Martínez, R., Vettori, L., Baranda, J., Mangues-Bafalluy, J. & Zeydan, E. (7-11 Dec. 2021). *Efficient restoration of simultaneous transport services within an NFV infrastructure* [proceedings]. 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain.

DOI: 10.1109/GLOBECOM46510.2021.9685511

Efficient Restoration of Simultaneous Transport Services within an NFV Infrastructure

Ricardo Martínez, Luca Vettori, Jorge Baranda, Josep Mangues-Bafalluy and Engin Zeydan
Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)
Castelldefels, Spain
ricardo.martinez@cttc.es

***Abstract*—In 5G networks, heterogeneous vertical services with different requirements are rolled out over a common multi-technology infrastructure. A resource orchestrator entity automatically coordinates the operations and functions to support the service's lifecycle management (i.e., creation, update and termination). Moreover, it is essential that service needs are continuously assured even if transport network anomalies (e.g., link failures) occur. Herein, we present an implemented resource orchestrator architecture integrating monitoring capabilities to attain *closed-loop* operations for: i) gathering monitored information; ii) detecting transport network anomalies; and iii) triggering the required action (e.g., restoration) to keep the service continuity. When a link failure happens, several transport services may be disrupted requiring to be immediately restored. To this end, we propose a novel on-line restoration algorithm called as Global Concurrent Optimization (GCO). The GCO algorithm aims at attaining an enhanced *restorability* performance compared to a more traditional restoration algorithm (referred to as *1-by-1*). Both algorithms are experimentally compared on top of the deployed resource orchestrator architecture. The evaluation is done upon both dynamic service arrival/departure and link failure generation using different performance metrics: the average restorability, the average network resource utilization, and the restoration computational time.**

***Index Terms*—Autonomous resource orchestration, network monitoring, on-line transport restoration**

I. Introduction

5G networks allow service providers offering dynamic, elastic, and customized deployments for vertical industrial-driven services (e.g., Industry4.0, automotive, eHealth, etc.) over a shared multi-technology infrastructure [1]. Vertical services are mapped into low-level network slices / network services (NSs) involving both computing and networking resources. Such resources indeed constitute the 5G infrastructure typically made up of a pool of data centers (i.e., Network Function Virtualization Infrastructure Point of Presence, Nfvi-Pops) interconnected by a transport network (Wide Area Network, WAN). The goal is to dynamically roll out NSs fulfilling their (computing and networking) requirements: i) Virtual Network Functions (VNFs) placed at the Nfvi-Pops; ii) WAN connectivity ensuring the bandwidth demands (b/s), maximum tolerated latency (ms) and reliability [2]. The life-cycle management (i.e., resource selection and allocation, re-optimization, and deletion) of the NSs is executed by a resource orchestrator entity. To this end, the orchestrator relies on the programmability and virtualization capabilities provided by both Software Defined Networking (SDN) and Network Function Virtualization (NFV) technologies.

In this 5G framework, the service assurance is a critical aspect to continuously meet the NSs' requirements in an *autonomous* way [3]. This entails: 1) gathering monitored network and service performance; 2) processing and analyzing the collected monitored information; and 3) deriving actions to preserve the NSs' requirements. These functions are carried out by the orchestration system in a *closed-loop* automation. In this context, we describe an implemented autonomous resource orchestration architecture with monitoring capabilities to dynamically manage WAN *network anomalies*. Typically, network anomalies can be classified into either *soft* or *hard* failures [4]. Hard-failures cause a complete disruption of the transport service (e.g., link or node failures). On the other hand, soft-failures are those degrading the connectivity service performance, e.g., increase of the packet losses, etc. For the latter, the use of Artificial Intelligence (AI) and Machine Learning (ML) mechanisms is seen as a key tool to proactively assist the resource orchestrator to trigger the required actions (e.g., re-routing) [5]. For the hard-failures, however, an alarm manager function in the monitoring platform can realize about these failures (e.g., link becomes failed), and report to the orchestrator so it can restore any disrupted transport service.

