Per Packet Distributed Monitoring Plane with Nanoseconds Measurements Precision

Guillaume Soudais[(1)], Sébastien Bigo[(1)], Nihel Benzaoui[(1)]

[(1)] Nokia Bell Labs, 91620 Nozay, France, guillaume.soudais@nokia.com

Abstract: *We propose a per-packet distributed monitoring plane for time-sensitive optical networking. Over a packet switched network and using our FPGA-based prototype, we demonstrate latency measurements of 6.4ns precision per hop with an offset of only 1.22µs.*

Introduction

5G and industry 4.0 opened the door for time sensitive applications, with growing requirement stringency. Extensive research efforts have focused on designing networking solutions[1] capable of strict performance guarantees. For scalability, cost and feasibility, these solutions should not be dedicated networks but need to expand into a single infrastructure transporting all types of applications, while ensuring all agreed service levels. Then, time-sensitive applications are more likely exposed to events altering performance. Therefore, a monitoring plane is crucial to 1) verify that service level agreements are met *per-application* and 2) reports guarantee violations, to allow the network control plane to react before service failure.

Early reports about performance monitoring provided insight on latency variations out of early indicators of network congestion such as traffic burst detector [2]. Other works derived one-way latency by halving the round-trip time [3], sometimes leveraging telemetry from a control plane [4], but all by probing the network performance with dedicated packets, instead of analyzing the packets of the flows of interest. Therefore, they could not reliably inform on the latency of each and every time-sensitive flow.

In this paper, we propose a monitoring plane, capable of reporting *precise* (ns scale), *per packet* latency measurement in any optical network. The monitoring plane consists of a network of stand-alone devices to be placed only on paths to monitor. In the following, we describe the proposed monitoring plane and its implementation. We evaluate the accuracy of the synchronization and the per-packet monitoring measurements precision.

Concept

We propose a monitoring plane (Fig. 1) composed 1) of standalone devices capable of measuring key performance metrics such as latency, jitter, and traffic pattern with a nano-s precision and at *packet granularity,* inserted at both ends of paths where time-sensitive flows travel, and at some intermediate networking nodes (e.g., at ToR level), 2) of dedicated point-to-point links between pairs of monitoring devices for synchronization and telemetry.

The first monitoring device, at the ingress, 1) intercepts the packet coming from the source, 2) inserts a timestamp in the payload of the packet, behind the Ethernet (802.3 [5]) header to allow for normal switching behavior, 3) recalculates the Forward Correction Sequence (FCS) to account for the new bits in the packet and 4) forwards the packet to the network so that it follows its original path. Note that timestamp insertion and packet forwarding are done in a cut through mode to minimize the added latency. The second monitoring device, at the egress, 1) receives the packet and extracts the timestamp from the packet payload to reconstruct it in its original format (including FCS recalculation), 2) uses the timestamp to calculate the delivery latency experienced by the packet in the network. Jitter can be derived from the measured latency variation, and traffic pattern by analyzing the packet arrival time statistics, 3) forwards the packet to the destination client. As described in Fig. 1, a single device can be used to monitor

