

Experimental Evaluation of Hierarchical Control over Multi-Domain Wireless/Optical Networks

Josep Mangues-Bafalluy*, José Núñez-Martínez*, Ramon Casellas*, Arturo Mayoral*, Jorge Baranda*, Josep X. Salvat†, Andres Garcia-Saavedra†, Ricard Vilalta*, Iñaki Pascual*, Xi Li†, Ricardo Martínez*, Raul Muñoz*

* Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA) - Castelldefels (Barcelona), Spain

† NEC Labs Europe - Heidelberg, Germany

Abstract—5G-Crosshaul aims at reducing network costs by designing an integrated transport (fronthaul/backhaul) network able to fulfill 5G requirements. Such transport networks will consist of heterogeneous technologies that need end-to-end orchestration. In this paper, we evaluate a hierarchical resource management framework for multi-domain wireless/optical networks. More specifically, we deploy a hierarchical 5G-Crosshaul Control Infrastructure (XCI) where child controllers deal with the specificities of each technology whilst the parent controller is in charge of offering to a resource management application (RMA) the appropriate abstraction level and an end-to-end view. To understand the end-to-end behavior related with service setup, we evaluate each network segment (wireless and optical), each plane (application and control planes), and each layer of the hierarchy inside the XCI. In particular, we evaluate the aggregated path setup time (in the order of seconds) as well as each component (wireless domain contributes with tens of ms and multi-layer optical network with hundreds of ms per layer for a total in the order of seconds). Path restoration results reveal the importance of leveraging control of child controllers when requiring fast response to unexpected data plane events, since an important part of the setup delay observed is due to the RMA-parent-child controller interaction and sequential message handling.

I. INTRODUCTION

Future 5G networks will handle multiple End-to-End (E2E) services with diverse requirements [1], whilst offering high deployment flexibility (e.g., various network densities) at a low-cost. 5G-Crosshaul aims at reducing network costs by integrating fronthaul (FH) and backhaul (BH) through common control and data planes. Yet, the variety of requirements and deployment scenarios will also result in a variety of transport technologies (optical and wireless) that must be orchestrated on an E2E basis. Such common control plane infrastructure is referred to as 5G-Crosshaul Control Infrastructure (XCI) [2]. On top of that, network management applications are in charge of efficiently using the network resources exposed by the XCI. Hierarchical control architectures have been proposed to split the E2E orchestration problem into smaller—more specific—problems, each handled by its own technology-aware transport controller. Then, a parent controller, sitting above them in the hierarchy, aggregates each partial view into a global E2E view, which is then exposed to the network management applications in charge of optimizing resource consumption [3], the resource management application (RMA) in our case.

This paper experimentally evaluates the various components of a hierarchical resource management framework for a multi-domain transport network. It involves two remote sites (CTTC and NEC) consisting of a heterogeneous data plane

featuring mmWave/Wi-Fi and multi-layer optical transport networks, a hierarchical control plane with three layers of hierarchy, and a resource management application in charge of optimizing network resource consumption during path setup. More specifically, the goal of the evaluation campaign is to characterize the E2E performance of such system by measuring various metrics of its control and application planes. In this direction, evaluations are carried out in multiple dimensions. First, each specific network segment (wireless and optical, packet switched and circuit switched) is evaluated. Second, control and application planes are characterized. Finally, the contribution of each hierarchical layer is also measured. We mainly focus on path setup time though domain-specific metrics, such as path restoration time (wireless) or blocking probability (circuit switched), are also evaluated. In fact, the measured path setup times in the order of seconds in such an E2E complex setup contribute to the 5G target of reducing service deployment times from months to minutes (or seconds). Such decrease is mostly due to switching from human involvement in current practice to automated path setup E2E. The concept of hierarchical distributed control of SDN networks has been proposed in the context of data centers [4] (as opposed to WAN environment in our case), mainly to pursue a high degree of scalability. In the context of Network Functions Virtualisation (NFV), ETSI also advocates the use of a hierarchical model to manage NFV Infrastructures [5]. Preliminary design/architectural ideas behind hierarchical SDN orchestration were presented in [6] as a feasible solution to manage and orchestrate the interconnection of wireless and optical transport domains. Such initial ideas were improved by defining the northbound API towards the RMA, by improving the interface between parent and child controllers, by supporting more complex wireless topologies, or by adding smarts to the wireless SDN controller for fast local event handling, among other things. But most importantly, in this paper, we exhaustively evaluate at the experimental level one such complex and heterogeneous network setup from both application and control planes perspectives.

