Towards Carrier Grade SDNs

Syed Naveed Rizvi[a], Daniel Raumer[a], Florian Wohlfart[a], Georg Carle[a]

[a] *Technische Universität München, Department of Computer Science, Network Architectures and Services, Boltzmannstr. 3, 85748 Garching, Germany*

Abstract

Driven by the expected benefits, SDN techniques prepare to dare the leap from datacenter- and small area networks to larger networks. SDN becomes more and more interesting in ISP networks, where new concerns about scalability, resilience and legacy functionality arise. Therefore, we analyze RouteFlow, a promising approach to SDN, that allows to gradually transition from conventional networks to SDNs. Based on the requirements for future ISP networks, we perform an experimental case study to validate the applicability of SDN techniques in carrier grade networks.

Keywords: SDN, ISP networks, carrier grade SDN, RouteFlow

1. Introduction

Network operators face challenges as the network traffic is expected to double each 2 to 3 years [1]. In addition, the unprecedented growth of virtualization and applications require frequent changes in network configuration. Therefore, automation and fine-granular control of the network infrastructure is required to cope with the demands of applications and services using it. Originally, researchers at Stanford and UC Berkeley invented OpenFlow to open their campus networks for research [2]. Today, their concept evolved into Software-Defined Networking (SDN), which promises to make networks more flexible and easier to operate. Successful SDN deployments have evolved from special purpose networks [3] over data center networks [4] to wide area networks [5].

Email addresses: naveed.rizvi@tum.de (Syed Naveed Rizvi), raumer@net.in.tum.de (Daniel Raumer), wohlfart@net.in.tum.de (Florian Wohlfart), carle@net.in.tum.de (Georg Carle)

In this paper we make an intellectual journey into possible future ISP networks and evaluate the SDN concept from the perspective of ISPs. We discuss the main challenges in future ISP networks in section 2 and present background information on RouteFlow, an SDN solution that mimics conventional routing protocols, in section 3. In section 4, we analyze and experiment with RouteFlow to assess the suitability of this approach for future ISP networks. We summarize our case study and conclude whether RouteFlow is able to approach the identified problems in ISP networks in section 5.

2. Challenges in ISP Networks

SDN deployments first emerged in data center networks, which are different from ISP networks in size, topology, geographical distribution, provided services, and the relationship with customers. ISPs offer services that range from simple IP connectivity to sophisticated services like VPN, VPLS, leased lines, DSL, or IPTV [6].

To provide these services, the network operator has to support configuration and chaining of internal functions including content caching, subscriber management, QoS provisioning, routing, traffic engineering, DNS, DHCP, AAA, DPI, firewalls, and monitoring. According to the Open Networking Foundation (ONF) – an organization formed to promote SDN – conventionally operated networks face the following challenges [7]:

1. **Device configuration costs.** The individual and often manual configuration of network devices impedes the swift provisioning of dynamic services. The configuration process cannot keep up with on-the-fly changes required by modern applications and does not scale with the requirements.
2. **Vendor lock-in.** When providing new services, ISPs are dependent on the development life-cycle of hardware vendors that introduce new capabilities. This vendor lock-in may also increase the cost to make changes to the current setup.
3. **Configuration complexity.** The complexity increases the risk of implementing inconsistent policies, as configuration tools are device-centric and require network operators to configure a large number of network nodes.
4. **Customization costs.** It is difficult to achieve individual customization with manual configuration in large-scale service provider networks.

5. **Labor costs.** Customized solutions for network configuration require a significant number of engineers to run these systems. This large scale human intervention in the manual configuration process results in higher OPEX.
6. **Overprovisioning.** Inefficient use of network resources requires overprovisioning which leads to higher CAPEX to meet customer demands.

Future SDN solutions promise to solve these problems by simplifying network administration, increasing bandwidth utilization, and enabling the rapid deployment of new services in the network.

3. Background

After looking at current and future problems in ISP networks, this section gives an overview of a specific SDN implementation.

3.1. RouteFlow

We study RouteFlow [8, 9], an approach that implements conventional IP routing on top of an OpenFlow network. Using existing distributed routing protocols (OSPF, BGP, etc.), RouteFlow can seamlessly integrate OpenFlow switches into conventional networks and reuse stable, well-tested code, while being able to benefit from increased scalability and vendor-neutrality. In comparison to progressive clean-slate approaches [10, 11], RouteFlow follows a more conservative approach with focus on backwards compatibility.

The case study targets RouteFlow, because it allows the gradual introduction of OpenFlow into an existing network, in contrast to clean-slate approaches that require rebuilding a new network from scratch. Migrating a large ISP network to SDN will only be feasible using a step-by-step approach, since replacing the whole network at once is too costly. RouteFlow is actually deployed in real networks [12, 13] and its source code is freely available under the terms of the Apache Public License, which gives us the opportunity to study it in detail.

