PhD Forum Abstract: Ultra-low Latency Communication in TSN-based Virtual Environments

PhD Forum Abstract: Ultra-low Latency Communication in TSN-based Virtual Environments

Andrea Garbugli
dept. of Computer Science and Engineering (DISI)
University of Bologna
Bologna, Italy
andrea.garbugli@unibo.it

***Abstract*—The extension of cloud computing concepts to edge devices will lead to the coexistence of a wide range of applications with heterogeneous quality of service (QoS) requirements. Hence the need to move towards a more fluid model based on a continuum of virtual resources. In this paper, we propose a network virtualization model to support applications with ultra-low latency communication requirements and finally compare our results with those of a physical network.**

***Index Terms*—time-sensitive networking, cloud continuum, network virtualization**

I. Introduction

The consolidation of cloud computing and Internet of Things (IoT) concepts is driving an exponential growth of connected smart devices in many application domains, such as Industry 4.0, Smart Cities, Healthcare, and connected vehicles. This evolution will lead to applications with different quality requirements to coexist with each other, using heterogeneous technologies and resources, i.e., communication protocols, storage, computing capacity, energy requirements, and security [1].

This coexistence can be achieved through the integration of different virtual/physical resources into a hierarchy of edge/fog nodes, giving rise to a more fluid computing model identified as Cloud-to-Thing Continuum (C2TC). Through the C2TC model, we provide slices of the resources to different applications to satisfy their requirements, guaranteeing isolation and distributing the workload at all levels of the infrastructure. Furthermore, a set of network functions and resources can be assigned to a specific service and then chained together to create an end-to-end (E2E) network able to support stringent QoS requirements. Such a model opens up space for decentralized and hierarchical programmable network architectures and the use of ultra-reliable, low-latency communication based on heterogeneous wired/wireless protocols, such as 5G, Wi-Fi 6, DetNet, and Time-Sensitive Networking (TSN) [2].

In this work in progress, we present a network slicing and virtualizing approach to supports ultra-low latency communications. To this end, let's first discuss virtualization and communication technologies to support real-time applications. Next, we propose a network virtualization model that allows multiple virtual machines (VMs) to use TSN-based communication, and finally, we compare the E2E latency supported by our approach with that achievable through communication that does not use virtualization.

II. Background and Related Work

In this section, we provide a concise introduction to virtualization and networking technologies for applications with stringent QoS requirements.

Such applications often need to rely on a synchronization mechanism. Usually, in networks with strict real-time QoS requirements, all communication participants need a unique time reference. In the context of TSN, there is a standalone protocol, namely IEEE 802.1AS, which specifies a specialized profile of the IEEE 1588 standard and extends the Precision Time Protocol (PTP). This extension called the generic Precision Time Protocol (gPTP) defines two main entities: a (i) Clock Master (CM) and (ii) Clock Slave (CS). Network participants have one of the latter associated with their network devices, which is used during the synchronization process of the device's clocks [2].

The second TSN standard aimed at supporting real-time traffic is IEEE 802.1Qbv. The latter relies on gPTP to introduce timed communication windows to support different types of time-critical flows. These windows, called time-aware traffic windows, are divided into multiple time slots that repeat cyclically. The latter are then used by different traffic flows associated with different traffic classes to have guaranteed low latency and jitter and prevent best-effort traffic from interfering with them. Specifically, packets belonging to a traffic class are buffered until a so-called guard band allows them to be transmitted. Implementatively, windows, and slots are expressed through a Gate Control List (GCL) which identifies the moments in time in which one or more queues, associated with the different traffic classes, are open for packet transmission [2].

Recent developments have made it possible to run complex real-time applications in virtual environments. In [3] it is shown that Xen and KVM are perfect candidates as real-time hypervisors. In addition, KVM's support for virtual PTP clocks and virtio-based, multi-queue network interfaces is perfectly suited to the virtualization of TSN-based applications.

In [4], three different approaches are proposed to enhance hypervisors and support TSN communication. The approaches are then evaluated through simulations and then compared with each other showing the pros and cons of each of them.

Gunda et al. [5] introduce a container-based architecture

that allows for flexible reconfiguration and redeployment of process control systems. Then, they evaluate their proposal through a PTP-synchronized testbed, demonstrating that deterministic QoS requirements can be met.

In this preliminary work, we assess the feasibility of TSN communication between VMs deployed in different physical hosts. This is achieved by providing the guests with a multi-queue virtual network interface (vNIC), and a virtual real-time clock for the synchronization.

