Guimarães, C., Oliva, A. de la y Contreras, L.M. (2021). Support for availability attributes in network slices in GANSO. *Wiley Internet Technology Letters*.

DOI: https://doi.org/10.1002/itl2.295

Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

# Support for Availability Attributes in Network Slices in GANSO

Carlos Guimarães[1] | Antonio de la Oliva[1] | Luis M. Contreras[2]

[1]Telematics Department,
Universidad Carlos III de Madrid,
Madrid, Spain
[2]Transport & IP Networks,
Telefónica I+D / Global CTIO Unit,
Madrid, Spain

**Correspondence**
*Carlos Guimarães
Email: cmagalha@pa.uc3m.es

**Present Address**
Av. de la Universidad, 30,
28911 Leganés, Madrid, Spain

**Summary**

Network slicing provides a way to define logically isolated networks over the same and shared physical infrastructure. This paper departs from previous work defining *GST And Network Slice Operator* (GANSO), a framework for automating the creation of network slices over SDN architectures, and proposes extensions to support the *availability* attribute as defined by the GSMA Generic Network Slice Template (GST). This new attribute is integrated and implemented within the GANSO framework, being validated through a set of experimental results.

**KEYWORDS:**
Network Slicing, GST, Availability, Transport, SDN

## 1 | INTRODUCTION

5G deployments are shaping the way in which telecom operators provision their services, due to the native use of slices for partitioning and isolating resources. Through network slicing the telecom operators allocate partitions of a physical infrastructure (including compute, storage and networking resources) to different vertical customers, which run their services as if the specific partition was a dedicated network. Network slices are defined end-to-end (E2E) comprising different network segments such as access, core and transport. In the case of 5G, the *3rd Generation Partnership Project* (*3GPP*) defines an overarching 3GPP Management System that interacts with different domain-specific controllers for each of those segments[1]. The overall slice is operated as a network slice instance, composed by network slice subnet instances (NSSIs), one per network domain or segment.

The slices are requested to the 3GPP Management System by means of templates that specify their concrete characteristics, reflecting the requirements of each vertical customer. The GSMA is defining a generic blueprint with the aim of being used by any vertical customer to specify its needs. Such a blueprint is known as Generic network Slice Template (GST)[2], containing 36 parameters on its latest version. The vertical customer, when specifying the values for this generic blueprint, forms what is known as NEtwork Slice Type (NEST), that is the actual input to the 3GPP Management System. From an operational point of view, the telecom provider supports both Standardized- and Private-NEST[1]. The former make reference to NEST elaborated in a normalized way by distinct organization (e.g., 5G-ACIA for Industry 4.0), while the latter are specific particularization created by the telecom provider. In any case, the vertical customer selects a specific template for slice instantiation.

When looking at the transport network segment, the overall NEST has to be processed in order to extract requirements that directly apply to the creation and instantiation of the transport slice. This consists of identifying parameters that either have a direct or indirect impact on the selection of transport resources chosen to form the E2E slice connectivity. Some attributes can be directly translated into transport network requirements, such as "slice throughput", while others have indirect implications, as for instance attributes like the "availability". An analysis of the type of parameters affecting the instantiation of a transport slice can be found in[3]. Thus, from the perspective of the transport network control entities, it is essential to account on mechanisms for assisting on such translation, resulting on the proper instantiation of the transport network slice.

Targeting this goal, the *GST And Network Slice Operator* (GANSO) framework is proposed in previous research[4] as a transport network slice provisioning solution for the transport segment, leveraging on the concept of GSTs. This work departs from the

initial implementation of GANSO framework by extending it to support the availability attribute during the implementation of the network slices. Section 2 introduces background concepts and the related work. Section 3 introduces the GANSO framework, focusing on the proposed extensions to support availability requirements in network slices, which are then validated through experimental results in Section 4. Finally, concluding remarks are provided in Section 5, pointing out future lines of work.