The rest of the paper is organized as follows. Section II describes the RMA application. In section III, we detail the resulting hierarchical architecture and its experimental setup for our evaluations. In section IV, we report and discuss the quantitative E2E evaluation carried out. Finally, section V concludes the paper.

II. THE RESOURCE MANAGEMENT APPLICATION

The Resource Management Application (RMA) runs on top of the hierarchical control infrastructure (see Section III), enabling us to dynamically adapt E2E 5G-Crosshaul flow paths between endpoints, e.g., Remote Radio Heads (RRHs) and BaseBand Units (BBUs). The RMA, developed in Java, is

Project partially funded by EU H2020 under grant 671598 (5G-Crosshaul) and MINECO grants TEC2014-60491-R and TEC2015- 69256-R.

physically located at NEC premises, and it is connected to the hierarchical control infrastructure, at CTTC premises, through Common Orchestration Protocol (COP)¹ via a VPN tunnel. In the initialization phase, the RMA obtains the current E2E topology (an abstraction of it) from the hierarchical control infrastructure and then computes the optimal paths between a pair of endpoints. In the event of a link failure, or in case a new link is added to the topology, the RMA updates the network topology view and searches for a new solution to route flows amongst endpoints. The different key architectural components are the following:

- *HTTP Client*: The HTTP Client class is used to create different HTTP request objects that will be used to communicate with the parent controller's REST API. It allows creating objects that encapsulate common HTTP methods such as GET, POST, DELETE and PUT. The bodies of the HTTP methods are filled out with JSON data objects that the ABNO controller understands;
- *Communication manager*: The communication manager class processes all the HTTP objects regardless its specific methods. That is, all the HTTP objects are enqueued in a FIFO queue while several threads process them. Furthermore, the set of threads use different callback objects to pass the results of the different HTTP objects processed back to the class that created them;
- *Network*: The network class provides the different data structures and algorithms that are needed to store the network state and to compute the different paths. Specifically, it provides graph data structures that allow us to store snapshots of the network. Thus, the data structures can be easily accessed and used to run different graph algorithms. Furthermore, it also contains path computation algorithms to calculate the best routes;
- *Manager class*: The manager class glues all other classes to provide the desired functionality. It contains the start and shutdown routines, the algorithms used to create a request to install a path (using an HTTP POST request) and to delete a path (using and HTTP DELETE method). Besides, this class manages the callbacks upon a link loss (upon the notification from the underlying controller) and initiates the recovery process so that a new path is computed and the data structures can be updated accordingly along with the actual network configuration.

These classes manage the following RMA processes. The *start* routine is further subdivided into three different parts. First, it loads the necessary classes, including an HTTP client class to encapsulate all the functionality to send HTTP messages, needed to communicate to the ABNO controller. Second, it sends an HTTP GET request to the controller to get the underlying topology, and creates the necessary data structures to save the current network state. Third, it computes the paths using the topology abstraction and creates the necessary JSON objects that will be sent within the HTTP POST requests in order to install each path.

The *find* routine belongs to the manager class previously described. In short, this function searches for the optimal paths between each source and destination. It begins by building necessary data structures required by the RMA's optimization

methods. Mathematical functions and data structures are built on top of the apache math3 common library². Next, data structures are passed on as inputs of the optimization functions. The result on the computed optimal paths is sent back to the manager class so that it can continue with the path installation routine.

The *install* function passes information on the paths computed in the *find* routing. Then, the installation routine loops through the different paths in the solution, and creates the JSON objects that will then be inserted in the HTTP POST methods to request their proper installation. The HTTP POST requests are sent to the communication manager class so that the parent ABNO controller interprets the JSON objects and installs the paths in the network.

