TELECOM INFRA PROJECT

OOPT-CANDI Whitepaper

Service provisioning & assurance with high-accuracy network monitoring and dynamic traffic engineering Proof of Concept 2021

Date September 30, 2021

Document version: v1.0

Authors

Olivier Dugeon
Senior Research Engineer, Orange
olivier.dugeon@orange.com

Aki Fukuda
Senior Research Engineer, NTT
aki.fukuda.cd@hco.ntt.co.jp

Oscar González de Dios
IP Transport and SDN Expert, Telefónica CTIO
oscar.gonzalezdedios@telefonica.com

Konomi Mochizuki
Senior Research Engineer, NTT
konomi.mochizuki.ps@hco.ntt.co.jp

Roberto Muggianu
Senior Network Architect and Lead Architect for Telco Cloud, Telia Company
roberto.muggianu@teliacompany.com

Hirotaka Yoshioka
Vice President, NTT
hirotaka.yoshioka.ec@hco.ntt.co.jp

Contributor

Erdem Cavusoglu
Network Engineer, Facebook
ecavusoglu@fb.com

Kentaro Ebisawa
Research professor, NTT
kentarou.ebisawa.xa@hco.ntt.co.jp

Sujay Gupta
Solutions Engineering, IP Infusion
sujay.gupta@ipinfusion.com

Sanat Nagaraju
Solutions Engineering, NTT
Sanat.Nagaraju@nttdata.com

Table of Contents

List of Figures

1 CANDI's roadmap

1.1 Introduction to CANDI

The Open Optical & Packet Transport group (OOPT) is a project group within Telecom Infra Project (TIP) that works on the definition of open technologies, architectures, and interfaces in optical and internet protocol (IP) networking. The primary aim of the OOPT group is to focus directly and contextually on enabling operators to plan and deploy disaggregated, open systems across a wide span of their infrastructure. Converged Architecture for Network Disaggregation & Integration (CANDI), a subgroup of OOPT, aims to construct an end-to-end reference network architecture that is conducive to disaggregation as well as evaluate logical integration points among disaggregated components within the reference architecture.

1.2 The final goal of CANDI

The goals of CANDI are to flexibly allocate features on IP & optical networks, achieve efficient IP & optical networks, and provide inputs for the OOPT sub-working group using disaggregated and open systems.

To achieve these goals, CANDI is taking the following actions.

1. CANDI clarifies issues to be solved for the future vision of packet optical transport networks
2. CANDI will take a step–by-step approach for achieving each use-case (Figure 1)
3. The main objectives of the Proof of Concept (PoC) are as follows:
   - To establish and test the basic architecture and workflow as a first step
   - To include both packet networks and optical networks, but not fully integrated networks

This whitepaper summarizes the PoC achieved by the CANDI working group and the lessons learned after the trial. CANDI will continue to contribute to conducting enhanced PoCs twice a year to test and evaluate agreed upon use-cases and find the issues for developing the technology required to validate the feasibility of an operator's use-cases.

1.3 Summary of operator's use-cases

In the first phase, CANDI collected the 10 use-cases shown in Figure 1 from five operators. All use-cases were analyzed in detail, and related use-cases were grouped together. For example, five of these use-cases (1, 3, 7, 9, and 10), were inter-related to provisioning of network termination in both optical and packet networks, hence these five use-cases are consolidated into "Provisioning of services in open optical and packet networks" as shown

on the right in Figure 1.

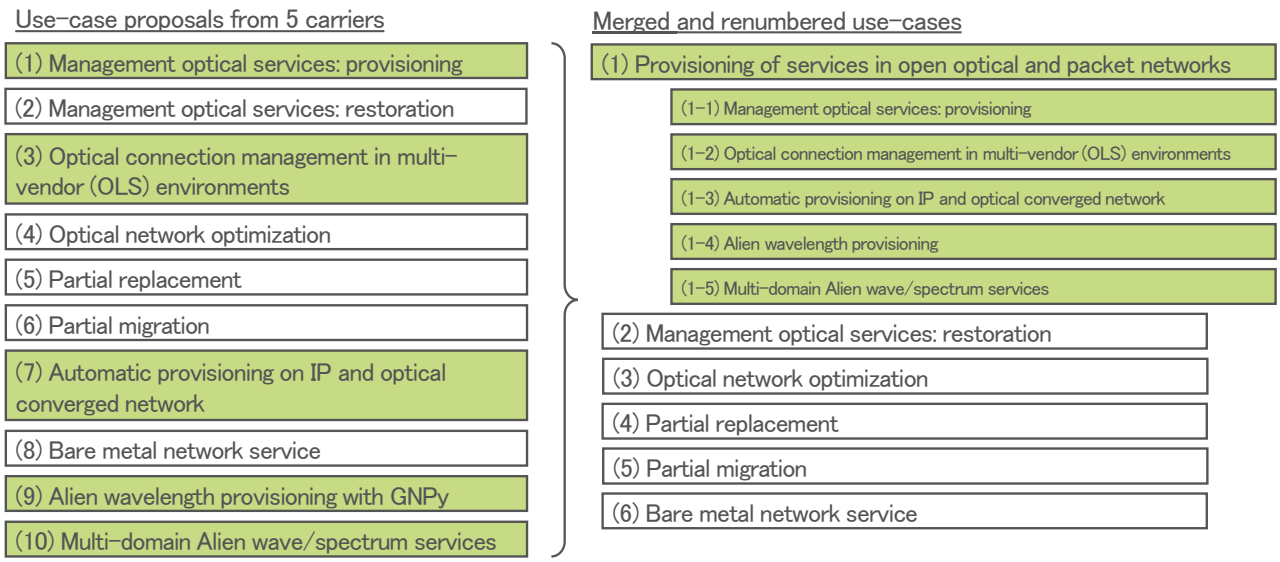

Figure 1. Use-case proposals

2 Experimental demonstration

2.1 Scope of demonstration scenario

CANDI PoC validates the end-to-end OOPT ecosystem. It is not only individual interfaces or products such as routers/transponders, but also a system view (devices, controllers, orchestrators, etc.). In CANDI PoC 2021, we have taken part of the CANDI use-cases shown in Figure 1, and mapped them to four staged and complementary scenarios. We carried out the experiments on the basis of their four scenarios.

1. Optical scenario (PoC #1): Provisioning including physical impairment information. This is corresponds to use-cases #1-1 in Figure 1. A Whitepaper with its results is available on the TIP web site [1].
2. Packet scenario (PoC #2): Data center interconnection with layer 3 virtual private network (L3VPN) service provisioning. This corresponds to use-cases #1-3 in Figure 1.
3. Packet scenario (PoC #3): L3VPN service provisioning in core network with service assurance using traffic engineering and high-accuracy delay monitoring. This corresponds to use-cases #1-3 in Figure 1.

4. Migration scenario (PoC #4): Remotely replace a box network operating system (NOS) (starting with white-boxes). A Whitepaper with its results will soon be available on the TIP web site.

