P4BFT: A Demonstration of Hardware-Accelerated BFT in Fault-Tolerant Network Control Plane

Ermin Sakic
Cristian Bermudez Serna
Siemens AG
Munich, Germany
{firstname.lastname}@siemens.com

Endri Goshi
Nemanja Deric
Wolfgang Kellerer
Technical University Munich
Munich, Germany
{firstname.lastname}@tum.de

CCS Classification: Networks -> Programmable Networks; Systems Security -> Distributed Systems Security;

1 INTRODUCTION

ONOS and OpenDaylight deploy RAFT consensus to enforce update ordering and leader fail-over in the face of controller failures. RAFT is, however, unable to distinguish malicious / incorrect (e.g., buggy [7]), from correct controller decisions, and can easily be manipulated by an adversary in possession of the leader [4]. Byzantine Fault Tolerance (BFT)-enabled controller designs support correct consensus in scenarios where a *subset of controllers is faulty due to a malicious adversary or internal bugs*. Existing proposals [1, 2, 4] base their correctness on premonition that controller outputs are collected by trusted configuration targets and are compared by the same for payload matching for the purpose of correct message identification. Namely, each controller instance of the administrative domain transmits its configuration to the target switch. In *in-band* [6] deployments, where application flows share the same infrastructure as the control flows, the traffic generated by controller replicas imposes a non-negligible network load [3]. Furthermore, comparing and processing controller messages in the switches' control plane incurs additional CPU load [2, 4] and reconfiguration time.

P4BFT introduces the concept of in-network processing nodes, which intercept and collect individually computed controller outputs for matching client requests. After collecting sufficient number of packets to identify the correct message payload, a processing node forwards the correct payload to the destined configuration target. By intercepting control flows in processing switches, and establishing point-to-point connections between processing switches and target destinations, it minimizes the network load imposed by BFT operation. P4BFT realizes the processing node functionality purely using a P4 pipeline, responsible for controller packet collection, correct packet identification and its forwarding to the destination nodes at line rate, thus effectively minimizing accesses to the switches' software control plane and vastly outperforming software-based BFT solutions.

2 P4BFT SYSTEM DESIGN

In P4BFT, network controller instances are configured as a set of replicated state machines, i.e., where each instance calculates its decision in isolation from other controllers, and transmits its decision to the destination switch. Control packets are intercepted by the processing nodes (i.e., processing switches) responsible for decisions destined for the target switch. Consider Fig. 1. Given the placement of controllers and the processing nodes' capacity, with objective to minimize the total control plane footprint and response time, incurred for target configuration switches, P4BFT's Reassigner component identifies the optimal processing node S_2 as the best fit processing node for S_4. The multi-objective formulation further considers the delay metric, available processing capacities at switches (e.g., a hardware-enabled P4BFT node has a higher throughput than the software-based one), and thus minimizes the total traversed critical path between the controller furthest away from the configuration target ($F_D = 3$ in the worst case in Fig. 1, assuming a delay weight of 1 per hop). The resulting assignment additionally minimizes the communication overhead to $F_C = 11$. This is compared to state-of-the-art works [1, 4] that default the processing node assignment to reconfiguration targets, thus resulting in higher F_D and F_C compared to P4BFT.

Reassigner is the component that dynamically reassigns the controller-switch connections based on the events collected from the detection mechanism of the P4BFT switches. Upon detection of faulty controllers, it excludes those from the assignment procedure. It determines a minimum number of required controllers, necessary to tolerate a configurable number of availability failures F_A and Byzantine failures F_M [4], and assigns the controllers to each switch of its administrative domain. Additionally, the Reassigner maps a processing node, in charge of controller messages' comparison, to each destination switch. Thus, switches declared as processing nodes gain the responsibility of control packets collection and forwarding. Reassigner executes once during network bootstrapping and for selected control plane changes (i.e., on

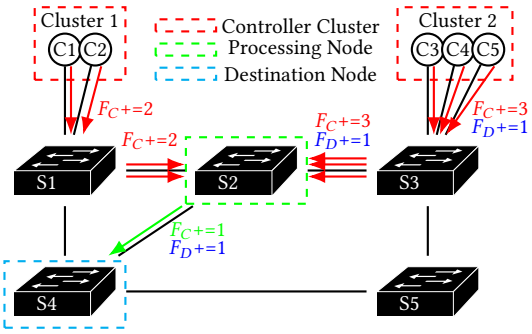

Figure 1: P4BFT's offloading of processing role capability to intermediate switches leads to decreased packet footprint and control flow delays on critical path, F_C = 11 and F_D = 3 hops, respectively.

addition / disconnection of a switch / controller). The optimization output is enforced upon the P4 match-action tables of the switches: i) the *Processing Table*, necessary to identify the switches responsible for comparison of controller messages; and ii) the *Forwarding Tables*, necessary for forwarding of controller messages to processing nodes and reconfiguration targets. Given the user-configurable parameter of required tolerated failures F_A and F_M, Reassigner reports to processing nodes the number of necessary matching messages that must be collected prior to marking a controller message as correct. The details of the P4 control flow, as well as the match-action pairs of P4 tables, are presented in [4, 5].

