Packet-Optical Differentiated Survivability Implemented by P4 Slices and gNMI Telemetry

Rossano P. Pinto[1], Kayol S. Mayer[2], Dalton S. Arantes[2],
Darli A. A. Mello[2], Christian E. Rothenberg[1]

[(1)] DCA, [(2)] DECOM, University of Campinas, Avenida Albert Einstein 400, Campinas, 13083-852, SP, Brazil.
rossano@dca.fee.unicamp.br

Abstract: We demonstrate a packet-optical differentiated survivability mechanism implemented in P4 switches using gNMI telemetry. The controller detects the premium slice interruption and switches it to an alternative path reducing the throughput of the regular slice.

OCIS codes: (060.1155) Software-defined optical networks, (060.4261) Networks, protection and restoration.

1. Introduction

In optical transport networks, path protection allocates a protection path as a contingency resource in case of a failure in the working path. Path protection can be implemented in reconfigurable optical add-drop multiplexers (ROADMs), using a single transponder at the transmitter and the receiver, or it can be implemented having packet switches as end-nodes, requiring duplicate transponders or dual transponders with duplicate client and line interfaces. Although increasing the number of required transponders, implementing path protection in packet switches has the benefits of increasing survivability and allowing to transmit and manipulate traffic simultaneously in both paths [1]. This feature is particularly interesting in modern elastic optical networks, which allow protection paths with lower rates in situations of lack of network resources (*squeezed protection*) [2] or because the connection is longer, yielding a lower optical signal to noise ratio (*soft protection*) [3]. In fact, having duplicate paths originated in packet switches allows to consider them as two paths with different priorities, or, better, to implement them as two or more network slices.

A network slice can be defined as a set of resources allocated to certain tenants. Bandwidth, topology, device CPU, storage, forwarding tables, and other control plane resources can be used to characterize network slices, providing service customization, isolation, and multi-tenancy support. Slices enable heterogeneous application services and product diversity in the physical network infrastructure [4–6]. In the recent years, the P4 programming language has gained traction as a promising alternative for the implementation of network slices [7]. In packet-optical networks, applications of P4 exceed simply slicing, including, e.g., monitoring and telemetry, latency-aware scheduling and forwarding, 5G function acceleration, and in-network AI [8] [9].

In this paper, we propose a hierarchical SDN packet-optical survivability solution based on network slicing offering differentiated reliability. Slices are implemented in the P4 programmable ASIC for traffic prioritization and protection switching. An SDN controller with a hierarchically organized fast-response application is used to detect failures and coordinate global and local controllers with the ability to write data plane pipelines using P4Runtime. The solution resorts to gNMI streaming telemetry as a timely monitoring mechanism.

2. Packet-Optical Differentiated Survivability: A Hierarchical P4 SDN Slicing Approach

The proposed solution features a global SDN controller connected to all optical and packet elements. Local controllers (agents), ideally running on-premise, are responsible for monitoring specific parameters, including link availability. The proposed architecture allows for several levels of hierarchy. In particular, we focus on three, namely a global controller, a local agent, and a data plane agent, as indicated in Fig. 1(a). When an anomaly is detected by the local controller agent or by the data plane agent, a fast-response event is issued. In this paper, global and local SDN controllers work in coordination to reconfigure the network and address the detected failure without introducing instability in the control plane or on established services (a data plane agent is not implemented). We evaluate our solution in an experimental testbed whose optical layer contains two P4 switches and two dual transponders, containing two bidirectional client- and line-side ports each.

Fig. 1(b) shows the complete experimental setup. Two Tofino P4 switches (DCS800 Wedge100BF-32X) are connected to 2 dual Padtec TD-100 boards, comprising two bidirectional optical links (in orange and in yellow). The two TD-100 boards are controlled by a SPVL supervisor accessible via NETCONF. The switches and