The PoC Network scenario consists of mixture of contributor's products and TIP's products (Physical Simulation Environment (PSE), Disaggregated Optical Systems (DOS), and Disaggregated Cell Site Gateways (DCSG)). CANDI has elaborated workflows to implement the use-cases using the set of TIP technologies showcasing the programmability of the IP/optical network.

This document describes the packet scenario (PoC #3) with experimental details, results, and discussion.

2.2 Location

We conducted the PoC testing at the TIP Community Laboratory owned by Facebook in London, United Kingdom. Facebook, 1 Rathbone Square, London, W1T 1 FB. United Kingdom. Except for setting up the physical environment, we did most of the work remotely via a VPN.

3 Service provisioning with service assurance

3.1 Test scope

3.1.1 Overview of related use-case

PoC#3 is one element that achieves use-case #1-3 in merged and renumbered use-cases of Figure 1. This use-case aims at automatic service provisioning, guaranteeing service assurance at all times in the IP and optical converged network. The network consists of multiple data center (DC) and core network, and the services between DCs are provided via the core network. In PoC#3, we focus on the managing IP packet core network. Moreover, its core network is assumed to be implemented using segment routing (SR), which has recently been considered by carriers as a useful technology for achieving traffic engineering (TE), especially VPN-TE, in the core network.

3.1.2 Motivation

Critical network services such as telemedicine and automatic operation require a secure network with low delay and high reliability. To achieve this type of network service, we must reduce the delay measurement and control cycle as much as possible. With dynamic and automated control, customers will be able to start using a network in a short period at a cost commensurate with network quality. Based on this, in this PoC's scenarios, the motivation is to prove a mechanism that can automatically provide customers with optimal networks that meet varying service level agreements (SLAs) based on required service. However, automatic control, i.e., closed-loop, has been discussed for some time, but there are currently few strong references on specific workflows, implementations, and feasibility checks.

3.1.3 Issues to be solved

In this PoC, we will guarantee the service requirements, i.e. SLAs, at all times by accurately monitoring the status of the network and promptly reflecting the results in the network control. Further, this process will be achieved in a closed-loop. Our target for the service requirements in this PoC is propagation delay.
In the future, the number of users with different requirements is expected to increase. Considering that, carriers should preferably have several options for combining the network element (NE) and the higher-level application so that a variety of management control can be achieved. Currently, vendors and manufacturers have proposed various solutions for network management and control, and NEs and high-level applications are generally operated by the same vendor/manufacturer solutions. However, this dependency on vendor/manufacturer specifications means that the way the system is operated is likely to change as service requirements change and equipment is upgraded, and this may impact OPEX. To achieve the objective mentioned above, the following issues need to be resolved:

- Open and disaggregated network
  NEs, controllers, and the interfaces (IFs) that connect them need to be open and disaggregated, i.e. utilizing white-box switches and open-source software (OSS). It is also important that there is a standard interface to support their interoperability.
- Rapid path control to follow network quality changes
  A mechanism is needed to reflect network quality changes captured by active monitoring and direct control, and it should be achieved automatically, i.e., closing the loop between the monitoring, controller, and network elements (closed-loop).
- High-accuracy monitoring of network quality
  There is a need for a highly accurate monitoring technique that captures minute variations in the quality of user traffic without relying on conventional methods (e.g., ping).

In the experiments described in this report, we were able to demonstrate an approach that successfully addresses the issues listed above.

3.2 Ideal specifications

In this section, we describe an ideal architecture and recommended protocols between the components to achieve the workflow.

3.2.1 System architecture

An ideal system architecture, shown in Figure 2, for this scenario typically consists of a network orchestrator, a packet software defined networking (SDN) controller, and a core network implemented using SR.
The orchestrator provisions the network slices (L3VPN) in accordance to the instructions

given by the operator. Moreover, the orchestrator requests the controller to design the desired optimal path and set configuration to NEs. Communication between the orchestrator and the controller is achieved by using either REST or RESTconf [2], which are standard protocols for obtaining/setting the network state in the controller. Communication between the orchestrator and NE (for configuring of network service paths, with measured SLA's) is achieved through Netconf [3], a standard protocol for network configuration.

The controller supports link state control and the traffic engineering database (TED). TED provides a graph that represent the network topology and serves as support for the calculation of paths based on link status, cost, and delay. Hence, the controller can trigger dynamic path change on the basis of prevailing network conditions. The communication protocol between the controller and NE is path communication element protocol (PCEP) [4] with a focus on prompt assured completion of network changes.

The network monitoring function measures the link/path delays with high accuracy by inserting probe packets in-channel into the network.

Finally, a component is also required to support a closed-loop process to autonomously and promptly reflect the change in the network path on the basis of prevalent user traffic conditions measured by the monitoring application. The closed-loop component works in conjunction with the orchestrator using REST or RESTconf.

