DesktopDC: Setting All Programmable Data Center Networking Testbed on Desk

Chengchen Hu, Ji Yang, Zhiming Gong, Shuoling Deng
Xi'an Jiaotong University
huc@ieee.org, {yangji, zhm.gong, dengshuoling2}@stu.xjtu.edu.cn

Hongbo Zhao
MeshSr Co. Ltd.
hongbo.zhao@meshsr.com

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design - Network communication, Network topology

Keywords

OpenFlow, data center, programmable

1. INTRODUCTION

Software Defined Networking (SDN) is an emerging network architecture, which decouples the control plane from the physical network infrastructure and operates network with global abstraction of lower level network functionalities. SDN becomes very attractive to design a flexible and customized Data Center Networks (DCN) since Google had recently achieved huge impact by applying SDN to manage the inter-DC traffic [2]. The successful story when SDN met DCN in Google not only demonstrates the feasibility for deployment of SDN to large scale network, but also stimulates a research of SDN, especially in the context of DCN. However, it becomes quite challenging to realize and verify the research progress into practice or to emulate a whole SDN-compatible DCN, since it is costly to operate a Data Center (DC) testbed in a research lab and it is hard to modify the processing logic of a data path for the research and innovation purpose.

Although OpenFlow [3] is the de factor SDN protocol nowadays, which defines the interface between the data plane switches and the control plane controllers, new SDN architectures and protocols are proposed. Even for the OpenFlow itself, it keeps evolving and updates its specification in every a few months. Software OpenFlow Switches (OFS), *e.g.*, Open vSwitch, are easy to deploy and modify, but they are hard to guarantee the performance for wire-speed processing. Commercial OFS provide stable performance and sufficient network interfaces. However, they cannot be updated with evolving OpenFlow specifications and modified innovated processing logic. NetFPGA [4] is quite successful to open a way for changing the hardware logic through FPGA, but it also has limitations, e.g., a host server is further required, the logic resource is not enough for OFS processing, and bottleneck PCI interface between host CPU and NetFPGA (NetFPGA 1G version).

We have designed an all programmable SDN switch (named as ONetSwitch) and a DCN testbed on the desktop (named DesktopDC) based on ONetSwitch. The merits of DesktopDC are its small size, low power, and flexible programmability. Among many building cases, we briefly describe two of them in this paper: one is SDN based routing and the other is Hadoop based computing.

2. DESIGN OF ONETSWITCH

ONetSwitch is a Zyqn-based embedded computing platform. The Zyqn chip is produced by Xilinx, which is constituted by both an ARM processor and a FPGA. Empowered by Zyqn, ONetSwitch is "all programmable", which means software programmable, gateware restructual, and hardware extensible.

Currently, ONetSwitch has two versions. The first version is ONetSwtich20. It has a 533MHz ARM Cortex-A9 dual core processor combined with a Artix FPGA in SoC, 512MB DDR3 SDRAM and 5 1G Ethernet ports. Four Ethernet ports are connected to FPGA and the rest is connected to the processor. Its size is about $13.5cm * 22.5cm * 1.5cm$. A second ONetSwitch45 model, which is first introduced in ONS 2014 [1], uses the Xilinx Zynq-7045 SoC integrating ARM Cortex-A9 dual core processor and Kintex-7 FPGA. ONetSwitch45 provides four SFP+ interfaces to support 10G links, four 1G Ethernet interfaces and commodity wireless adapter modules for 802.11 a/g/n. Its size is $18cm * 18cm * 1.5cm$.

An OpenFlow switch is implemented on ONetSwitch. As shown in Fig.1, this design of OpenFlow switch has a datapath with hybrid software and hardware design. The hardware part provides 8Gb/s packet header matching ability while the software part enables almost "unlimited" matching table size. The flow table lookup operation starts at the first table in hardware, which contains the hottest flow entries. If table miss in hardware, the packet will be sent to software datapath, which contains all the flow entries from controller. An OpenFlow agent is ported from the reference OpenFlow software switch to translate OpenFlow messages. Hardware Abstraction Layer (HAL) works between agent and hardware, which translates original flow entries from OpenFlow messages into the semantic-equivalent formats that optimize the switch performance.

Besides switch ability, ONetSwitch also has more resources for further use. A 64Gb/s DMA channel in the SoC connects software and hardware in various ways. The first way, configurable virtual Ethernet devices send packets to hardware datapath, which reduce latency and unnecessary matches. Another way helps to share computing resources by sending raw data from software to hardware.

SIGCOMM'14, August 17–22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2836-4/14/08.
http://dx.doi.org/10.1145/2619239.2631472.

