)

$$\min \sum_{a \in A} w^0 t_a + \sum_{i \in Q} \big(w^1 y_i + \sum_{a \in A} w_a x_a^i \big) - \sum_{r \in R} \sum_{p \in P^r} w_p^r z_p^r \tag{6}$$

$$\textit{Constraints} \qquad (2), (3), (5)$$

$$x_a^i \le t_a \qquad \forall a \in A, i \in Q, \tag{7}$$

$$z_p^r \le t_a \qquad \forall r \in R, p \in P^r, a \in p, \tag{8}$$

$$t_a \le \sum_{i \in Q} x_a^i + \sum_{r \in R} \sum_{p \in P^r | a \in p} z_p^r \qquad \forall a \in A, \tag{9}$$

$$\vartheta(t) \le 0, \tag{10}$$

$$\text{where} \qquad \vartheta(t) = \max \sum_{a \in A' \cap \bar{A}} q_a \tag{11}$$

$$\alpha_a : \; t_a \sum_{p \in S \ni a} x_p^1 = t_a \sum_{p \in S \ni a} x_p^2 \qquad \forall a \in A, \tag{12}$$

$$\beta_a : \; - \sum_{p \in S \ni a} x_p^1 + \sum_{p \in S \ni a} x_p^2 \le -q_a \qquad \forall a \in A, \tag{13}$$

$$\theta_a^i : \; \sum_{p \in S | a \in p} x_p^i \le b_a \qquad \forall a \in A, i = \overline{1, 2}. \tag{14}$$

where $w^0 \in \mathbb{R}_+$ represents the weight given to the number of monitored links in the objective function, $b_a \in \mathbb{R}^+$ represents the bandwidth capacity of link $a \in A$ and α, β and θ represent the dual variables associated with Constraints (12), (13) and (14), respectively. Constraints (7)-(9) guarantee that if a link is monitored it must belong to a new monitoring path or an alternative path used for monitoring. Thanks to the follower, Constraints (10) ensure that all traffic matrices satisfying the loads on the monitored links, give the same load on the recovered links. For the follower, Objective (11) maximizes the total link load differences between two traffic matrices. Note that only links belonging to both OSPF and alternative paths are considered in this objective function. Constraints (12) ensure that, for every monitored link (i.e., $t_a = 1$), the two matrices give the same load. Constraints (13) computes the load difference for every link and finally, Constraints (14) guarantee that each traffic matrix respects the link capacities.

5.2 Single-level model

In the literature, several methods have been developed to solve the Bilevel optimization problems. One approach, called "Single-Level Reduction", consists in including the KKT conditions of the follower as constraints of the leader problem. In our case, we can exploit the fact that the follower is a maximization problem and $\vartheta(t) \geq 0$ for any $t \in \{0,1\}^{|A|}$ (as $q(t) \geq 0$), to guarantee the optimality of the follower. Let us consider the dual of the follower, given as follows:

$$\min \quad \sum_{a \in A} \sum_{i \in Q} \theta_a^i b_a \tag{15}$$

$$\sum_{a \in p} (\alpha_a t_a - \beta_a + \theta_a^1) \geq 0 \qquad \forall p \in S, \tag{16}$$

$$\sum_{a \in p} (-\alpha_a t_a + \beta_a + \theta_a^2) \geq 0 \qquad \forall p \in S, \tag{17}$$

$$\beta_a \geq 1 \qquad \forall a \in A' \cap \bar{A}, \tag{18}$$

$$\beta_a, \theta_a \geq 0 \qquad \forall a \in A. \tag{19}$$

Claim 5.2.1. *All variables θ can be set to 0.*

Proof. From Constraints (10), $\vartheta(t) = \sum_{a \in A} \sum_{i \in Q} \theta_a^i b_a = 0$. Since $\theta \geq 0$ and $b_a \geq 0$ for all $a \in A$, the result follows. □