As depicted in Figure 1, RouteFlow implements a virtual control plane by using virtual machines (VMs). Each VM represents the control unit of a logical node. These VMs run any Linux-based routing engine to update their forwarding information base (FIB). A RouteFlow daemon running in each VM listens to updates in the OS routing table and translates the routing table entries into OpenFlow rules that control the forwarding behavior of

the switches. It is possible to configure the virtual control plane to either exactly replicate the physical topology of the OpenFlow switches or to define an arbitrary mapping between the router VMs and switches. Separating the logical topology (VMs) from the physical topology (OpenFlow switches) allows to simplify the logical network. The virtual routers exchange the reachability information with each other through the OpenFlow switches in the underlying network. In turn RouteFlow collects the routing tables from these routers and instructs the OpenFlow controller to translate the routing table entries into corresponding OpenFlow rules for installation into the OpenFlow switches. The installation of specific flow table entries in switches enable the control traffic (for example OSPF, BGP, ICMP packets, etc.) to traverse the data plane and reach the desired virtual router.

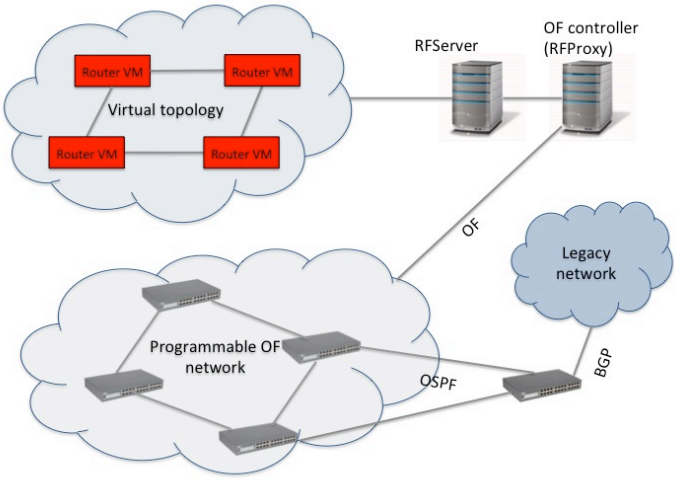

Figure 1: RouteFlow system overview

RouteFlow consists of multiple components which play different roles. They facilitate the collection of routing tables from virtual routers, conversion to flows, installation of flows in the switches, and manage the mappings between routing engines and OpenFlow switches. RouteFlow's architecture aims for the following goals:

- Compatibility with conventional networks by using existing routing protocols in the control plane

- Decoupling of the control and data plane topologies to enable network virtualization
- Utilization of open source routing software
- Capability to work without a global view of the network, i.e. each router makes locally optimized decisions

RouteFlow does not make routing decisions by itself. Instead, it relies on the routing protocols running in the emulated nodes. The routing performance of RouteFlow is therefore limited by the performance of these conventional routing protocols.

The controller implementation can take form of either a monolithic entity or a modular system consisting of constituent control functions for example topology discovery, event manager, state collector, routing etc. These modules aid the controller in creating a global network view. The controller exposes this view to external applications for implementing network services [14].

Control functions can also be implemented as external applications using the northbound interface or more appropriately the Application-Controller interface of the controller. The external application can both invoke the services exposed by the controller and provide its services to the controller. RouteFlow is implemented as an application that utilizes a northbound OpenFlow controller interface.

RouteFlow implements OpenFlow Version 1.0, which provides only limited match fields and operations on packet headers. Since RouteFlow only requires matching the destination IP address for routing, OpenFlow Version 1.0 provides sufficient match fields but lacks support for operations like TTL decrement or support for IPv6. While these are substantial shortcomings, they are not inherent to the RouteFlow design and can be solved by migrating RouteFlow to recent versions of OpenFlow.

3.2. SDN Criteria

Jarschel et al. [15] define four requirements for a network architecture to fully benefit from all advantages connected to SDN: separation of control and data plane (1), logically centralized control (2), open interfaces (3) and programmability (4). In this section we evaluate RouteFlow using the requirements defined by Jarschel et al.

1. RouteFlow provides separation between the control and data plane. It uses the OpenFlow protocol to connect OpenFlow switches to the control plane consisting of an SDN controller and multiple VMs.
2. The use of distributed routing protocols running inside VMs does not conform to the notion of a logically centralized control plane. Rather than having a central application that has a global view of the network, control is spread throughout multiple VMs, each running a distributed routing protocol as a separate process.
3. RouteFlow implements open interfaces. These are northbound, southbound, east-, and westbound interfaces. RouteFlow utilizes OpenFlow as southbound interface, distributed routing protocols (e.g. OSPF, BGP) as the east/westbound interface to communicate with conventional networks and with other SDN domains.
4. Finally, RouteFlow cannot satisfy the programmability requirement. Like conventional networks, RouteFlow is controlled by distributed independently operating subprocesses. This means RouteFlow does not offer the possibility to program the whole network as a single entity instead it is only able to control individual nodes in the network.

