Improving PC-based OpenFlow Switching Performance

Voravit Tanyingyong
KTH Royal Institute of Technology
SE-100 44 Stockholm, Sweden
+46-8-790 42 08
voravit@kth.se

Markus Hidell
KTH Royal Institute of Technology
SE-100 44 Stockholm, Sweden
+46-8-790 42 67
mahidell@kth.se

Peter Sjödin
KTH Royal Institute of Technology
SE-100 44 Stockholm, Sweden
+46-8-790 42 55
psj@kth.se

ABSTRACT
In this paper, we propose an architectural design to improve lookup performance of OpenFlow switching in Linux using a standard commodity network interface card based on the Intel 82599 Gigabit Ethernet controller. We describe our design and report our preliminary results that show packet switching throughput increasing up to 25 percent compared to the throughput of regular software-based OpenFlow switching.

Categories and Subject Descriptors
C.2.6 [**Computer-Communication Networks**]: Internetworking – *Routers*

General Terms
Performance, Design, Experimentation

Keywords
OpenFlow, Flow-based switching, Performance Evaluation

1. INTRODUCTION
In recent years, there has been a strong trend in the networking community to address network ossification using a clean slate approach. Large-scale projects such as FEDERICA [3] and GENI [4] are good examples of this trend. To break away from the traditional IP forwarding, new flexible architectures emerge. Network virtualization plays a major role in this effort thanks to its nature that allows innovation through disruptive technologies [2].

OpenFlow [7] is a novel technology that enables flow tables in switches and routers to be programmed via a standardized interface; namely the OpenFlow protocol. OpenFlow switching is flow-based—a forwarding paradigm based on multiple fields of packet headers—which is more flexible compare to traditional IP forwarding. It is a promising technology that can enhance network virtualization towards a more flexible future Internet architecture.

An OpenFlow reference implementation is publicly available and purely software-based. Although it is portable, it inherits lookup performance penalties that come with software processing. In addition, OpenFlow switching implies that more information needs to be taken into account in the packet lookup. To enhance software-based OpenFlow lookups, an extension to accelerate the performance is suggested here.

In this paper, we propose an open architecture based on the Intel 82599 10 Gigabit Ethernet (GE) controller [5] to improve the lookup performance of the OpenFlow Linux kernel module implementation. The idea to offload the packet processing from host CPU level to the Network Interface Card (NIC) level is similar to earlier work [6][8], but we use a different approach based on a regular commodity NIC rather than specialized NICs with FPGAs or network processors. The commodity NIC might not be as flexible, but it is easier to program, which makes it straight forward to implement our design. We also make use of multi-core CPUs, which are located on the motherboard instead of on the NIC. Moreover, we use higher line speed (10GE rather than 1GE) and measure the results in terms of throughput improvements.

2. ARCHITECTURAL DESIGN
In our research, we aim at keeping our design open and accessible by using open source software and standard PC hardware components. In this context, the Linux OpenFlow implementation matches well with our purpose. We focus on the kernel module implementation since it offers better performance compared to the user space implementation. The lookup mechanism in the OpenFlow kernel module is straight forward. Once the received packet arrives to the OpenFlow module, it will be matched against the flow table. An action will be performed as defined in the flow table entry. This action can be to drop or to forward the packet (including forwarding to the controller).

With our design objectives in mind, we propose an enhancement of the packet lookup process by utilizing features supported by the Intel 82599 10 GE controller. This is a very recent chipset, which Intel uses in all new 10 GE NICs. It is rich in features and well-suited to our needs. The key feature we employ in our enhancement is the Flow Director, which is used for directing incoming packets to received queues based on the flow signatures. In our architecture, we use the Flow Director as a lookup shortcut. This is done by pre-assigning receive queues to output ports and use the Flow Director to direct actual traffic flows accordingly. To achieve the desired behavior, the Flow Director extension table is introduced as an additional lookup table to keep track of which receive queue maps to which output port. It is added just before the standard OpenFlow lookup, which allows compatibility

ANCS'10, October 25–26, 2010, La Jolla, CA, USA.

and keeps all the modifications within the OpenFlow module leaving the kernel intact. In addition, the multiple queues feature of the NIC makes it possible to load-share packet processing among multiple CPU cores. This is incorporated in our design to utilize the symmetric multiprocessing (SMP) system as depicted in Figure 1.

