Demo: Resilient Integration of Distributed High-Performance Zones into the BelWue Network Using OpenFlow

Mark Schmidt*, Robert Finze†, Daniel Reutter*, Michael Menth*
* Chair of Communication Networks and † Zentrum für Datenverarbeitung, University of Tuebingen, Tuebingen, Germany

***Abstract*—The Internet service provider (ISP) BelWue interconnects higher education and research institutions in Baden-Wuerttemberg, Germany. Its 10 Gb/s backbone among university campuses has been augmented with an additional optical 100 Gb/s high-speed network called NeIF. The bwNET100G+ project sets up special high-performance zones (HPZ) at a few university campuses and seeks for cost-efficient interconnection over the NeIF and integration into the legacy infrastructure. We propose for that purpose SDN-NeIF, a software-defined network architecture leveraging OpenFlow and iBGP. The demo illustrates this concept for three university locations with a special focus on resiliency. It leverages a virtualized testbed consisting of four desktop PCs and three consumer switches. The demo validates the SDN-NeIF architecture and serves as a first proof-of-concept.**

I. Introduction

The Internet service provider (ISP) BelWue [1] interconnects higher education and research institutions in Baden-Wuerttemberg, Germany. BelWue offers universities a redundant 10 Gb/s uplink to its core network and the Internet. The core network was recently expanded with a new 100 Gb/s optical platform, called NeIF (Netzwerk fuer Innovation und Forschung / network for innovation and research). A 10 x 10 Gb/s switching matrix in any university location enables the setup of 10 Gb/s optical point-to-point connections between any two locations. To leverage the increased network capacities, novel high-performance zones (HPZ) with high-performance servers and switches are installed within the bwNET100G+ project at university locations in Karlsruhe, Tuebingen, and Ulm. One objective is to seek for cost-efficient interconnection of HPZs over the NeIF and their integration into the legacy infrastructure which can be achieved by the use of SDN-capable switches instead of full routers.

In this paper we describe an architecture (SDN-NeIF) that leverages OpenFlow [2] and iBGP [3] for that purpose. We describe a demo implementing SDN-NeIF on a virtualized platform and illustrate the data flow under working and failure conditions.

The design meets some important real-world requirements. Optical 10 Gb/s links must be efficiently used as they do not suffice to support a full mesh; furthermore, some of them need to be reserved for special services. While campuses have a redundant uplink to the existing BelWue network, the NeIF typically connects neighboring locations only with a single link. In case of a link failure in the NeIF, failover mechanisms should guarantee communication with and among HPZs over the legacy network. The solution should be independent of the internal structure of the campus network and the HPZ because these structures may vary among locations. Moreover, the design must respect that border routers and border switches are administered by BelWue while other campus and HPZ infrastructure is operated by universities. We choose a software-defined networking (SDN) approach because it facilitates cost-effective IP-based forwarding using switches and allows for advanced traffic engineering and security solutions in the future.

The remainder is structured as follows. Section II reviews related approaches interconnecting "Science DMZs" and research networks. In Section III, we introduce the SDN-NeIF architecture. Section IV explains a virtualized testbed implementing SDN-NeIF and Section V describes the demo to be shown. Section VI concludes this work.

II. Related Work

We briefly review other projects that provide high-speed networks for research and how they are integrated into a campus infrastructure.

ESnet Science DMZ [4], [5] proposes high-speed zones within a university that form a dedicated network for research data in addition to a general purpose network at a university campus. A high-speed DMZ is a central place for storage and computation servers which has special security policies and enforcement. Typically, the access switch for a high-speed DMZ is directly connected to the university border router. SDN facilitates the communication within a DMZ and among different DMZs of a university. The research network within a university provides a bandwidth of 100 Gb/s to the DMZs which allows to use resources across DMZs without relaying traffic through the campus network. A high-speed network among the DMZs of different universities is not mentioned.

Internet2 [6] provides US educational and research centers with a typical uplink of 10 Gb/s through an optical 100 Gb/s backbone. Its goal is to provide high-speed access for collaborative applications, distributed research experiments, grid-based data computation, and analytics. In particular, Internet2 can interconnect Science DMZs at different locations.

McCahill [7] described at an Internet2 technical meeting a bypass of the campus core network of a university for special traffic, e.g., scientific data. Different departments of a

This work has been supported in the bwNET100G+ project by the Ministry of Science, Research and the Arts Baden-Wuerttemberg (MWK). The authors alone are responsible for the content of this paper.

978-0-9883045-1-2/16 $31.00 © 2016 ITC

university are connected to the campus core network via SDN-capable switches and the bypass among them is implemented with dedicated high-speed links.

