Design and evaluation of a hierarchical SDN control plane for 5G transport networks

Dimitris Giatsios, Kostas Choumas,
Paris Flegkas, Thanasis Korakis
School of Electrical and Computer Engineering
University of Thessaly

Joan Josep Aleixendri Cruelles, Daniel Camps Mur
Mobile and Wireless Internet Group
i2CAT Foundation

***Abstract*—Software-defined networking is at the root of future 5G network design. Among others, it allows for automated network reconfiguration and network slicing support. In this paper we present an implementation of a hierarchical control plane in the transport network architecture envisioned by the 5G-PICTURE project. We analyze the implementation of the slicing mechanism at the network edge and the end-to-end path establishment procedure, which involves interactions within a hierarchy of controllers. We also evaluate the latency performance of our control plane solution, by means of testbed experimentation.**

I. Introduction

Transport networks for 5G cellular services are expected to accommodate multiple tenants through virtualization and, at the same time, be able to satisfy stringent QoS requirements, such as those related to the transport of fronthaul and backhaul interfaces for supporting centralized and distributed radio access network (RAN) deployments. The increased fluctuations in demand patterns due to network densification call for flexible transport architectures based on software defined networking (SDN) [1].

5G-PICTURE project [2] is a European project under the umbrella of the 5G-PPP initiative, focusing on challenges in the design of 5G transport networks. Its scope is to provide a flexible transport network consisting of multiple technologies, including wireless and optical segments, which will facilitate the creation of virtual and isolated overlay networks belonging to different tenants. Its ambition is to develop an architecture where hardware and software components are disaggregated across the optical, wireless and compute/storage domains.

In this paper, we present a control plane architecture for multi-tenant multi-domain 5G transport networks, as envisioned by this project. Figure 1 gives an overview of this architecture. To the best of our knowledge, our proposed solution is the first hierarchical SDN control plane supporting virtualization in a multi domain environment. We evaluate scalability of our proposed architecture experimentally, measuring path provisioning latency for different number of domains, and find it to be well below 1 second for most scenarios of practical interest for 5G transport networks.

The remainder of this paper is organized as follows. In section II, we briefly summarize the dataplane abstraction in our architecture. In section III, we explain the structure of our control plane implementation, while in section IV we analyze the interactions among different controllers in the hierarchy. In section V we present an evaluation of the latency of our

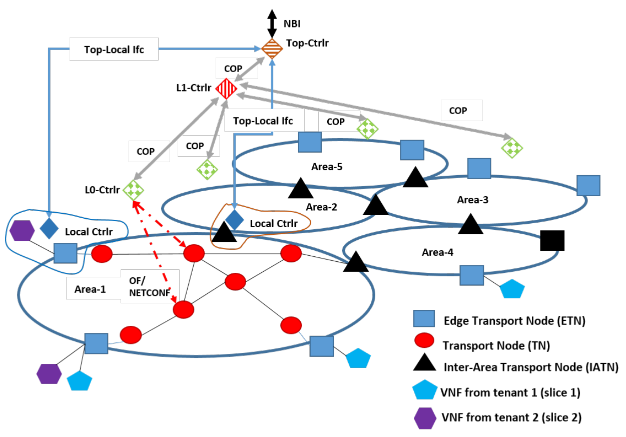


Fig. 1: Overview of the proposed transport network architecture and associated control plane hierarchy.

control plane solution, obtained via testbed experimentation. In section VI we mention the most relevant work, with emphasis in standardization efforts. Section VII concludes the paper.

II. 5G-PICTURE Dataplane Abstraction

A. Layer 2 Overlay to support virtualization

In terms of the dataplane abstraction, the 5G-PICTURE transport network can be seen as a virtual network, where each tenant brings its *virtual entities*, namely Virtual Network Functions (VNFs) and virtual DataPaths (vDPs). VNFs are the end points and the vDPs are the tenant controlled datapaths.