## 2 | BACKGROUND AND RELATED WORK

### 2.1 | Network Slicing of Transport Network

Network slicing is a concept introduced by the 3GPP organization's Release 15 for the standardization of 5G, which development has continued in successive releases as a critical concept tight to the 5G system. The physical infrastructure is virtualized such that its resources can be distributed and allocated among logical independent networks that simultaneously serve different customers with specific requirements. This concept is supported by programmable transport networks where different network services can be accommodated and coexist simultaneously. Such design is being leveraged by several SDN/NFV-based E2E network slicing solutions. 5Growth[5] is an example of such approach that aims to bring the network slicing paradigm into SDN/NFV-based mobile transport networks by provisioning and managing slices tailored to specific requirements. A similar work[6] proposes a control, management, and orchestration platform that forwards requests to the respective controllers that implement the data path on the underlying SDN-switches. Other works[7,8] also leverage on the usage of SDN mechanisms to configure the data plane of network slices. Finally, SliMANO framework[9] aims at automating the instantiation of E2E network slices over the different network segments. To the best of our knowledge, none of the previous works support *availability* as a configurable parameter for the network slices.

### 2.2 | Network Availability and Reliability at the Transport Network

Path diversity is used to protect traffic delivery in the events of link or node failure. This is typically done by identifying shared risk group at resource level, computing disjoint routes to enforce protection. Alternatively, replication mechanisms can be considered for increasing reliability on transport networks. In this way, possible failures or packet drops are uncoupled and the probability of receiving a packet successfully increases. Under this category, an important effort has been done towards the standardization of these mechanisms. First, at the IEEE 802.1 Time Sensitive Networking (TSN) group, the standard IEEE 802.1CB, also known as Frame Replication and Elimination for Reliability (FRER), was completed in 2017[10]. However, it can only be applied to IEEE 802 link technologies. As a consequence, an IP layer solution, complementary to FRER, is developed at the DetNet WG of the IETF. The service sub-layer of the DetNet architecture (specified in RFC8655[11]) is used to provide DetNet service protection (e.g., by the Packet Replication and Elimination Functions (PREF)) and reordering of packets.

## 3 | GANSO: GST AND NETWORK SLICE OPERATOR

The *GST And Network Slice Operator* or *GANSO* is a proof-of-concept framework developed as a means to automate the creation, deployment and instantiation of network slices over Software Defined Networking (SDN) infrastructures.

### 3.1 | GANSO Framework Overview

Due to the flexibility and the degree of network customization provided by SDN, GANSO has been developed to instantiate network slices over architectures that follow this networking paradigm. As such, it exploits many features enabled by SDN, like reliable and fast network programmability.

As depicted in Figure 1, GANSO is provided as an application residing on the SDN Application Layer. The creation and instantiation of a new network slice is triggered by the user by issuing a request towards the GANSO application (step 1). Whenever a new request is received, GANSO provides a GST form to be filled by the user (step 2) and expects a NEST to be returned with the desired network slice requirements (step 3). Upon receiving the NEST, GANSO computes the required network slice configurations (step 4) and triggers the instantiation of the requested network slice towards an SDN controller (step 5), which in turn implements it in the underlying forwarding devices (step 6).