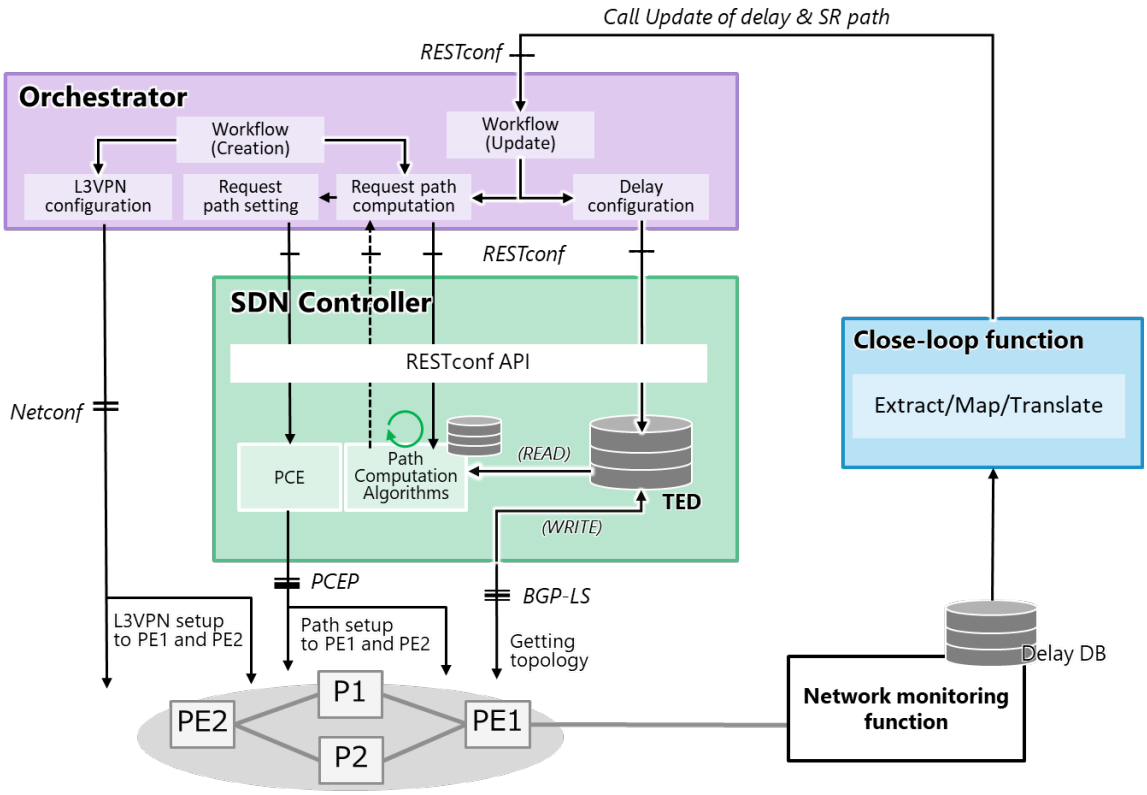

Figure 2. Ideal system architecture

As a side note, Figure 3 shows a comparison with the SDN architecture for the packet network defined by Mandatory Use Case Requirements for SDN for Transport (MUST) [5], a subgroup of TIP OOPT. The packet SDN controller we defined corresponds to the "IP/MPLS Multi-Vendor SDN Controller" in the MUST architecture. The definitions of APIs/IFs and assumptions about the functions to be retained are largely consistent between the two. A characteristic difference is related to the network condition monitoring. Network status monitoring in MUST assumes telemetry or simple network management protocol (SNMP) based status monitoring in the "Alerts & Events." CANDI, on the other hand, assumes high precision monitoring in the μ seconds order using in-channel monitoring. In addition, the function defined by CANDI as an orchestrator is considered to be equivalent to the hierarchical SDN controller in the MUST definition. However, since the current study in CANDI is focused on the controller, we think that requirements, specification etc. of the higher-level function need to be further analyzed and studied.

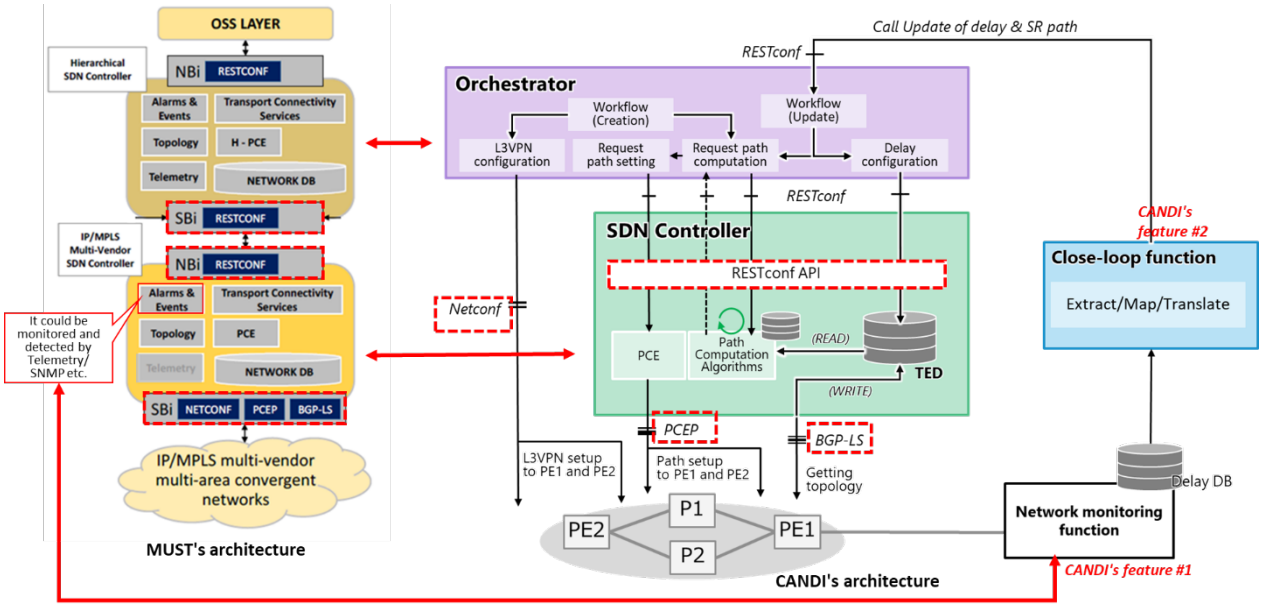

Figure 3. Comparing the ideal architecture with MUST's architecture

3.2.2 Workflow

Figure 4 shows the ideal workflow executed in the system architecture described in 3.2.1.

First, the network monitoring function is activated to probe all the links in the network. Next, the orchestrator configures the L3VPN to the appropriate endpoint on the basis of the request from the user.

After L3VPN configuration is completed, the next step is to configure the SR path that will be connected the VPN infrastructure. The orchestrator notifies the packet controller of the endpoints as well as the SLA requirements for the path. With this information, the controller calculates a preferred path that meets the requirements (i.e., creates different SR paths including a failover path) and submits the configuration to the appropriate NEs via PCEP. When the above settings are completed, the network is now ready to exchange user traffic between the endpoints.

During service delivery, the network monitoring function probes the quality of the SR network by frequently inserting probe packets. The probe packets measure the propagation delay of each link. When there is any variation to the propagation delay, the network monitoring function alerts the closed-loop function.
The closed-loop function provides a link between active network monitoring and network operation for fast and autonomous network control. The closed-loop function notifies the orchestrator that due to variation in propagation delay, a new path must be recalculated. In parallel, the closed-loop function also instructs the controller to update the amount of delay for each link associated with the topology in TED. The orchestrator then instructs the controller to recalculate the path. Based on the new information available in TED, the controller is able to recalculates the path to fit the SLA requirements and submits the configuration to the appropriate NE. In this way, the orchestrator, the controller, and the network monitoring function can work together through the closed-loop function to provide a path that is always SLA guaranteed during service delivery.

