P4 Postcard Telemetry Collector in Packet-Optical Networks

F. Alhamed, D. Scano, P. Castoldi
Scuola Superiore Sant'Anna, Pisa, Italy

F. Paolucci, F. Cugini
CNIT, Pisa, Italy

I. Verschkov, J.J. Vegas Olmos
NVIDIA, Israel

***Abstract*—Next generation 5G services will require accurate service level verification including the packet-optical metro/aggregation segment. Network telemetry enables such accurate monitoring exploiting in-band and postcard telemetry technologies. However, the massive collection of telemetry data from the network and its processing represent a potential bottleneck subject to scalability issues. In this paper, we propose a two-stage telemetry collector by offloading the postcard telemetry reports processing and aggregation to a programmable P4 switch performing aggregation and correlation to be potentially performed at wirespeed. The experimental evaluation highlights the benefits in terms of reduced CPU and bandwidth requirements at the telemetry server.**

I. Introduction

Accurate monitoring capabilities enabled by pervasive telemetry across the entire packet-optical metro/aggregation network is a key requirement for next generation QoS-guaranteed 5G services. For example, detailed network telemetry enables constant verification of the provided network service level, rapid recovery upon service degradation, and even proactive provisioning based on AI forecasting [1].

Data-plane programmability and the P4 language [2] have been applied to provide accurate latency and congestion monitoring. Two P4-based telemetry technologies have been so far proposed. The first one, named *in-band telemetry* (INT), exploits data plane programmability to extend packet headers with metadata such as the time spent in each traversed switch queue [1]. In this case, at the egress node (only), an INT packet report is generated and delivered to a telemetry collector, providing it with the cumulative performance parameters experienced by the data packet across the network. INT has been extensively investigated in the scientific literature, also in the context of metro/aggregation networks for 5G [3]. The second telemetry technology, named *postcard telemetry*, generates a dedicated packet containing a telemetry report (postcard) every time a data packet to be monitored traverses a node, providing the telemetry collector with relevant per-node and per-data packet metadata parameters [4]. So far, although available in off-the-shelf network nodes, postcard telemetry has not been investigated in the scientific literature.

Either technology has pros and cons. On one hand, INT consumes bandwidth resources due to the introduced extra header, while postcard telemetry does not modify the packet size and format. On the other hand, INT delivers to the telemetry collector just one report per packet while postcard telemetry generates N reports, one for each of the N traversed nodes (see Fig. 1). Furthermore, ingress-to-egress service statistics are directly provided by INT reports while dedicated processing and correlation of N reports is needed at the telemetry collector in case of postcard telemetry. However, per-node performance can be easily retrieved by postcard telemetry while this might not be feasible using INT due to the complex processing needed. Albeit telemetry is applied to selected high-class flows only, a telemetry collector -traditionally implemented in a server- can easily become overloaded with either telemetry technology in terms of amount of received bytes/packets and subsequent processing of report information.

So far, no scientific work has focused on how to efficiently implement the network telemetry collector and improve its scalability and performance. In this work, we focus on the postcard telemetry technology and we design and implement a two-stage telemetry collector where the first stage, addressed in this paper, is designed and implemented using P4 data-plane programmable technology.

II. P4 Telemetry collector

Figure 1 shows the reference network scenario. Packet-optical metro nodes supporting postcard telemetry deliver telemetry reports towards the proposed collector, whose first stage is implemented using a P4 switch (the collector switch). Indeed, by leveraging on its multiple high-speed interfaces, P4 programmability, and stateful capabilities, the P4 switch can efficiently perform a first set of pre-processing operations at wire speed when implemented in hardware ASIC, thus significantly reducing the scalability issues at the subsequent telemetry server. Many different solutions for aggregation and/or correlation can be performed by the P4 switch. The following solutions are considered.

The first solution, named $Agg(R)$, consists in the aggregation of R telemetry reports originated from multiple network nodes in a single packet to be delivered to the telemetry server. In this case, a single register is used at the P4 collector switch to store all the received metadata. When the register reaches R payloads (or a timer expires), a packet including all stored R report payloads is delivered to the server.

The second solution, named $Agg_N(R)$, aggregates data on a per-node basis. That is, up to R reports, originated from the same network node, are appended in a single packet. This solution requires N registers, but it is expected to simplify subsequent per-node correlations at the telemetry server.

978-3-903176-44-7 © 2022 IFIP

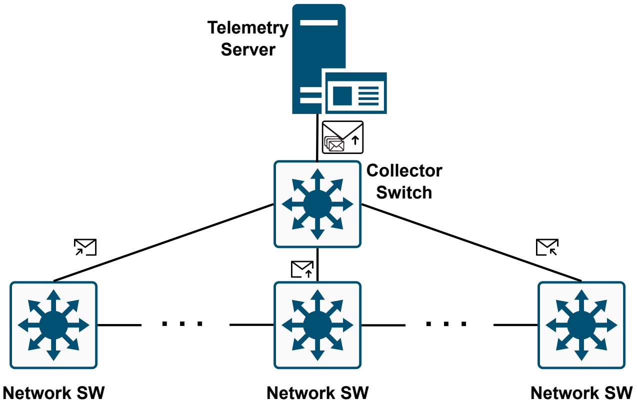

Fig. 1. A telemetry collector aggregating multiple telemetry postcards in one packet before forwarding it to the telemetry server.