In summary, RouteFlow provides only partial conformance to the SDN definition presented in [15]. This is not surprising considering the conservative approach of RouteFlow. Nevertheless, RouteFlow is able to provide the elasticity of control plane resources and protocol independence because of the use of virtualization, while the features like network programmability, flow granularity, and dynamic configuration are not present.

4. Case Study: RouteFlow

After presenting the theoretical background in section 3, this section provides the findings of our hands-on evaluation of RouteFlow, the main contribution of this paper. Our case study is based on multiple tests designed to analyze the suitability of RouteFlow for future ISP networks: we perform data- and control-traffic forwarding, build a model for the relation between routing and flow tables, estimate the flow table size and memory requirements, and finally test the reaction to failure in the data plane, control plane or OpenFlow connection.

4.1. Data Traffic Forwarding

To emulate the forwarding behavior of a conventional router on top of OpenFlow switches, RouteFlow programs the switches with flow table rules, that approximate routing behavior. An OpenFlow switch matches incoming packets against the ruleset in its flow table. Each rule specifies actions such as forward the packet to an outgoing port, drop the packet, forward the packet to the controller, or more advanced actions. The non-matching packets are either dropped or forwarded to the controller.

The controller faces a trade-off between performance and flexibility when installing flow rules in the switches. Handling packet flows autonomously on the OpenFlow switches increases the data plane performance, but limits the packet processing to the OpenFlow actions supported by the switch. Forwarding a packet flow to the controller introduces a performance penalty, but allows arbitrary packet processing on the controller. Forwarding IP packets based on their destination IP addresses can be achieved with the features that are readily available in all OpenFlow switches. Therefore, IP packets can be forwarded by OpenFlow switches without involving the controller.

RouteFlow makes use of proactive, reactive, and interpreted forwarding rules. It installs proactive and reactive rules for IPv4 traffic addressed to a remote network destination that can either be a host or a virtual router. A proactive rule is installed in an OpenFlow switch beforehand and it is used for all the reachable network destinations learned by the routers. Reactive rules are installed in an OpenFlow switch in reaction to a received packet for which no matching rule was present. They are used for directly connected hosts and next hop routers. The reactive flows have IP address level granularity while the proactive flows are for the whole subnet. The third category – interpreted flows – means that each packet in the flow is forwarded to the controller, for example ICMP packets. Interpreted forwarding is only used for out-of-band control traffic. Table 1 presents a summary of different types of flow table rules, installation methods and their granularity levels.

Figure 2 shows the path of an IP packet (solid line: steps 2,3,4,6) through a RouteFlow controlled network with proactive rules already installed on the intermediate nodes A and C and reactive rules installed after an ARP request to get the next hop (dashed line: step 5) on node B. Preliminary to the initial transmission of the IP packet on host H1 was an ARP request (dashed line: step 1). The use of proactive flows results in only one OpenFlow Packet-in message from the egress switch at the start of the first flow to a particular destination. Therefore, reactive installation for the hosts in edge switches can

Packet type	Destination	Flow installation	Flow granularity
IPv4	remote router	proactive	coarse
IPv4	neighbor router	reactive	fine-grained
IPv4	remote host	proactive	coarse
IPv4	neighbor host	reactive	fine-grained
ARP, ICMP, routing protocols	any	interpreted	-

Table 1: Types of flow rules installed by RouteFlow.

happen in two ways: if the host initiates a packet flow or if it is the destination of a packet flow. Similarly, the reactive flows for neighboring routers are installed. The overall effect is that the IPv4 data traffic is forwarded in the data plane without interacting with the OpenFlow controller. Therefore, we do not expect an effect on the forwarding performance.

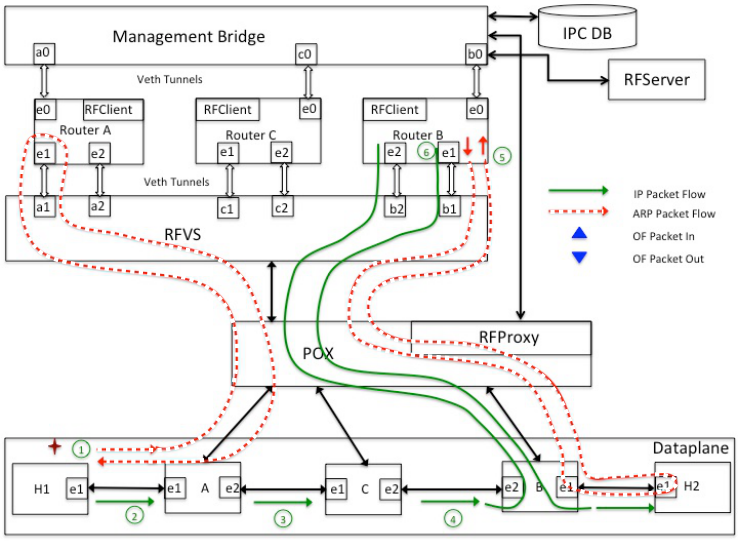

Figure 2: IP Packet flow through a RouteFlow controlled network.