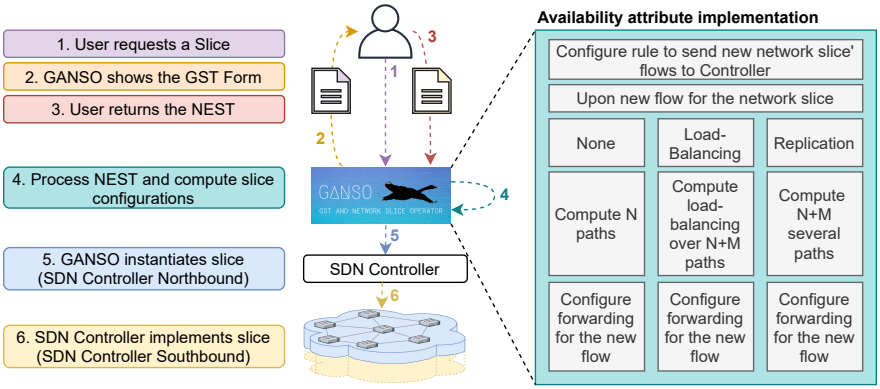


**FIGURE 1** GANSO Overview, including Workflow and Availability Attribute Implementation

Throughout this process GANSO executes a set of internal procedures. First, it reads the values of the attributes configured in the NEST, mapping them into an XML file that contains the information of the network slice and that is subsequently stored. Alternatively, GANSO allows users to directly upload their own NEST XML if these are properly formatted. The NEST XMLs are then translated into JSON format and processed such that the corresponding network slice is instantiated. Currently, two parameters are supported while creating the network slices in GANSO: *(i) User Data Access*; and *(ii) Rate Limit*. While the former defines the level of connectivity allowed within the network slice (e.g., Internet, local edge, etc), the latter defines the maximum throughput, either in downlink or uplink, allowed to a given device (i.e., UE). These two parameters result from mapping five GST attributes, namely *User Data Access*, *Maximum Downlink Throughput per UE*, *Maximum Uplink Throughput per UE*, *Guaranteed Downlink Throughput per UE* and *Guaranteed Uplink Throughput per UE*.

Additional details about GANSO framework, bootstrap and communication mechanisms, and implementation of the aforementioned parameters can be found in previous work[4].

## 3.2 | Extensions to Support Availability Attribute in GST

The GST proposes an optional character attribute (i.e., an attribute that characterizes a slice and is independent of the Network Slice Customer and the Network Slice Provider) to define the communication service availability of a network slice. This attribute specifies the time the E2E communication service is delivered according to an agreed QoS, in relation to the time the system is expected to deliver the E2E service according to the specification in a specific area[2].

Although GST provides typical and exemplary classes with different availability requirements such as low (<90%), medium (90-95%), high (>95-99.999%), very high (>99.999%), the way they are implemented is open to the underlying controller. In GANSO, availability is achieved by employing different path redundancy models.

### 3.2.1 | Availability Estimation

The availability of the network slice at the transport network (i.e., $A_{TS}$) can be estimated by using *N+M* and *N:M* redundancy models, where *N* defines the number of disjoint paths needed to serve the workload without protection and *M* the number of paths that can fail. Additionally, a path is composed by a variable number of *L* switches, where if one fails the whole path fails. As such, the availability of the network slice is given by:

$$A_{TS} = \sum_{k=0}^{M} \binom{N+M}{N+k} A_{path}^{N+k} \cdot (1 - A_{path})^{M-k} \quad , \quad N > 0, M >= 0 \tag{1}$$

$$A_{path} = \prod_{y=0}^{L} A_{switch_y} \quad , \quad Z > 0 \tag{2}$$

Both *N+M* and *N:M* redundancy models follow the previous estimation, reflecting the availability with respect to the resources provided by the transport network.