The third solution, named $Agg_{N,F}(R)$, aggregates on a per-flow and per-switch basis. This solution requires $F{\cdot}N$ registers, where F is the number of flows to which postcard telemetry is applied. This solution is expected to simplify subsequent per-flow correlations at the telemetry server, also guaranteeing per-node visibility.

Three additional solutions are then derived from the aforementioned ones. In these cases, correlation through P4 computations is performed on the collected metadata directly within the switch. Indeed, simple operations like computing the average, or identifying minimum and maximum values can be efficiently operated using P4 technology at wire speed.

The solutions named $Cor(R)$, $Cor_N(R)$ and $Cor_{N,F}(R)$ expand the previously-introduced related solutions by computing a single metadata representative of the whole set of collected R reports. For example, $Cor(R)$ relies on $Agg(R)$ but instead of sending all R metadata appended in a single packet, it generates a unique message including e.g. the metadata average value. This significantly reduces the number of bytes flowing to the telemetry server. Moreover, these solutions are expected to further simplify correlation operations at the telemetry server.

III. Testbed and P4 program description

A testbed reproducing the reference scenario of Fig. 1 is used to evaluate the performance of the telemetry collector. The topology shown here is a bus topology, however this approach can be extended to any topology and any number of switches as the postcards are normal IP packets that carry the telemetry data, and these telemetry packets (postcards) are forwarded within the network based on their destination IP address. The three network switches implement a standard P4 forwarding pipeline with postcard telemetry capabilities, resorting to the implementation in [1]. In addition, the testbed includes the telemetry collector P4 switch, performing the processing and aggregation of postcards received on different interfaces and from different sources. Finally, the telemetry server is a simple python script that receives, dissects and displays data contained in the postcards. Each node is implemented using a bmv2 software switch running inside an Ubuntu server (CPU AMD EPYC 7262 8-core @3.4GHz, 16GB RAM) and is connected to the actual physical interfaces of that server. The flow-rules are installed in every switch as well as the telemetry Access Control List (ACL) that triggers the postcard reports generation. A traffic generator is used to generate a custom number of flows. The aforementioned solutions are studied to evaluate the impact on the telemetry server by aggregating a different number of postcards, and by pre-processing the postcards before sending them to the server. To demonstrate the scalability of the telemetry-collector switch, the postcards are generated at every network switch at a rate of 10,000 packet per second (pps) and carry telemetry data that includes ingress and egress timestamps of each packet, queue length at the network interface, as well as switch and flow IDs.

The baseline P4 collector program follows the standard bmv2 architectural model and includes a parser, two pipelines (i.e., ingress and egress) and a deparser block. The ingress flow table matches the destination IP address and the Report UDP port. Upon each report reception, the table triggers actions on telemetry data collection in dedicated P4 registers, increase a dedicated P4 counter and drops the packet. When the counter reaches the pre-defined value R, the pipeline triggers the counter reset and an additional action that loads and encode all the stored telemetry data in a single Report packet to be forwarded to the server. In the Agg solutions, telemetry data are stored in the registers, while in the Cor ones they are subject to additional processing to compute correlations and statistics, while only overall correlated data (e.g., max, min, average latency) are stored in registers. The egress pipeline is not used and left as bypass.

IV. Experimental Results

Figure 2 shows the experimental results achieved by the $Agg(R)$ solution in terms of CPU load at the telemetry server for three different aggregation levels (i.e., different R values). In the first baseline case (red dashed line), the collector switch operates as a normal switch by forwarding the postcards to the telemetry server without any aggregation or processing (i.e., R=*1*). The result is that the telemetry server is overwhelmed by the amount of packets received per second, which in this case amounts to 30,000 pps and the CPU load is constant at a 100% during the experiment run-time. In the second case, $Agg(15)$ (represented by the green continuous line), the collector-switch is programmed to collect the telemetry data from R=*15* postcards in a first come first serve manner regardless of the source or the interface on which the postcard was received, and then aggregates the collected data in one packet before sending that packet to the telemetry server. In the aggregated packet, the headers (e.g Ethernet, IP, and UDP headers) are removed from the postcards and only the telemetry data is aggregated in the payload field of the new packet. That is, the content of R=*15* packets, each of 90 bytes, is aggregated in a single packet of 370 bytes. Despite the larger packet size, a significant improvement in the loading of the telemetry server is achieved: the CPU load dropped to around 50% given the fact that now the amount of received packets is reduced by a

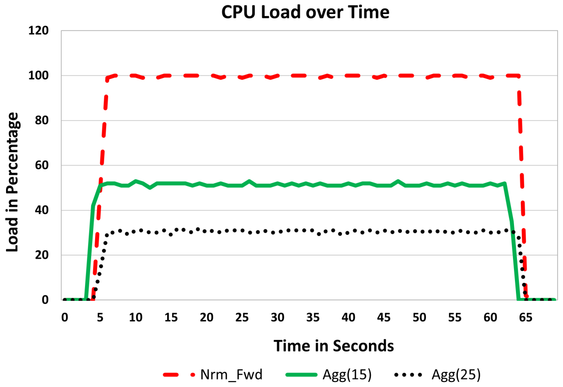

Fig. 2. CPU usage of the telemetry server.