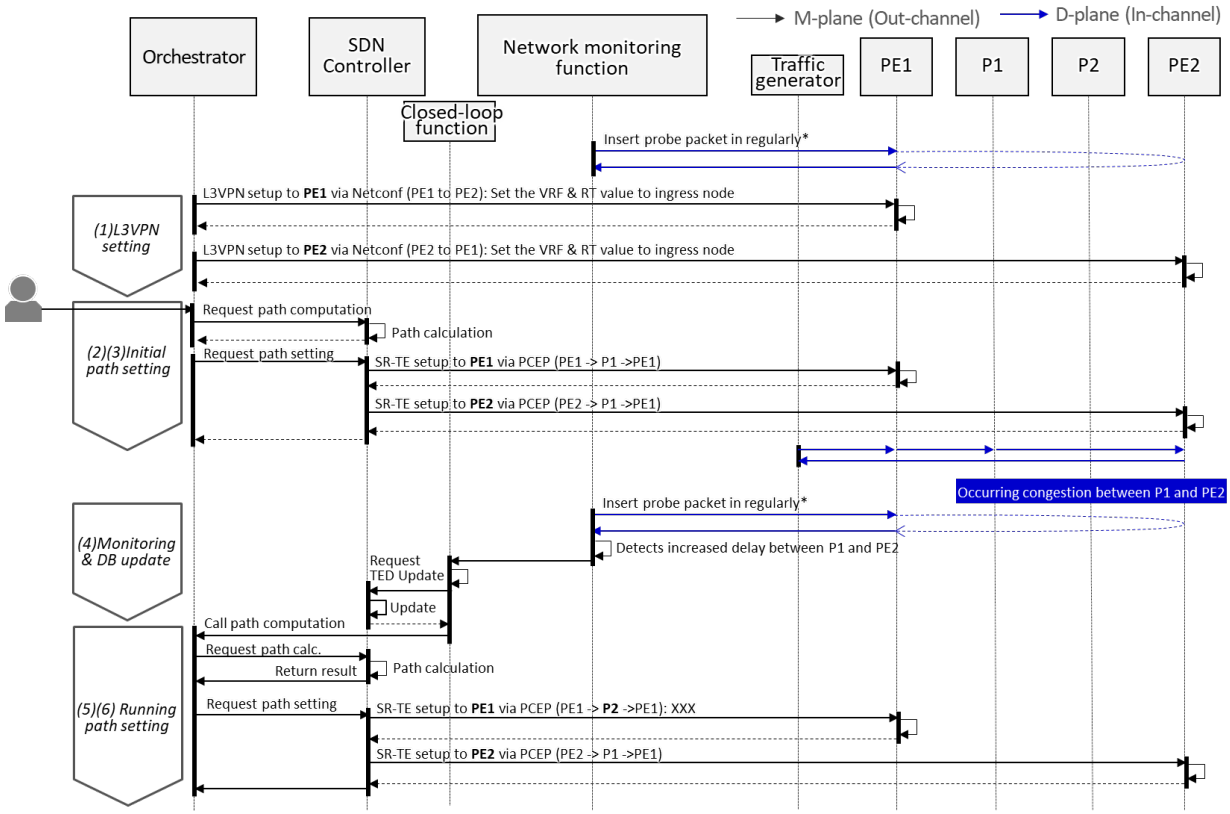

Figure 4. Ideal workflow

3.3 Test specifications

3.3.1 PoC implementation

In this section, we describe the software components used to realize the architecture and workflow described in section 3.2

1. Packet Transport network
   An experimental testbed simulating a core network is implemented with white-box switch based IP packet transport as shown in Figure 5, and this network supports SR and open shortest path first (OSPF) as the underlay protocol. Further, MP-BGP is implemented for exchanging information related to VPN settings (e.g., VRF). Carriers are studying SR as a useful technology achieving TE, especially VPN-TE, in the core network. This network is built up with four IP white-box switches from Edgecore hardware and open-networking software from IP Infusion OcNOS [6].

2. Orchestrator function
   Ansible/AWX [7] in used as the orchestrator function. All configuration items required

for the controller, TED metrics, NE setup, and L3VPN can be centrally setup in AWX and triggered through the Ansible "playbook." In our implementation, the communication between the orchestrator and NE's was carried out using NetCLI. All communication between the orchestrator and controller was carried out using RESTconf APIs from the controller. Similarly, the communication between the orchestrator and closed-loop was carried out using RESTconf APIs.

3. Network control function
   The PoC's network is mainly managed by the packet controller implemented by open-source OpenDayLight (ODL) [8]. ODL has the following capabilities: network topology discovery (using BGP-LS), TED capability, Path Computation Element (PCE). ODL also implements the SAMCRA algorithm [9] for path calculation. Once a change in a path is calculated, ODL can dynamically reconfigure the path in NEs using PCEP. The above functions have been implemented using the APIs provided by the BGPCEP project [10], which is one of an ODL subproject.

4. Network monitoring function (HANMOC)
   We adopted high-accuracy monitoring and control (HANMOC) [11] to monitor the network state of the core network with high-accuracy. HANMOC consists of an application on a server and a P4 switch and enables probing of different links in the network. After configuring topology information of the target network for the system and connecting the system to that network, HANMOC measures delay in both directions for each link on a regular basis.

5. Closed-loop function
   This function is achieved with python script, and works in conjunction with controller and HANMOC software.

6. Delay router
   To simulate real-life congestion scenarios, a congestion router was placed in the user traffic path. Load traffic is added to create congestion to the user traffic between P1 and PE2. The congestion is controlled externally to simulate no-congestion-load and congestion scenarios. With the no-congestion-load, user traffic is carried on the default SR path set by the controller. In the congestion scenario, the controller is expected to dynamically change the SR path (monitor detecting and triggering closed-loop. Closed-loop updates the orchestrator/the controller with the delay metric. The controller uses the delay metric to recalculate the path and updates NEs configuration to change the SR path).

A PoC setup and a revised system architecture (implemented system architecture) are shown in Figures 5 & 6.