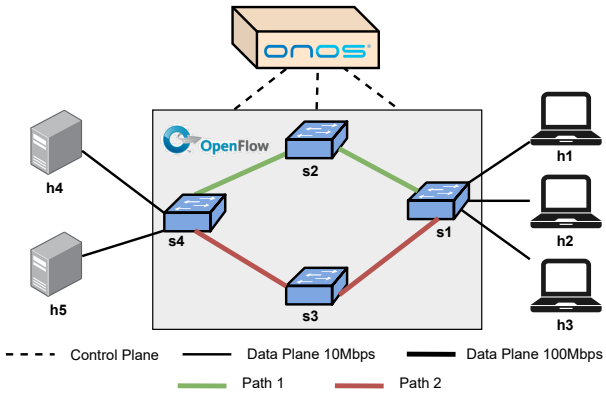


**FIGURE 2** Experiment Topology

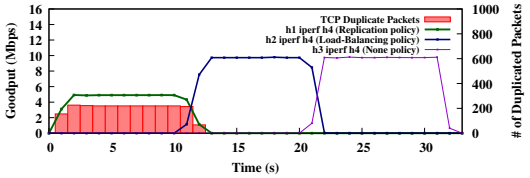


**FIGURE 3** Experiment 1: Baseline Performance

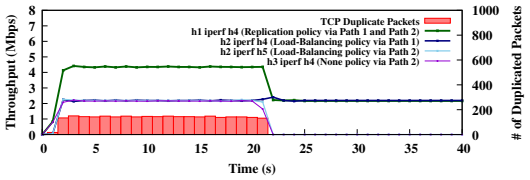


**FIGURE 4** Experiment 2: Resilience against Failures

### 3.2.2 | Attribute Implementation

Upon estimating the availability paths that meet the requested parameter, GANSO implements the paths in the transport network using different policies at flow level. Each policy targets fault tolerance in different degrees for the entire network traffic handled by the network slice, providing the required protection so that there is no QoS degradation in case one path fails or misbehaves.

First, **flow replication policy** forwards all network slice flows through $N + M$ disjoint paths, improving the resilience against failures. Second, **flow load balancing policy** distributes the network slice flows between $N + M$ disjoint paths, improving reliability, redundancy, and performance. Different load-balancing algorithms can be employed, such as (weighted) round robin, (weighted) least connection, (weighted) response time, source/destination IP hash, etc. Third, **non-redundant flow delivery policy** forwards network slice flows through the same path, without the implementation of additional redundancy mechanisms.

Note that, fail-recover mechanisms are out of the scope of this work, since the transport network is assumed to implement them as part of the baseline operation[12]. However, detection of failures and establishment of alternatives paths might not be instantaneously and, therefore, the proposed policies aim to add another layer of resilience for the network slices.

## 4 | VALIDATION RESULTS

This section provides an assessment of the proposed extensions over the GANSO framework to support the availability GST attribute. To do so, a proof-of-concept is implemented and tested over Mininet, as the emulator of an SDN network topology, and using ONOS controller, as the SDN controller managing the network. A set of experiments are conducted to showcase and validate the correct implementation of the availability attribute as well as to evaluate the trade-off of each option.

### 4.1 | Testbed Setup and Implementation

The validation presented in this section has been carried over an emulated network (as depicted in Figure 2), composed by three clients (i.e., $h_1$, $h_2$, $h_3$), two servers (i.e., $h_4$, $h_5$), and a transport network composed by two disjoint paths. The switches (i.e., $s_1$, $s_2$, $s_3$, $s_4$) composing the transport network are OpenFlow-compliant[13] instantiated via Open vSwitch (OVS) and controlled by ONOS controller. Hosts (i.e., clients and servers) are connected through virtual links configured with a bandwidth of 10 Mbps, while the transport network is interconnected through virtual links configured with a bandwidth of 100 Mbps. The E2E communication service is assumed to be delivered according to an agreed QoS in order to isolate the availability attribute. For example, QoS can be ensured through the implementation of other GST attributes, such as the *Guaranteed Uplink / Downlink Throughput per UE* already implemented in GANSO[4].

GANSO takes advantage of the *Northbound API* defined by ONOS to directly instruct the creation and instantiation of network slices. Therefore, a request for a network slice is interpreted by GANSO and then translated into JSON format to be requested via the ONOS REST API. Finally, ONOS configures the flow tables of each switch using OpenFlow protocol.