4.2. Control Traffic Forwarding

The virtual routers running in the control plane need to exchange control traffic via protocols like OSPF, BGP, ICMP, ARP, etc. Control traffic needs to be forwarded to the virtual destination router. RouteFlow uses interpreted flows to forward these packets to the virtual routers. In RouteFlow

the control traffic is forwarded via the control plane at each hop, even if it is destined for another virtual router. This behavior does not affect the control traffic between neighbors (e.g. OSPF or ARP), but it adds extra delays for the control protocol packets (e.g. ICMP or BGP), that need to transit through multiple OpenFlow switches before reaching their intended destination. As shown in Figure 3, an ICMP packet from host $H1$ to host $H2$ is sent to the first hop switch A (1), encapsulated into OpenFlow packets (2,3) for transmission to the virtually abstracted topology, treated and forwarded by the first hop VM (4), and sent back via OpenFlow packets to the first hop switch A. The same happens at switch B until the ICMP packet reaches its destination host H2.

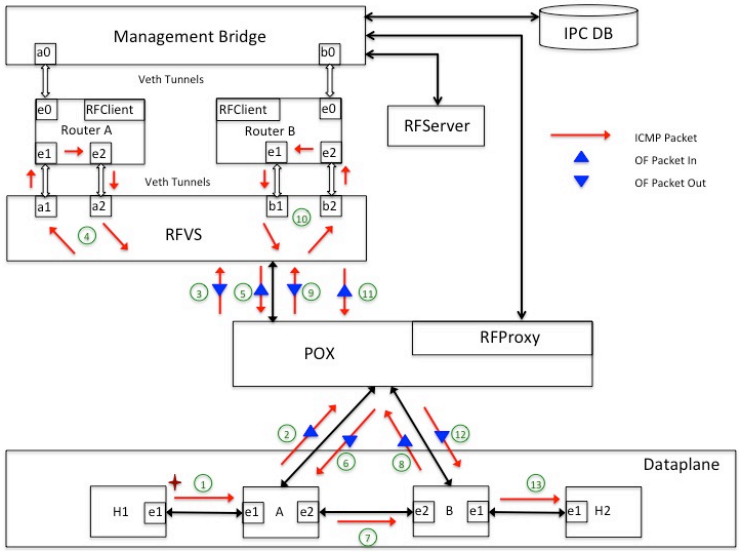

Figure 3: ICMP Packet flow through a RouteFlow controlled network.

Similarly, in a non-MPLS core, multiple internal BGP sessions are created between the routers to exchange reachability information learned via external domains. In a RouteFlow controlled network these BGP packets will traverse through the controller at each hop to reach the intended destination router. For a network with an MPLS enabled core the edge routers still need to form internal BGP sessions that will be forwarded through virtual routers.

The control traffic handling in RouteFlow leads to another issue in addition to extra delays which is the use of four OpenFlow Packet-in and Packet-out messages per packet at each hop. These messages on the link between

the controller and OpenFlow switches along with packet handling is an additional load on the controller. We propose to decrease the priority value of the interpreted flows for multiple hop control traffic below the proactive and reactive flow rules. Therefore, the control packets can be forwarded entirely in the data plane and are only sent to the intended virtual router through its controlled OpenFlow switch.

Another important issue is the lack of the TTL decrement action in OpenFlow Version 1.0, which leads to the inability to use traceroute and measure delays in a RouteFlow controlled network. If packets are not sent to the router VMs at each hop, these hops are not visible in a traceroute. Also any measured delays are a false estimate if packets are sent to the control plane. It can deviate from the real values if the controller is placed at a distance comparable to the distance of the next hop switch. This way latency measurements in the network will show large differences. So RouteFlow should support newer OpenFlow versions with a TTL decrement feature in the switch specifications to solve this issue.