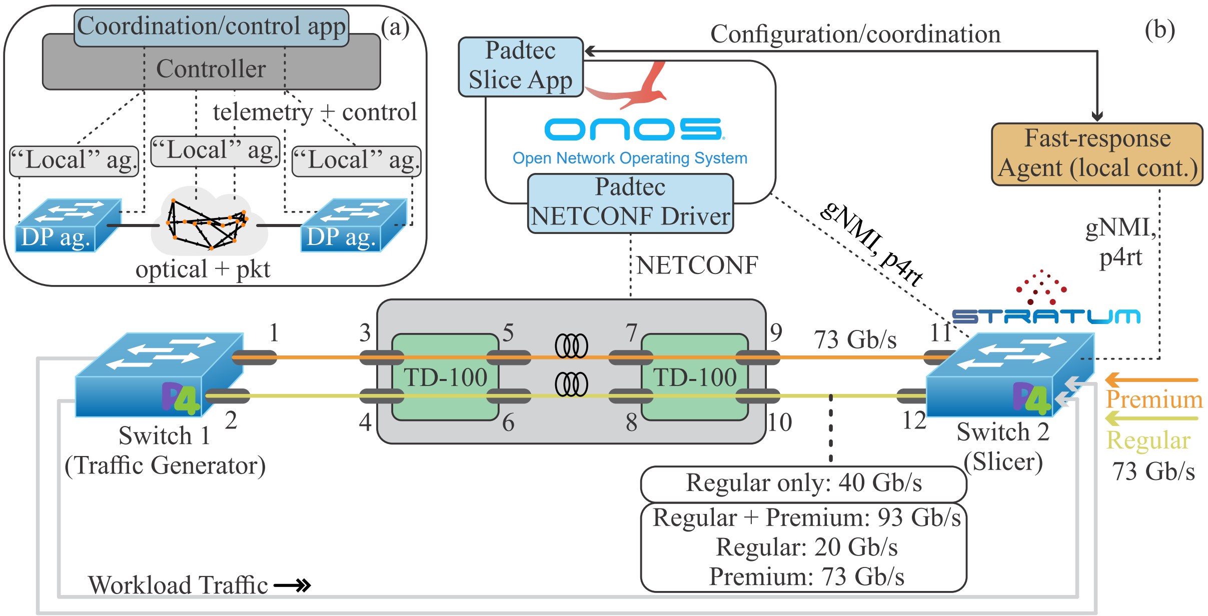

Fig. 1. (a) Hierarchical control plane architecture. (b) Physical experimental setup.

transponders are interconnected by 12 QSFP28 pluggable transceivers configured to operate at 100 Gbps (see numbers 1 through 12 in Fig. 1(b)). All transceivers are connected directly with patchcords. An optical path carrying a premium slice traverses transceivers 1, 3, 5, 7, 9, and 11, and is represented by the orange line. A second optical path carrying a regular slice traverses transceivers 2, 4, 6, 8, 10, and 12, and is represented by the yellow line.

The P4 switch located on the left (P4 Switch 1) acts as a *Traffic Generator (TG)* for the workload production of both the regular and premium slices, each at 73 Gbps throughput. The setup is intended to emulate traffic coming from two tenants connected to the P4 switch on the right (P4 Switch 2). The P4 Switch 2 acts as a network slicer (P4 Slicer) that directs traffic according to link availability. It also shapes the regular slice throughput from 73 Gbps to 40 Gbps before optical layer transmission. When the orange path is down, the P4 Slicer redirects the premium slice from the orange to the yellow path. The premium slice is kept at 73 Gbps, while the regular slice throughput is reduced from 40 Gbps to 20 Gbps, resulting in a combined throughput of 93 Gbps.

The P4 switch that implements the P4 Slicer features *Stratum* as operating system with *gNMI* and *P4Runtime* protocol support. We use ONOS with a custom application as the global SDN controller. We developed a fast-response application that acts as a local controller (agent). Both controllers are hierarchically connected and work in coordination using custom SSH-based IPC messages. In order to emulate a link failure, the laser of transponder 7 is turned ON and OFF in intervals of 30 seconds. Turning OFF the laser of interface 7 generates the sequential shutdown of all lasers in the bidirectional link (orange line). Analogously, reactivating the laser of interface 7 generates the sequential reactivation of all lasers in the bidirectional link.

Both ONOS (the global controller) and the local agent monitor link availability through gNMI. P4Runtime is used to configure P4 table entries of the P4 Slicer switch defining the rules to forward incoming packets to target output ports. When the fast-response agent detects a failure (orange path is down), it changes the table entries in the P4 Slicer pipeline to redirect traffic to the port connected to the yellow path. Also, the local controller instructs the P4 Slicer to change the table entry actions to guarantee 73 Gbps for the premium slice and reduce the regular slice throughput down to 20 Gbps.

3. Results and Discussion

We carried out experiments using two different configurations: *i)* global controller only, *ii)* and global controller acting in tandem with the fast response agent.

Figs. 2(a-c) show the results obtained with the global controller. Fig. 2(b) shows the UP and DOWN states of the link status (yellow curve) and of the protection switch command (blue line). We observe very slow response times, varying from a few seconds to dozens of seconds. Moreover, some detected failures are not adequately handled, as in case IV. Fig. 2(c) shows the rates of the orange and yellow paths in several operation conditions. Fig. 2(a) shows the histogram with the switch times of an evaluated time frame (note that this time frame does not include the 39s event). In the evaluated time-frame, switch times can reach up to 10s.

Figs. 2(d-f) show the results when the fast-response agent is also in use. As expected, the response time is much