The single-level model can be obtained from PMPCP by replacing (10)-(14) by the constraints of the dual of the follower together with $\sum_{a \in A} \sum_{i \in Q} \theta_a^i b_a = 0$. By Claim 5.2.1, the single-level model is equivalent to the following the mixed integer non-linear model:

$$\min \quad \sum_{a \in A} w^0 t_a + \sum_{i \in Q} (w^1 y_i + \sum_{a \in A} w_a x_a^i) - \sum_{r \in R} \sum_{p \in P^r} w_p^r z_p^r \tag{20}$$

$$\textit{Constraints} \quad (2), (3), (5)$$

$$\textit{Constraints} \quad (7), (8), (9)$$

$$\sum_{a \in p} \alpha_a t_a = \sum_{a \in p} \beta_a \qquad \forall p \in S, \tag{21}$$

$$\beta_a \geq 1 \qquad \forall a \in A' \cap \bar{A}, \tag{22}$$

$$\beta_a \in \mathbb{R}^+ \qquad \forall a \in A. \tag{23}$$

The above model can be linearized easily as follows.

Claim 5.2.2. *For $a \in A$, let $f \in \mathbb{R}$ be a new variable. For an enough big value $M \in \mathbb{R}^+$, Constraints (21) can be replaced by*

$$\sum_{a \in p} f_a = \sum_{a \in p} \beta_a \qquad \forall p \in P,$$

$$\alpha_a - M(1 - t_a) \leq f_a \leq \alpha_a + M(1 - t_a) \qquad \forall a \in A,$$

$$-Mt_a \leq f_a \leq Mt_a \qquad \forall a \in A.$$

6 NUMERICAL EXPERIMENTS

We now evaluate the monitoring paths computation on randomly generated networks. For benchmarking, we compare our solution to the "Naive approach" which consists in monitoring all alternative paths with path-tracing. Except for Figure 9, all experiments are on the partial links monitoring given in Section 5. In our tests, the models are solved using IBM ILOG CPLEX 12.6 solver [6]. All implementations are in Python on a machine with an Intel(R) Xeon(R) CPU E5-4627 v2 at 3.30GHz and 504GB RAM, running Linux 64 bits. A maximum of 32 threads has been used for CPLEX, and a time-limit of 3 hours. The instances have been generated by varying the following parameters: the number of nodes: $\{50, 100, 200\}$, the number of alternative paths k: $\{2, 4\}$, the number of "PE" nodes: $\{30\%, 60\%\}$ of the number of nodes, the network density: $\{20\%, 40\%\}$, the maximum number of hops in the monitoring paths $L_{\max}$: $\{6, 12\}$ and the maximum number of monitoring paths q: $\{5, 10\}$.

We consider the following weights in the objective function

- $w^0 = 10^3, \qquad w^1 = 10^2, \qquad w_a = 10^2 \ \forall a \in A,$
- $w_p^r = 10^2 \times (|p| + 1), \ \forall r \in R$ and $p \in P^r$.

These weights give the highest priority to minimizing the number of monitored links. 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 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. The OSPF paths have been computed between each pair of nodes in the network using the Dijkstra algorithm [7]. For a given node $u \in V$, for every "PE" node $r \in R$, and for every outgoing arc $\bar{a} \in \delta(u)$, the k associated alternative paths are generated by solving mathematical model (ILP). The model maximizes the disjointness between the k paths without crossing $\bar{a}$.

Fig. 7 displays the reduction ratio on the number of monitored links when using the monitoring paths described in Section 5 compared to the "Naive approach". Each color corresponds to a distinct parameter, each of which has two values (see above). In Plot 7.(a), on instances with 50 nodes, we notice that with few "PE" nodes (i.e., 30% of number of nodes), the improvement on the number of monitored links is much higher (83% instead of 65%). This is to be expected, as the greater the number of "PE" nodes, the more alternative and OSPF paths there are. On the other hand, we notice that a high value of k gives a better improvement in the number of monitored links. Indeed, when the number of alternative paths is high, the links are crossed multiple times. This leads to redundant monitoring via the "Naive solution". These two behaviours are similar on networks with 100 and 200 nodes (Plots 7.(b) and 7.(c)). On 50 node instances, we also notice that the density, the maximum monitoring path length "P-Length" and the maximum number of monitoring paths "M-Paths" positively impact the number of monitored links. These three parameters allow to design of efficient monitoring paths, avoiding repeated crossed links. On instances with 100 and 200 nodes, The behaviour is ambiguous, and due to a significant optimality gap, a definitive conclusion cannot be drawn. Fig. 8 displays a comparison of the following ratios:

- Recovered Links "R": $\frac{\#\text{Recovered links}}{|A'|} \times 100$
- Optimality Gaps "G": $\frac{\text{Upper bound} - \text{Lower bound}}{\text{Lower bound}} \times 100$
- Paths for monitoring "M": $\frac{\#\text{Alternative paths} - \#\text{Paths for monitoring}}{\#\text{ Alternative paths}} \times 100$

for instances of 50, 100 and 200 nodes. The higher the number of nodes, the lower the ratio of recovered links. This is due to

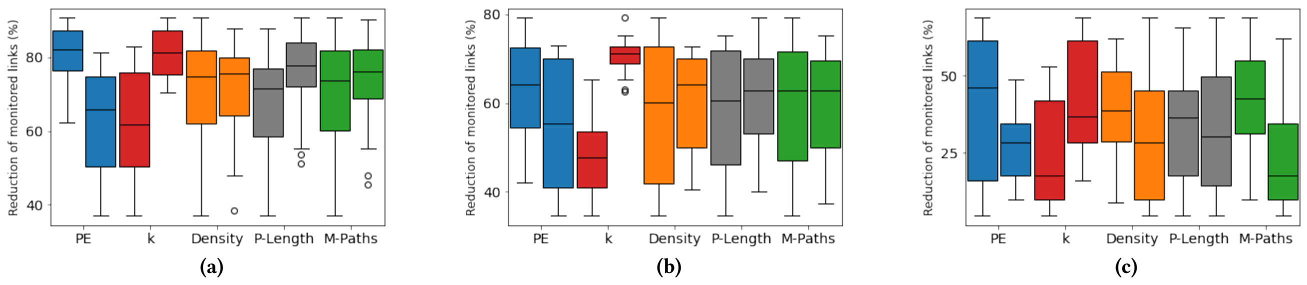

Figure 7: Reduction ratio on the number of monitored links (%) for 50, 100 and 200 nodes respectively.

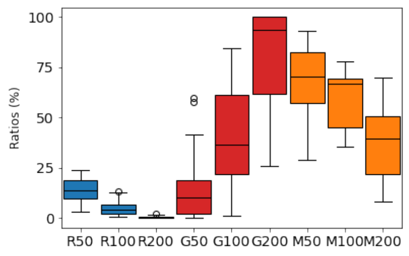

Figure 8: Ratio of Recovered links, Optimality gap and Reduction of paths used for monitoring.

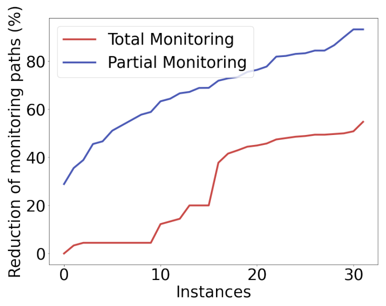

Figure 9: Reduction comparison of the number of paths used for monitoring between Total and Partial monitoring.

the optimality gap. Indeed, this latter increases with the number of nodes impacting the quality of the solutions. We see the same impact on the number of paths used for monitoring. We save 68%, 58% and 39% of paths, in average, for instances of 50, 100 and 200 nodes, respectively. These results depend mainly on the quality of the solutions obtained within the time limit. The last Fig. 9 shows a comparison of the reduction ratio on the number of paths used for monitoring between "Total Monitoring" version given in Section 4 and the "Partial Monitoring" given in Section 5. The second version performs much better than the first one. This was expected since in the second version, the paths used for monitoring can avoid crossing through Recovered links and Links out of OSPF paths.

7 CONCLUSION

In this paper, we propose a way to exploit the in-band telemetry for accurate congestion mitigation. We have shown that a lot of redundancies in the measurements appear when alternative paths are monitored. We proposed to design extra "monitoring paths" to help drastically decrease the number of monitored links. Moreover, we have shown that up to 25% of links in alternative paths can be recovered without monitoring. Indeed, the load over these links can be recovered from the measurement of others. From a practical perspective, an efficient heuristic needs to be developed to solve the problem in a short time in order to be used in practice.