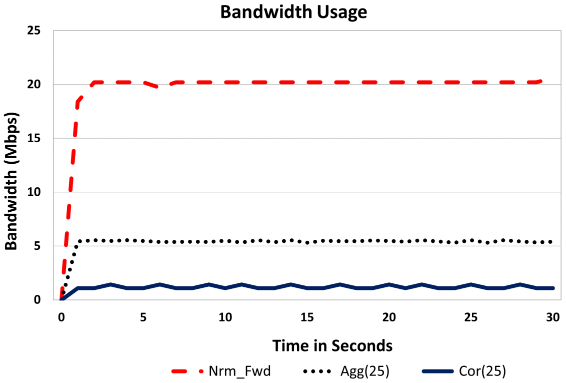

Fig. 3. Bandwidth usage measurement in the switch-server link.

factor of 15 down to 2,000 pps. The third case (the black dotted line) is similar to the second except that the aggregation level is increased to R=25 postcards in one packet (i.e., $Agg(25)$), and the CPU load of the telemetry server is further reduced to around 30% during the experiment run-time. However, it is worth to highlight there are two constraints to be considered on the number R of aggregated postcards: a) the length of the postcard itself, and b) the delay that might be introduced in the arrival of the telemetry reports to the telemetry server that results from the store-and-forward behaviour at the collector switch. The latter is the time spent in the P4 register by the first postcard metadata until all R metadata are received and a report is delivered to the telemetry server. It depends on the arrival rate of the flows. This latency may become relevant for the $Agg_N(R)$ and $Agg_{N,F}(R)$ solutions. In the considered experiments, an $Agg(25)$ scenario added a delay of $\sim$ 830µs and in the case of $Agg_{3,4}(25)$ the added delay is $\sim$ 10ms.

In this work, given the basic implementation of the telemetry server, no remarkable differences have been measured comparing the two types of Agg and Cor solutions in terms of CPU load. However, significant differences can be measured in terms of bandwidth usage.

Fig. 3 demonstrates the difference in the bandwidth usage of the telemetry packets that are forwarded from the collector switch to the telemetry server. The first case is shown in red dashed line and is the same baseline case $Agg(1)$ with normal forwarding and no processing of the postcard telemetry packets. The resulting traffic in this case is more than 20 Mbps. This number depends on the number of the network switches that are generating telemetry postcards, on the rate of generating these postcards, and on the size of the telemetry data that is being collected. The second case (the black dotted line) is when the collector switch aggregates 25 postcards in one packet without further processing (i.e., $Agg(25)$)- The bandwidth is reduced to around 5 Mbps as not only the number of packets is reduced, but also the overhead from the headers of the postcards which, in our scenario, was longer than the payload that contains the telemetry data. In the third case, $Cor(25)$ in Fig. 3 (shown in blue continuous line), the collector-switch performs an extra processing on the collected postcards and sends a summary of the data (e.g. max and min latency at each switch, max and min queue length). This approach leads to further reduction in the bandwidth to around 1 Mbps, with clear potential of further reducing the CPU usage on the telemetry server. However, this last case depends on the number of switches and monitored flows and may require more memory at the collector switch to sort the collected postcards as well as extra calculation power.

V. Conclusion

The proposal in this paper demonstrated the potential of a P4-capable switch to significantly offload a telemetry server by processing and aggregating the telemetry data, and showed promising results as the CPU load of the telemetry server was decreased by 70%. At the same time, the bandwidth usage decreased by around 75% with the aggregation of 25 postcards, and can be decreased even further by performing the correlation of the postcards at the collector switch. Even though the ideas discussed in this paper are related to telemetry scenarios, they are still applicable to other applications and services within the network that can tolerate the small delay emerging at the collector switch, freeing some resources on the servers that then can be allocated to handle other services.

Acknowledgment

This work received funding from the ECSEL JU project BRAINE (grant agreement No 876967). The JU receives support from the EU Horizon 2020 research and innovation programme and the Italian Ministry of Education, University, and Research (MIUR).

References

[1] D. Scano, F. Paolucci, K. Kondepu, A. Sgambelluri, L. Valcarenghi, and F. Cugini, "Extending P4 in-band telemetry to user equipment for latency- and localization-aware autonomous networking with AI forecasting," *Journal of Optical Communications and Networking*, vol. 13, no. 9, pp. D103–D114, 2021.

[2] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[3] F. Paolucci, F. Cugini, P. Castoldi, and T. Osiński, "Enhancing 5G SDN/NFV edge with P4 data plane programmability," *IEEE Network*, vol. 35, no. 3, pp. 154–160, 2021.

[4] H. Song and et al, "In-Situ OAM Marking-based Direct Export," IETF, Internet Draft, draft-song-ippm-postcard-based-telemetry-11, Nov. 2021.