Improving TRILL Mesh Network's Throughput Using Smart NICs

Danilo Cerović*†, Valentin Del Piccolo*, Ahmed Amamou* and Kamel Haddadou*
†Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France
*Gandi SAS Paris, France, Email: {danilo.cerovic, valentin.d.p, ahmed, kamel}@gandi.net

***Abstract*—TRILL is a Layer 2 protocol which allows the establishment of a layer 2 mesh network while improving infrastructure utilization in comparison with Ethernet. Indeed, TRILL does not use STP and keeps all links active. It always uses the shortest path between any two nodes and can dispatch the flows following ECMP. However, the problem is that throughput wise the performances of TRILL are not on par with Ethernet. This is the consequence of the fact that Linux does not have a fast path solution for TRILL and, moreover, the fact that CPU and memory resources have to be used in order to process the frames. The goal of this paper is to propose a fast path solution for TRILL that is based on a smart NIC. We experimented with our solution and the results show that we successfully increased the throughput of TRILL by 100%, thus reaching the network interface limit of 10Gbps while reducing latency.**

I. Introduction

With the growth of cloud computing, data centers tend to be replaced with hyperscale data centers. In fact, a report published by Cisco [1] predicts that, by 2021, hyperscale data centers will represent 53% of all installed data centers. More and more resource-consuming tasks are being processed in the cloud and more users are adopting cloud computing. This implies more data to transfer between a user and the cloud, but mostly more data to transfer inside the cloud infrastructure. This infrastructure must scale and increase network bandwidth without increasing latency. In this paper, we introduce a solution to increase bandwidth and reduce latency of a mesh network based on the Layer 2 TRILL (TRansparent Interconnection of Lots of Links) protocol [2] and smart NICs. The remainder of this article is organized as follows. Section II introduces the TRILL protocol and the smart NIC used in our experimentations . Then, our solution is presented in section III and its performances are discussed in section IV. Section V briefly presents the related work, and finally, section VI concludes the paper with a description of our future work.

II. Background

A. TRILL protocol

TRILL protocol [2] is an IETF protocol which aims to overcome the limitations of conventional Ethernet networks. TRILL does not use STP and keeps all the links of the network active, thus forming a mesh network (Figure 1). In fact, TRILL uses the shortest paths between any two points and with its multipath capability, it uses ECMP to choose the paths. TRILL also uses new devices called RBridges. They implement TRILL and are similar to conventional customer bridges. They each have a unique 2-octet nickname within the TRILL network (campus) and can be implemented in standard Linux hosts. RBridges may be interconnected with IEEE 802.3 technology, or with some other layer 2 technology like PPP (Point-to-Point Protocol). The first RBridge that a frame encounters in the campus will encapsulate a frame with a TRILL header (Figure 2) and an outer header which has as a destination the MAC address of the next hop. This header helps to maintain retro-compatibility with conventional customer bridges as they can be located between two RBridges. Each RBridge has to process the TRILL header and the outer Ethernet header. The Ethernet header must be replaced and the TRILL header has to be updated. This processing is TRILL throughput's bottleneck because of the speed limit of memory transactions in the host. To solve it we propose to offload this processing onto a smart NIC.

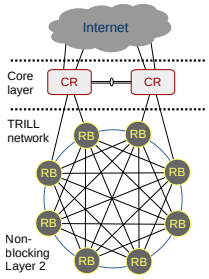

Fig. 1: TRILL mesh network

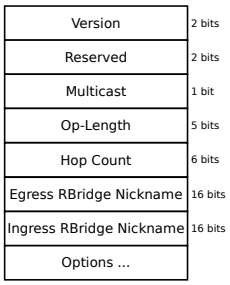

Fig. 2: TRILL header

B. Smart NIC

The smart NIC we use is Kalray MPPA2-256 (Bostan) card which is a Massively Parallel Processor Array (MPPA) [3] with 256 user-programmable cores and 32 system cores. All of the cores can run in parallel with high-performance and low power consumption. They are clustered in 16 groups of 16 cores (Figure 3). A cluster is an elementary processing unit and runs a specialized, low footprint OS, called NodeOS. To program this NIC we use the ODP (OpenDataPlane) API [4]. ODP [5] is a cross-platform set of APIs used for programing the networking data plane.

III. Proposed solution

In our previous work [6], we implemented TRILL at the host level of each server - source code available [7]. This allows us to have a complete mesh network inside a data center. Each RBridge node is implemented and deployed in

978-1-5386-2542-2/18/$31.00

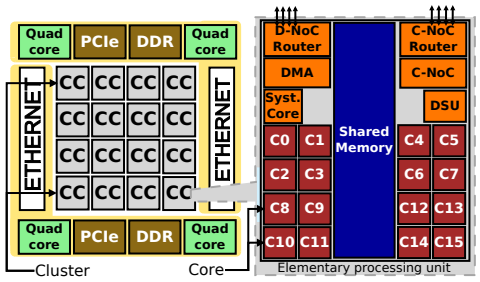

Fig. 3: MPPA architecture

the Linux bridge module of a server. However, throughput performances are limited because of the lack of a fast path for TRILL frames. Indeed, at each RBridge on the path the outer Ethernet header is switched and the TRILL header is updated. This process requires a lot of system interruption and memory changes which impact the time to process a frame, thus reducing throughput. To solve this issue we propose to offload the management of TRILL frames to Kalray's MPPA2-256 smart NIC. We implemented the forwarding process of TRILL frames on the NIC using ODP and using the NIC's smart dispatcher to evenly spread the load on each cluster. The forwarding process has three main steps:

1 Take off the outer MAC header
2 Update the TRILL header. If the *Hop count* is zero the frame is dropped, otherwise the *Hop count* is decreased.
3 Add the new MAC header based on the egress nickname of the TRILL header.