SciPass [8] describes a security-enhanced Science DMZ. An intrusion detection system (IDS) is connected to an SDN-capable switch and classifies traffic as trusted and untrusted. An example for trusted traffic may be scientific data exchanged between universities. Once a flow is classified as trusted, the network is configured so that this flow is routed around potential bottlenecks and can bypass firewalls. The goal of this approach is to enable the utilization of a 100 Gb/s connection between different campuses.

GEANT2 [9] is a European research network. The GEANT project already exists for several years and its backbone has been upgraded from 100 Gb/s in 2013 to currently 500 Gb/s. Its core is based on dark fibres with an OTN switching layer and a GMPLS control plane.

III. SDN-NeIF Architecture

We introduce the physical interconnection of the BelWue core, campus networks, and HPZs, explain the integration of the HPZs using OpenFlow and iBGP, and present failover mechanisms.

A. Physical Interconnection

Figure 1 shows that a campus router (CR) confines the border of a campus network and connects to a BelWue border router (BR) on the premises of the university. The BR provides a 10 Gb/s uplink to the BelWue core network and access to the Internet. All HPZs together have an IPv4 */22* address space out of which each HPZ has its own */24* prefix that is administered by the corresponding university. A HPZ is attached to a BelWue border switch (BS) on the premises of the university which interconnects via the NeIF to BSs of other locations. The BS has a direct local link to the CR and the BR.

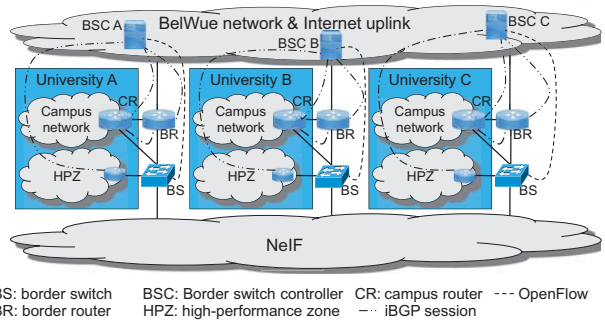

Fig. 1: Topology of SDN-NeIF with three universities connected to the legacy BelWue network and the NeIF.

B. Integration with OpenFlow and iBGP

The BS integrates the HPZ, the campus network, the BelWue network, and the NeIF. It is configured via OpenFlow by its BS controller (BSC) which is operated by BelWue [1]. The BS connects to its BSC via a BelWue transit network attached to the BR so that OpenFlow control messages are forwarded from the BS over the BR to the BelWue core. To that end, the BS has a static route towards its BSC via the BR, and the BR announces a route towards the BS to the BSC.

Within a HPZ, devices are directly connected to the BS or indirectly via an IP gateway which is responsible for a subnet within the */24* network of the HPZ. For directly connected devices in the HPZ, the BSC installs dynamic flow entries on the BS as follows. If the BS receives a packet for a device in the HPZ which does not match any existing flow entry, the BS informs the BSC. The BSC sends an encapsulated ARP request to the BS which then broadcasts the ARP request to all of its neighbors within the local HPZ. The corresponding device replies, the BS relays the response to the BSC, and the BSC installs an appropriate forwarding rule on the BS. The dynamic installation of flow entries for devices directly attached to the BS may be optimized, e.g., by leveraging information from a local DHCP server. The BSC can install flow entries for IP prefixes for which connected gateways in the HPZ are responsible. To that end, the BSC has an iBGP speaker running and may be contacted by a gateway via iBGP to learn about it and its address space.

All BRs in the BelWue network exchange routing information via iBGP and so does a BR with the same university's CR. The objective is that all campus networks can reach each other and that they are reachable from the Internet. In SDN-NeIF, the BS announces the */24* prefix of the local HPZ via iBGP towards the BR. As the BS is only an SDN-capable switch and not a router, it cannot speak BGP itself. Therefore, the corresponding BSC acts as proxy for its BS, i.e., the BSC maintains an iBGP session with the BR and announces the availability of the local */24* HPZ prefix via the BS.

In a similar way, the BSC maintains an iBGP session with the CR and informs the CR that the BS is the next hop for the */22* prefix of all HPZs. Therefore, traffic from a campus network to a remote HPZ is normally forwarded through the NeIF instead through the BelWue core due to a shorter AS path. In addition, the CR announces the prefix of the campus network to the BSC which then installs an appropriate forwarding rule for campus traffic on the BS.

The BSs of different locations are interconnected with multiple 10 Gb/s point-to-point connections which are aggregated to a single virtual link by the Link Aggregation Control Protocol (LACP) [10]. They constitute the core network of SDN-NeIF. To relay traffic destined to remote HPZs, the BSCs configure their BSs with flow rules using the */24* IP prefixes of the remote HPZs as match patterns and appropriate neighboring BSs as forwarding action. The same is done for the address spaces of other campus networks so that traffic originating in a HPZ is forwarded through the NeIF to destinations in remote campus networks.

