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 offer 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 acts 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
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 base-case setup with Hardware Acceleration.

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 base-case setup with Hardware Acceleration. 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.

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

\[ \text{Normalized index} = \frac{\text{Nbench index at no traffic load}}{\text{Nbench index at no traffic load}} \]  

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

\[ \text{Normalized index} = \frac{\text{Nbench index at no traffic load}}{\text{Nbench index at no traffic load}} \]  

4. REFERENCES