Each virtual network is a slice of the transport network infrastructure. A transport slice is composed of virtual Layer 2 segments where virtual entities (VNFs/vDPs) are attached. A virtual Layer 2 segment emulates a broadcast domain and is identified by a *Layer 2 Segment Identifier (L2SID)*. L2SIDs are unique system wide, meaning that L2SIDs cannot be reused within or across slices. VNFs are identified within a slice by a Media Access Control (MAC) address scoped to a single Layer 2 segment. vDPs contain custom network control logic defined by the tenant, for instance they may correspond to a virtual switch. Unlike a VNF, a vDP may have several interfaces, each one connected to a different virtual Layer 2 segment. Each interface of a vDP is identified again by a MAC address scoped to the L2SID where it is attached. Note that, in our architecture, Layer 3 forwarding between different virtual Layer 2 segments is the responsibility of the tenant and can take place at its vDPs, that is, only a Layer 2 overlay is provided to the tenants as a service.

B. *SDN based multi-domain transport underlay*

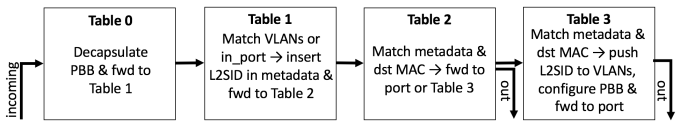
The 5G-PICTURE architecture is deployed over a transport network infrastructure consisting of domains, often technology-specific, to which we also refer as areas. It is composed of three main functions in the dataplane, namely the Transport Nodes (TNs), depicted as circles in Figure 1; the Edge Transport Nodes (ETNs), depicted as squares in Figure 1; and the Inter-Area Transport Nodes (IATNs), depicted as triangles in Figure 1.

All transport nodes embed one major function, the *Forwarding Information Base (FIB)*. FIB is in charge of forwarding packets between VNFs/vDPs, which are either colocated in a single ETN or bounded to different ETNs. In the second case, packets are inserted into pre-instantiated transport *tunnels*, implemented with use of encapsulation. Traffic from multiple slices can be combined into a single tunnel. The *Transport network Adaptation Function (TAF)* is responsible for pushing (or popping) the corresponding transport header before (or after) injecting the packets into the transport network, whereby the transport header signals at least three major pieces of information: i) the address of the destination ETN (which might be located in the same or in another area), ii) the transport slice ID and iii) the *tunnel ID*. Tunnel IDs determine how encapsulated packets are routed throughout the underlay. They are area specific, meaning they generally change from area to area, assuming that the tunnel spans multiple areas. Transport slice IDs can be used to apply differentiated policies depending on the slice, such as binding slice traffic to a particular Quality of Service (Qos) class. Only ETNs/IATNs feature a TAF, corresponding to the transport technology used in the area where the tunnel is located.

The TNs act as regular tenant-agnostic transport nodes, and could be instantiated by different technologies, such as wireless or Time Shared Optical Networks (TSON). The TN datapath forwards incoming packets according to their tunnel ID, and optionally applies differentiated policies according to the transport slice ID. ETNs and IATNs are interconnected through transport tunnels, which are based on the forwarding services of the TNs.

An ETN would typically be implemented as a software datapath in a network hypervisor, and holds all tenant related state thus enabling virtualization/multi-tenancy over the 5G-PICTURE transport network. The main function of each ETN is to host virtual entities from several tenants, as well as to offer a datapath abstraction that connects the hosted virtual entities to the transport network. ETN implements FIB and TAF. In Figure 2 we see the flow table structure of an ETN, following the OpenFlow v1.3 specifications and using Provider Backbone Bridging (PBB) as the transport encapsulation technology.

Finally, an IATN stitches pre-defined connections between two or more areas. Each IATN is in charge of collecting all incoming packets from one area and forwarding them to the appropriate tunnels of another area. For each incoming packet, IATN first executes TAF to decide which tunnel will be used for the forwarding of this packet (tunnel ID and possibly even


incoming
Table 0
Decapsulate PBB & fwd to Table 1
Table 1
Match VLANs or in_port → insert L2SID in metadata & fwd to Table 2
Table 2
Match metadata & dst MAC → fwd to port or Table 3
out
Table 3
Match metadata & dst MAC → push L2SID to VLANs, configure PBB & fwd to port
out
Fig. 2: ETN as a datapath.

the transport technology could be different between adjacent areas), and then changes its tunnel ID and forwards it through the appropriate interface completing the FIB action.