III. Virtualized TSN Communication

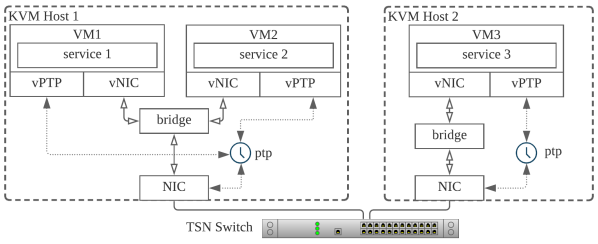

Fig. 1. Virtualization model architecture for TSN-based networks.

Figure 1 shows the components involved in our TSN networks virtualization approach. As an example, we can think of an industrial control application composed of multiple virtual Programmable Logic Controllers (PLCs) distributed as composable services.

As previously explained, a communication that supports ultra-low latencies, such as those we can obtain with IEEE 802.1Qbv, requires a complete synchronization between all the clocks of the TSN network participants. The approach we have chosen is to deploy a PTP service on the hosts, which will then be synchronized with the rest of the network. At this point, the VMs will be provided, via KVM, with a clock that keeps track of the real-time clock of the host, effectively creating a virtualized PTP clock (vPTP) synchronized with the whole network and usable as a clock associated with the virtualized network device which we will discuss next. Finally, an NTP daemon runs in the VM that uses vPTP as a reference to synchronize the system clock.

As for the implementation of time-aware traffic windows, as in IEEE 802.1Qbv, we need a virtual network interface with support for multiple transmission queues, so as to associate each queue with a different traffic class. In our architecture, VMs have associated a virtio-based vNIC with multi-queue transmission support. Usually, a single queue allows for separate support for receiving (RX) and transmitting (TX) packets. To work, a vNIC with multiple queues must be associated with a VM with 2 or more virtual CPUs (vCPUs), this is because each queue (RX and TX) must have its associated thread.

At this point, vNICs are connected to the physical network through a virtual bridge which is in turn associated with 1 or more physical network interfaces. In this way, we can provide the VMs of a TSN-based communication regardless of the physical NIC and the communication network used at the underlying level.

IV. Preliminary Results and Conclusions

To evaluate the effectiveness of our approach to the virtualization of a TSN-based communication, we set up a latency test between two VMs distributed on two different UP Xtreme boards equipped with TSN network interface (hosts) and connected via a physical TSN switch. The VMs and the hosts are configured according to the approach described in the previous section. Then, we used an application that sends UDP packets with a publishing cycle of 1ms between the two VMs and leveraging the vNICs. The test was then repeated directly on the TSN physical network using the same application deployed on the physical hosts. Figure 2 shows the comparison of E2E latency times in the two scenarios. As you might expect, given the overhead introduced by virtualization, communication between hosts has lower latency than communication between two virtual machines. This difference is probably introduced by the steps a packet has to go through between kernel and userspace. However, even in the virtualized case, the measured latencies meet the QoS requirements required by ultra-low latency communication such as those based on TSN.

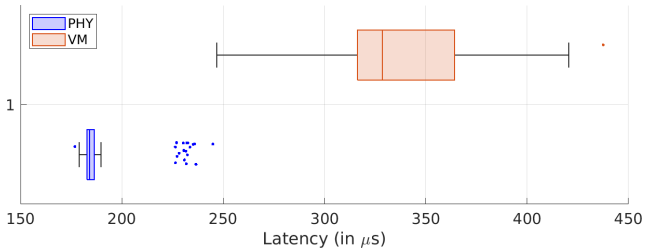

Fig. 2. End-to-end latency comparison.

In this paper, we introduced our approach to virtualizing communications with ultra-low latency constraints. We subsequently demonstrated its effectiveness through a real testbed. Currently, our proposal is based on a single virtualization and communication technology, so in the future, we plan to extend our approach to containers and other communication protocols as well. Furthermore, we plan to take advantage of Open vSwitch's virtualization and kernel bypassing features, so as to create overlay networks that extend our approach across the cloud continuum.

References

[1] L. Bittencourt, et al., "The internet of things, fog and cloud continuum: Integration and challenges," *Internet of Things*, vol. 3-4, pp. 134–155, 2018.

[2] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ull) networks: The ieee tsn and ietf detnet standards and related 5g ull research," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 88–145, 2019.

[3] L. Abeni and D. Faggioli, "Using Xen and KVM as real-time hypervisors," *Journal of Systems Architecture*, vol. 106, no. July 2019, 2020.

[4] L. Leonardi, L. L. Bello, and G. Patti, "Towards time-sensitive networking in heterogeneous platforms with virtualization," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1155–1158.

[5] M. Gundall, C. Glas, and H. D. Schotten, "Introduction of an Architecture for Flexible Future Process Control Systems as Enabler for Industry 4.0," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 2020-Septe, 2020, pp. 1047–1050.