4.3. Priority Levels of the Flow Table Entries

Flow table entries in OpenFlow are matched from high to low priority. Then the first matching rule is applied. This is different from conventional routing, where longest prefix matching is used. Therefore, RouteFlow implements a priority level calculation mechanism based on the subnet mask length so that the more specific flows are used before generic flow rules. RouteFlow defines a base priority level, drop priority level, a high priority level for control traffic, and a multiplier for each bit in the subnet mask for proactive and reactive flow rules. Trying to match more fine grained subnets in the data plane increases the multiplication factor.

$$Priority = Base\ Priority + Offset \tag{1}$$

$$Offset = Multiplying\ Factor \times Subnet\ Prefix\ Length \tag{2}$$

RouteFlow uses a predefined priority level for the interpreted flow rules used for RIP, ICMP, ARP, IPv6, BGP, LDP, and OSPF. The priority of these flows is higher than any other flows. Therefore, these packets are always forwarded to the controller and incur forwarding delays while being processed in the control plane. In section 4.2, we propose a modification of the predefined priority to achieve line rate forwarding for control traffic.

4.4. Flow Table Size and Memory Requirements

This section addresses the issue of using OpenFlow switches instead of legacy routers and their capability to hold flows with equivalent forwarding behavior as routers. The size of the routing table depends on the role of a router in the network (i.e access, edge, core, etc.) and the network topology.

We have found that RouteFlow translates a single route in a virtual router to multiple flow rules in an OpenFlow switch. The reason for this one-to-many relation is the use of the input port number of received packets and the MAC address of the destination node as match fields in the flow table rules. The impact of this relationship has effects on replacing all types of routers in different roles. Usually, the access routers have high port density while the core routers have large routing tables. The available OpenFlow switches are capable of holding only a few thousand flows, while the routers used in the core or the edge of a service provider network hold a few hundred thousand routes [16]. In addition to large memory requirements for flow tables it is also potentially problematic to add a new switch in a running network. It can be predicted that the introduction of a new switch will suddenly increase the OpenFlow Flow Mod messages which will put stress on the link between the controller and OpenFlow switches.

We derived Equations 3 and 4 to estimate the number of flow rules that are installed in an OpenFlow switch. The total number of flow table entries (TF) is the sum of fixed flows (FF) and variable flows (VF). The fixed flow rules represent the interpreted flows for control traffic and their number is fixed for all the OpenFlow switches therefore the term fixed flows (FF). The number of variable flow rules (VF) is dependent on multiple parameters specific to a switch and represent both the reactive and the proactive flows. It depends upon the following parameters: 1. Number of ports in OpenFlow switch (NP) controlled by RouteFlow, 2. Number of directly connected nodes (DC), 3. Number of indirectly connected subnets (IC).

$$TF = FF + VF \tag{3}$$

$$TF = FF + (NP - 1) \times (DC + IC) \tag{4}$$

Using the simple example networks in Figure 4 and 5, we have calculated the number TF for switch A. In these diagrams, circles represent OpenFlow switches controlled by RouteFlow and rectangles represent hosts or simple

Ethernet switches not controlled by RouteFlow. The values of different parameters are shown in Table 2.

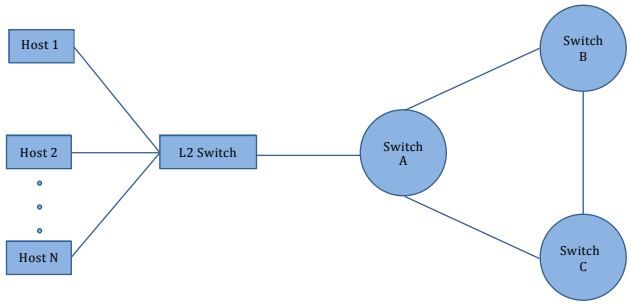

Figure 4: Switch A with DC = N+2 and IC = 1

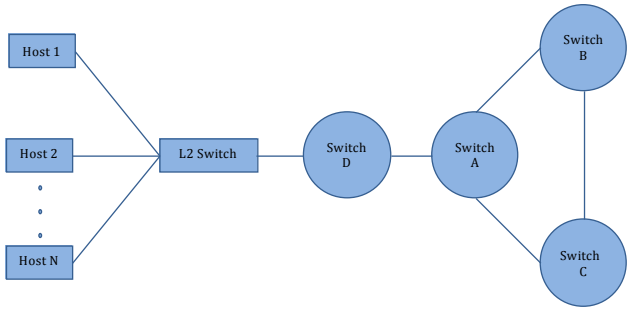

Figure 5: Switch A with DC = 3 and IC = 2

Network	**NP**	**DC**	**IC**	**TF**
Figure 4	3	N+2	1	FF + (3-1)(N+2 + 1)
Figure 5	3	3	2	FF + (3-1)(3 + 2)

Table 2: Total Number of Flows in OpenFlow Switch A for Networks in Figures 4 and 5

In Figure 4, OpenFlow switch A is connected to N hosts via a Layer 2 switch and with the OpenFlow switches B and C, therefore switch A has a value of DC equal to N+2. IC is 1 as the subnet between switches B and C is the only indirectly connected subnet. In Figure 5, OpenFlow switch A is now connected to three directly connected switches B, C and D, so DC equals 3. Now the value of IC equals 2 as all the N hosts are now advertised as a single subnet plus the subnet between switches B and C.