## 4.2 | Experiment 1: Baseline Performance for Availability Policies

This experiment analyses the baseline performance of the proposed availability policies and the consequences of implementing each one in the total goodput. Each client (i.e., $h_1, h_2, h_3$) sequentially generates a TCP traffic flow using the maximum capacity of their local virtual link (i.e., 10 Mbps) towards server $h_4$ during approximately 10s. Moreover, the network slices of clients $h_1$, $h_2$ and $h_3$ are configured with a reliability of *Very High* (i.e., replication), *High* (i.e., load-balancing following an source/destination IP hash) and *Low* (i.e., *None*), respectively. The obtained results are presented in Figure 3.

All traffic generated by client $h_1$ is replicated in the transport network. Unless aggregation points in the network can identify duplicated packets, all packets are forwarded to the next hop as if they were actually different packets. Vanilla OpenFlow Switches cannot detect duplicated packets and forwarding packets to the Controller for additional processing is not efficient. Therefore, the destination endpoint (i.e., server $h_4$) receives twice the traffic destined to it, making the total goodput between $h_1$ and $h_4$ to be halved to approximately 4.90 Mbps (green line in Figure 3). While for TCP traffic duplicated packets are identified and discarded by the TCP/IP stack (as shown in Figure 3), for UDP traffic the applications need to handle duplicated packets themselves. Notwithstanding, note that the virtual links towards the hosts are being saturated, which otherwise would allow a goodput of approximately 10 Mbps at the cost of doubling the total throughput. Support for solutions like FRER or PREF have the potential to mitigate this issue by dropping duplicated packets in the aggregation points, but their support in OpenFlow is currently lacking and novel extensions are required.

As for the *load-balancing* and *none* policies, the previous issue is not verified since only one copy of each packet is being sent over the transport network. As a result, the server receives packets at the same rate (i.e., both with a goodput of approximately 9.75 Mbps) as they are issued by the clients (respectively, blue and purple lines in Figure 3).

## 4.3 | Experiment 2: Resilience against Failures

This experiment targets a further analysis of the trade-offs of each policy regarding their resilience against failures. Each client generates a TCP traffic flow of 2 Mbps towards server $h_4$. Client $h_2$ generates a second TCP traffic flow towards server $h_5$. Network slices follow the same configuration as in the previous experiment. Finally, a failure of switch $s_3$ is triggered at time 20s, making *path 2* to be unavailable. Considering the switches switch $s_2$ and switch $s_3$ as the one subject of the availability analysis, even lower individual availability levels (e.g., 95%) can result on high availability levels (i.e., 99.75%) when combined. Clearly, more complex topologies will produce different availability levels. It would be a matter of the GANSO to logically map the expected availability level to the specific topology for supporting it. The obtained results are presented in Figure 4.

Since links are not saturated, the flow of client $h_1$ is received at server $h_4$ with double of the throughput of the remaining flows. Notwithstanding, every flow is reflected in the same goodput of approximately 2 Mbps. When the failure of switch $s_3$ is triggered (i.e., at time 20s):

- the network slice of $h_1$ is able to keep the goodput towards server $h_4$. As packets are being duplicated through both paths, the transport network can still deliver a copy of each packet to their destination.
- the network slice of $h_2$ is only capable of delivering flows sent by *path 1*, since *path 2* becomes unavailable. Load-balancing solutions at the packet level would allow both flows to be delivered with minimal impact.
- the network slice of $h_3$ uses the same path to deliver all its flows. As such, upon a failure in this given path all the flows are interrupted and not delivered to their destination.

Even though the transport network implements mechanisms to recover from failures, the delay to detect the failure and to enforce corrective actions in the network are likely to impact the availability of the network slice. This is especially critical in ultra-reliable low latency communication slice/service types, which availability up to six 9's is required by industrial use cases [14].