III. Hierarchical Control Plane Architecture

The control plane of 5G-PICTURE transport network is designed based on the principles of full address space virtualization and scalability. It is composed of a hierarchy of logical controllers, as illustrated in Figure 1. The top level controller, referred to as the *Top controller*, is responsible for provisioning per tenant slices and orchestrating the required connectivity across different areas (e.g. optical transport domain, wireless transport domain). The *Level-0 controller*, also referred to as Area controller, is responsible for the provisioning and maintenance of transport tunnels between ETNs and IATNs of a given area; a Level-0 controller operates at the level of individual network elements. A set of Level-0 controllers, which are technology-specific, are logically organized under a Level-1 controller. The latter is technology-agnostic, is in charge of maintaining connectivity between the corresponding areas, and operates with a higher level of abstraction, namely maintains state at the area level. Finally, ETNs and IATNs, which lie at the edges of transport areas, are directly controlled by *Local Agents* which are the glue between their datapaths and the Top controller with which they interact. TNs are directly controlled by a Level-0 controller, and do not feature such Local Agents.

The main principle adopted in the design of the higher layers of the 5G-PICTURE control plane is the separation of responsibilities between the L1 controller and the Top controller, whereby the Top controller interfaces with the ETNs, e.g. in order to provision a new VNF, and with the IATNs in order to stitch domains. On the other hand the L1 controller's job is to act as an aggregator of L0 controllers, thus interacting only with these controllers, which end up programming the TNs of each domain.

A local ETN agent's main responsibility is to maintain mappings from virtual entity addresses to the remote ETNs hosting them, whenever these entities are attached to the same Layer 2 segment as at least one virtual entity hosted locally at the agent's ETN. For these remote ETNs of its interest, it also maintains mappings to the respective tunnel IDs that must be used for forwarding encapsulated traffic towards them. Here we note that in cases where a tunnel traverses multiple areas, only the first tunnel ID is stored, the one leading to the first IATN along the route. This is in accordance with abstracting out unnecessary information; the ETN does not have to care about whether the destination ETN lies in the same or another area, in fact it does not know anything about areas. The ETN Local Agent also maintains mappings from local ports to the respective L2SIDs, and from local virtual entity addresses to local ports. These mappings are summarized in Table I.

A local IATN agent manages a table of mappings, which allows the underlying IATN to forward packets across areas using the appropriate interface, as well as to modify the tunnel ID information, when required. This mapping can be viewed in Table II.

TABLE I: ETN Local Agent mappings

Key	Notes	Value
port	All local ports	L2SID
{L2SID, MAC}	All local virtual interfaces	port
{L2SID, MAC}	Remote virtual interfaces with an L2SID locally present	hosting ETN
destination ETN	ETNs hosting virtual entities attached to at least one L2SID locally present	tunnel ID

TABLE II: IATN Local Agent mapping

Key	Value
{in port, in tunnel ID}	{out port, out tunnel ID}

IV. Inter-controller communications

There are three main interactions taking place among the controllers in the hierarchy. The Top controller communicates with the ETN/IATN Local Agents, the Top controller communicates with the L1 controller, and the L1 controller communicates with L0 controllers.

A. Top controller - Local Agents Interaction

The Top-Local interfaces are different for the Local Agents hosted at ETNs and those hosted at IATNs, as these types of nodes are responsible for different functionalities. Both of them, however, are Representational State Transfer (REST) based interfaces.

