Offloading Packet Forwarding in a Combined Router/Server

Voravit Tanyingyong Markus Hidell Peter Sjödin

School of Information and Communication Technology
KTH Royal Institute of Technology
Kista, Sweden
{voravit, mahidell, psj}@kth.se

ABSTRACT

Despite the performance limitation with minimum-sized packet processing, a modern PC-based router can provide as competitive service as a specialized hardware router while offering more flexibility and possibility to extend beyond routing. We focus on a use case in which the PC-based router also functions as a server. In this paper, we propose an architecture to boost overall performance of the PC-based router by offloading packet processing tasks to the NIC. We introduce a fast path for packet forwarding based on caching of flow entries in on-board classification hardware on the NIC. We describe our design and present an experimental evaluation.

Categories and Subject Descriptors

C.2.6 [**Internetworking**]: Routers

General Terms

Design, Experimentation, Performance

Keywords

PC-based router, Commodity Hardware Classifier, Flow-based switching

1. INTRODUCTION

Advancement in PC technology allows a modern PC-based router to offer as competitive service as a specialized hardware router. Although a PC-based router might still not be able to perform at wire-speed when it comes to minimum-sized packet forwarding, recent research [1, 2] has shown that the gap is closing in. This enables a PC-based router to offer as competitive service as a specialized hardware router while having more advantages in terms of price, accessibility, and programmability. These attributes of a PC-based router offers more flexibility and fosters new innovations. For instance, it can be extended to offer more services beyond solely routing. One example is data centers using BCube [5], DCell [4] and FiConn [7] interconnection structure in which a server act as an end host as well as a relay host for other servers. Another example is community-level gateways in residential networks in which a PC-based router responsible for forwarding aggregated traffic to/from numerous building blocks in a residential area also serves as a server providing local services such as community web portal, mail, media streaming, and directory services.

One way to improve overall performance of a PC-based router is by offloading packet forwarding task to a hardware component. Modern commodity hardware components are very capable yet relatively cheap making them an attractive choice for this purpose. By exploiting this fact, recent research have proposed to bring forth better performance to a PC-based router through the help of commodity hardware components. Within these commodity hardware assisted approaches, there are a wide spectrum of methods ranging from using purely software-based [6], modifying the actual hardware [8], to consolidating hardware [3, 10]. In this paper, we take the purely software-based approach to improve the PC-based router forwarding performance. We aim at using unaltered commodity off-the-self hardware that can be inserted into a PC-based router and only make changes in software. This method is easy to adopt and requires no change in the existing infrastructure.

The idea for this work comes as a spin-off from our previous work [11] that aims at improving lookup performance of PC-based OpenFlow [9] with the use of NIC hardware classification to offload the CPU from the lookup processing task.

2. ARCHITECTURAL DESIGN

For a combined router/server, incoming packet is classified into two types: a pass-through packet to be forwarded further and a local delivery packet to be handled by an application running on the PC-based router. In Linux, the pass-through packet is forwarded by the network stack in the kernel space while the local delivery packet is passed on to the user space and handled by a local application. Our goal is to introduce a hardware classification to assist CPU on processing the pass-through packets.

We propose an architecture based on caching of flows as depicted in Figure 1. The applications in the user space are the services running on the PC-based router. The forwarding engine, or the routing engine, is a software process on the PC-based router that makes forwarding decision for pass-through packets. To offload the CPU from packet processing task, we introduce a fast path in the lookup process to bypass the forwarding engine in the software. This is done by caching active flow table entries in the commodity NIC with hardware classification support, which functions as a lookup accelerator. In general, commodity NICs have no capability to forward the packet by themselves. Thus, the Quick Path Selector is introduced as a decision point to determine which path a received packet should take. A packet belonging to a cached flow in the lookup accelerator will find a match in the Quick Path Selector and gets forward directly, while a packet that does not belong to a cached flow will find

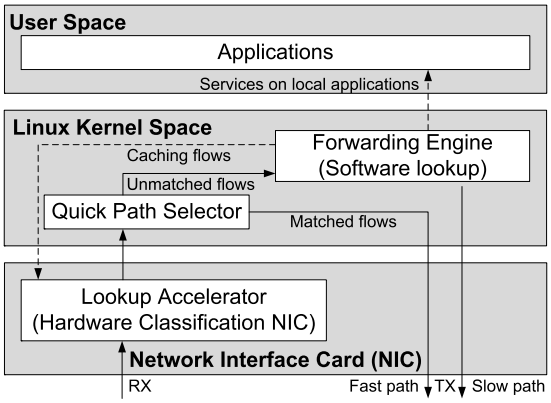

Figure 1: Architectural Design

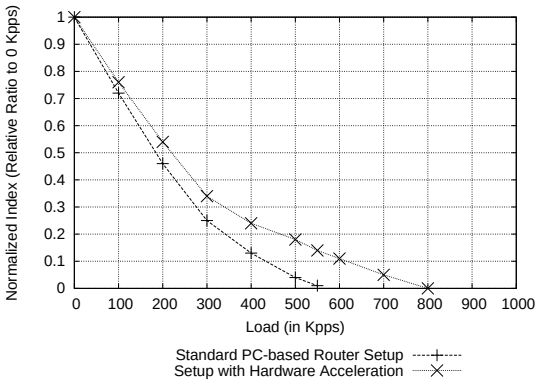

Figure 2: Normalized Integer Index

no match and follows a standard path through normal software lookup. A packet destined for a local application on the system will be identified in the forwarding engine and will be passed on to the application on the user space. The architecture design in Figure 1 is intended to be generic to allow flexibility and should support any types of hardware classification NICs as well as any types of lookup process in the forwarding engine. The actual implementation should be able to adopt this design effortlessly.

