An Open Router Virtualization Framework using a Programmable Forwarding Plane

Zdravko Bozakov
Institute for Communication Technology
Leibniz Universität Hannover, Germany
zdravko.bozakov@ikt.uni-hannover.de

ABSTRACT

Network virtualization promises to spur innovation and add flexibility to the Future Internet infrastructure. Routers supporting virtualization allow the deployment of concurrent virtual networks, and can be employed to consolidate resources and improve energy efficiency in data centers. The closed nature of commercial router systems poses a significant problem for research in the field of virtual network architectures. On the other hand, the performance of software-based, open routing solutions is typically limited. In this work we outline an open router virtualization framework utilizing OpenFlow enabled hardware as a fast, programmable forwarding plane.

Categories and Subject Descriptors

C2.6 [**Computer Communication Networks**]: Internetworking—*Routers*

General Terms

Design

Keywords

Routers, Virtualization, OpenFlow, Commodity Hardware

1. INTRODUCTION

In the past decade the virtualization paradigm has been remarkably successful in the server domain, and significant benefits are expected from the virtualization of network resources. To date, the feasibility of executing software routers within system virtualization platforms has been investigated (e.g. [1]). While hypervisor performance has steadily improved in most areas, I/O throughput remains a major bottleneck for software based virtualization solutions, giving rise to research into hardware supported virtualization (e.g. [2]). We believe that performance and interoperability are crucial factors for the success of a virtualization layer in the Internet. Therefore, we propose a framework for router virtualization which employs a combination of open software components as well as hardware supporting the emerging OpenFlow protocol, which enables packet forwarding at line speeds.

SIGCOMM'10, August 30–September 3, 2010, New Delhi, India.
ACM 978-1-4503-0201-2/10/08.

2. FORWARDING PLANE ACCESS

The OpenFlow project [3] has recently released a specification enabling external access to the forwarding tables of compliant switches. The project aims to allow researches to deploy and evaluate novel networking concepts in parallel to production networks. The work is backed by a number of hardware vendors, and switches implementing the specification are already commercially available. The elegant and highly flexible design of the OpenFlow architecture make it an ideal basis for an open router virtualization framework.

OpenFlow-enabled switches (OF) contain a flow table, which is accessible over a secure channel using the OpenFlow protocol. Forwarding entries are inserted by a controller running on remote host. The OpenFlow protocol is based on the notion of rules and actions. Each rule identifies a packet flow using the following 10-tuple:

in port	eth src	eth dst	eth type	VLAN ID	IP src	IP dst	IP proto	src port	dst port

A rule is associated with one or more actions, triggered whenever an incoming packet matches a flow table entry. The specification defines a number of required (e.g. *drop*, *forward*, *send to controller*) and optional (e.g. *enqueue*, *modify-field*) actions. Flow rules may contain wildcards for one or more fields, which match any header value. Packets not matching any rule, are sent to the OpenFlow controller for further processing. Additionally, each flow is assigned a priority.

3. ROUTER VIRTUALIZATION FRAMEWORK

Our goal is the development of a virtual router architecture, comprising of a virtualized control plane running on commodity hardware, and off-the-shelf, OF-enabled switches, functioning as a programmable forwarding plane. Moreover, we aim to minimize the VR configuration overhead. In the following, we present an suitable framework, necessary data plane configuration procedures, as well as our prototype implementation.

3.1 Architecture

The architecture of our virtual router (VR) framework is depicted in Fig. 1. Each VR instance is represented by a virtual environment (VE), implemented using standard software components, which provides the control plane functionality. The VE host is connected to the forwarding plane hardware over a dedicated link.