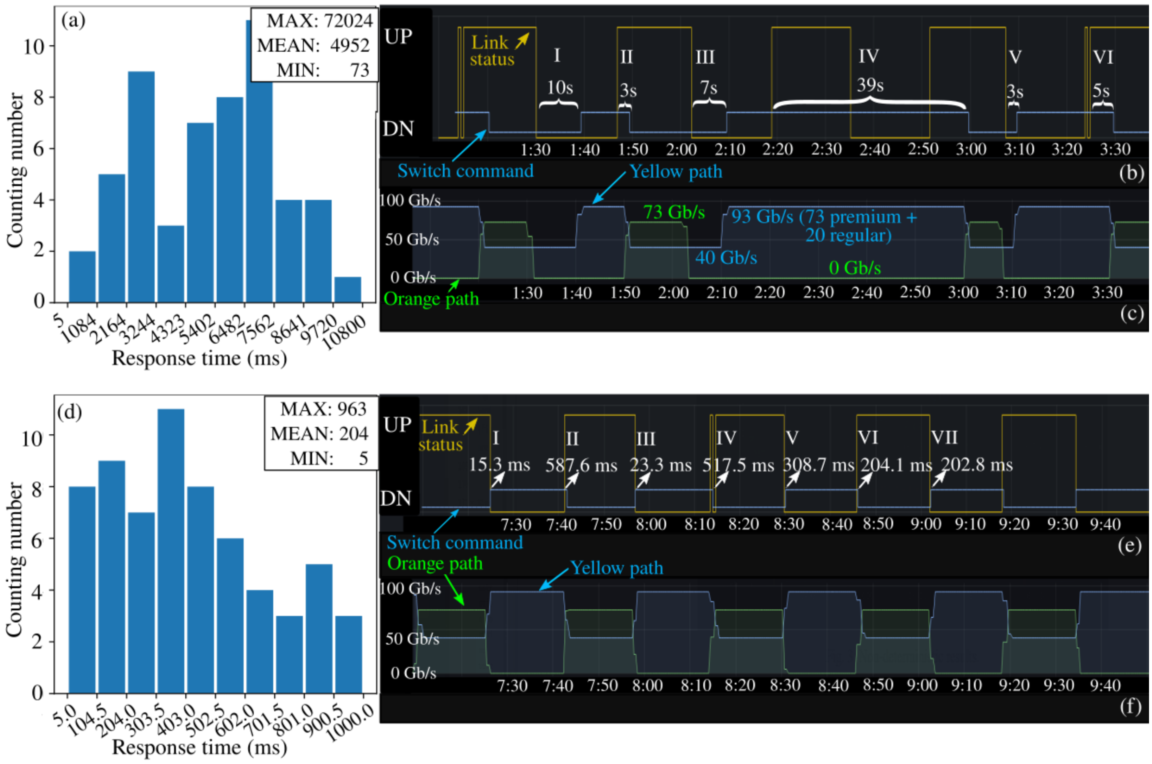

Fig. 2. Response time and throughput. (a-c) Global controller results. (d-f) Fast-response agent results.

faster in comparison with the first scenario, yielding results in the millisecond scale. Fig. 2(e) presents switch time events using the fast-response agent during the experiment. In this case, the switch command follows the link status closely. The throughput curves in Fig. 2(f) also follow the link status behaviour. Fig. 2(d) shows the switch time distribution for the scenario with the fast response agent. The best result, although an outlier, can be seen in case I (15.27 ms). Both the response time and the throughput are much more regular, indicating a viable solution.

Although the obtained switch times exceed in most cases the typical 50 ms expected for optical layer switching, the benefits obtained by the packet-layer switching, e.g., differentiated survivability, can still favor the packet-based solution. The requirement of a 50-ms switch time is controversial for most data traffic applications. In any case, as pointed out in Section 2, Fig. 1(a), another layer of the hierarchical control plane can be implemented in the data plane using P4 constructs, i.e., using a data plane agent (DPA), delivering further improvements in response time, eventually under 50 ms.

4. Conclusion

We demonstrate the implementation of packed-optical differentiated protection based on network slices implemented in P4 switches with gNMI telemetry. Two SDN architectures featuring a global controller and a global controller acting in tandem with a fast-response agent have been investigated. The results indicate that the fast-response agent provides successful differentiated protection switching in sub-second scale.

Acknowledgement. This work was supported by Padtec S/A.

References

[1] A. Sgambelluri et al., “Coordinating pluggable transceiver control in SONiC . . . ,” Proc. of OFC, paper W1G.3, (2021).
[2] R. Goscien et al., “Protection in elastic optical networks,” IEEE Netw. **29**, 88–96 (2015).
[3] D. A. A. Mello et al., “Soft protection in optical networks . . . ,” Proc. of ICTON’2014, (2014), pp. 1–4.
[4] F. Fitzek et al., *Computing in communication networks: From theory to practice,*, 1st ed. (Elsevier, 2020).
[5] I. Afolabi et al., “Network slicing and softwarization: A . . . ,” IEEE Commun. Surv. Tutor. **20**, pp. 2429–2453, (2018).
[6] R. Vilalta et al., “End-to-end interdomain transport network slice . . . ,” Proc. of OECC/PSC, paper TuF3-5, (2022).
[7] E. Hauser et al., “Slicing networks with P4 hardware and software targets,” Proc. of 5G-MeMU ’22, pp. 36–42, (2022).
[8] F. Cugini et al., “Applications of P4-based network programmability in optical . . . ,” Proc. of OFC, paper M4F.3, (2022).
[9] F. Cugini et al., “Telemetry and AI-based security P4 applications . . . ,” J. Opt. Commun. Netw. **15**, A1–A10 (2023).