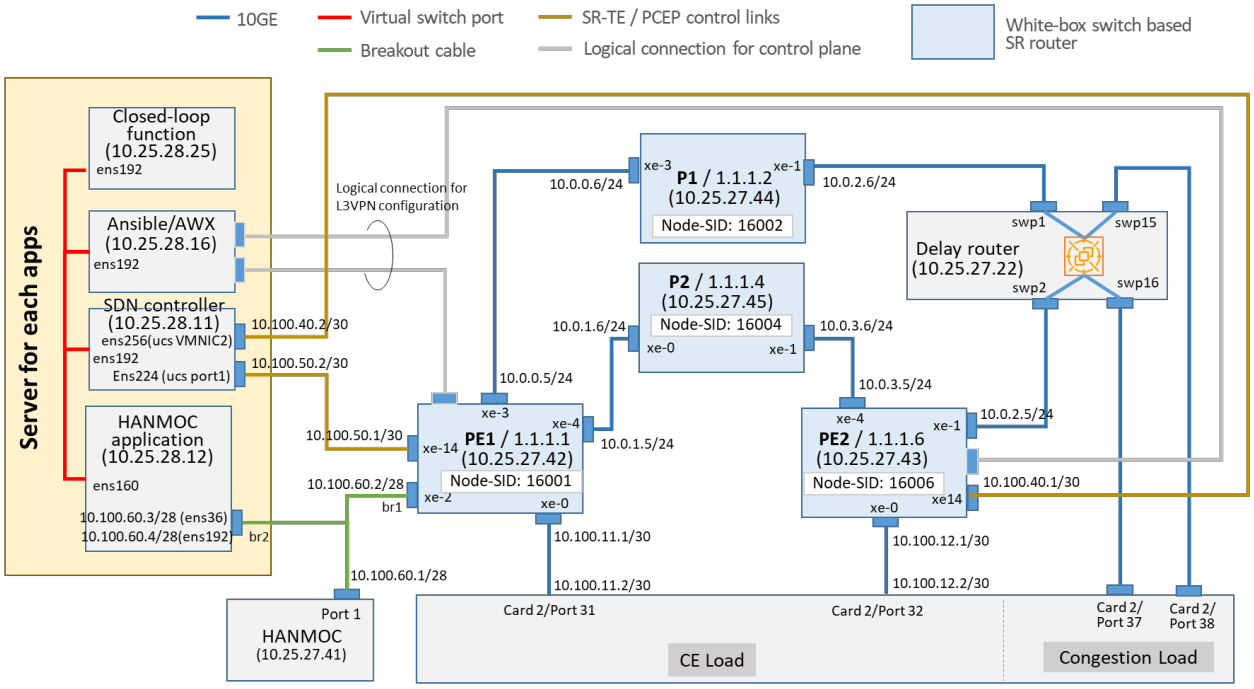

Figure 5. PoC setup

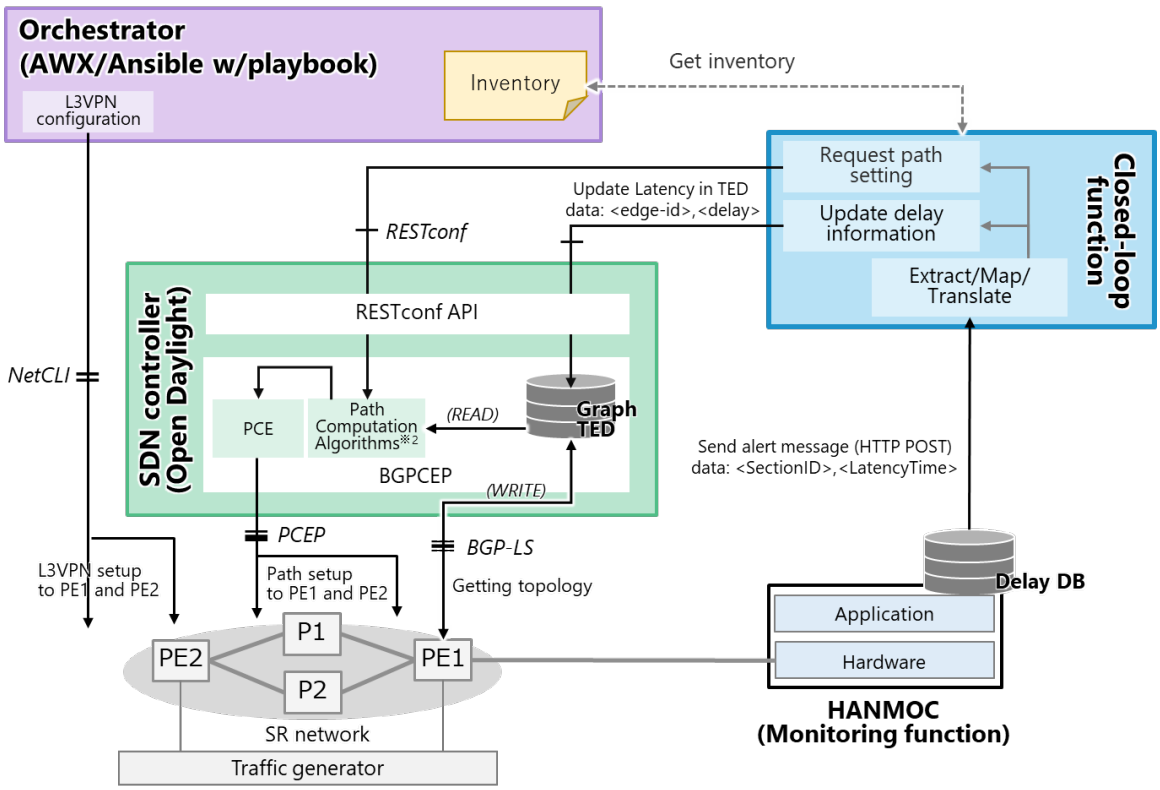

Figure 6. Implemented system architecture

3.3.2 Observations / Limitations seen

Three limitations were observed in the process of implementation:

- ODL – NE communication: We discovered there is issue in terms of interoperability between ODL and NE. Because of that, the ideal workflow (i.e., the ODL setting paths to NE on the basis of requests from Ansible/AWX) did not enable us to work around this issue due to limitations in the Ansible/AWX message format. Therefore, in this PoC, implementing all path settings starting from a python script enabled us to avoid the interoperability issue and still achieve the path settings. This workaround is now available as an ODL patch [12].
- Orchestrator – NE communication: There is the constraint regarding interoperability of Ansible and NEs using Netconf supporting the L3VPN configuration. Hence, the Ansible playbooks for L3VPN configuration were modified to use NetCLI to communicate with NE.
- To reduce the delay between congestion detection and triggering path change, the closed-loop function directly instructs the controller to update the delay for each link in the TED as well as recalculate the path.

Note that Orange also setup a preliminary PoC in its Lab with Juniper MX routers as NEs and did not observe such limitations.

The revised workflow (implemented workflow) is shown in Figure 6.

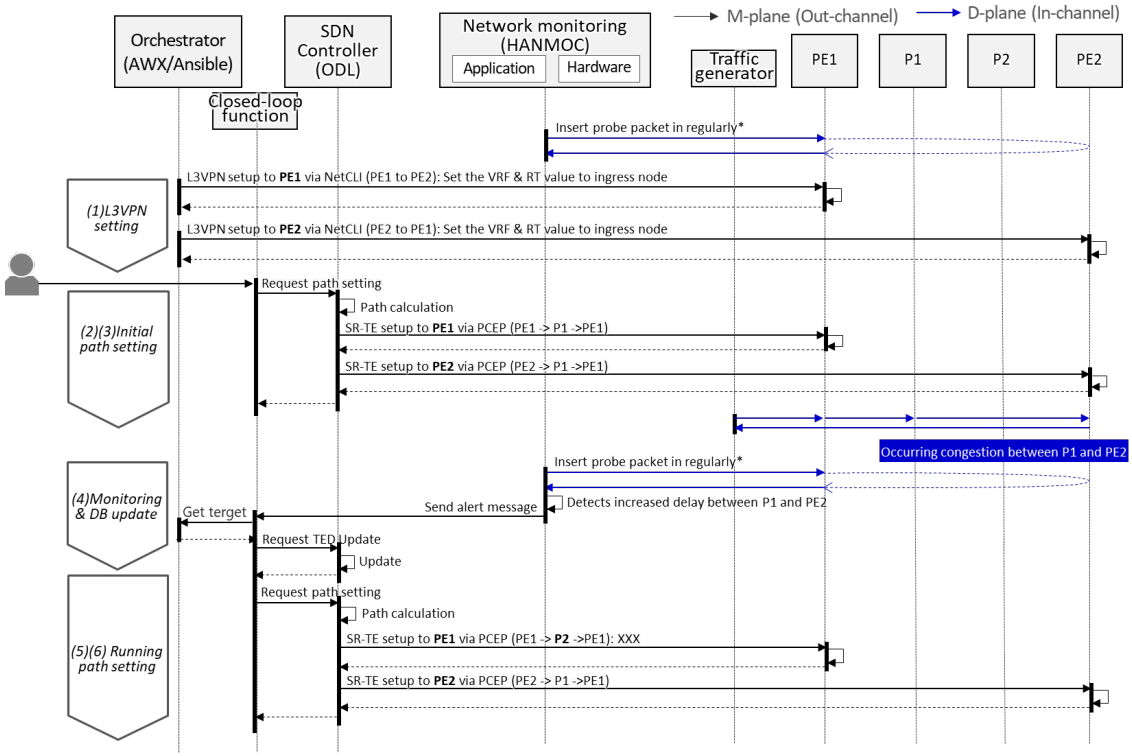

Figure 7. Implemented workflow

3.3.3 Components/Products

Table 1. Management systems

Management systems			
Component	**Product**	**Quantity**	**Provider**
Server for applications	Commercial products	1	Facebook
Packet controller (SDN controller)	OpenDaylight Aluminium-SR3	1	(OSS)
Orchestrator	Ansible/AWX	1	(OSS)
Closed-loop function	Internal production	1	Orange
HANMOC application	Internal production	1	NTT
HANMOC hardware	Commercial products	1	NTT