The *shutdown* function unloads the classes that have been loaded in the *Start* process, and deletes the current installed paths. It creates an HTTP DELETE object for each current path installed and sends them to the parent ABNO controller so that it erases all the forwarding rules.

III. HIERARCHICAL CONTROL TESTBED

The testbed is composed of building blocks coming from two testbeds located at the CTTC premises, namely ADRENALINE³ and EXTREME⁴, which form the core of the CTTC E2E 5G infrastructure [7]. The proposed hierarchical control plane shown in Fig. 1 is part of the 5G-Crosshaul Control Infrastructure (XCI) [2]. Its main features are:

- Multi-layer and multi-domain network, encompassing 1) 802.11ad and 802.11ac technologies at the edge packet-switched wireless transport domain/segment, and 2) wired packet-switched layer on top of an optical circuit-switched layer in the core transport domain;
- Hierarchical network management and orchestration: each domain is managed and orchestrated by its own technology-aware controller.

In what follows, we provide the main details behind each of the controllers/orchestrators:

E2E SDN Orchestrator. The hierarchical SDN orchestrator is based on the IETF Applications-Based Network Operations (ABNO) architecture. The parent controller leverages the Common Orchestration Protocol (COP) to bidirectionally interact with its per-technology child SDN controllers. In the same way, COP is also used as API with the ultimate goal of offering an abstracted global network view to the RMA (described in II). A complete description of the ABNO can be found in [8].

Wireless SDN Controller. A complete description of the wireless SDN architecture can be found in [9]. The wireless SDN controller, which is based on the Ryu framework, acts as acts as child controller. As such, it is responsible for the network control of the edge transport segment featuring a dense mmWave and WiFi mesh data plane. The Southbound Interface (SBI) implements REST-based services and OpenFlow (OF) to interact with the wireless transport nodes, whereas the

²<http://commons.apache.org/proper/commons-math/>

³<http://networks.cttc.cat/ons/adrenaline/>

⁴http://networks.cttc.cat/mobile-networks/extreme_testbed/

¹<https://github.com/5G-Crosshaul/COP>

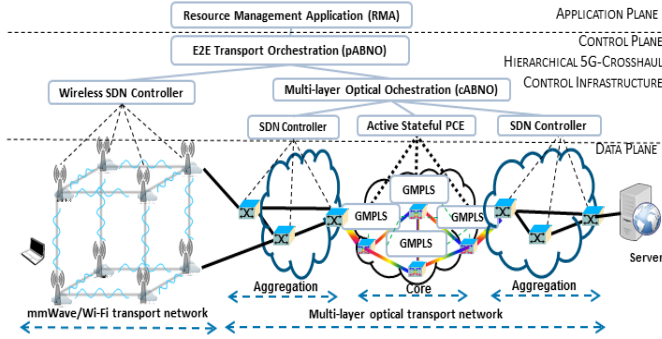


Fig. 1. Architecture and experimental setup at CTTC and NEC premises.

Northbound Interface (NBI) is based on COP to interact with the E2E SDN Orchestrator. At the data plane level, each wireless transport node features several wireless Gigabit interfaces based on IEEE 802.11ad and IEEE 802.11ac. These nodes can use xDPd and OpenvSwitch as software switches running OF and host a REST server for reconfiguration of mmWave/WiFi BH/FH interfaces. More specifically, REST interfaces will be used for the less dynamic aspects such as interface configuration.

SDN Transport Orchestrator. The child ABNO (cABNO) provides E2E multi-layer (packet and optical) and multi-domain provisioning across two heterogeneous control domains: SDN packet domain at the aggregation segment and GMPLS/Active Stateful (AS) Path Computation Element (PCE) managing circuit domains at the core segment. For the core segment, the AS PCE manages an all-optical DWDM mesh network, with two colorless Reconfigurable Optical Add/Drop Multiplexer (ROADM) nodes and two Optical Cross-Connect (OXC) nodes, interconnected with 5 bidirectional optical links with a total of 610Km of optical fiber. The network provides reconfigurable (space and frequency) end-to-end lightpaths, transparent to the format and payload of client signals. Each optical node has multiple DWDM transceivers up to 2.5Gbps and one at 12.5Gbps with fully tuneable laser sources. As for the aggregation segment, the aggregation nodes are packet-switched, also controlled by an OpenFlow controller (acting as child of the transport orchestrator, hence the third layer of the hierarchy), with the specific constraint that edge nodes interfacing to the optical transport network have a tunable interface and they also need to be programmed. For further details of the cABNO and the multi-layer optical network, we refer the reader to [8].