Focusing on WAN hard-failures, this work experimentally tackles the autonomous detection and restoration of a *bulk* of transport services affected by a link failure. For the restoration, the resource orchestrator relies on an on-line algorithm whose objective is to find feasible paths fulfilling the requirements (i.e., bandwidth and maximum latency) of any disrupted connectivity service. Typically, a *1-by-1* algorithm is adopted which in a sequential way triggers a specific restoration path computation for each transport service affected by a link failure. That is, the algorithm starts trying to restore the first transport service in the received bulk. If this succeeds, the network resources (i.e., link bandwidth) on the computed restored path are set to occupied prior to continue restoring the next bulk transport service [6] [7]. Otherwise, the restoration of the transport service is declared as failed, and the next bulk' service is attempted to be restored. Thus, this is a simple mechanism addressing the concurrent restoration of

Work supported in part by EU Commission H2020 5Growth project (Grant No. 856709), Spanish MICINN AURORAS (RTI2018-099178-B-I00) and Spanish MINECO 5G-REFINE (TEC2017-88373-R) projects and Generalitat de Catalunya grant 2017 SGR 1195.

several disrupted transport services. Nevertheless, it is likely that some of these transport services cannot be eventually restored despite a more efficient solution exists. Thus, with the aim to improve the *restorability* of a bulk of disrupted transport services, a novel restoration algorithm called Global Concurrent Optimization (GCO) is devised. The GCO restoration algorithm does enlarge the space of explored candidate solutions when restoring a bulk of disrupted transport services. To do this, the ordering of the received bulk of transport services is randomized several times to derive different solutions. Among the resulting solutions, GCO selects the one attaining the higher restorability and/or more efficient use of the network resources (e.g., lowest amount of used bandwidth for the restored paths).

Both GCO and 1-by-1 restoration algorithms are experimentally evaluated and compared into the implemented autonomous resource orchestration architecture with monitoring functionalities. For this, we consider i) dynamic arrival and departure of transport service requests with heterogeneous bandwidth and latency demands; and ii) dynamic and random generation of network link failures. The considered performance metrics are: a) the *blocked bandwidth ratio* (BBR) showing the ratio between the aggregated non-served connections' bandwidth over the total requested bandwidth during the provisioning; b) the *restorability*, defined as the amount of connections' bandwidth successfully recovered over the total connections' bandwidth affected by link failures; c) the average network bandwidth occupation; and d) the average restoration path computation time.

The rest of the paper is structured as follows. Section II presents the implemented architecture and interactions of the resource orchestrator to autonomously handle WAN hard-failures. Section III details the potential *inefficient* solution attained by 1-by-1 approach upon multiple disrupted transport services to be concurrently restored. Then, Section IV describes the proposed GCO algorithm to improve the service restorability. Section V compares the performance of both 1-by-1 and GCO algorithms under dynamic connection requests and link failure generation. Section VI concludes this work.

II. Resource Orchestrator with Monitoring Capabilities: Architecture and Workflow

Figure 1 shows the key architectural elements of the resource orchestrator with monitoring capabilities to autonomously handle network anomalies (e.g., hard-failures). The Resource Layer (RL) is the orchestrator executing the operations/functions supporting the NSs lifecycle management: i) creation and termination (i.e., resource allocation/removal); and ii) re-routing/restoration. A detailed description of the RL is found in [8] whose open source implementation, referred to as *mobilE transport pLatform for multi-tEChnology neTwoRk infrAstructure* (ELECTRA), is available in [9].

The *transport service generator* application has been deployed to dynamically create/terminate transport service requests demanded to the RL emulating the NSs' needs. This application interacts with the RL via a defined REST API [8]. In the transport service request, it is specified the service name *servName*, the source (*src*) and destination (*dst*) endpoints (i.e., NFVI-PoPs Gateways), as well as the service requirements such as bandwidth (*bw*) in b/s, and maximum end-to-end latency (*l*) in ms. For the removal of an active transport service, it is simply determined by its *servName*.

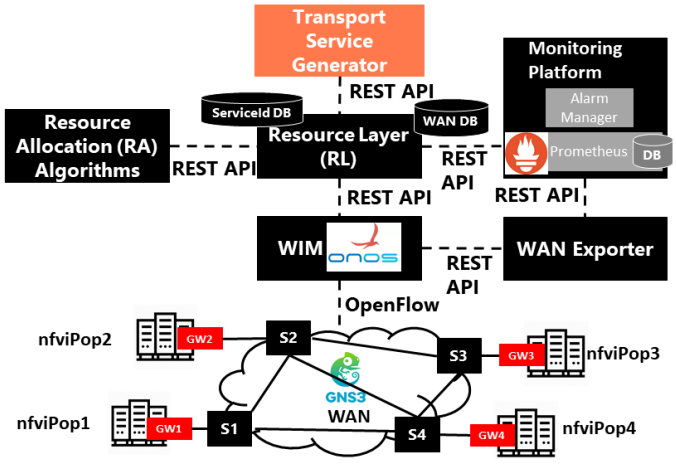

Fig. 1: Overall Architecture of the Resource Orchestration with Monitoring Capabilities

