FIBIUM: Towards Hardware Accelerated Software Routers

Nadi Sarrar Anja Feldmann Steve Uhlig
TU Berlin / Deutsche Telekom Laboratories, Germany
{nadi,anja,steve}@net.t-labs.tu-berlin.de

Rob Sherwood Xin Huang
Deutsche Telekom Inc. R&D Lab, USA
{r.sherwood,xin.huang}@telekom.com

1. INTRODUCTION

We propose an alternative router design to fill the gap between pure software routers that suffer from low forwarding performance and commercial routers. We describe a prototype implementation and show that it is in principle capable of handling the traffic requirements of a carrier aggregation network.

Open source routing software [1, 3] is already deployed within enterprise and ISP networks for specialized tasks. Its quality is getting closer to what is commonly referred to as "carrier grade". However, PC based routers with open source routing software do not yet offer the throughput needed for ISPs' purposes. Still, recent advances in both hardware and software capabilities motivate us to revisit the question of what may be a good architecture for a hardware accelerated software router. Current Ethernet switches can often be used as layer-3 switches. Such layer-3 switches have limits as they are not designed to replace Internet backbone routers. Nevertheless, current switches from vendors such as HP, NEC, and Cisco, contain components, e.g., TCAMs, for performing longest prefix matching at line rate and offer multiple Terabit throughput.

In this paper, we leverage the decoupling of the traditional tasks of a router: maintaining up-to-date routing information and forwarding packets. We propose an alternative hardware accelerated software router, called **FIBIUM**, that couples a commodity PC running an open source software router with lower cost programmable switching hardware. We are using the switch as a flexible forwarding engine for high-performance forwarding. The PC serves both as a route controller as well as for handling the traffic that was chosen not to be forwarded by the switch.

The advantage of our proposal is that we (a) separate the development cycle for hardware-based forwarding and control software, (b) gain flexibility as routers are increasingly expected to do more than route packets, (c) have a hardware accelerated software based alternative to traditional routers such as those by Juniper and Cisco, especially as switches are becoming a commodity. We do not claim that our approach will lead to alternatives comparable to current high-end switches and routers available on the market, neither in terms of forwarding performance nor features supported.

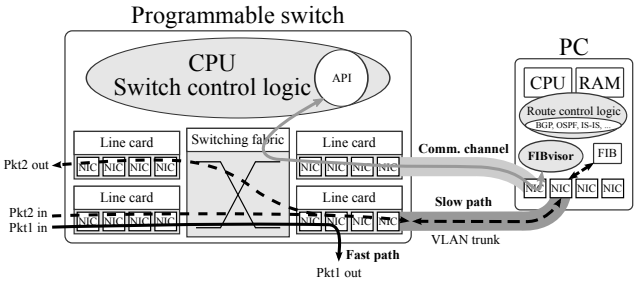


Figure 1: FIBIUM design with its slow path (dashed black arrow), fast path (solid black arrow), and communication channel (grey arrow).

However, we believe that our approach has the potential to bring the flexibility of software routers in previously unexplored contexts, e.g., data centers and aggregation networks.

2. DESIGN / IMPLEMENTATION

The switch provides a hardware accelerated forwarding path as well as a high port density. We use it as our forwarding engine for most of the traffic. The PC serves not only as a route controller, but it also monitors the traffic and maintains a communication channel with the switch to populate its FIB with the most popular destination prefixes. In addition, it provides a slow path to forward the remaining traffic not handled by the switch.

2.1 FIBvisor

Our switch controller **FIBvisor** creates virtual siblings of the switch ports on the PC, providing visibility of all physical interfaces to the software router. Those sibling interfaces see only slow path traffic not handled by the switch, as well as all traffic necessary for routing protocols to operate. As a consequence, any adjacent router connected to one of the physical switch ports can establish, e.g., a BGP peering session with the route controller. To the outside world, the switch-PC combination acts as if it was a conventional IP router.

In addition, **FIBvisor** monitors the routing table as computed and updated by the software router and decides for a

set of prefixes to be installed into the switch FIB. For that, it collects traffic statistics from both the slow path and the fast path. A smart prefix selection strategy implemented within **FIBvisor** then uses these statistics to periodically compose a selection of prefixes which are expected to contribute most of the traffic.

FIBvisor monitors if changes to the FIB on the PC require a modification of the switch FIB. For example, the removal of a prefix in consequence of an incoming BGP withdrawal should immediately be propagated to the switch. Also, **FIBvisor** should be able to detect malicious activity and provide countermeasures against attacks.