# 5 | CONCLUSIONS

The instantiation of E2E services, comprising virtual functions distributed across multiple data centers (i.e., edge, centralized or public clouds), requires network slicing to be applied into different network segments, including the transport network interconnecting them. This paper extends the GANSO framework, developed to automate the creation of customized transport slices, adding the capability of mapping the requirement of availability contained in a slice template. Different policies are evaluated

to extract guidelines for attribute mapping. Three different policies are considered, namely flow replication, flow load balancing and non-redundant flows. The replication case results to be the more robust at the cost of requiring increasing the capacity allocated to the slice (as much as the number of flow replicas generated) to guarantee the slice throughput, which can compromise the scalability of the approach. The load balancing schema can maintain the negotiated availability levels at the cost of reducing the slice throughput, unless complementary mechanisms such as path protection are in place (i.e., to enable a new path setup on failure events). Finally, the non-redundant approach clearly fails on guaranteeing high levels of availability.

Future work will focus on the support of the remaining GST attributes and a fully network slice lifecycle management. These extensions will pave the way for the implementation of intelligent and automated transport slice management on top of GANSO.

## ACKNOWLEDGMENTS

This work has been (partially) funded by H2020 EU/TW 5G-DIVE (Grant 859881) and H2020 5Growth (Grant 856709). It has been also funded by the Spanish State Research Agency (TRUE5G project, PID2019-108713RB-C52PID2019-108713RB-C52 / AEI / 10.13039/501100011033).

## References

1. 3GPP . Management and orchestration; Provisioning. Technical Specification (TS) 28.531, ; : 2020. Version 16.5.0.
2. GSM Association (GSMA) . Official Document NG.116-Generic network Slice Template, Version 4.0.; 2020.
3. Contreras LM, Homma S, Ordonez-Lucena JA. IETF Network Slice Use Cases and Attributes for Northbound Interface of IETF Network Slice Controllers. Internet-Draft draft-contreras-teas-slice-nbi-04, Internet Engineering Task Force; : 2021.
4. Infiesta JT, Guimarães C, Contreras LM, de la Oliva A. GANSO: Automate Network Slicing at the Transport Network Interconnecting the Edge. *IEEE Conference on Network Function Virtualization and Software Defined Networks* 2020.
5. Li X, Garcia-Saavedra A, Perez XC, et al. 5Growth: An End-to-End Service Platform for Automated Deployment and Management of Vertical Services over 5G Networks. *IEEE Communications Magazine* 2021; 1(1). (*in-press*).
6. K. Chartsias P, Amiras A, Plevrakis I, et al. SDN/NFV-based end to end network slicing for 5G multi-tenant networks. *2017 European Conference on Networks and Communications* 2017.
7. Afolabi I, Bagaa M, Taleb T, Flinck H. End-to-end network slicing enabled through network function virtualization. *2017 IEEE Conference on Standards for Communications and Networking* 2017.
8. Giatsios D, Choumas K, Flegkas P, Korakis T, Cruelles JJA, Mur DC. Design and Evaluation of a Hierarchical SDN Control Plane for 5G Transport Networks. *IEEE International Conference on Communications* 2019.
9. Meneses F, Fernandes M, Corujo D, Aguiar RL. SliMANO: An Expandable Framework for the Management and Orchestration of End-to-end Network Slices. *2019 IEEE 8th International Conference on Cloud Networking* 2019.
10. Finn N. Introduction to Time-Sensitive Networking. *IEEE Communications Standards Magazine* 2018; 2(2).
11. Finn N, Thubert P, Varga B, Farkas J. Deterministic Networking Architecture. RFC 8655; 2019.
12. Cominardi L, Gonzalez-Diaz S, Oliva d. lA, Bernardos CJ. Adaptive Telemetry for Software-Defined Mobile Networks. *Journal of Network and Systems Management* 2020.
13. Open Networking Foundation (ONF) . OpenFlow Switch Specification, Version 1.5.1.; 2015.
14. 5G-ACIA - 5G Alliance for Connected Industries and Automation . 5G for Connected Industries and Automation (White Paper - Second Edition).; 2019.