The Resource Allocation (RA) server is a standalone process assisting the RL for the computation of WAN paths (i.e., nodes and links) fulfilling the service requirements. Both RL and RA create a client/server relationship (supported by a defined REST API [10]) to request/respond path computation demands. The path computation request specifies: i) the set (bulk) of transport services to be computed with their own *src* and *dst* endpoints; ii) the algorithm identifier to be executed (e.g., either provisioning or restoration); iii) the constraints/requirements to be fulfilled (i.e., *bw*, *l*, failed links to be avoided); and iv) an updated WAN view (i.e., topology and links' attributes such as the available bandwidth and delay). In the successful RA response sent to the RL, it is determined (for each bulk transport service) the computed path (i.e., nodes and links). Next, for each node in the computed path, the RL communicates with the WAN Infrastructure Manager (WIM) implemented by an SDN controller (ONOS) instance [11]. This allows programming (via OpenFlow protocol) each node along the computed WAN path. Herein, the WAN infrastructure (nodes and links) is emulated by the GNS3 tool [12].

The monitoring platform is based on the open source monitoring system Prometheus [14]. This collects periodically WAN physical and performance information/metrics such as link and node status, link packet losses, etc., reported by the WAN exporter. Thus, the WAN exporter acts as an intermediate element which aggregates and filters monitoring information derived from the WIM (i.e., ONOS controller). To do so, the WAN exporter uses the ONOS REST API to request the controller about WAN links' status (i.e., active/inactive). This information is then exposed (and stored) at the monitoring

platform constituting the input to detect whether a link failure occurred. The element handling such decisions is the alarm manager. If this happens, the RL is immediately notified to trigger the restoration action for any affected transport service.

Figure. 2 describes the workflow (i.e., control interactions) to accomplish the closed-loop operations for autonomously restoring disrupted transport services. Once a WAN link fails, the ONOS SDN controller immediately updates its WAN topology reflecting such a failure (step 1). Next, using the periodical programmed queries (every *time polling*, t_p, e.g., 1 s) triggered by the Prometheus monitoring platform, it requests the WAN exporter to acquire updated WAN monitored information. The exporter asks the SDN controller to report the updated WAN state. The aggregated/filtered WAN information (e.g., link active/inactive) by the exporter is eventually passed (step 2) to the Prometheus tool to be stored in a dedicated database (i.e., DB). If the WAN link state of a previous DB instance differs from the new collected one, e.g., from active to inactive, the alarm manager reports that to the RL.

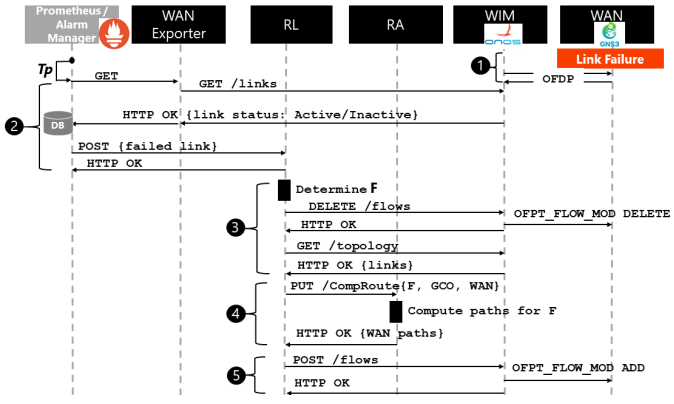

Fig. 2: Control workflow for restoring transport services.

In step 3, the RL processes the received alarm with the failed WAN link/s to determine whether any active transport service (i.e., F) is affected, hence needing restoration. If this happens, the RL applies a *break-before-make* (BBM) strategy: first, all the allocated resources (i.e., link bandwidth) bound to each connection (r_i) in F are de-allocated (via the ONOS controller); second, the RL queries to the RA server (step 4) the path re-computation/restoration of all r_i in F. Finally, the RA server responds to the RL indicating for each r_i the successfully restored path (i.e., nodes and links) or that a feasible path was not found since the requirements could not be fulfilled. Those r_is whose restoration path is found are then allocated via the ONOS controller (step 5).