Table 2. Network elements

Network elements			
Component	**Product**	**Quantity**	**Provider**
White-box switch	Commercial products (DCSG Products)	4	Facebook
Network OS	OcNOS-SP 4.2	4	IP Infusion

	(OcNOS-SP-MPLS)		

Table 3. Test environment

Management systems			
Component	**Product**	**Quantity**	**Provider**
Traffic generator	Commercial products	1	Facebook
Router for congestion	Commercial products	1	Facebook

ODL and Ansible are open-source software, but Orange is the main contributor of Path Computation server used in this PoC. Second, Orange provides all Ansible Playbook for this PoC.

3.3.4 Contributors

- NTT: Provided test scenario / Define issues / Complete test / Provide P4 switch for high-accuracy network monitoring system (HANMOC)
- Telefónica, Orange, and Telia: Contributed to PoC analysis and theoretical setup
- Orange: Implemented the packet controller with OpenDayLight (ODL), the Ansible Playbook for Path Computation and SR-TE path setup and python script for achieving the closed-loop
- IP Infusion: Provided OcNOS with SR features & Netconf, and implemented the Ansible playbook for L3VPN setting
- Facebook: Provided Test environment / On-site support on Community lab

3.4 Test

3.4.1 PoC schedule

Table 4. Schedule for PoC testing

Items	Due date
Start to develop/prepare environment	January 11, 2021
Complete PoC environment	June 30, 2021
Start PoC	July 1, 2021
Complete all PoC testing	July 14, 2021

3.4.2 Test items and results

Table 5. Test items and results for SLA driven SR path provisioning

#	Main item	##	items	Description	Result
1	Basic operation verification	1	Topology information getting by BGP-LS	• Verify that the BGP-LS session is working correctly between controller and NE. • Verify the contents of the Graph TED to make sure it matches the actual topology.	☑
		2	Setting the infrastructure and L3VPN to NEs	Verify the setting OSPF and L3VPN.	☑
		3	Initial path setting	Verify the following behavior regarding path setting. – Initialize PCEP session – Initialize delay info. in topology Graph – Calculate the optimize route/path – Set SR-MPLS path using PCEP	☑
		4	High-accuracy delay monitoring operation	Verify of operation (monitoring, storing results in a DB) using HANMOC in white-box switches based SR network.	☑
		5	Path setting during service	Verify the auto path switching in cooperation with delay variation detected by monitoring function.	☑
		6	Whether the path can be reverted after SR-TE	Verify if it is possible to return to the original path after SR-TE.	☑
2	Performance verification	1	Comparison of delay	The propagation delay between PE1 and PE2 in the transport network shown in Figure 5 is measured using HANMOC/Ping/traffic	☑

			measurement performance	generator method and the results are compared.	
		2	Impact on user traffic when path switching	Verified degree of packet loss on user traffic when the path was switched from PE1-P1-P2 to PE1-P2-PE2 in the transport network shown in Figure 5.	☑ No packet loss
3	Verify the turn-around time	1	Initial path setting	Verify the time from invoking SDN until the SR-TE setting using PCEP is completed from result of #1-3 performing.	☑ Around 3,000ms
		2	Path setting during service	• Verify the time between the occurrence of the delay increase and the completion of the SR-TE configuration in OcNOS from the HANMOC log and the OcNOS log in (1-4). • Verify the processing time required for performing (1-7).	☑ Around 3,000ms

☑: Approved, ☑: Partially Approved, ☐: NG

3.5 Analysis

All testing items and results were agreed on by the participant carriers. The following conclusions are based on our analysis in accordance with the items in Table 5.

1. Basic operation verification

We successfully tested and verified all basic operation scenarios. With regard to #1-5, this includes congestion and no-congestion scenarios. In the no-congestion scenario, the user traffic is routed on the default SR path selected by the controller. In the congestion scenario, congestion is detected by the HANMOC probe packets and triggers the path change at the controller. The controller recalculates the optimum path in the congestion scenario and initiates SR path change in the NEs. With the network setup, we observed network flapping in the congestion scenario. To prevent this, appropriate guard times for switching need to be set on the basis of the service and the traffic conditions on the network.

2. Performance verification

- Comparison of delay measurement performance (#2-1)
  As shown in Figures 6, 7, and 8, we verified that high-accuracy delay monitoring by HANMOC. The HANMOC application shows the measurement in 6 μs. While the traditional ping like tool takes 300-600 μs, but HANMOC tool detects the delay significantly better and is closer to the controlled traffic generator average lag of 8.5 μs. This ability to capture delays in such a high accuracy enables the control to react rapidly

to fluctuations. In addition, the control can be adjusted flexibly and rapidly by setting appropriate guard times depending on the use-case.