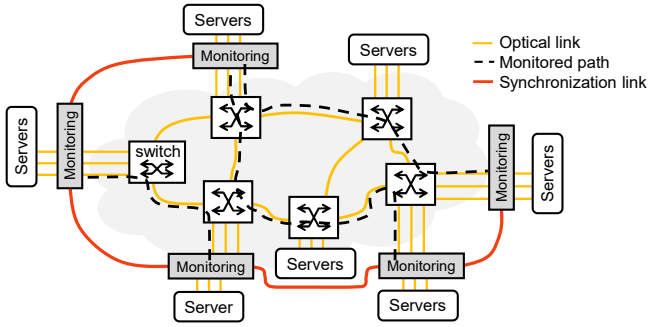

Fig. 1: Monitoring device placement in network

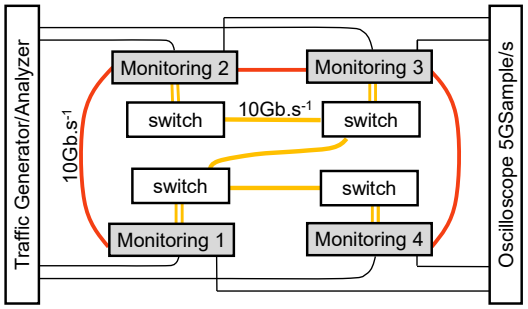

Fig. 2: Experimental setup

978-1-6654-3868-1/21/$31.00 ©2021 IEEE

several flows, as long as they are discriminated with their port addresses (e.g. MAC) contained in the packet header.

In order to accurately compute latencies; all monitoring devices need to share a common time reference. For this purpose, we select one monitoring device (master) to be the clock reference. This device can be connected to a GPS clock or a trusted reference clock. Should this device fail, the monitoring plane could recover from another master using a kind of spanning tree protocol, not implemented here. We propagate synchronization hop by hop through dedicated point to point fiber links, where each adjacent (slave) device synchronizes its clock and frequency[6] with respect to the previous one. We perform clock synchronization using Precision Timing Protocol (PTP, IEEE 1588[7]) down to sub µs range, whereby the master device initiates a PTP timestamp exchange for assessing the time difference with the next device. Once the slave device is synchronized, it acts as a master, and starts a new PTP exchange with the next device in the daisy chain. It can be predicted that the quality of synchronization increases with the frequency of PTP exchanges, which is upper-bounded by the inverse of twice the round-trip time. We therefore recommend to use our monitoring plane in edge cloud scenarios, where propagation distances are moderate. To increase the quality of synchronization further, we perform frequency matching (syntonization) between adjacent devices. We retrieve the master clock frequency from the Phase Lock Loop (PLL) of the slave device receiver and use it to tune the Digitally Controlled Oscillator (DCO) of the slave device transmitter. After Syntonization, the master-slave clock frequency mismatch is essentially determined by the accuracy of the DCO, i.e. 10 part per billion. Advantageously, the mismatch remains contained after a link failure for a few 10s, enough time to update the synchronization scheme before further damage.

Experiments

We implemented our prototype monitoring device on HTG930 development board using the Xilinx sdevices where we assess synchronization and monitoring precision. We use 10G Ethernet ports available in our FPGA for both the data plane and the monitoring plane, but only ~1-100Mbps rate would be required for the monitoring plane. For PTP implementation, we use hardware (FPGA) timestamping, providing a lower-bound precision of 6.4ns, i.e. the FPGA clock cycle. We perform syntonization every 1s, which offers frequency mismatch down to 1Hz, as evidenced by the detection of at most one clock cycle discrepancy. Using the testbed described in Fig. 2, we evaluate 1) the accuracy of the synchronization of monitoring devices and 2) the precision of the per-packet monitoring measurements.

1. Synchronization accuracy

At each monitoring device, we generate a periodic rectangular waveform at a rate of 1kHz, and 1Hz, and record short-term (minute-long) and longer term (day-long) drift of synchronization, respectively. In Fig. 3, we report oscillograms showing the phase drift over 32s between pairs of devices of the chain, with one (top) or three hops (bottom), with and without syntonization, at PTP frequencies of 1.2Hz or 600Hz. The oscilloscope is triggered by the leftmost device (Fig. 2) from the pair. We can observe that the erratic behavior of the phase shift obtained with PTP only is drastically reduced with syntonization and that combining syntonization with more frequent PTP exchanges brings quite efficient phase locking

Fig. 4 shows the distribution of the phase shifts over a 15-hour period between all pairs of chained devices and for multiple PTP exchange frequencies, as provided by the oscilloscope. On a single point to point link (1 hop), using high PTP frequencies (1192Hz, or even 19Hz), the phase difference is found uniformly distributed within the time range of one clock cycle, i.e. 6.4ns. The syntonization is accurate enough to keep synchronization within the PTP exchange period. When the frequency of PTP decreases (1.2Hz, 0.3Hz) we can see the edges of the distribution turning less steep, as evidence that syntonization by itself is not sufficient to maintain accurate phase synchronization, but it is capable of