III. Problem Statement: Example

This section deals with the step 4 of the workflow depicted in Fig. 2. The objective is to illustrate the potential inefficient restorability accomplished by traditional 1-by-1 algorithm when several disrupted transport services (i.e., F) need to be recovered simultaneously. Fig. 3(a) depicts a packet-switched WAN where each link describes its available capacity (in Mb/s) and (propagation) delay in ms. Three transport service are requested: $r1$ and $r2$ requesting both a bw of 50 Mb/s and l of 3 ms between switches S1 and S6, and $r3$ demanding a bw of 50 Mb/s and l of 4 ms between S2 and S6. For each r_i, the RA server executes a routing algorithm seeking for a path fulfilling the r_i's bw and l requirements. The algorithm relies on a K Constrained Shortest Path First (K-CSPF) mechanism detailed in a previous work [13]. As shown in Fig. 3(b), $r1$ and $r2$ are established over S1-S7-S6 path, whilst $r3$ is set up through S2-S7-S6 path.

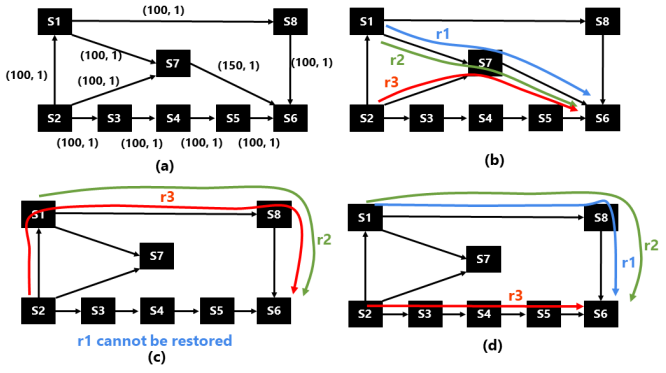

Fig. 3: Inefficient multiple transport service restoration.

Let's assume that S7-S6 link fails. Hence, all transport services (r_1, r_2 and r_3) need to be restored. Applying the 1-by-1 restoration algorithm, all r_i are tried to be re-computed sequentially according to the ordering in the received F. By doing so, it is very likely that despite there is a feasible solution that allows restoring all (or most of) the failed transport services, the restoration of a particular r_i precludes to successfully restore subsequent transport services (i.e., r_{i+1}, r_{i+2}, etc.). We use the example to reflect this as illustrated in Fig. 3(c). Assume that the F set received at the RA is [r_2, r_3, r_1]. Applying the K-CSPF algorithm (where the failed link is removed from the WAN view), r_2 is computed over the path S1-S8-S6. Next, r_3 is computed through the path S2-S1-S8-S6. After these two path computations and locally occupying the used resources for them, the algorithm is unable to find a feasible path for r_1 since its latency requirement cannot be fulfilled. However, as shown in Fig. 3(d), there is indeed a solution that allows restoring all transport services: r_1 and r_2 are set up over the path S2-S8-S6, whilst r_3 is accommodated on the path S2-S3-S4-S5-S6.

In light of the above, the goal and novelty of this work is to enhance the restorability of traditional 1-by-1 algorithm upon simultaneous and multiple disrupted transport services. This is accomplished by the devised GCO restoration algorithm.

IV. GCO Restoration Algorithm

The GCO's macroscopic objective is to attain the highest restorability among several disrupted transport services in F. The GCO implementation is available in [10]. The algorithm aims at exploring a broader solution space than traditional 1-by-1. Moreover, GCO is also devised to attain a more efficient

use of the overall network resources (e.g., link bandwidth). The latter objective is pursued to facilitate the establishment of subsequent new arriving transport services and/or the restoration of existing ones. The GCO algorithm inputs are: i) F; ii) updated WAN view (i.e., G); and iii) the maximum number of solutions to be explored (i.e., S_{max}). Each derived solution s in S encompasses a particular restoration path for every r_i. To do that, each s is computed randomizing the received r_i ordering in F, referred to as F_s. Once all the targeted s are computed, the GCO selects the s attaining the highest restorability and/or the more efficient use of the network resources (i.e., lowest amount of traversed links among all the successfully computed paths).