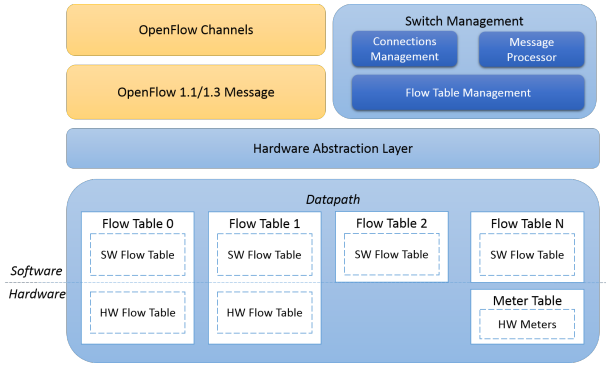

Figure 1: OpenFlow switch implementation

3. DESKTOPDC AND BUILDING CASES

Connected multiple ONetSwitches, we assemble a DCN testbed called DesktopDC, whose size makes it possible to put on the desktop. In DesktopDC, each ONetSwitch can work as both a computing node and a networking node, and both the computing and the networking functions can be fit either in ARM processor or FPGA, depending on the tradeoff between performance and flexibility. A DesktopDC can also be built with ONetSwitch45 only or mixing with two ONetSwitch models. In the example in Fig.2, the nodes were following a Fattree(4) topology.

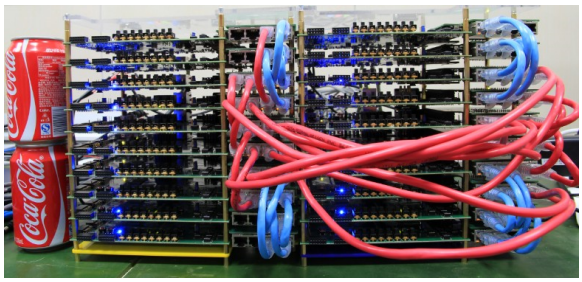

Figure 2: DesktopDC with Fat-Tree topology

With this DesktopDC testbed, we set two building cases. In the first one, we emulate an OpenFlow based fault tolerance routing for DCN. Each ONetSwitch node in this experiment is served as an OpenFlow switch, as mentioned in the last section. All the ONetSwitches are directly connected to a Ryu OpenFlow controller where configurations and our fault tolerant routing application are performed. Two more servers are connected as client. In the first step, the routing application will collect physical topology of the DCN dynamically. In the second step, it maps the collected physical network topology with the blueprint. Even there are little malfunctions, the mapping process can identify the good ones for networking functions. Also, it equipped an ID learning mechanism, which helps to learn custom servers' physical ID (like MAC), logical ID (like IP) and its related physical locations. After completing of all the above work, the routing application will calculate a path based on the routing algorithm, and assign flow entries to switches. We have measured the average delay within 100 experiments. The time of topology collecting, ID mapping from blueprint to real network, and ID/address/flow table issuing time experiments are 1.5ms, 3.2ms and 0.29ms respectively. We simulate the network failure by disabling different links in the DesktopDC. A port-statistics OpenFlow message will be sent by the affected switches.

Table 1: DesktopDC's Hadoop performance comparison

	ONetSwitch	x86 Server
Core Frequency	533MHz	3.3GHz
Boot	<10s	50s
Job Duration	192794ms	13314ms
Node Power	6.7W	56.9W
Switch Power	–	20W
Size	$13.5 * 22.5 * 1.5cm^3$	$37.0 * 43.5 * 4.5cm^3$

On receiving the messages in the controller, routing application will figure out whether routing policy would suffers from the failure and if so, it recalculates all the lapsed paths and replace their entries with new ones. The path will recover in less than 500ms.

The second experiment tests Hadoop computing on DesktopDC, where we configure each node in DesktopDC as both computing node and networking node. For the testbed in Fig. 2, our Hadoop computing system will have at most 4Gbps bandwidth Ethernet for each node. In order to compare the power and computing with one single x86 based server, four ONetSwitch20 nodes are utilized. In the first stage, all the computing tasks are finished by ARM processors, meanwhile the rest resources form the OpenFlow based network connecting all ONetSwitches. A high bandwidth virtual Ethernet port transfers data to OpenFlow datapath, meanwhile, OpenFlow controller monitors and manages the data transfer to optimize bandwidth for jobs. Table.1 demonstrates that with elaborate design of computing management algorithm in controller, we can run a data center with hundreds of nodes on the desktop, without any concern about the power and space. In addition, the computing ability and power saving will be further improved by moving some dedicate computing to FPGA, which is our future work.

4. CONCLUSION

In this poster, we propose the DesktopDC, an all programmable and energy efficient SDN compatible innovation platform. Our initial experiments and demo exhibit the features of flexibility, capability, power saving. It provides the SDN datapath designs for fast SDN prototyping and verifications in both software and hardware.

5. ACKNOWLEDGEMENT

This paper is supported by 863 plan (2013AA013501), National Sci. and Tech. Major Project (no.2013ZX03002003-004), NSFC (61272459), Research Plan in Shaanxi Province (2013K06-38).

6. REFERENCES

[1] C. Hu, J. Yang, H. Zhao, and J. Lu. Design of all programmable innovation platform for software defined networking. In *Open Networking Summit 2014*, Santa Clara, CA, 2014.

[2] S. Jain, A. Kumar, S. Mandal, and Etal. B4: Experience with a globally-deployed software defined wan. In *Proceedings of the ACM SIGCOMM 2013*, pages 3–14, 2013.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[4] J. Naous, G. Gibb, S. Bolouki, and N. McKeown. Netfpga: reusable router architecture for experimental research. In *Proceedings of the ACM PRESTO '08*, pages 1–7, New York, NY, USA, 2008. ACM.