3. EXPERIMENTAL EVALUATION

We setup a prototype of a PC-based router to evaluate our architectural design. We use a NIC with the Intel 82599 10 Gigabit Ethernet controller to provide a lookup acceleration function. We create a Quick Path Selector using a simple index lookup table with receive interface and receive queue as the lookup key to identify the outgoing interface and outgoing queue. We use OpenFlow as our forwarding engine to provide flexible forwarding. We use nbench[1] as a local application on the PC-based router.

Nbench is a Linux/Unix ported version of release 2 of BYTE Magazine's BYTEmark benchmark program. It runs through ten different tasks, each produces a result in term of the numbers of iteration per second. Nbench uses these numbers to calculate geometric mean to produce three overall indexes: Integer index, Memory index, and Floating-point index. These indexes are relative scores compared to a baseline system based on an AMD K6/233 with 32 MB RAM and 512 KB L2-cache running Linux 2.0.32 and using GNU gcc version 2.7.2.3 and libc-5.4.38. To provide a better representation for how much relative gain we acheive, we normalize each nbench index into a ratio relative to when there is no traffic load (and only nbench occupies the CPU) according to (1) and use them as our performance metric.

$$\text{Normalized index} = \frac{\text{Nbench index to normalize}}{\text{Nbench index at no traffic load}} \quad (1)$$

To investigate how much we gain from offloading the lookup processing task with hardware classification, we carry out an experiment to compare the application processing performance of a standard PC-based router with a PC-based router with our architecture. To keep the test simple, we set up the DUT to use only one CPU core and two receive queues. Both queues are mapped to the CPU core. The hardware classifier sorts pass-through packets to one queue and local delivery packets to the other. The traffic generator sends randomized 64-byte UDP packets at a constant rate to the DUT on one port. All packets will match entries in the lookup table on the DUT and are forwarded to another port to the sink. We vary the traffic load in each round of test from no load up to 1 Mpps and observe how traffic load affects the normalized nbench indexes.

From the experiment, it turns out that the results are almost identical for all indexes. Thus, we present only the normalized integer index in Figure 2. When compared with the standard PC-based router setup, the setup with our architecture achieves an increasing gain as the traffic load increases as can be seen from the increasingly widen gap in Figure 2. The largest gap is at the point when the standard PC-based router is saturated (at 550 Kpps). The normalized indexs are 0.01 for the standard PC-based router and 0.14 for our architecture. This translates to more than ten times relative gain for our architecture. In addition, the maximum throughput of our architecture reaches up to about 800 Kpps, which is equivalent to 45% increase compared to standard setup. This is expected since the lookup accelerator should reduce the CPU cycles required to process the packets making it possible to handle more packets in general.

[1]source from http://www.tux.org/ mayer/linux/bmark.html

4. REFERENCES

[1] R. Bolla and R. Bruschi. Pc-based software routers: high performance and application service support. In *PRESTO '08*.

[2] N. Egi et al. Towards high performance virtual routers on commodity hardware. In *CoNEXT '08*.

[3] K. Fall et al. Routebricks: enabling general purpose network infrastructure. *SIGOPS OSR*, 45:112–125, February 2011.

[4] C. Guo et al. Dcell: a scalable and fault-tolerant network structure for data centers. *SIGCOMM CCR*, 38:75–86, August 2008.

[5] C. Guo et al. Bcube: a high performance, server-centric network architecture for modular data centers. *SIGCOMM CCR*, 39:63–74, August 2009.

[6] S. Han et al. Packetshader: a gpu-accelerated software router. *SIGCOMM CCR*, 40:195–206, August 2010.

[7] D. Li et al. Scalable and cost-effective interconnection of data-center servers using dual server ports. *IEEE/ACM Trans. Netw.*, 19:102–114, February 2011.

[8] G. Lu et al. Serverswitch: a programmable and high performance platform for data center networks. In *NSDI'11*.

[9] N. McKeown et al. Openflow: enabling innovation in campus networks. *SIGCOMM CCR*, 38(2):69–74, 2008.

[10] N. Sarrar et al. Fibium: Towards hardware accelerated software routers. Technical report, Deutsche Telekom Laboratories.

[11] V. Tanyingyong et al. Using hardware classification to improve pc-based openflow switching. In *HPSR*, 2011.