C. Failover Mechanism

For resilience reasons, every university campus is in fact connected by two BRs with identical IP address over non-overlapping paths to the BelWue core network.

Likewise, there are two CRs with identical IP address within a campus network, and a full mesh interconnection among the two BRs and CRs. The Virtual Router Redundancy Protocol (VRRP) [11] is used between the two CRs so that they act as one virtual router. This ensures connectivity even if

[1] As the BSC can maintain iBGP sessions for only a single BS, every BS requires its own BSC.

one fails. Similar applies to the two BRs. Thus, any CR, BR, or path towards the BelWue core network may fail without compromising the uplink of a location.

As a result, we can consider the connection between the BS and its BSC as highly reliable because the BSC is located in the BelWue core network. SDN-NeIF implements out-of-band signalling since OpenFlow control traffic is not carried over the NeIF. A potential failure in the NeIF is detected by adjacent BSs which inform their BSCs about this event.

In such a case, the BSCs reconfigure the BSs to send affected traffic for remote HPZs via the corresponding BRs from where the traffic is attracted through the BelWue core network and the BR of the destination to the remote BS.

If traffic between two HPZs is affected by a non-adjacent failure in the NeIF, the BS detecting the failure reroutes the traffic. This may be problematic since the uplink of another university is used for rerouting. Therefore, we are currently working on BSC-to-BSC communication so that a BSC can notify the other BSCs about that failure. If a BSC is informed about a failure within the NeIF, it can reconfigure its BS to reroute affected traffic. Thereby, affected traffic is rerouted through the uplink of the originating network.

IV. SDN-NeIF Testbed

The SDN-NeIF testbed implements a topology as shown in Figure 1 with three university locations (Site A, B, and C) and the BelWue site. Each university site consists of a high-performance host (HPH) in the HPZ, a campus host (CH), a CR, and a BR. The BelWue site accommodates the BSCs for the three BSs, an Internet host (IH), and a BelWue core router (BCR). The implementation of the BSCs is based on the Ryu SDN framework [12] which supports all OpenFlow versions from 1.0 up to 1.5. We chose Ryu because it is implemented in Python which allows for rapid prototyping and it is relatively lightweight compared to other SDN controllers. As the large number of physical entities is expensive and difficult to manage, we model most of them as virtual machines (VMs) and implement the BSs and additional auxiliary switches (AUX) on switch partitions. In the following, we map virtualized entities to hardware, discuss the hardware platform of the testbed, and explain the virtualization approaches.

A. Mapping Virtualized Entities to Hardware

Figure 2 depicts the testbed which consists of four desktop PCs, three SDN-capable switches, and one unmanaged consumer switch. CH, CR, BR, and HPH of a university location are realized as VMs on a dedicated desktop PC and the BS is realized on a partition of an SDN-capable switch. As the BSCs and the IH do not require dedicated external Ethernet interfaces and computational power, they are realized as lightweight containers instead of VMs on an additional desktop PC. They are connected via virtual network interfaces to the host which acts as BCR and is connected to the unmanaged switch. This switch represents the BelWue core network connecting the BRs and the BSCs and giving access to the Internet.

Figure 3 illustrates the organization of the VMs of a university location on a desktop PC with a quad-port NIC. Each port of this NIC is used for one VM. As some VMs require multiple interfaces, VLAN tags are used to identify

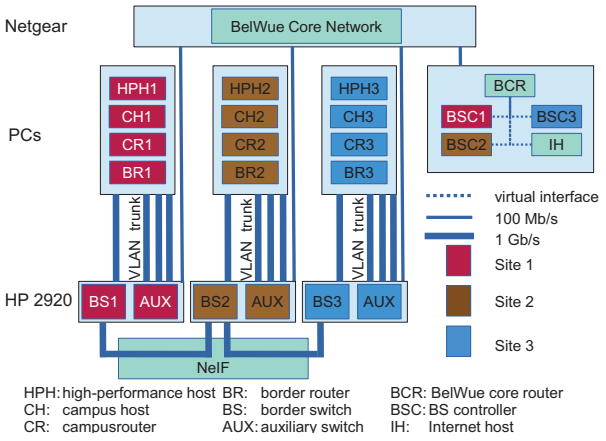

Fig. 2: Relevant nodes illustrating the operation of SDN-NeIF are implemented in the demo as VMs and switch partitions and are grouped on desktop PCs and consumer switches according to topology.

their traffic which is multiplexed over a single port. The HPH is directly connected to the BS using a 1 Gb/s link. All other ports are connected to the AUX partition of the SDN-capable switch. The SDN functionality is not enabled for the AUX partition of the switch because it is only used for multiplexing and demultiplexing VLANs, and possibly for connecting them to other VLANs or physical 100 Mb/s ports. The BS is controlled over a configured VLAN (BR/BS-Ctrl) which is exclusively used for that purpose. Therefore, the BR requires one VLAN for data traffic (BR/BS-Data) and one for control traffic (BR/BS-Ctrl) towards the BS.