2.2 Forwarding

Given that we want to rely on commodity switches that have only limited FIB memory, we cannot install all FIB entries that current routers have to keep into the switch. Accordingly, we introduce a fast path and a slow path for packet forwarding, see Figure 1. If a packet, Pkt1 in Figure 1, is forwarded using the fast path, the relevant FIB entry has already been loaded into the switch FIB by **FIBvisor**. In this case the switch can forward the packet on its own.

If the destination of a packet, Pkt2 in Figure 1, is unknown by the switch FIB, it is directed towards the PC via a dedicated network interface. Upon receiving this packet, the PC can use any forwarding software, e.g., an in-kernel packet forwarding stack, to determine the appropriate output interface on the switch. After rewriting the layer-2 header and decrementing the IP time-to-live field, the packet is sent back to the switch where it is then forwarded via the switching fabric to the appropriate network interface.

A limitation of this approach is the slow path capacity. The prefix selection strategy of **FIBvisor** must make an effort to keep the probability low of overwhelming the slow path with traffic amounts above its capacity.

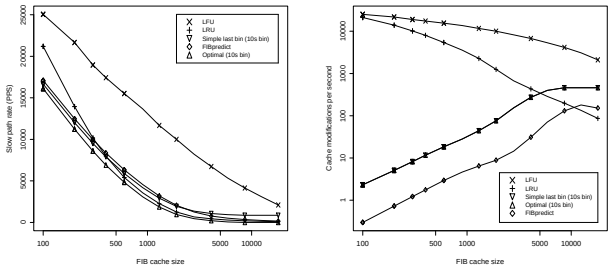
2.3 Programmable switch

FIBvisor relies on an advanced control interface provided by the switch. OpenFlow enabled switches [2] provide a sufficient API. Crucial for **FIBvisor** are control facilities for adding and removing individual FIB entries as well as for retrieving traffic statistics per FIB entry. For our prototype, we use the HP ProCurve 5406zl which supports up to 2000 wildcarded FIB entries. It takes roughly 11ms to obtain statistics of 1000 FIB entries, and the modification of a single FIB entry is completed in 1ms on average.

3. PERFORMANCE

In this section we show that our **FIBIUM** prototype performs sufficiently well for handling traffic demands of a carrier aggregation network. We base our study on an anonymized 48 hours packet-level trace of 20.000 residential DSL lines collected in 2009 at an aggregation point within a large European ISP.

We assume the fast path capacity to be sufficient for handling the full traffic volume. However, the slow path is expected to be limited to a few 100*k* PPS. Also, the communication channel between the switch and the PC also has a limited capacity. Hence, we focus our study on (a) the required slow path capacity and (b) the number of FIB modifications needed, both as a function of FIB cache size.



(a) Slow path requirements (log-linear). (b) FIB cache churn (log-log).

Figure 2: FIBpredict evaluation.

Figure 2(a) shows the impact of FIB size on the resulting slow path rate for different prefix selection strategies. The optimal one updates the FIB every 10s and has perfect knowledge of future traffic. The simple last bin strategy uses the most popular prefixes as seen during the last 10s bin. **FIBpredict** is our strategy which relies on both short-term and long-term trends and also reduces churn by performing FIB modifications only if the expected benefit is significant. For a reasonably sized FIB, all three strategies keep the slow path rate low. Figure 2(b) however shows that **FIBpredict** outperforms both the optimal and the simple last bin strategies substantially in terms of FIB modifications per second.

On Figure 2, we also compare against LRU and LFU, whose applicability is questionable since for each cache miss, a FIB modification has to be done until the packet can be forwarded to its destination, whereas the other strategies immediately forward each packet through either the fast path or the slow path.

4. SUMMARY AND FUTURE WORK

In this work we explore how to take advantage of the significant capabilities of commodity switches and open source software routers to build a hardware accelerated software router called **FIBIUM**. Our prototype, which is based on Quagga, OpenFlow, and our controller **FIBvisor**, is capable of inter-operating with conventional routers and its performance is sufficient to handle the traffic requirements of a carrier aggregation network. In future work we will be investigating the applicability of **FIBIUM** to different deployment scenarios, e.g., data centers.

5. REFERENCES

[1] HANDLEY, M., HODSON, O., AND KOHLER, E. Xorp: an open platform for network research. *ACM CCR 33*, 1 (2003).

[2] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. OpenFlow: enabling innovation in campus networks. *ACM CCR* (2008).
[3] Quagga Routing Suite. `http://www.quagga.net`.