In the case of ETNs, the basic resources made available to the Top controller are called *virtual interfaces*, as an abstraction that covers both interfaces directly connected to a VNF, and interfaces of a virtual datapath (vDP) in a tenant's slice topology. These are uniquely identified by the L2SID where they are attached and the corresponding MAC address, scoped at the specific L2SID. Each virtual interface is hosted at one and only one ETN at a given time (migrations are of course allowed), and this information is a property of the virtual interface resource. The other type of managed resource are *tunnels*, also called paths, whose source node is the particular ETN, and destination is some remote ETN. These are identified by the (area-specific) tunnel ID attached to packets of that path leaving the ETN, which in general is a VLAN ID field. They feature at least one associated value, indicating the destination ETN, but they can also optionally feature information about the QoS guarantees provided by that path, if any, thus allowing differentiated tunnel selection decisions for flows of special requirements.

In the case of IATNs, the resources exposed and managed by the Top controller are tunnel mappings between areas attached to the IATN. The tunnels are organized by incoming IATN port, since in our design each IATN port corresponds to an area. They are uniquely identified by this port and the incoming tunnel ID, while the associated properties are the outgoing tunnel ID and the outgoing IATN port. Remember that these translations are required to keep full independence of different areas (domain stitching functionality). So, in fact the IATN Local Agent is a simple thin layer acting as a proxy between the Top controller and the IATN datapath.

Based on the interaction with the local agents of ETNs and IATNs just described, the Top controller is responsible for performing a number of operations. The first one is deployment of VNFs and vDP interfaces at specific ETNs, thereby instantiating a given tenant slice virtual topology step by step. Another operation is to provide the functionality of simple migration of virtual interfaces from one ETN to another. The important thing to realize is that the Top controller makes sure that each ETN is kept up to date about the location of all virtual interface addresses it might need to send packets to. This requires a global view of the location of all virtual interfaces, and could not be addressed locally. Yet another operation is informing the local agents of all tunnel IDs they should be aware about. This comes as the subsequent step after an initial step of actual path establishment, which involves the interaction between the Top controller and L1 controller. This interaction is analyzed in the following subsection.

B. Top - L1 and L1 - L0 controllers interactions

The interactions between Top controller and L1 controller, as well as between L1 controller and L0 controllers, make use of the Control Orchestration Protocol (COP), originally defined in the Strauss project [3]. COP is a REST based protocol, which defines a set of data models to allow REST endpoints to offer network related services. COP has been proposed as a research-oriented transport API, technology and vendor independent, that permits to abstract technology specifics of a given transport domain. It provides a multi-layer hierarchical control plane approach using YANG and RESTconf.

The following two main services are offered by the L1 controller towards the Top controller:

- A topology dissemination service, able to retrieve topologies from individual L0 controllers and then aggregate them into an end to end topology.
- A path provisioning service, able to receive a path provisioning request involving nodes in different areas and resolve it into separate path requests for each of the involved areas. If there are multiple paths between two areas, the L1 controller performs routing at the area level.

In addition, the L0 controller offers the same two services towards the L1 controller:

- A topology dissemination service, in which the L0 controller exports the topology specific to its domain in COP format. We note that the L0 controller might choose to report only a summarized version of its topology, where the only critical information to be exposed to the L1 controller are the nodes connecting to an ETN or an IATN.
- A path provisioning service, whereby the L0 controller receives a request to connect to TNs under its control,

and the L0 controller responds with the corresponding tunnel identifiers.

The COP service-topology data model consists of a list of nodes, representing network devices, and edges, representing network links. In COP, nodes embed multiple edge-ends, representing a port or an interface, and edges refer to the nodes at each side of the link.

Since the L1 controller only interacts with the L0 controllers, not ETNs or IATNs, the COP topology will only report TNs in its list of nodes. However, the Top controller needs to be able to resolve an ETN or IATN into a TN in order to issue a path request to the L1 controller. Since the mapping between ETN/IATNs and TNs is expected to be something fairly static, we opt to manually provision the L1 controller with this information. In particular, we enable an additional REST endpoint in the L1 controller that allows to specify information about IATNs and ETNs connecting to one of the L0 domains under the control of this L1 controller. Then, the L1 controller exposes the IATN/ETN information as a COP edge. This information is sufficient for the Top controller to match ETN/IATNs with TNs and issue a path request.