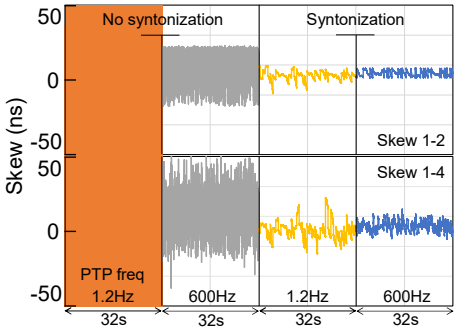

Fig. 3: Short-term evolution of synchronization

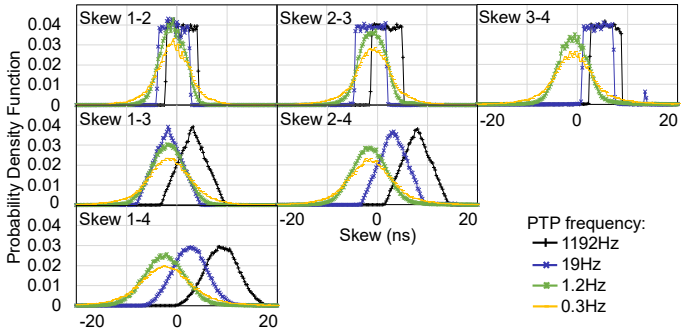

Fig. 4: long-term (15h) distribution of skews between different devices

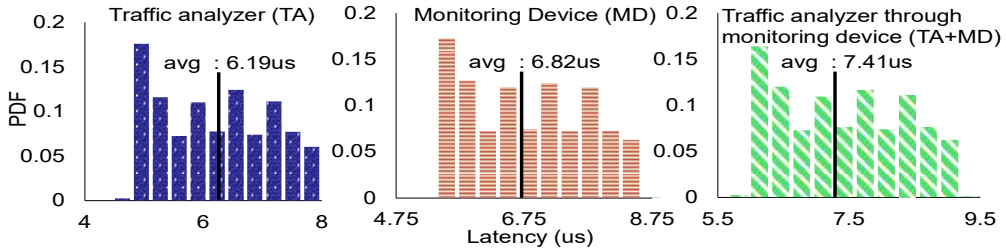

Fig. 5a: Latency distribution measured by the traffic analyzer

Fig. 5b: Latency distribution measured by the monitoring device.

Fig. 5c: Latency distribution measured by the traffic analyzer through the monitoring device

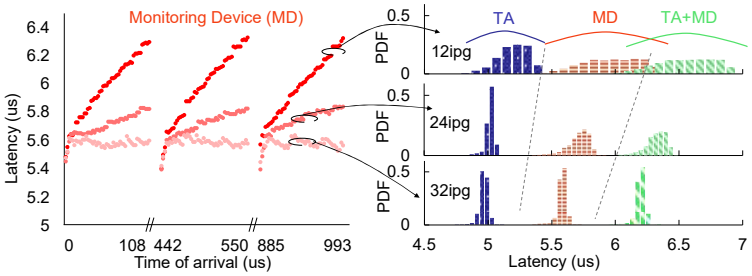

Fig. 6: Latency evolution of burst of 50 packets for different interpacket gap

Fig. 7: Latency distribution of 50 packets for different interpacket gap

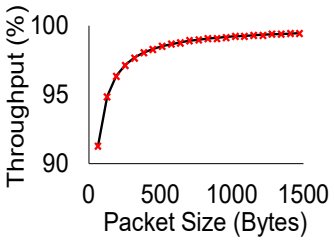

Fig. 8: Throughput for CBR traffic of different size

matching the frequency to less than 10ppb (i.e. within one clock cycle).