REFERENCES

[1] Lirong Bai, Liang Zhang, Geng Zhang, Lin Zhang, Paolo Medagliani, and Sébastien Martin. 2023. A Distributed Congestion Mitigation Mechanism Based on Neighboring Nodes Traffic Steering. In Proc. IEEE NoF.
[2] Frank Brockners. 2017. Next-gen Network Telemetry is Within Your Packets: In-band OAM. http://events17.linuxfoundation.org/sites/events/files/slides/In-band_OAM.pdf.
[3] Alexander Brundiers, Timmy Schüller, and Nils Aschenbruck. 2022. Midpoint Optimization for Segment Routing. In Prox. IEEE INFOCOM. 1579–1588.
[4] Alexander Brundiers, Timmy Schuller, and Nils Aschenbruck. 2023. Tactical Traffic Engineering with Segment Routing Midpoint Optimization. In IFIP Networking.
[5] Ariel G. Castro, Arthur F. Lorenzon, Fábio D. Rossi, Roberto I. T. da Costa Filho, Fernando M. V. Ramos, Christian E. Rothenberg, and Marcelo C. Luizelli. 2021. Near-Optimal Probing Planning for In-Band Network Telemetry. IEEE Communications Letters 25, 5 (2021), 1630–1634.
[6] CPLEX. [n.d.]. https://www.ibm.com/products/ilog-cplex-optimization-studio.
[7] Edsger W Dijkstra. 2022. A note on two problems in connexion with graphs. In Edsger Wybe Dijkstra: His Life, Work, and Legacy. 287–290.
[8] Dennis Ferguson, Acee Lindem, and John Moy. 2008. OSPF for IPv6. RFC 5340. https://doi.org/10.17487/RFC5340
[9] Clarence Filsfils, Ahmed Abdelsalam, Pablo Camarillo, Mark Yufit, Thomas Graf, Yuanchao Su, Satoru Matsushima, Mike Valentine, and Amit Dhamija. 2023. Path Tracing in SRv6 networks. Internet-Draft draft-filsfils-spring-path-tracing-05. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-filsfils-spring-path-tracing/05/ Work in Progress.
[10] Michael R Garey and David S Johnson. 1979. Computers and intractability. A Guide to the Theory of NP-Completeness (1979).
[11] The P4.org Applications Working Group. 2020. In-band network telemetry (INT) dataplane specification version 2.1. https://github.com/p4lang/p4-applications/blob/master/docs/INT_latest.pdf.
[12] Youngho Kim, Dongeun Suh, and Sangheon Pack. 2018. Selective in-band network telemetry for overhead reduction. In Proc. IEEE CloudNet. IEEE, 1–3.
[13] Tian Pan, Enge Song, Zizheng Bian, Xingchen Lin, Xiaoyu Peng, Jiao Zhang, Tao Huang, Bin Liu, and Yunjie Liu. 2019. Int-path: Towards optimal path planning for in-band network-wide telemetry. In Proc. IEEE INFOCOM.
[14] Leonardo Rodoni. 2022. Delay Measurement, Path Tracing, and Telemetry Data Correlation in Segment Routed Networks. Technical Report. ETH and Swisscom.
[15] J Shin and SK Telecom. [n.d.]. In-situ Flow Information Telemetry Framework draft-song-opsawg-ifit-framework-00.
[16] Goksel Simsek, Doğanalp Ergenç, and Ertan Onur. 2021. Efficient network monitoring via in-band telemetry. In Proc. IEEE DRCN.
[17] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. 2018. A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications. IEEE Transactions on Evolutionary Computation 22, 2 (2018), 276–295.
[18] Lizhuang Tan, Wei Su, Wei Zhang, Jianhui Lv, Zhenyi Zhang, Jingying Miao, Xiaoxi Liu, and Na Li. 2021. In-band Network Telemetry: A Survey. Computer Networks 186 (2021), 107763. https://doi.org/10.1016/j.comnet.2020.107763
[19] Shaofei Tang, Deyun Li, Bin Niu, Jianquan Peng, and Zuqing Zhu. 2019. Sel-INT: A runtime-programmable selective in-band network telemetry system. IEEE transactions on network and service management 17, 2 (2019), 708–721.