In particular, the L1 controller maps 5G-XHaul functions into COP topology objects in the following manner:

Node	TN_A
Regular edge	$TN_A : PortA \rightarrow TN_B : PortB$
IATN edge	$IATN_A : PortA \rightarrow IATN_B : PortB$
ETN edge	$ETN_A : PortA$

C. End to end path establishment

The basic operation that requires interactions among all controllers in the hierarchy is that of an end-to-end path establishment between two ETNs that are located in different areas. The time sequence of interactions required for this procedure is summarized in Figure 3 for an example with two adjacent areas.

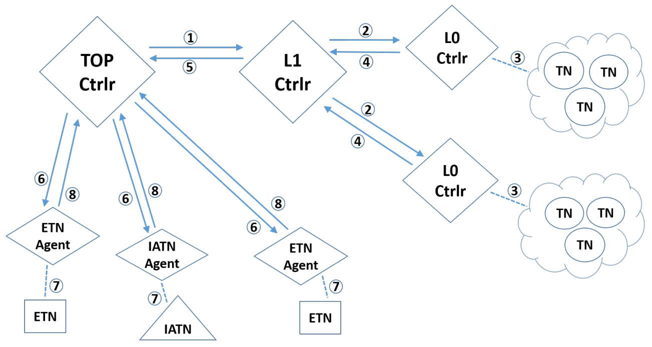


Fig. 3: Interactions among controllers for end to end path establishment.

At step 1, the Top controller sends a request to the L1 controller, using COP/REST for establishing a path between two ETNs. The L1 controller examines the request and finds out that the path must traverse two adjacent areas. It then breaks the original request into two sub-requests, intended for the L0 controllers of these areas. Essentially, these sub-requests, sent in parallel via COP/REST at step 2, are for

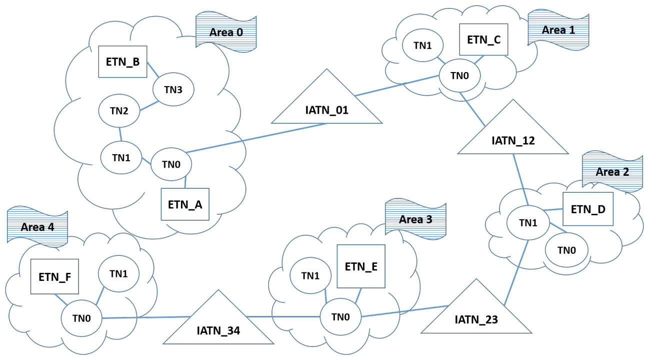


Fig. 4: Experiment setup for latency evaluation.

provisioning of connections between each ETN and the IATN stitching these areas. The L0 controllers build the connections by programming the required flows at their subordinate TNs during step 3, using OpenFlow, NETconf or another standard protocol. At steps 4 and 5, confirmation of path establishment travels from the L0 controllers to the L1 controller, and from there to the Top controller. The procedure is not yet finished, because the ETNs and the IATN need to be instructed as well. Therefore, at step 6 and using the respective REST interfaces, the Top controller informs the ETN Local Agents of the tunnel IDs they must associate to the remote ETN, and the IATN Local Agent of the tunnel translation it must perform. These messages are sent at step 6. At step 7, the Local Agents install the required flows into the underlying datapaths, typically using OpenFlow. Finally, at step 8, the Local Agents send confirmations to the Top controller. After this point, the tunnel is ready for use.

V. Experimental Evaluation of Control Plane Latency

In this section we evaluate the performance of the 5G-PICTURE hierarchical control plane looking at the time required to provision an end-to-end tunnel between two separate ETNs, involving multiple control plane areas. To carry out this evaluation we have conducted repeated experiments at the NITOS testbed [4], where we implemented the topology depicted in Figure 4, with OpenVSwitch being used for the data plane.