Therefore, switch A has an IC value that depends on neighboring switches. The switches B, C, and D can be connected to multiple subnets directly or via other switches, therefore increasing the value of IC for switch A. In general, for a large service provider the value of IC will be large, as neighbor switches will advertise thousands of routes.

The issue of large memory requirement and the unavailability of OpenFlow switches that can hold such a number of flows is going to hamper the adoption of RouteFlow as a SDN solution. We propose, instead of installing multiple flows for each route, to use only two flows per route. One low priority flow for all the possible input ports and the other flow with higher priority to block IP packet received through the outgoing port to avoid loops. In this way, the flow table size can be limited to twice the size of the routing table and not dependent on the number of ports in the switch. It is necessary to mention that this approach cannot be used with the router multiplexing mode of operation available in RouteFlow to divide a physical switch between multiple virtual routers.

4.5. Resilience

To evaluate the resilience of RouteFlow and its ability to recover from faults we derive multiple test cases. These test cases deal with the failures in data plane, control plane and the OpenFlow connection between them.

In the data plane, link failure between two OpenFlow switches or port failure on an OpenFlow switch may occur. We observed that RouteFlow is unable to differentiate between failures that are explicitly reported by the switches to the controller and the ones that are not reported, since RouteFlow exclusively relies on routing updates to detect changes or failures in the network. The recovery time from both types of failure is similar and in the range of multiple routing updates. In our case the routing updates are sent at an interval of 4 seconds and the recovery times are about 10 seconds. The routing update interval can be reduced to shorten the recovery time but it has its own negative impacts on the network stability. These recovery times are obviously too large for production service provider networks where 50 ms recovery time is already achievable using the MPLS fast reroute mechanism [17].

The router VMs can fail unexpectedly resulting in route modifications in neighboring routers and flow modifications in the switches. Our analysis revealed that RouteFlow is unable to recover from such failures in an automated way and it requires manual intervention and configuration to recover

failed router VMs. We froze and unfroze the running VMs to analyze how RouteFlow reacts to maintenance or migration of VMs. In this case RouteFlow was able to automatically reintegrate the router VMs. The only effect were the temporary changes in the routes in neighboring routers and flows in their controlled switches. The use of virtualization is beneficial in a sense that it can be used to migrate an active router to another server for scaling the processing resources on demand or planned maintenance.

Another test showed that a failure of communication between the OpenFlow controller and the switches stops data forwarding even if there are flow table entries that are still intact in the effected switches. The main cause is the unsuccessful ARP requests from the end hosts that can only be answered by the routers controlling the edge switches. If the communication failure occurs only between core switches and the OpenFlow controller, it is possible to support data traffic via alternative paths available in the core.

To increase its reliability RouteFlow needs to incorporate the automated router VM recovery and OpenFlow event reporting mechanisms. The dependence on routing protocols to recover from failures in the data plane is an unsuitable option for service providers. It was shown [18] that it is possible to achieve less than 50 ms recovery time using OpenFlow event reporting mechanism and BFD protocol by the OpenFlow controller. This requires RouteFlow to support OpenFlow specification version 1.1 or above and precalculate multiple independent paths for each route. The OpenFlow Version 1.3 specification [19] includes the option for OpenFlow switches to connect to backup controllers in addition to a primary controller. This feature will help in providing redundancy for the control plane and reduce the number of interruptions in the network.

4.6. Migration and Communication with Conventional Networks

The challenge of replacing the conventional networks with SDNs is multifaceted, for example the ISPs have a large installed network infrastructure which they rely on for their business. Their network infrastructure is a large investment, that cannot be replaced at once. Also the network operators are trained to deploy vendor-specific solutions over the years. At the moment, some OpenFlow switches do not execute all actions in line rate, thus lacking performance compared to hardware routers. Therefore, economic, technological, and human resource constraints should be properly addressed to achieve a transition to an all software defined network.

Autonomous Systems (ASes) use BGP to exchange global routing information and announce their reachable IP address space. Therefore, it is necessary to support BGP to establish connectivity with other ASes. RouteFlow supports a gradual approach to transform conventional networks to SDN enabled networks by supporting the isolation between the edge and core routing functions through router aggregation mode. In our study, we evaluate this feature of RouteFlow by controlling all the external BGP-speaking edge switches using a single router VM.