Moreover, it is worth noting that the child controllers can make autonomous local decisions at a shorter time-scale than the parent controllers and external SDN applications, as they are located closer to and are more aware of technological specificities of the data plane devices. For instance, the mmWave/WiFi mesh controller can quickly recompute paths locally within the mmWave/WiFi mesh transport network in case of failures to maintain connectivity.

IV. EXPERIMENTAL EVALUATION

This section presents the results characterizing our hierarchical resource management: we first study the behavior of each individual domain and, after that, we present the E2E results, including the control and application planes.

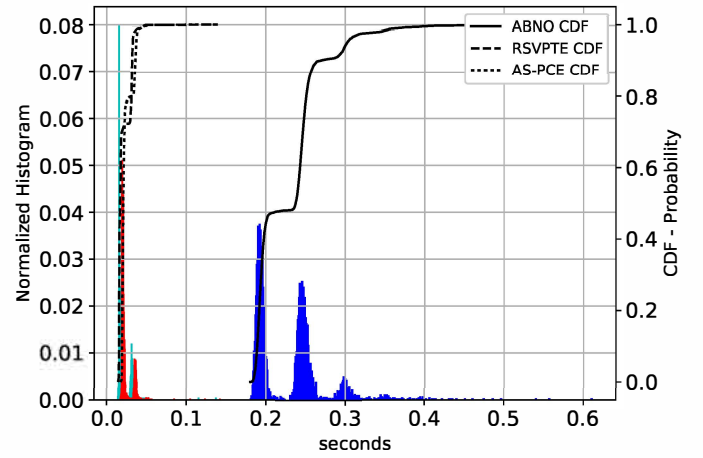


Fig. 2. Setup delay histogram and CDF from the GMPLS/RSVP-TE controllers, from the AS-PCE, and from the cABNO.

A. Single domain characterization

1) *Control plane of the optical network:* Experiments of 10000 requests have been conducted, where connections arrive following a Poisson process with negative exponential holding times, fixing inter-arrival average time and increasing holding times depending on the offered traffic in Erlangs. (e.g., one connection arriving on average every 3 seconds, and lasting on average 60 seconds for an offered traffic of 20 Erlangs). For the evaluation, we consider that requests are being uniformly distributed among the four optical nodes (optical cross connects, or OXCs) within the network scenario. Each link is characterized by having 8 bidirectional wavelengths supporting 10Gbps client rates. Requests go from one OXC to a destination OXC and do not involve client transceivers due to hardware limitations that would constraint the results, since only a limited set of client transceivers are available, well below the theoretical maximum supported by the optical network. Therefore, in this section, only the circuit-switching layer is characterized.

Path Setup Time. This includes the analysis of path setup time as measured from the cABNO, AS-PCE, and GMPLS entities. In particular, we obtain the histogram and the CDF of the different entities. Notice that since we focus on the optical domain only, we study the hierarchy up to the cABNO. Path setup time is affected by the hardware configuration delay, which involves configuring the OXCs and is vendor dependent. We have performed different measurements with and without hardware configuration.

Without optical hardware configuration. To exclusively characterize the control plane behavior, we configure the hardware optical nodes in emulation mode, hence requiring a negligible configuration time. Fig. 2 depicts the histogram and CDF of the setup delay seen at different reference points: from the GMPLS/RSVP-TE connection controllers, from the AS-PCE, and from the cABNO:

- From the GMPLS control plane: this means the setup delay considering the signaling process, from the RSVP-TE connection controller of the ingress node to that of the egress node, and roughly corresponding to the round-trip time (RTT) of the signaling messages with forward Path and reverse Resv messages across the different transit