Not surprisingly, over two (resp. three) concatenated links, the recorded distributions are well approximated by the sum of the distributions of two (resp. three) individual links, therefore spanning over a width of 2x6.4ns (resp 3x6.4ns). After two links, the distribution exhibits a typical triangular shape, as particular case of an Irwin-Hall distribution for n hops, as predicted by theory. The distributions are also found shifted by one clock cycle after each additional link, which we attribute to the synchronization latching to successive clock cycles. We therefore believe that we can safely extrapolate the performance of this synchronization method for a chain of any length.

2. Latency accuracy

We now evaluate the measurement precision of the monitoring device, while highlighting the impact of the monitoring plane onto the performance of the data plane. We run 2 competing flows (constant bit rate CBR at 2.5Gb/s each with respectively 256B and 1412B packet sizes) over a switched network of 4 nodes. The network is fed with traffic from a traffic analyzer (Spirent, 10ns time precision) which we also use for performance measurement as reference for benchmarking with our prototype. In Fig. 5 a) and b) we report the distribution of latencies of the (256B) CBR flow. We observe that the distributions in a) and b) are very similar (<1% in the histograms), except a deterministic shift of 0.63 µs, which corresponds to the serialization/deserialization time of the FPGA for timestamping. When interfacing inserting our device in the traffic analyzer path, we can extract the latency added by our device. We obtain 1.22 µs, i.e. twice the deterministic shift, as expected from back and forth timestamping.

Further insight on how our device impacts the system latency by timestamping is provided by periodically sending bursts of 50 consecutive packets, with inter-burst time of 330us, long enough to reset the FPGA queues, while varying the interpacket gap (ipg) inside the burst. We report in Fig. 6 the packet-by-packet latency evolution provided by our device and the latency distribution measured in the same conditions as Fig. 5. With 12B of interpacket gap, the minimum gap allowed by the Ethernet protocol, we observe that the latency increases linearly as new packets come in, until the queues are reset. This is due to the delay introduced by timestamping operation. This alteration on the traffic pattern is visible only at for very short interpacket gaps but having a packet separation of 32B is enough to absorb the timestamping delay.

In a last experiment, we assess the impact of the device on the bandwidth utilization. We loop back the device on the traffic analyzer and feed it with CBR traffic. In Fig. 8, we measure the relative throughput, with varying packet sizes. It is found reduced because of (constant) monitoring overhead. Smaller packets experience higher restriction, down to 91.25% (+-0.1%) in the worst case, i.e. with 64B packets.

Conclusion

In this paper we proposed a distributed monitoring plane for time-critical traffic. Precision is guaranteed by the tight synchronization at 6.4ns per link with offset of 1.22µs. Our per-packet method of monitoring slightly dilates latency of bursty traffic, but this effect vanishes by increasing the minimum interpacket gap.

References

[1] IEEE, Inc. Time-Sensitive Networking Task Group, 2016.

[2] R. Joshi et al., "BurstRadar: Practical real-time microburst monitoring for datacenter networks", *Proc. 9th Asia–Pac. Workshop Syst.,* pp. 1-8, 2018.

[3] A. Atary and A. Bremler-Barr, "Efficient Round-Trip Time monitoring in OpenFlow networks", *Proceedings - IEEE INFOCOM*, pp. 1-9, July. 2016.

[4] Lingxia Liao *et al.*, "An efficient and accurate link latency monitoring method for low-latency software-defined networks", *IEEE Trans. Instrum. Meas*., vol. 68, no. 2, pp. 377-391, Feb. 2019.

[5] IEEE Standard for Information Technology–Telecommunications andInformation Exchange Between Systems–Local and Metropolitan AreaNetworks–Specific Requirements Part 3: Carrier Sense Multiple AccessWith Collision Detection (CSMA/CD) Access Method and PhysicalLayer Specifications - Section Three, IEEE Std. 802.3-2008, 2008.

[6] Timing characteristics of a synchronous Ethernet equipment slave clock (EEC), ITU-T Std. G.8262, 2007.

[7] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Std. 1588-2008, 2008.