The topology consists of five Ethernet based areas connected in a chain by IATNs. The 5G-PICTURE control plane is implemented in a set of Vitual Machines, including the five L0 controllers, the L1 controller, the Top controller, and Local Agents for all ETNs and IATNs in the topology. All VMs are hosted in NITOS physical nodes, and are connected to the same local testbed network. Each area was emulated by a Mininet instance running at the same VM as the respective L0 controller.

Using this setup we carry out a set of measurements to estimate the overall time required to establish an end-to-end tunnel. In this experiment, we examine the establishment of a bidirectional connection, which is very usual in practice, but our solution also supports unidirectional connections.

The overall time can be broken into two sequential components. The first component involves the Top controller's request to the L1 controller for the latter to provision an end-to-end bidirectional path between two given ETNs. The response of the L1 controller contains, among other information, the tunnel IDs to be used in each direction in every area used by the path. The second component involves the Top controller's instructions to the Local Agents of the two ETNs and of the IATNs traversed by the returned path. This component must strictly follow the first one, as the Top controller must instruct the Local Agents of the specific tunnel IDs to be used, and also it must be aware of which IATNs need to be updated. Therefore, we take separate measurements for the two components.

We are particularly interested in examining how our solution behaves for an increasing number of areas between the two ETNs, as we want to study the scalability of our solution. Our topology was designed to allow this. By keeping one of the two ETNs fixed, specifically ETN_A in the first area, and changing the other ETN to be ETN_B, ETN_C, ETN_D, ETN_E and ETN_F, we essentially examine the scenarios where the two ETNs are in the same area, or lie at the edges of an increasing number of areas, from 2 to 5, connected in chain.

For each of the five scenarios, we execute 25 different tests. The resulting cumulative distribution functions (CDFs) of the latencies for the two components of path establishment are reported respectively in Figure 5 and Figure 6. For both components, we observe a linear increase of the latency with the number of areas between the ETNs. This makes sense. The L1 controller must partition the request and distribute it to a respective number of L0 controllers, which in turn need some time to program the TNs under their control. Increasing the number of areas also means the Top controller must instruct an increasing number of IATNs to install rules for tunnel ID translations. For each added area, the increase is approximately 50ms for the first component and 70ms for the second, i.e. a total of approximately 120ms on the average.

We have seen how the path establishment latency scales with the increasing number of areas. Yet another latency scaling result of interest is how the size of each area affects the latency. Specifically, how it is affected by the number of TNs within each area along the end-to-end path. We have examined this scenario using a fixed number of areas. Our results indicated that there was no significant fluctuation in the latency of a L0 controller for provisioning the path in its area with respect to the number of TNs involved. This is because the L0 controller has all the topology information allocated in its memory and can instantiate the paths using multiple threads.

Note that in the experimental evaluation above, all controllers were hosted at VMs connected to the same local network, therefore propagation delays were quite low. The TNs were also in the same local network as the respective L0 controllers. Therefore, for implementations where the architectural elements are distributed over a wide area, we should add the respective propagation delays to the above results, wherever they apply. These can be easily measured with the ping utility. As an example, we mention preliminary experimental results (without scaling) we obtained in [5], where areas and controllers were split between two sites, located at University of Thessaly (UTH) and i2CAT. Specifically, one area per site was deployed, the Top controller was running at UTH, the L1 and L0 controllers at i2CAT, the IATN Local Agent at UTH,

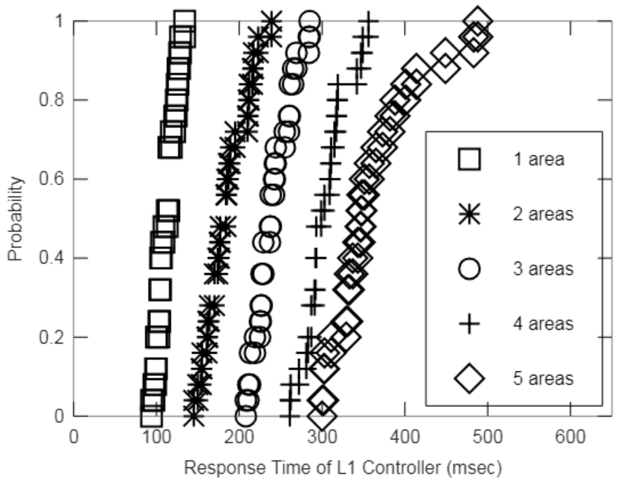


Fig. 5: Scaling of L1 controller response latency CDFs with increasing number of areas in provisioned path.