The following notation describes the GCO in Algorithm 1.

F	Failed transport services, $[r_1, r_2, ..., r_{\|F\|}]$;
$\|F\|$	Number of failed transport services in F;
S_{max}	Maximum number of solutions;
s	Solution iterator, $s : 1, .., S_{max}$;
$randomized(F)$	Function randomizing F;
F_s	Randomized set of transport services for s;
r_i^b, r_i^l	Bandwidth and latency requirements of r_i;
G	WAN graph view;
G_s	Temporal WAN graph view for s;
$f(r_i, G, K)$	K WAN paths for r_i fulfilling both r_i^b and r_i^l. K paths are sorted by: 1) paths with largest available bandwidth on the most congested link; 2) if tie, it is fostered paths with the lowest end-to-end latency; 3) if tie, it is prioritized paths with less number of links. The first sorted path ($p_{r_i}^0$) is returned.
P_s	Set with the $p_{r_i}^0$ for $i : 1, .., \|F\|$ in s;
P	Set with all the computed P_s;
P_o	P_s returned to the RL;
$g(G, p_{r_i}^0)$	Local G update occupying resources for $p_{r_i}^0$;
$h(P)$	Fitness function sorting P by: 1) P_s with the largest amount of restored bandwidth; 2) if tie, the one with the lowest average path length (i.e., number of links) is sorted firstly;

S_{max} is a design parameter to limit the total number of explored solutions by the GCO algorithm. First, it is checked whether the number of potential different ordering of F (i.e., $|F|!$) is lower than S_{max}. If yes, the targeted number of solutions (S) is set to $|F|!$. Otherwise, S is set to S_{max}. Observe that the above is done to unnecessarily compute solutions if the number of transport services (F) is low (e.g., 1 or 2 disrupted transport services). Next, the algorithm starts computing each solution s. To this end, r_is in F are shuffled to generate a particular F_s. Next, the K-CSPF algorithm is triggered for each r_i in the resulting F_s. The computed (up to) K WAN paths are sorted and the first one ($p_{r_i}^0$) is chosen. The criteria to sort the computed K paths is: first, prioritizing those having the largest available bandwidth on the most congested link; if two or more paths have the same available bandwidth, it is prioritized those with the lowest path latency; finally, if the tie holds, the path traversing the less number of links is sorted first. Prior to compute the restoration path for the next transport service (i.e., r_{i+1}) in F_s, those resources (i.e., link bandwidth) computed for previous r_is are occupied in the *temporal* WAN view (i.e., G_s) executing g. Finally, the computed path set for solution s (i.e., P_s) is added to P. The final P_o in P is sent to the RL. To determine P_o, a fitness function h allows choosing the P_s achieving the highest amount of restored bandwidth. If there is tie between two or more P_ss, it is selected the P_s traversing (in average) the lowest number of links. Note that the fitness function leads to first increase the overall restored bandwidth, and second attain an efficient network bandwidth utilization.

Algorithm 1 GCO pseudocode

Input: F, G, S
Output: P_o
if $|F|! < S_{max}$ **then**
  $S = |F|!$
else
  $S = S_{max}$
end if
$s \leftarrow 0$
for $s < S$ **do**
  $F_s \leftarrow randomized(F)$
  $G_s \leftarrow G$
  for r_i in F_s **do**
    $p_{r_i}^0 \leftarrow f(r_i, G_s, K)$
    $P_s \leftarrow P_s + p_{r_i}^0$
    $G_s \leftarrow g(G_s, p_{r_i}^0)$
  end for
  $s \leftarrow s + 1$
  $P \leftarrow P + P_s$
end for
$P_o \leftarrow h(P)$
return P_o

V. Experimental Evaluation

The experimental evaluation compares the proposed GCO with respect to the 1-by-1 restoration algorithm on top of the implemented autonomous resource orchestrator with monitoring capabilities. The GNS3 application is used to emulate the packet-switched core WAN depicted in Fig. 4. This WAN infrastructure is made up of 14 packet nodes interconnected through 1 Gb/s bidirectional links. The delay (in ms) between node pairs is labeled on each link, see Fig. 4. Each node can act as source, destination or transit element of any bidirectional transport service. Service requests (r) arrive according to a Poisson process with mean inter-arrival time (IAT_r) set to 10 s. The duration of r is exponentially modelled whose mean Holding Time (HT_r) varies between $[250, 300, 350, 400]$ s offering different traffic loads. Although no real data traffic is transported, node flow rules are programmed by the ONOS SDN controller (via OpenFlow protocol) when setting up an r. For each r, both the source and destination nodes are randomly chosen among the 14 WAN nodes. The bandwidth

and latency requirements of r are uniformly chosen among the sets $[50, 100, 150, 200]$ Mb/s and $[10, 15, 20]$ ms, respectively.