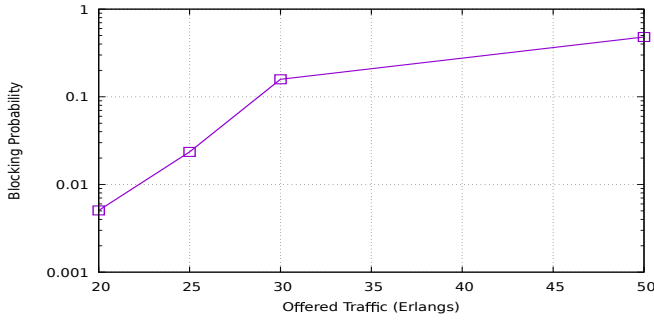


Fig. 3. Blocking probability for the optical network.

nodes. We see that on average, this shows two peaks, roughly corresponding to whether 2 or 3 optical nodes have been involved in the provisioning. Avg.=20,72ms.

- From the AS-PCE: the AS-PCE adds a small component to the setup delay, associated to the processing of requests from the child ABNO and dispatching of requests to the corresponding head-end node. Avg.=23,80ms
- Finally, we see the setup time from child ABNO, which adds an additional time due to the COP protocol and the use of text-based REST interfaces. Average setup delay seen from ABNO is 259,01ms.

With optical hardware configuration. In this case, an additional set of 100 requests has been performed. The main finding is that the hardware configuration adds an additional 200ms to the provisioning time. As a summarizing guideline, the setup delay for the optical domain, seen from the cABNO controller can be probabilistically bound to below 500ms for our scenario.

Blocking Probability. To better characterize the wavelength-switched layer of the optical transport network, we generate dynamic arrivals of connection requests, where each connection requires the establishment of an optical lightpath between two client ports. If a connection cannot be served, for example, due to no more wavelengths being available that can be provisioned to the service, we consider the connection blocked. Blocking probability (for the selected service of lightpath establishment) has been experimentally evaluated as a proportional ratio of successful establishment over offered ones. Likewise, Fig. 3 has been obtained empirically by performing random setup requests and a Dijkstra-based path computation algorithm ensuring wavelength continuity. We present data points from 20 to 50 Erlangs. The blocking probability ranges from 0,5% at 20 Erlangs to 48% at 50 Erlangs. This test has been done to stress the system and evaluate the BP trend; in real operation scenarios, networks are dimensioned in operating ranges that keep the BP at acceptable levels.

2) Control plane of the Wireless domain: We focus on two control channel setup options, namely wireless and wired. In the former, control plane messages are wirelessly exchanged between switches and the controller by using 802.11ac links. As for the later, used as benchmark, control plane messages are exchanged by using Gigabit Ethernet links. We measured path setup and restoration time distributions (10 repetitions per experiment).

Path Setup Time. We analyzed the path setup time by varying the number of wireless hops traversed by the injected traffic. As for the wired case (tagged with prefix *Wrd* in Fig. 4), one

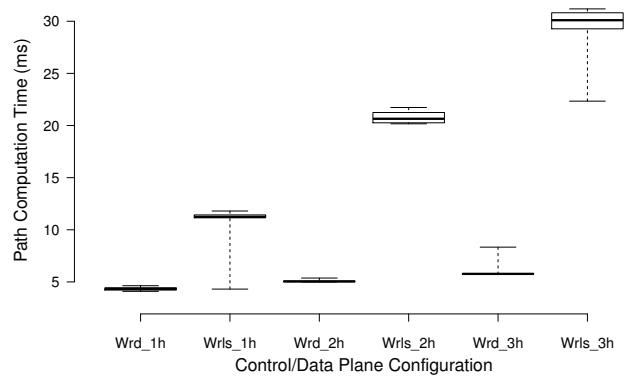


Fig. 4. Path Computation/Setup Time in the wireless domain.