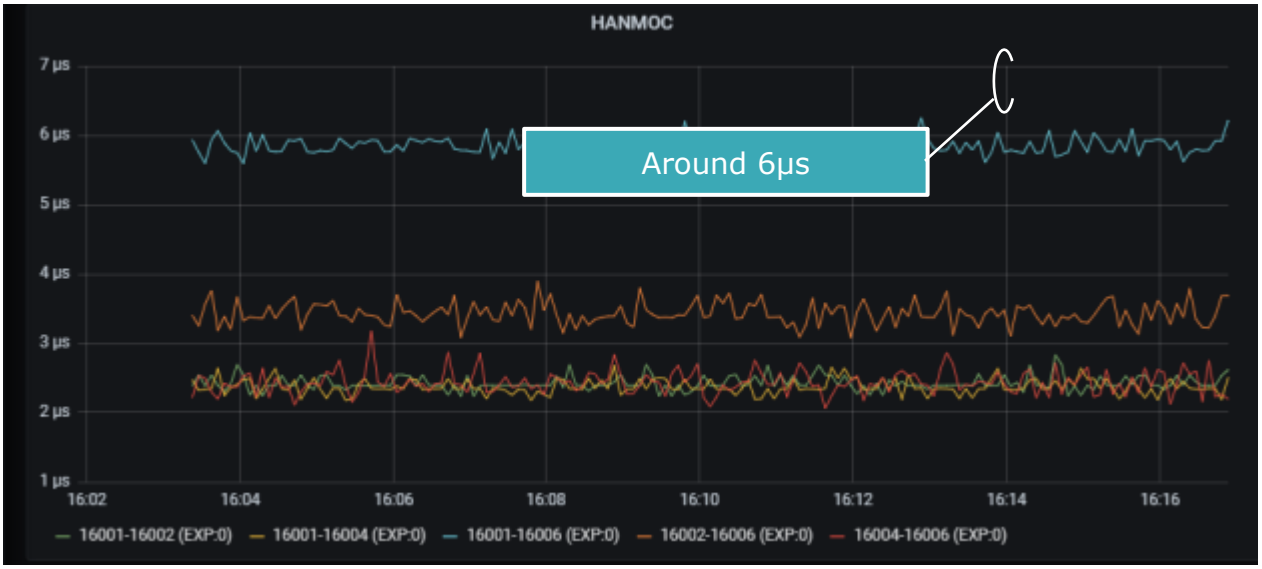

Figure 8. Delay monitoring result by high-accuracy monitoring system

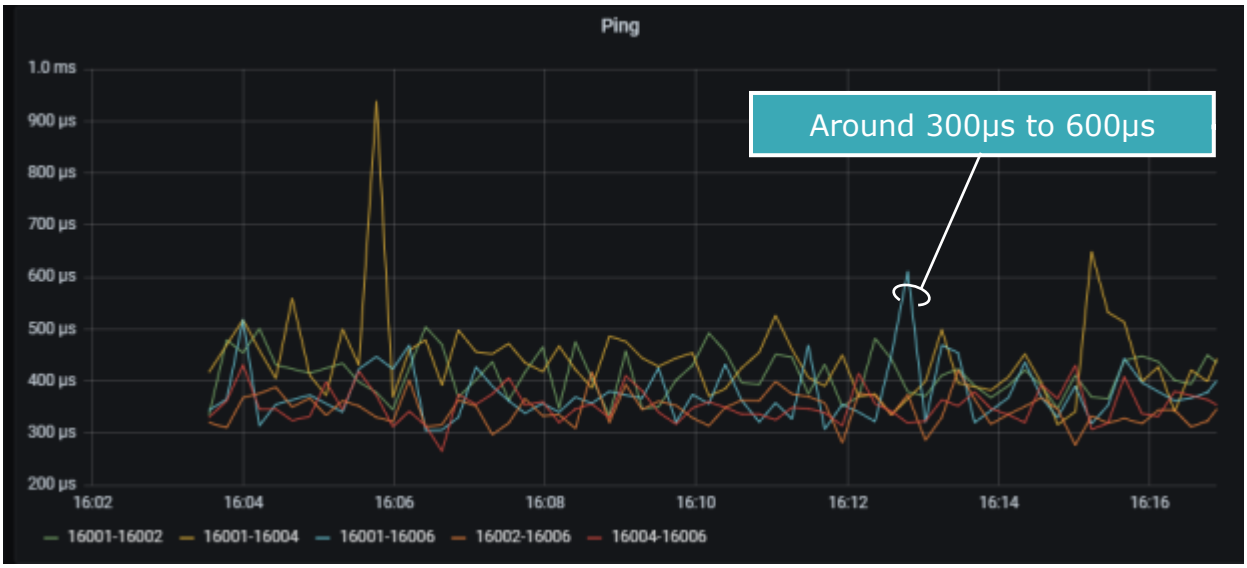

Figure 9. Delay monitoring result by Ping

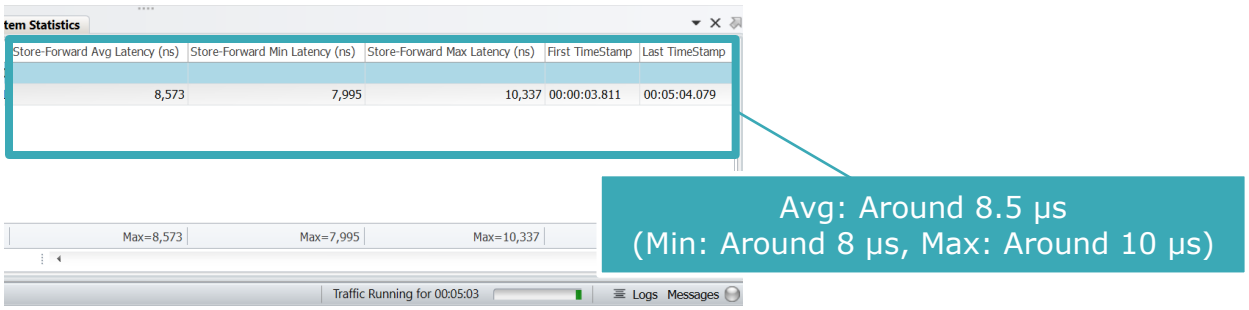

Figure 10. Delay monitoring result by traffic generator

- Impact on user traffic when path switching (#2-2)

  We monitored the status of packet loss in the traffic generator and observed no packet loss during the path switch (i.e., SR-TE). During SR-TE path switching, no packet loss is normal behavior (unlike RSVP-TE). This means that even if the path switching takes some time, there is no concern about service interruption and we have verified that the white-box switch based solution can ensure service continuity.

Table 6. Summary of verification of the packet loss with path switching

Measurement Summary		
Item	Before path switch ODL trigger	After path switch ODL trigger
Traffic generator reported loss [%]	0	0
Traffic generator reported store-forward average latency [nsec]	8,575	8,209
Traffic generator reported store-forward min latency [nsec]	8,000	6,982
Traffic generator reported store-forward max latency [nsec]	10,550	10,550

3. Verify the turn-around time

- Verify the turn-around time of initial path setting (#3-1)
  We verified that the process, i.e. path calculation/configuration, took around 3,000 ms (3 sec) to complete as shown in Figure 9.

- Verify the turn-around time of path setting during service (#3-2)
  We verified that the process, i.e. updating of TED's delay information and path calculation/configuration, took around 3,000 ms (3 sec) to complete as shown in Figure 10.

  Further, we kept both turn-around time within the expected range (10 sec). Regarding the turn-around time of path setting during service (#3-2), what is truly remarkable is

the time between the manifestation of the delay variation and the completion of the path reconfiguration, so turn-around time will be 4~5 sec. However, due to implementation reasons, a series of measurements could not be achieved, so the turn-around time will need to be verified further in the future.

Moreover, because many sections will have to be processed in the commercial network, we will need to further develop the architecture and implementation to ensure and path control can be completed in a shorter time in the future. In addition, we think another issue is whether those processing times can be efficiently maintained during scaling up (e.g., when the number of paths to be monitored is much higher).