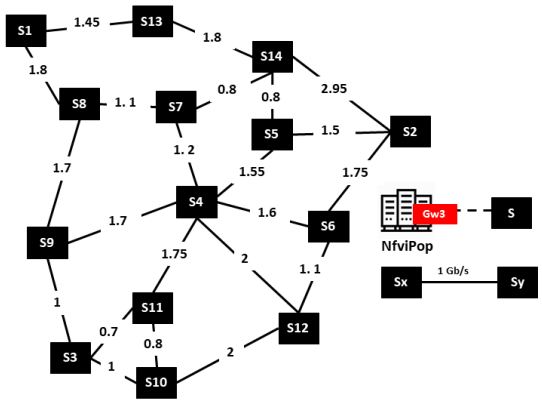

Fig. 4: Emulated core WAN using GNS3.

The developed link failure generator communicates with the GNS3 application leveraging the GNS3 API. This generator randomly sets a WAN link to failed for a determined time. To do that, a Poisson process with inter-arrival time IAT_f of 80 s triggers the failed link event. The failed link duration is exponentially modelled with a mean (HT_f) of 120 s.

The performance comparison relies on the following metrics: i) the *restorability* quantifying the amount of failed connection's bandwidth that are successfully restored; ii) the blocked bandwidth ratio (*BBR*) showing the amount of bandwidth of new transport services that cannot be provisioned; iii) the average network bandwidth occupation; and iv) the average elapsed time by the restoration path computation (i.e., 1-by-1 or GCO). Each test entails the generation of 3000 transport services, and is repeated 8 times. The K value for the K-CSPF algorithms is set to 5. The S_{max} for GCO algorithm is set to 6. The rationale behind this is that in average the number of simultaneously disrupted transport services (i.e., $|F|$) varies from 3 to 4 depending on the offered traffic load. Thus, setting S_{max} = 6 allows achieving a good trade-off between the number of GCO explored solutions and avoiding to increase the restoration path computation time too much.

A. Results and Discussion

The GCO primary objective is to improve the restoration performance of 1-by-1 upon multiple disrupted transport services (i.e., F). This is measured by the restorability metric where the higher the restorability is, the larger amount of restored bandwidth is achieved. As shown in Fig. 5, GCO does outperform the restorability obtained by 1-by-1 for all traffic loads. At low traffic load (i.e., HT_r=250 s), the average restorability is 95.0% and 95.8% for 1-by-1 and GCO, respectively. In bandwidth terms, this improvement means that GCO with respect to 1-by-1 (upon the test is completed) allows restoring a higher aggregated bandwidth, approximately 1.2 Gb/s. On the other hand, as the traffic load grows, more transport services coexist which makes network resources (e.g., link bandwidth) more occupied. Thereby, regardless of the applied algorithm, it becomes more troublesome to restore affected transport services which worsens the overall restorability performance. That said, we observe that GCO still improves the restorability attained by 1-by-1. Particularly, at the highest traffic load (HT_r=400 s), the attained restorability is 84.9% and 85.5% by 1-by-1 and GCO, respectively. This improvement is achieved because GCO algorithm explores over broader restoration path solution space than 1-by-1, which leads to find a more efficient restoration solution for F.

The above improvement, however, is attained at the expenses of increasing both the algorithm complexity and computational time (see Table I). Indeed, since GCO explores more solutions, its computational time is larger (around 3 times) than 1-by-1, e.g., for HT_r = 300 s, this time is 9.4 ms and 29.1 ms, for 1-by-1 and GCO, respectively. Hence, a trade-off between restorability and restoration path computation time arises when adopting either GCO or 1-by-1 algorithm. Moreover, as traffic load grows, the restoration path computation time in both GCO and 1-by-1 algorithms is increased. Indeed, more transport services are established and when a link failure occurs, this entails that larger number of services (F) need to be restored increasing the overall algorithm computational time.