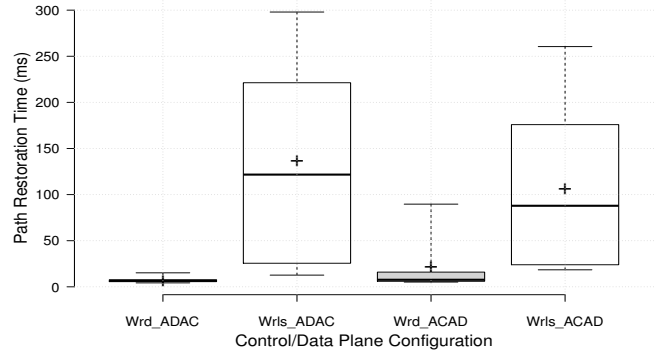


Fig. 5. Path Restoration time in the wireless domain.

may observe that the average path setup time slightly increases with the number of hops due to the fact that more flow mods must be handled, but since they are sent through reliable wired links, all values remain ~ 5 ms. As for the wireless type (tagged *Wrls*), there is a substantial impact of around 10ms per additional hop on the average path setup time due to sharing of the wireless (control) medium. Additionally, the variability is also much higher than in the wired case and also increases with the number of hops, since the impact of traversing a more dynamic and less reliable control plane link is suffered more times as the number of hops increases.

Path Restoration Time. We fixed the number of traversed hops of injected traffic to three. In this case (see Fig. 5), an experiment is defined by the type of control plane and the wireless link suffering failures. The tag *ADAC* corresponds to a failure in the 802.11ad link and a consequent switch to the 802.11ac as alternative, whilst tag *ACAD* refers to an interface switch in the opposite direction. A failure in one of the links causes path recalculation and rerouting of ongoing traffic through the non-failed interface. As observed in Fig. 5, the values in the wired case are higher than those for path setup time, which is mainly due to the time it takes to detect the failure in the interface and to notify the controller through OF port status messages and path recalculation. The rest of the process (i.e., flow mod message handling) is similar to the previous case. What is also observed is the much higher variability compared to the wired path setup time, again due to the detection time variability. As for the wireless control plane, average values are much higher (in the order of hundred ms, as opposed to tens for path setup). Additionally, variability due to detection is added to that coming from the dynamicity of the wireless medium, hence resulting in a wide margin of

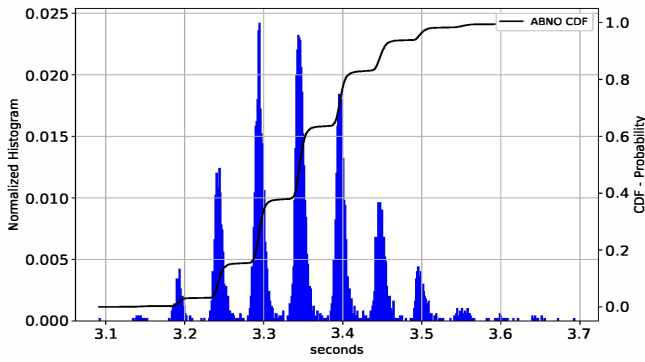


Fig. 6. Setup delay and histogram at the parent ABNO.

TABLE I. SETUP DELAY. PARENT ABNO VS. RMA

	Average	Min.	25-percentile	75-percentile	Max.
Parent ABNO (s)	3,349	3,092	3,294	3,398	3,693
RMA (s)	3,971	3,667	3,804	4,046	5,281

possible values. Note also that processing time at the nodes when switching is relevant, as different interface switching directions give different values of path restoration times. This study confirms the interest of hierarchical setups in which a child (technology-aware) controller handles local network impairments faster (100s of ms) than if such decisions climb up the hierarchy for a fully centralized decision (seconds).

B. Multi-domain experiments

Path setup time. The histogram and CDF of the E2E multi-domain multi-layer path setup time as seen from the parent ABNO are presented in Fig. 6. The multiple peaks in the histogram reflect an implementation artifact through which the ABNO (child and parent) wakes up every 50ms and processes the received requests during that period. However, the most remarkable observation is the increase of the average setup delay from tens or hundreds of milliseconds in the wireless and single-layer optical domains as seen from the child controllers to seconds as seen from the parent. This is due to various factors. First, previous sections only presented unidirectional single-layer values. On the other hand, this section presents values for multi-layer (i.e., Ethernet over wavelength-switched optical network) and bidirectional connection setup, out of which an average of 2,867s. are spent in the multi-layer optical network. Second, there is the interaction and message processing between the parent ABNO and child controllers. Third, there is the sequential handling of some of the messages to set up the E2E path. As far as the application plane is concerned, Table I compares some statistics of the setup delay as seen from the parent ABNO and from the RMA. Recall that there is a tunnel set up between the CTC premises (in Barcelona, Spain) and the NEC premises (in Heidelberg, Germany) with an approx. RTT of 60ms. However, the difference is of 600ms approximately, which is due to the processing carried out at the RMA.