For the third step it is mandatory to look in the FIB to find the next MAC address on the path to the egress RBridge whose nickname is the egress nickname in the TRILL header.

TABLE I: Throughput comparison

	64 Bytes frame		1518 Bytes frame	
Throughput	Gbps	Mpps	Gbps	Mpps
TRILL offloaded	10	11.57	10	0.81
Ethernet offloaded	10	14.89	10	0.81
TRILL	#	#	4.5	0.37
Ethernet	#	#	6	0.49

IV. Performance analysis

To test our offloading solution we used a simple test bed composed of two servers - a Dell PowerEdge R310 and a Dell PowerEdge C6100 both acting as TRILL RBridges - connected with a 10Gbps link to the MPPA2 card. The MPPA2 card was responsible for realizing the forwarding process between the two RBridges. Both servers, running Ubuntu 16.04.3 LTS, generated Ethernet and then TRILL traffic using Pktgen-dpdk traffic generator (Pktgen version 3.4.9 with DPDK 17.08.1) and sent it to the MPPA card. Then, the MPPA card processed (forwarded) the traffic and sent it to the other server.

Thanks to parallel computing offered by the MPPA card, we have improved the TRILL's frame forwarding throughput. Indeed, we have split the computing load on 16 clusters each having 16 cores and each being responsible for one sixteenth of the destination nicknames. In Table I we can see that when the NIC forwards the frames, the throughput of both TRILL and Ethernet is significantly increased. In fact, using the smart NIC allows reaching the 10 Gbps throughput limit of the interface for both frame sizes. For TRILL (Ethernet) frames forwarding our solution increases the throughput in Gbps by 122% (66%) and the packet rate in Mpps by 118% (65%).

V. Related works

To the best of our knowledge our solution is the first solution which offloads TRILL frames management on a smart NIC. Other fast packet processing solutions offload packet processing on either a NetFPGA card or a GPU such as ClickNP [8], Snap [9], PacketShader [10] and GRIP [11].

VI. Conclusion

In this paper we presented our solution based on a smart NIC that improves the throughput of a TRILL mesh network. The results of our solution experimentations show that both Ethernet and TRILL benefit a lot from offloading the forwarding processes on the smart NIC. However, in our experimentations we are limited by the 10Gbps link and therefore cannot assess the full potential of our solution. This is why our next work will focus on improving our test bed in order to have 40Gbps links between the card and both servers. Additionally, our final goal being to offload all TRILL data plane on the smart NIC, we will work on implementing both TRILL frames encapsulation and decapsulation at the NIC level. Then we will test our solution again to see if we have performance gains in each case compared to both TRILL without offloading and Ethernet.

References

[1] "Cisco global cloud index: Forecast and methodology, 20162021 white paper," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html#_Toc503317520, Fbruary 2018, accessed: 2018/03/15.

[2] D. E. E. 3rd, D. G. Dutt, S. Gai, R. Perlman, and A. Ghanwani, "Routing Bridges (RBridges): Base Protocol Specification," RFC 6325, Jul. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6325.txt

[3] B. D. de Dinechin, "Kalray MPPA : Massively parallel processor array: Revisiting dsp acceleration with the kalray mppa manycore processor," in *2015 IEEE Hot Chips 27 Symposium (HCS)*, Aug 2015, pp. 1–27.

[4] "OpenDataPlane port for the MPPA platform," https://github.com/kalray/odp-mppa, accessed: 2018/03/15.

[5] "OpenDataPlane (ODP) Project," https://www.opendataplane.org/, accessed: 2018/03/15.

[6] A. Amamou, K. Haddadou, and G. Pujolle, "A TRILL-based multi-tenant data center network," *Computer Networks*, vol. 68, pp. 35 – 53, 2014, communications and Networking in the Cloud. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614000851

[7] "ktrill - TRILL implementation in the Linux Kernel," goo.gl/MfqUd7, accessed: 2018/03/15.

[8] B. Li, K. Tan, L. L. Luo, Y. Peng, R. Luo, N. Xu, Y. Xiong, P. Cheng, and E. Chen, "ClickNP: Highly Flexible and High Performance Network Processing with Reconfigurable Hardware," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: ACM, 2016, pp. 1–14. [Online]. Available: http://doi.acm.org/10.1145/2934872.2934897

[9] W. Sun and R. Ricci, "Fast and flexible: Parallel packet processing with GPUs and click," in *Architectures for Networking and Communications Systems*, Oct 2013, pp. 25–35.

[10] S. Han, K. Jang, K. Park, and S. Moon, "PacketShader: A GPU-accelerated Software Router," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 195–206. [Online]. Available: http://doi.acm.org/10.1145/1851182.1851207

[11] P. Bellows, J. Flidr, T. Lehman, B. Schott, and K. D. Underwood, "GRIP: a reconfigurable architecture for host-based gigabit-rate packet processing," in *Proceedings. 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2002, pp. 121–130.