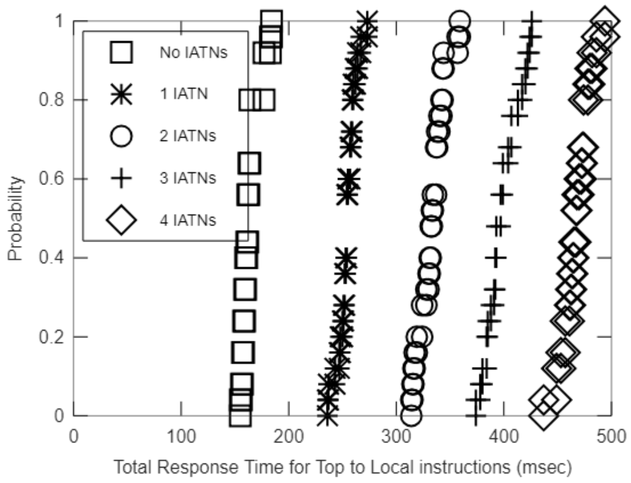


Fig. 6: Scaling of total response time of Local Agents of ETNs and IATNs in provisioned path.

and the ETN Agents at the site of the area where they were attached to. Through ping, an additional average round trip latency of 50 ms was measured between the two sites, which aggravated the measured control plane latency. The CDF of the overall path provisioning time measured in this experiment can be seen in Figure 7.

Summarizing, the key observation to make is that the overall connection establishment time is below 1 second, which is negligible compared to service provisioning time in current IP networks, which often require manual intervention. We also point out that, for 5G transport networks, ETNs are expected to be separated by no more than a few areas, for instance due to data plane latency concerns. For two areas, expected latency is less than 500ms. Finally, we believe we can significantly reduce the path establishment latency through optimization of the software implementations of our controllers, by further employing multi-threading wherever possible, which is part of our ongoing work.

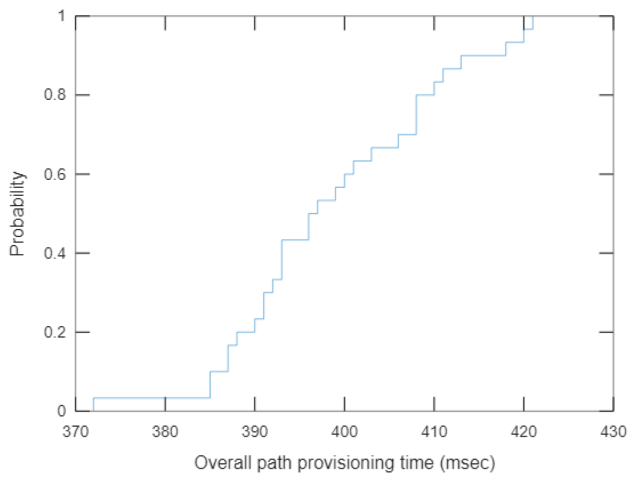


Fig. 7: CDF of overall delay for path provisioning in two-area experiment with areas and controllers distributed at two remote sites.

VI. Related Work

In this section we describe related work in the areas of network virtualization, SDN architectures applied to transport networks, and compute and network integration.