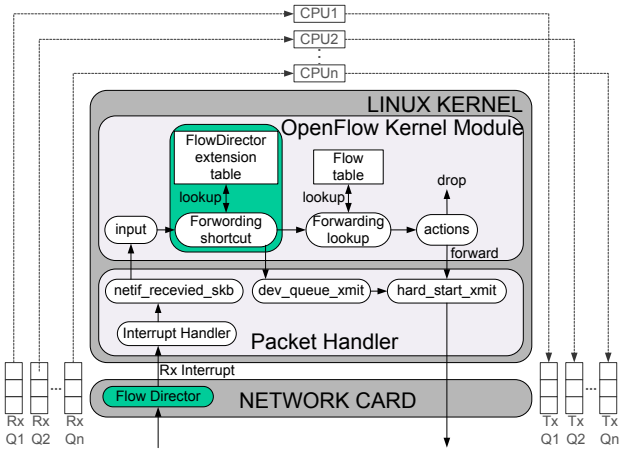

Figure 1. Architectural Design

3. EXPERIMENTAL EVALUATION

We use a simple setup with two directly connected PCs; PC1 used for generating/receiving traffic and PC2 used as a dedicated OpenFlow switch. Pktgen [9] is used as a packet generator on PC1 and OpenFlow kernel module implementation is used as our device under test on PC2. Both have identical hardware; TYAN S7002 Motherboard with Intel Xeon Quad Core 5520, 2.26 GHz, 3GB RAM, one 10GE dual-port card with Intel 82599 10GE controller. The Operating System we use is Bifrost/Linux [1], which is a Linux distribution optimized for routing. We use Bifrost 6.1 with net-next 2.6.32-rc2 kernel.

The OpenFlow reference system has two types of lookup tables: hash and linear. We test them with various configurations and compare their performance with the modified version that incorporates our design. The performance metric we focus on is the packet switching rate in term of packet per second (pps). In each test, PC1 sends randomized 64-byte packets of 100 UDP flows. PC2 receives all traffic on one port and forwards them back to PC1 on another port. The test is performed repeatedly, each round using different number of CPUs, to illustrate the effect of the SMP process. The result is shown in Figure 2.

We make some observations in the results. First, standard OpenFlow switching performs better with hash table than linear table even when there is only 1 entry in the linear table. This implies that a hash index can be calculated faster than the time it takes to process a lookup with normal packet header. Second, OpenFlow switching performance decreases as linear table size increases while hash table size as well as Flow Director table size do not affect OpenFlow switching performance. Moreover, our modified OpenFlow improves switching throughput up to 25% (from 528,077 pps to 668,705 pps in 1 CPU case). The overall gain is roughly 100,000 pps per CPU core. One interesting irregularity from our experiment worth pointing out is that when we increase the number of CPU cores to 5, 6 and 7 (not included in Figure2), the overall performance degraded. This is likely caused by unevenly distributed load over physical CPU cores with hyper-threading. Some CPUs receive many more packets than the others and become overloaded resulting in an adverse performance impact. This problem does not occur when 8 CPU cores are used. Further research to mitigate this problem is needed.

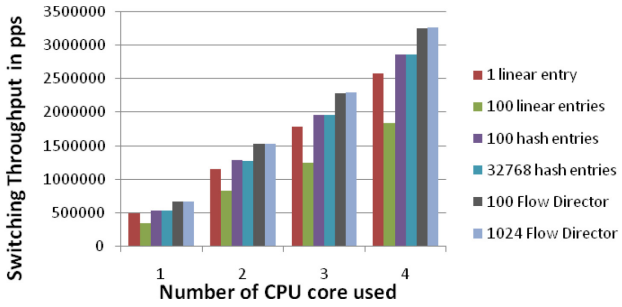

Figure 2. Switching Throughput of OpenFlow with different configurations

4. ACKNOWLEDGMENTS

This paper is supported by a project within the Sino-Swedish Strategic Cooperative Programme on Next Generation Networks: IMT-Advanced and Beyond.

5. REFERENCES

[1] Bifrost Network Project. http://www.bifrost-network.org/.

[2] Chowdhury, N. M. and Boutaba, R. 2009. Network Virtualization: State of the Art and Research Challenges. *Comm. Mag*. 47, 7 (Jul. 2009), 20-26. DOI= http://dx.doi.org/10.1109/MCOM.2009.5183468.

[3] FEDERICA Project. http://www.fp7-federica.eu/.

[4] GENI Project. http://www.geni.net/.

[5] Intel. Product Brief - Intel® 82599 10 Gigabit Ethernet Controller. 2009. http://download.intel.com/design/network/prodbrf/321731.pdf .

[6] Luo, Y., et al. 2009. Accelerating OpenFlow Switching with Network Processors. In *Proceedings of the 2009 ACM/IEEE Symposium on Architectures For Networking and Communications Systems* (Princeton, New Jersey, USA, October 19-20, 2009). ANCS'09.

[7] McKeown, N., et al. 2008. *OpenFlow:* Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.* 38, 2 (Mar. 2008), 69-74. DOI= http://doi.acm.org/10.1145/1355734.1355746.

[8] Naous, J., et al. 2008. Implementing an OpenFlow Switch on the NetFPGA Platform. In *Proceedings of the 4th ACM/IEEE Symposium on Architectures For Networking and Communications Systems* (San Jose, California, November 06 - 07, 2008). ANCS '08. ACM, New York, NY, 1-9. DOI= http://doi.acm.org/10.1145/1477942.1477944.

[9] Olsson, R. 2005. Pktgen the Linux Packet Generator. In *Proceedings of the Linux Symposium* (Ottawa, Ontario, Canada, July 20 – 23, 2005). Volume two, 11-24.