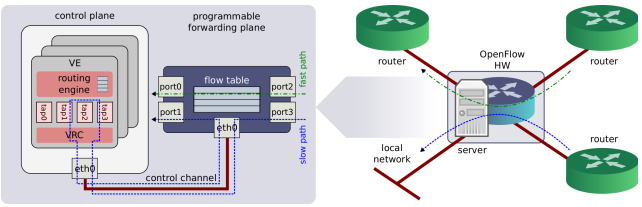

Figure 1: Virtualization Framework

The VR controller component (VRC) is the central element of the architecture. Each VE contains a VRC, which provides a transparent mapping between the virtual and physical resources of the router. To this end, a set of dummy network devices are configured within each VE and associated with specific data plane ports. The VRC uses the control channel to redirect packets written to the virtual interfaces to the corresponding physical port, and vice versa. Moreover, the VRC mirrors the VE routing tables onto the forwarding plane by installing OpenFlow rules, as described below. As all OpenFlow control operations are exposed only to the VRC, the framework enables the use of arbitrary routing daemons to populate the VR routing tables. Furthermore, the operating system networking stack is utilized for the processing of control and address resolution messages. Finally, standard UNIX network configuration commands can be used to configure the router.

3.2 Flow-Table Configuration

In order to mirror the VR routing tables onto the forwarding plane, each VRC installs a number of flows which we define in the following.

Localhost flow set Each VR instance must respond to ARP, routing, and configuration messages destined to its virtual interfaces. Therefore, for each of the VE's virtual interfaces, a rule matching against the corresponding MAC address is generated. All remaining fields are set to wildcards. The *send to controller* action is applied to all matched packets. This rule set has the lowest priority and hence only matches packets for which no other rule is defined.

Gateway flow set A set of rules is inserted for all IP destinations, reachable over gateway routers. In addition to the MAC address of the virtual interface, the flow entry includes the destination network address and netmask. Matched packets are forwarded to the data plane port associated with the virtual interface specified in the VE routing table. Additionally, the source and destination MAC fields are modified. Flow priorities must be proportional to the netmask length in order to implement longest prefix matching. Note, that packets matching this rule set are processed at line speed in the forwarding plane without traversing the control plane.

Broadcast domain flow set Packets destined to hosts attached directly to a VR require separate handling. Flow entries are inserted on a per host basis in order to allow rewriting of the layer two source and destination addresses. The control plane ARP table is used to determine the destination MAC address. If the destination MAC address is unknown, the packet is processed in the VR slow path, i.e. it is sent to the controller, where an ARP query is generated, an appropriate flow entry is inserted and finally the packet is sent out to the network. Subsequent packets are forwarded directly in the data plane. Again, the output interface lookup for the forwarding action is performed using the VE routing table.

Note, that all rules above contain the MAC addresses of the interfaces associated with a specific VR instance. As a consequence, multiple VRs can share the same forwarding plane interfaces, as long as the corresponding virtual interfaces are assigned unique MAC addresses. Furthermore, access control can be enforced by limiting the flows installable by a VR to a specific address set, using an OpenFlow proxy such as [4].

Broadcast messages do not match any of the above flow sets, therefore they are forwarded to the controller where the VE kernel can decide if and how to respond.

3.3 Prototype Implementation

To demonstrate the feasibility of the approach outlined above, we implemented a prototype using commodity PC hardware and a NetFPGA card with a reference bitfile implementing version 0.9 of the OpenFlow protocol. We used the NOX framework [5] to implement the VRC component. Our prototype uses and the KVM hypervisor for the control plane virtualization and the XORP suite for routing. Within each VE we used Debian Linux as well as the TUN/TAP framework to instantiate the virtual network interfaces.

4. CONCLUSION AND OUTLOOK

Our framework demonstrates that virtual software routers can be transparently extend with an OpenFlow-based forwarding plane. A major advantage of the proposed approach is that readily available, well tested components (e.g. the Linux networking stack) and open specifications (OpenFlow) are leveraged, resulting in a fast, stable, and flexible virtual router architecture. The proposed framework provides a basis for research in novel network strategies, such as live router migration or distributed router architectures. Although a number of questions, such as QoS and strict resource isolation remain to be addressed, we believe that the design is applicable for a wide range of networking scenarios.

5. REFERENCES

[1] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, and T. Schooley. Evaluating xen for router virtualization. in *Proc. of ICCCN 2007*, pages 1256–1261, Aug. 2007.

[2] M. B. Anwer and N. Feamster. Building a fast, virtualized data plane with programmable hardware. *SIGCOMM Comput. Commun. Rev.*, 40(1):75–82, 2010.

[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, 2008.

[4] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, N. McKeown, and G. Parulkar. Flowvisor: A network virtualization layer. Technical report, 2009.

[5] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.*, 38(3):105–110, 2008.