Network virtualization solutions based on overlays have seen a large adoption in data-center environments, where a common compute infrastructure, connected with an underlay Layer 2 or Layer 3 network, is used to support multi-tenant services. The most representative solution is Virtual Extensible LAN (VXLAN), which is an overlay technology providing virtual Layer 2 networks over a Layer 3 infrastructure, tunneling VXLAN packets over UDP/IP. Initial VXLAN standards [6] implemented an in-band flood-and-learn scheme, like a traditional Ethernet bridge, without an explicit control plane. In particular, each virtual Layer 2 network is associated with a multicast group, and all servers hosting VMs attached to that virtual network are subscribed to that group. A drawback of flood-and-learn schemes is that they introduce too much broadcast traffic, due to protocols like ARP. To enhance VXLAN scalability, the Multiprotocol Border Gateway Protocol (MP-BGP) Ethernet Virtual Private Network (EVPN) has been defined by the IETF as a standards-complying control plane for VXLAN. In EVPN, MP-BGP is used to disseminate the bindings between the MAC addresses of a particular virtual Layer 2 network and the corresponding IP end-point of the underlay. This is in contrast to the 5G-PICTURE virtualization solution where a Layer 2 overlay is used in coordination with an SDN controlled Layer 2 underlay.

Multi-domain SDN control planes for transport networks have also been a topic of intense research in the last years. The Control Orchestration Protocol (COP) first proposed in [7] was one of the first approaches to propose a generic interface to allow inter-controller coordination in hierarchical SDN control planes. The work in COP evolved into the definition of the ONF's Transport API (T-API) [8], which offers a true multi-vendor interface, with extensions to support optical and microwave equipment. The inter-controller architecture proposed in this paper is based on extensions of COP, but can be easily ported to T-API. It is worth noticing though that neither COP nor T-API natively support network virtualization, as it is done by 5G-PICTURE.

Finally, some initial work on SDN/NFV integration for integrated network and compute infrastructures is reported in [9] for mobile networks, and in [10] for 5G slicing. Although this paper has not explicitly discussed network-compute integration, our proposed solution can be integrated in an NFV MANO system, by integrating an ETN endpoint in the hypervisor of each compute node, and integrating the north-bound interface of the 5G-PICTURE Top controller with the VIM network controller (e.g. neutron for OpenStack).

VII. Conclusion

We presented an implementation for the basic functionalities of a hierarchical control plane in the 5G-PICTURE architecture for 5G transport networks. Our design targets in providing scalability and efficiency. Future extensions of our work will focus on QoS support at the transport nodes and its implications in designing efficient policies.

Acknowledgment

This work has been financially supported by the EU Horizon 2020 project 5G-PICTURE under grant agreement No 762057. The European Union and its agencies are not liable or otherwise responsible for the contents of this document; its content reflects the view of its authors only.

References

[1] "5GPPP architecture working group, View on 5G architecture," https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-For-public-consultation.pdf.

[2] "5G-PICTURE project," https://www.5g-picture-project.eu/.

[3] "Straus project, deliverable 3.2: Preliminary report on the building blocks for the network virtualization, openflow control and sdn orchestrator," June 2015.

[4] "NITOS testbed," https://nitlab.inf.uth.gr/NITlab/nitos.

[5] "5G-Xhaul project, deliverable 3.3: 5g-xhaul algorithms and services design and evaluation," August 2018.

[6] M. M. et al., " Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," Internet Requests for Comments, RFC Editor, RFC 7348, August 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7348.txt

[7] R. Muñoz, A. Mayoral, R. Vilalta, R. Casellas, R. Martínez, and V. López, "The need for a transport API in 5G networks: The control orchestration protocol," in *Optical Fiber Communications Conference and Exhibition (OFC), 2016*. IEEE, 2016, pp. 1–3.

[8] C. Janz, L. Ong, K. Sethuraman, and V. Shukla, "Emerging transport sdn architecture and use cases," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 116–121, 2016.

[9] V.-G. Nguyen, A. Brunstrom, K.-J. Grinnemo, and J. Taheri, "Sdn/nfv-based mobile packet core network architectures: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1567–1602, 2017.

[10] A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, and V. López, "Cascading of tenant sdn and cloud controllers for 5g network slicing using transport api and openstack api," in *Optical Fiber Communications Conference and Exhibition (OFC), 2017*. IEEE, 2017, pp. 1–3.