RouteFlow supports an aggregation mode, where multiple OpenFlow switches in the data plane can be aggregated into one virtual router in the control plane. One possible use-case would be to use one central routing function for a core network, since aggregating all switches into one virtual router results in a logically centralized control plane. The single router would then set the received routes and in turn install flows in core OpenFlow switches. Then, the core network does not require BGP support. Another advantage that can be achieved by this approach is to install flow entries in only those switches that will be used to route the traffic, thus limiting the growth of flow tables in core switches. It is similar to using MPLS which also makes the core network BGP free. Although the current distributed implementations of MPLS still require per-node configuration there has been some effort on centralized MPLS [20, 21] and centralized routing functionality [22].

The current RouteFlow implementation only supports proper aggregation for fully meshed OpenFlow switches. In other topologies, RouteFlow relies on the aggregated virtual router to forward transit traffic. This approach is inefficient and requires further development of RouteFlow to accommodate non-meshed topologies.

In the transitioning phase to SDN the use of available solutions to enable the compatibility and the idea of isolating the core network by using edge router aggregation functionality seems to be most promising. The aggregated network will appear to the outside world as a single router therefore hiding the internal details of the network. The use of a centralized SDN routing solution eliminates the need for distributed routing protocols in the network core. It is also possible to implement MPLS-like functionality inside the core network in a centralized way or any form of routing or switching in general. To efficiently manage the core switches, we can utilize the approach presented in [23], where multiple controllers can control a specific portion of the core network. These controllers can cooperatively make routing decisions, provide easier management, and better resilience in the event of controller failure.

5. Summary

The RouteFlow project provides a proof of concept to gradually deploy SDN in real world scenarios. The use of well-tested routing protocols along with the flexibility of network virtualization makes it a commendable starting point. However, our case study revealed numerous problems that the RouteFlow implementation faces in practice.

Out of the six challenges in ISP networks we discussed in section 2, RouteFlow is only able to address **vendor lock-in**. RouteFlow relies on switches implementing the OpenFlow protocol and Linux VMs, so it does not require buying from a specific vendor. Moreover, it enables network operators to gradually replace vendor-specific routers with commodity OpenFlow switches. It remains unclear whether RouteFlow will reduce **device configuration costs**, **customization costs** and, as a result, **labor costs**. On one hand, RouteFlow facilitates the network configuration by virtualizing the control plane, so it can run on a single machine. This makes it easy to track the global configuration state and troubleshoot problems. On the other hand, RouteFlow does not deploy a logically centralized control plane. Control is split among multiple cooperating VMs. This means device configuration does not improve. In fact, the deployment of RouteFlow requires additional work to setup all necessary components. The deployment of RouteFlow adds **additional complexity** to an existing network. Finally, RouteFlow even aggravates the problem of **overprovisioning** by adding additional control traffic on top of the traffic carrying actual payload.

Some of these problems can be addressed with some effort. We proposed possible solutions in our case study. RouteFlow would benefit greatly from an update to support OpenFlow Version 1.3 or later, since OpenFlow Version 1.3 allows decrementing the TTL value, and supports IPv6 [19]. It also improves network resilience through a new feature to configure multiple controllers in a switch [19]. If the primary controller is not reachable, the switch can use another controller as a fall-back option. Additionally, the resilience can be improved by utilizing the OpenFlow link failure reporting mechanism instead of waiting for the routing protocol to converge. As far as the control plane is concerned, an automated virtual router recovery mechanism could restore VMs without manual intervention. RouteFlow stores a large number of flow table entries on the OpenFlow switches. Removing the matching on incoming ports can help reduce the number of flow table entries in typical scenarios.

Other problems are inherent to the design of RouteFlow and therefore

hard to fix. This includes its lack of a logically centralized path selection and its limited performance. Instead of making control plane decisions in the controller, RouteFlow relies on decentralized routing protocols. RouteFlow runs a whole network of virtualized routers with full featured OSes, although it needs only a small fraction of their functionality. Therefore, compared to a logically centralized control plane, the RouteFlow control plane is complex, limits performance, and complicates the network configuration.

Acknowledgments

This research has been supported by the DFG (German Research Foundation) as part of the MEMPHIS project (CA 595/5-2), the EU as part of KIC EIT ICT Labs on SDN, and the BMBF (German Federal Ministry of Education and Research) under EUREKA-Project SASER (01BP12300A).

References

[1] Cisco, Cisco Visual Networking Index: Forecast and Methodology, 2012–2017, Whitepaper, Cisco (May 2013).

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: Enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74.

[3] Big Switch Networks Inc, Big tap monitoring fabric, Datasheet.

[4] OpenStack, `http://www.openstack.org/`, last visited 2014-10-15.

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, A. Vahdat, B4: Experience with a globally-deployed software defined wan, SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 3–14.