RMA characterization To evaluate the latency incurred in path computation and setup (and other RMA-related processes) in a more generic way, we randomly select 5 source-destination pairs and set up a flow of 5 Mbps for each of the pairs. Table II presents the mean and standard deviation of the processes described in Section II when 5 connections are

TABLE II. RMA PROCESSING CHARACTERIZATION.

	Start	Find	Install	Shutdown
Mean (ms)	880,45	47,00	11625,65	11793,6
St.Dev. (ms)	29,81	1,59	456,43	387,63

handled. *Start* characterizes the interaction with the parent controller to get the topology abstracted by the parent ABNO. *Find* characterizes the path calculation time based on the received topology, and so, it corresponds to internal processing inside the RMA. *Install* corresponds to the E2E path setup time of the 5 connections. Notice, however, that five times the average value shown in Table I is much higher than the mean value of the install process. This is due to the setup of the optical lightpath, which is only done for the first connection. The remaining ones reuse this same optical path, and so, there is less setup delay. Finally, the *shutdown* process has a duration that is similar to that of the install process due to the sequential deletion of each of the paths, which is quite similar in terms of interaction between the RMA and parent ABNO.

V. CONCLUSION

The results confirm the interest of a hierarchy of controllers in which some technology-specific local decisions can be taken by the closest controller in the hierarchy, hence saving processing and propagation time. In our case, and depending on the domain, there may be differences in path setup/restoration values observed at child vs. parent controller ranging from ~500ms to one order of magnitude (up to units of seconds). For instance, the path restoration time values in the wireless domain (100s of ms) and the path setup time at the child controllers (~2.5s.) is smaller than that observed by the parent ABNO (3,349s.) and RMA (3,971s.). For the scenario under study, the main components of the path setup delay are: 1) RMA processing, 2) RMA-to-parent ABNO latency (~60ms of RTT) for each message exchange, 3) bidirectional multi-layer connection setup in the optical network (~2.5s.), and 4) wireless domain (tens of ms) assuming a wireless control channel. Overall, experimental results provided show an average of 3,971 seconds for E2E path setup delay, hence contributing to the 5G target of lowering the service deployment time, in this case, multi-domain path setup, from months to minutes.

REFERENCES

- [1] "Small cell virtualization functional splits and use cases, small cell forum," 2015.
- [2] A. D. L. Oliva *et al.*, "Xhaul: toward an integrated fronthaul/backhaul architecture in 5g networks," *IEEE Wireless Communications*, vol. 22, no. 5, October 2015.
- [3] "Open Networking Foundation, SDN Architecture Overview, Version 1.1, onf tr-504, palo alto, ca, usa," 2014.
- [4] L. Fang *et al.*, "Hierarchical sdn for the hyper-scale, hyper-elastic data center and cloud," in *ACM SIGCOMM Symposium on Software Defined Networking Research*. ACM, 2015.
- [5] "ETSI Network Function Virtualization: Architectural Framework," 2013.
- [6] R. Vilalta *et al.*, "Hierarchical SDN Orchestration of wireless and optical networks with E2E provisioning and recovery for future 5g networks," in *OFC*, 2016. IEEE, 2016.
- [7] R. Muñoz *et al.*, "The CTC 5G end-to-end experimental platform: Integrating heterogeneous wireless/optical networks, distributed cloud, and iot devices," *IEEE Vehicular Technology Magazine*, March 2016.
- [8] "5G-Crosshaul D5.1, report on validation and demonstration plans," 2016.
- [9] J. Núñez-Martínez *et al.*, "WiseHAUL: An SDN-empowered Wireless Small Cell Backhaul testbed," in *2016 IEEE WoWMoM*, June 2016.