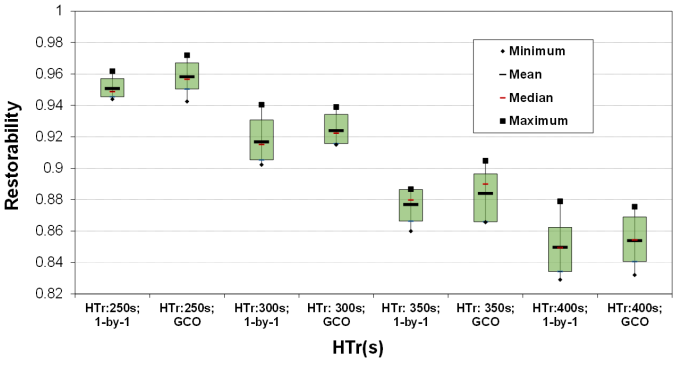

Fig. 5: Restorability performance: 1-by-1 vs. GCO

Table I also shows the average network bandwidth occupation ratio for both GCO and 1-by-1 algorithms for the different HT_rs. This ratio is measured as the amount of used bandwidth over the total available bandwidth through all WAN links. This measurement is repeated whenever a link failure occurs. All collected measurements through the test are averaged. For each individual measurement, it is retrieved the bandwidth occupation ratio of every active (not failed) link. Then, it is averaged the (active) link's bandwidth occupation ratio to derive the targeted network bandwidth occupation. We observe that such a ratio is relatively low (ranging 37.0%-53.6%). This is mostly due to the stringent latency requirement bound to the transport services. To meet that, both the provisioning and restoration algorithms tend to route transport services over a specific subset of WAN links whose bandwidth becomes highly used. These links are the most network centered and/or having lowest delay. Thus, transport services are not accommodated where available network capacity is larger. As seen in Table I, the average

bandwidth occupation ratio of the most used links is larger than the obtained ratio for the whole network, ranging between 70.2% and 87.2% depending on the traffic load and restoration algorithm. This ends up with a notable unbalanced use of the WAN link bandwidth. That said, GCO attains a higher average network bandwidth occupation since more transport services are restored and thus, bandwidth resources remain more used.

TABLE I: Average occupied network and most used links bandwidth, restoration computation times for 1-by-1 and GCO

HTr(s)	Rest. algorithm	Av. Network Bw Occupation Ratio (%)	Av. Most Used Links Bw Occupation Ratio (%)	Av. Rest. Path Comp. time (ms)
250	1-by-1	37.0	70.2	8.8
	GCO	37.4	71.8	24.5
300	1-by-1	43.7	77.9	9.4
	GCO	44.2	78.6	29.1
350	1-by-1	49.3	82.3	9.7
	GCO	49.7	84.1	30.9
400	1-by-1	53.4	87.0	9.9
	GCO	53.6	87.2	32.9