[6] O. M. Heckmann, The Competitive Internet Service Provider: Network Architecture, Interconnection, Traffic Engineering and Network Design, Wiley Series on Communications Networking and Distributed Systems Ser., John Wiley & Sons, Ltd., 2006.

[7] ONF, Software-Defined Networking: The New Norm for Networks, White Paper.

[8] M. R. Nascimento, C. E. Rothenberg, M. R. Salvador, C. N. A. Corrêa, S. C. de Lucena, M. F. Magalhães, Virtual Routers As a Service: The RouteFlow Approach Leveraging Software-defined Networks, in: Proceedings of the 6th International Conference on Future Internet Technologies, CFI '11, ACM, New York, NY, USA, 2011, pp. 34–37.

[9] C. E. Rothenberg, M. R. Nascimento, M. R. Salvador, C. N. A. Corrêa, S. Cunha de Lucena, R. Raszuk, Revisiting routing control platforms with the eyes and muscles of software-defined networking, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12, ACM, ACM, New York, NY, USA, 2012, pp. 13–18.

[10] M. Casado, T. Koponen, S. Shenker, A. Tootoonchian, Fabric: a retrospective on evolving sdn, in: Proceedings of the first workshop on Hot topics in software defined networks, ACM, 2012, pp. 85–90.

[11] B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, S. Shenker, Software-defined internet architecture: decoupling architecture from infrastructure, in: Proceedings of the 11th ACM Workshop on Hot Topics in Networks, ACM, 2012, pp. 43–48.

[12] J. P. Stringer, Q. Fu, C. Lorier, R. Nelson, C. E. Rothenberg, Cardigan: Deploying a distributed routing fabric, in: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, ACM, 2013, pp. 169–170.

[13] A. Vidal, F. Verdi, E. L. Fernandes, C. E. Rothenberg, M. R. Salvador, Building upon RouteFlow: a SDN development experience, in: XXXI Simpósio Brasileiro de Redes de Computadores, SBRC'2013, 2013.

[14] Open Networking Foundation, SDN architecture, `https://www.opennetworking.org`, last visited 2014-10-15.

[15] M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia, W. Kellerer, Interfaces, attributes, and use cases: A compass for SDN, Communications Magazine, IEEE 52 (6) (2014) 210–217.

[16] D. Kreutz, F. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-Defined Networking: A Comprehensive Survey, arXiv preprint arXiv:1406.0440.

[17] C. R. Kalmanek, S. Misra, Y. R. Yang (Eds.), Guide to Reliable Internet Services and Applications, Computer Communications and Networks, Springer, 2010.

[18] S. Sharma, D. Staessens, D. Colle, M. Pickavet, P. Demeester, Enabling fast failure recovery in OpenFlow networks, in: Design of Reliable Communication Networks (DRCN), 2011 8th International Workshop on the, IEEE, 2011, pp. 164–171.

[19] ONF, OpenFlow Switch Specification 1.3.0, `https://www.opennetworking.org`, last visited 2014-10-15.

[20] A. R. Sharafat, S. Das, G. Parulkar, N. McKeown, MPLS-TE and MPLS VPNs with Openflow, in: ACM SIGCOMM Computer Communication Review, Vol. 41, ACM, 2011, pp. 452–453.

[21] S. Das, A. Sharafat, G. Parulkar, N. McKeown, MPLS with a simple OPEN control plane, in: Optical Fiber Communication Conference, Optical Society of America, 2011.

[22] G. Khetrapal, S. K. Sharma, Demystifying Routing Services in Software Defined Networking, White Paper.

[23] X. T. Phan, N. Thoai, P. Kuonen, A collaborative model for routing in multi-domains OpenFlow networks, in: Computing, Management and Telecommunications (ComManTel), 2013 International Conference on, IEEE, 2013, pp. 278–283.

Abbreviations

AAA – Authentication Authorization Accounting
ARP – Address Resolution Protocol
AS – Autonomous System
BFD – Bidirectional Forwarding Detection
BGP – Border Gateway Protocol

CAPEX – capital expenditures
DC – directly connected nodes
DHCP – Dynamic Host Configuration Protocol
DNS – Domain Name System
DPI – Deep Packet Inspection
DSL – Digital Subscriber Line
FF – sum of fixed flows
IC – indirectly connected nodes
ICMP – Internet Control Message Protocol
IPTV – IP based TV streaming
IP/IPv4/IPv6 – Internet Protocol
LDP – Label Distribution Protocol
NP – number of ports
MPLS – Multiprotocol Label Switching
OPEX – operating expenditure
OS – Operating System
OSPF – Open Shortest Path First
QoS – Quality of Service
RIP – Routing Information Protocol
SDN – Software Defined Network
TF – flow table entries
TTL – Time To Live
VF – sum of variable flows
VM - virtual machine
VPLS – Virtual Private LAN Service
VPN – Virtual Private Network