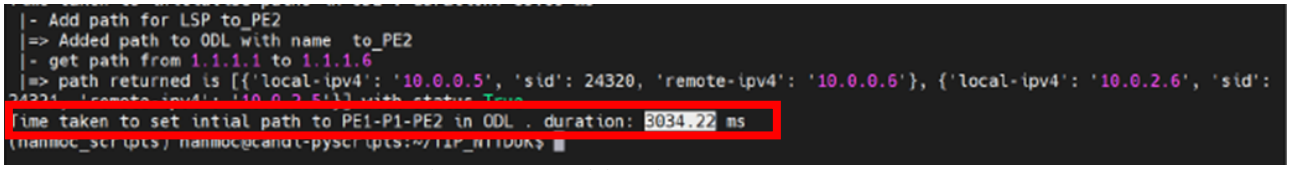

Figure 11. Verification result of #3-1

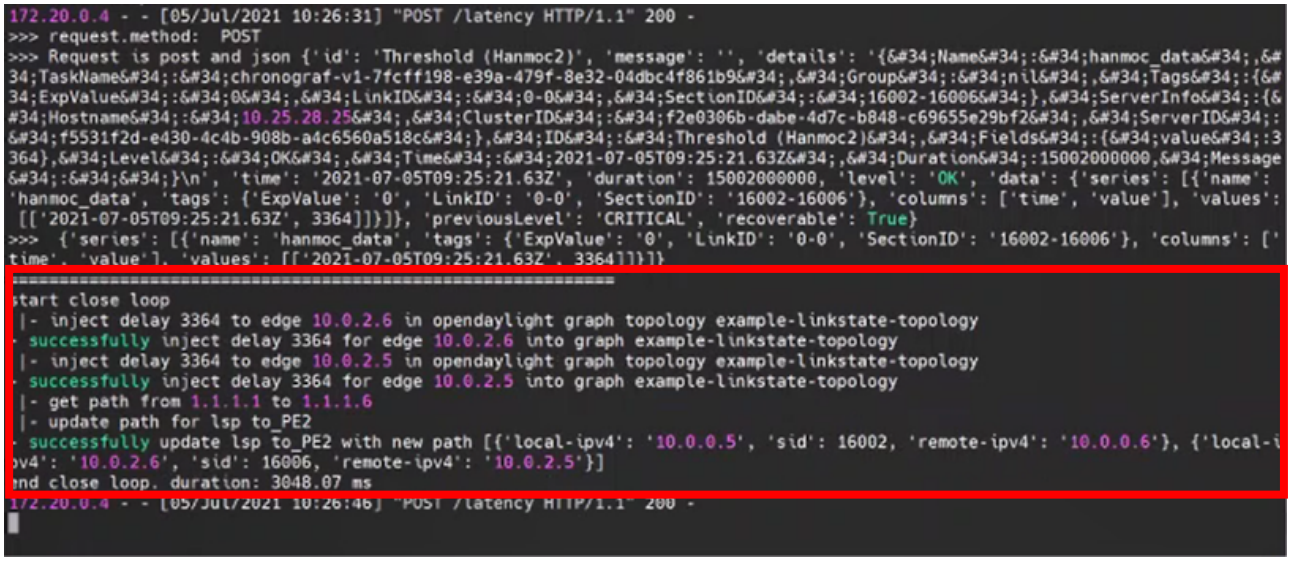

Figure 12. Verification result of #3-2

The results and issues gained from this experiment, we are looking into providing feedback to the community (e.g., the MUST subgroup, which specifies the TIP standards within the TIP OOPT) and to external communities such as ODL and the Open Compute Project (OCP).

4 Future plans

In the future, for a final experiment, we aim to demonstrate integrated control of IP and optical transport. We will further promote open disaggregation in both the IP and optical transport areas and develop automatic path control in optical networks.

5 References

[1] PoC#1 whitepaper (OOPT-CANDI Whitepaper Optical Proof of Concept 2021)

https://cdn.brandfolder.io/D8DI15S7/at/xtpvqj8pb3m569m8bcsx2s/TIP_OOPT_CANDI_Optical_PoC_White_Paper_2021_FINAL_GREEN_ACCESS.pdf

[2] RESTCONF Protocol

https://datatracker.ietf.org/doc/html/rfc8040

[3] Network Configuration Protocol (NETCONF)

https://datatracker.ietf.org/doc/html/rfc6241

[4] Path Computation Element Communication Protocol (PCEP)

https://datatracker.ietf.org/doc/html/rfc5440

[5] TIP OOPT MUST (Mandatory Use Case Requirements for SDN for Transport)

https://telecominfraproject.com/oopt/

[5] OcNOS
https://www.ipinfusion.com/products/ocnos/

[6] Ansible/AWX

https://github.com/ansible/awx

[7] OpenDaylight (ODL)
https://www.opendaylight.org/

[8] SAMCRA (Self Adaptive Multiple Constraints Routing Algorithm)
Piet Van Mieghem and Fernando A. Kuipers, “Concepts of Exact QoS Routing Algorithms”, IEEE/ACM Transactions on Networking, Volume 12, Number 5, October 2004.

[9] BGPCEP project
https://docs.opendaylight.org/projects/bgpcep/en/latest/index.html

[10] HANMOC (High-accuracy monitoring and control)
https://www.rd.ntt/e/research/NS0030.html

[11] Correction: https://git.opendaylight.org/gerrit/c/bgpcep/+/96978 and bug, Report: https://jira.opendaylight.org/browse/BGPCEP-974)

TIP Document License

By using and/or copying this document or the TIP document from which this statement is linked, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to copy, display and distribute the contents of this document, or the TIP document from which this statement is linked, in any medium for any purpose and without fee or royalty is hereby granted under the copyrights of TIP and its Contributors, provided that you include the following on ALL copies of the document, or portions thereof, that you use:

1. A link or URL to the original TIP document.

2. The pre-existing copyright notice of the original author, or if it does not exist, a notice (hypertext is preferred, but a textual representation is permitted) of the form: "Copyright 2019, TIP and its Contributors. All rights Reserved"
3. When space permits, inclusion of the full text of this License should be provided. We request that authorship attribution be provided in any software, documents, or other items or products that you create pursuant to the implementation of the contents of this document, or any portion thereof.

No right to create modifications or derivatives of TIP documents is granted pursuant to this License. except as follows: To facilitate implementation of software or specifications that may be the subject of this document, anyone may prepare and distribute derivative works and portions of this document in such implementations, in supporting materials accompanying the implementations, PROVIDED that all such materials include the copyright notice above and this License. HOWEVER, the publication of derivative works of this document for any other purpose is expressly prohibited.

For the avoidance of doubt, Software and Specifications, as those terms are defined in TIP's Organizational Documents (which may be accessed at

https://telecominfraproject.com/organizational-documents/), and components thereof incorporated into the Document are licensed in accordance with the applicable Organizational Document(s).

Disclaimers

THIS DOCUMENT IS PROVIDED "AS IS," AND TIP MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

TIP WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name or trademarks of TIP may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with TIP and its Contributors.
This TIP Document License is based, with permission from the W3C, on the W3C Document License which may be found at https://www.w3.org/Consortium/Legal/2015/doc-license.html.