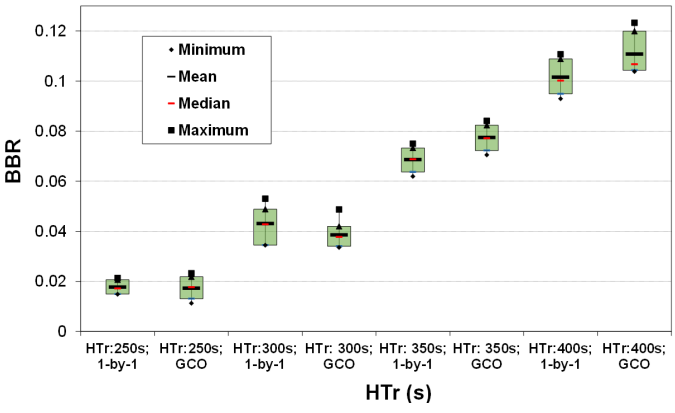

Fig. 6: BBR performance: 1-by-1 vs. GCO

Figure 6 depicts the BBR performance vs. HT_r for both GCO and 1-by-1 algorithms. This metric allows assessing how the adopted restoration algorithm impacts on the path and resource selection when serving new incoming transport services. In general, the aim is to lower the BBR. For low traffic load (i.e., HT_r= 250 and 300 s), observe that GCO algorithm attains better BBR than 1-by-1 algorithm. As said, the GCO targets to improve the restorability as a primary objective, but also aims at finding the path restoration solution encompassing the more efficient use of the network bandwidth. This second objective leads to favour the establishment of upcoming new transport services, i.e., lower BBR. For example, in HT_r=300 s, the obtained BBR upon applying GCO or 1-by-1 is 0.038 and 0.043, respectively. Nevertheless, as the HT_r grows (larger than 300 s), the advantage brought by GCO in the BBR is lost. As aforementioned, GCO restores better and achieves higher average network bandwidth occupation than 1-by-1. Consequently, at high traffic loads, the provisioning mechanisms when applying the GCO algorithm experiences more problems than 1-by-1 to find feasible paths for new transport services, worsening the BBR performance.

In short, the proposed GCO algorithm always enhances the restorability compared to 1-by-1. Additionally, for low traffic load, GCO also favours the provisioning of new transport services. However, as traffic load grows and network resources become more used, 1-by-1 lowers BBR because disrupted services have more problems to be restored which, in turn, favours the establishment of new transport services.

VI. Conclusions

This work presents an implemented resource orchestration with monitoring capabilities architecture to autonomously support closed-loop operations for the dynamic restoration of multiple disrupted transport services. The key elements and their functions (i.e., collecting monitoring information, determining and restoring affected transport services) are discussed detailing the designed workflow and used APIs. Upon a WAN link failure, several transport services may be disrupted and need to be restored. To this end, we devise the GCO restoration algorithm aiming at improving the restorability performance obtained by traditional 1-by-1 algorithm. From the experimental results, GCO always outperforms the restorability regardless of the offered traffic load. This is achieved because GCO seeks for the most efficient restoration through deriving different candidate solutions. This restorability improvement is attained at the expenses of increasing the path computational time. For low traffic load, it is also seen that GCO favours the provisioning of new incoming transport services thanks to the more efficient use of the network bandwidth resource when computing the transport service restoration paths.

References

[1] A. Zafeiropoulos, et. al., "Enabling Vertical Industries Adoption of 5G Technologies: A Cartography of Evolving Solutions", in Proc. of IEEE EuCNC 2018, June 2018.

[2] H. D. Chantre and N. L. S. da Fonseca, "Reliable Broadcasting in 5G NFV-Based Networks", IEEE Commun. Mag., vol. 56 (3), March 2018.

[3] X. Xie, et. al., "Service Assurance in 5G Networks: A Study of Joint Monitoring and Analytics", in Proc. of IEEE PIMRC, Sept. 2019.

[4] S. Shahkarami, F. Musumeci, F. Cugini and M. Tornatore, "Machine-Learning-Based Soft-Failure Detectionand Identification in Optical Networks", in Proc. of OSA 2018, March 2018.

[5] R. Boutaba, et. al., "A comprehensive survey on machine learning for networking: evolution,applications and research opportunities", J. of Internet Service and Applications, vol. 9 (16), 2018.

[6] Q. Li, et. al., "BOND: Flexible failure recovery in software defined networks", Elsevier Computer Networks, vol. 149, Feb. 2019.

[7] S. A. Astaneh and S. S. Heydari, "Optimization of SDN flow operations in multi-flow restoration scenarios", IEEE Trans. on Network and Service Management, vol. 13(4), Sept. 2016.

[8] J. Baranda, L. Vettori, R. Martínez and J. Mangues-Bafalluy, "A Mobile Transport Platform Interconnecting VNFs over a Multi-Domain Optical/Wireless Network: Design and Implementation", in Proc. of IFIP ONDM 2020, May 2020.

[9] Open Source CTTC ELECTRA Implementation, https://github.com/5growth/5gr-rl/tree/cttc-rl.

[10] Open Source CTTC Resource Allocation Server Implementation, https://github.com/5growth/5gr-rl/tree/master/rl/RL_RA_Server_R2/src.

[11] Open Network Operating System (ONOS), https://opennetworking.org/onos/

[12] GNS3 Network Emulator, https://www.gns3.com/.

[13] R. Martínez, et. al., "Experimental Validation of Compute and Network Resource Abstraction and Allocation Mechanisms within an NFV Infrastructure", in Proc. of IFIP/IEEE IM 2021, May 2021.

[14] Open source monitoring solution Prometheus, https://prometheus.io/.