P4BFT's Reassigner can be deployed as a trusted component of switch, or as a replicated component of the network controller, i.e., in at least $2F_M + F_A + 1$ instances, so to tolerate F_M Byzantine and F_A availability faults of the Reassigner.

3 P4BFT DEMONSTRATION

The accompanying demonstration showcases the practical advantage of P4BFT in 34-switch Internet2 and exemplary topologies (ref. Fig. 1) in a testbed equipped with software and physical P4-enabled switches. The significant load footprint and reconfiguration delay improvements over state-of-the-art works are visualized on a real-time dashboard, similarly to Fig. 2. Furthermore, it is shown how a hardware-based packet comparison can lead to a lowered total reconfiguration delay in scenarios where the capability of a processing role is centralized in a single P4-enabled hardware node, due to the decrease in number of accesses to the software-based control plane. The software switches are instances of the open-source *bmv2* reference switch, adhering to $P4_{16}$ language specification. The hardware-based $P4_{16}$-enabled P4BFT data plane node comprises the *Netronome Agilio CX 10GE SmartNIC*. The option to disable the offloading of processing node capability is implemented for the purpose of comparison to methods presented in existing works. The configurable weight parameters allow for fine-tuning of multi-objective optimization, and thus provide the user with an interface to prefer either minimized communication overhead or resulting reconfiguration latency. The special case, where all control plane packets stemming from replicated controller instances are traversing a single $P4_{16}$ hardware node, demonstrates the advantages of hardware-accelerated packet hashing and comparison, and thus undermines a case for hybrid deployments where the control-plane relies on state-of-the-art control protocols (e.g., OpenFlow / NETCONF+YANG), whereas the P4BFT-equipped edge nodes internalize the processing node capabilities.

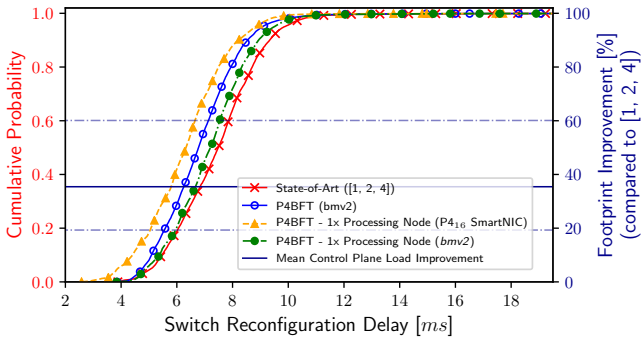

Figure 2: P4BFT's performance gains in terms of control plane load and reconfiguration latency.

Acknowledgment: This work has received funding from the EU in the context of the H2020 project SEMIOTICS (grant agreement number 780315).

REFERENCES

[1] He Li et al. 2014. Byzantine-resilient secure SDN with multiple controllers in cloud. *IEEE Transactions on Cloud Computing* 2, 4 (2014).
[2] Purnima Murali Mohan et al. 2017. Primary-Backup Controller Mapping for Byzantine Fault Tolerance in SDN. In *2017 IEEE Global Communications Conference (IEEE Globecom 2017)*. IEEE.
[3] Abubakar Siddique Muqaddas et al. 2016. Inter-controller traffic in ONOS clusters for SDN. In *2016 IEEE International Conference on Communications (IEEE ICC 2016)*. IEEE.
[4] Ermin Sakic et al. 2018. MORPH: An Adaptive Framework for Efficient and Byzantine Fault-Tolerant SDN Control Plane. *IEEE Journal on Selected Areas in Communications* 36, 10 (2018).
[5] Ermin Sakic et al. 2019. P4BFT: Hardware-Accelerated Byzantine-Resilient Network Control Plane. *CoRR* abs/1905.04064 (2019). arXiv:1905.04064 http://arxiv.org/abs/1905.04064
[6] Liron Schiff et al. 2016. In-band synchronization for distributed SDN control planes. *ACM SIGCOMM CCR* 46, 1 (2016).
[7] Petra Vizarreta et al. 2017. An empirical study of software reliability in SDN controllers. In *2017 13th International Conference on Network and Service Management (IEEE CNSM 2017)*. IEEE.