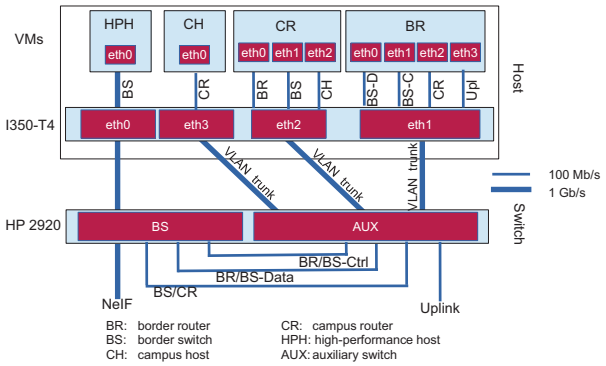

Fig. 3: The virtualized HPH, CH, CR, and BR on the host communicate over an auxiliary switch (AUX) with each other and/or with the BS on the switch partition. The BS gives access to the NeIF and the BR is connected via the AUX to the BelWue core network.

B. Experimental Platform

The computers hosting VMs are equipped with an Intel Core i7 CPU, 16 GB RAM, and an Intel i350-T4 NIC with 1 Gb/s interfaces to provide sufficient computational power and virtual network interfaces for the VMs. The PC holding the containers is equipped with an Intel Core i5 CPU, 4 GB RAM, and a 100 Mb/s NIC as it does not need that much resources. The hosts, VMs, and containers run a minimal Ubuntu as operating system with additional tools for debugging and testing. The virtual routers, namely BR, CR, and the BCR modelled by the host use the Quagga [13] routing suite with the BGP module enabled.

Like already mentioned in section Section IV-A, we use VLAN tags to distinguish multiple virtual interfaces per NIC. The igb [14] driver of the NIC can be configured in a way that automatically adds a VLAN tag for traffic from VM to host and strips the VLAN tag for traffic from host to VM at the transition between VM to host. This way the VMs do not see any VLAN tags or even have to know about them.

C. Virtualization Platform

To use hardware-accelerated virtualization on the Intel x86 platform, some extensions are needed as the architecture itself is not virtualizable. Intel VT-x [15] enables basic hardware acceleration on that platform. VT-d [16] provides an IOMMU which allows to pass-through devices, e.g., NICs from the host to the VM. VT-c [17] comprises Virtual Machine Device Queues (VMDq) [18] which enables multiple queues and Single Root I/O Virtualization (SR-IOV) [19] which is an extension to the PCI standard. With these extensions it is possible to provide multiple virtual NICs on the host, each with its own queue per physical NIC, and to individually pass them through to a VM.

These hardware features must be supported by the hypervisor running the VMs. As hypervisor we use kvm [20], which is part of the Linux kernel, in conjunction with the qemu [21] tool. The VMs are managed with the libvirt [22] framework. Regular desktop PCs can provide these features, but in contrast to servers, the features are mostly disabled by default. Some additional kernel parameters need to be set to enable all required features and libvirt has to be configured to use a special pass-through method. As a result, we run multiple VMs per host with virtual NICs having an overall I/O performance close to the one of a dedicated physical machine.[2]

D. Container Platform

Linux containers (LXC) [23] provide a method for virtualization at the operating system level that enables multiple isolated Linux systems (containers) on a host. All containers share the same Linux kernel with the host. For resource limiting, process prioritization, and access control to hardware, LXC relies on Linux kernel cgroups [24]. Isolated namespaces [25], [26] allow containers to have their own process and network space within the host system. LXC and containers save computational and resource overhead compared to a hypervisor and VMs. Running containers as unpriviledged users prevents them to damage the host system and access hardware directly.

V. Demo

The demo illustrates traffic forwarding in SDN-NeIF in failure-free scenarios and in case of link failures in the NeIF. The topology, statistics, and the flow tables of the BSs are visualized by a web app. The demo serves as validation of the architecture and a first proof-of-concept.

VI. Conclusion

We proposed SDN-NeIF as an SDN-based network architecture that integrates university HPZs into university campuses, the BelWue network, and the high-speed NeIF network. OpenFlow controllers act as iBGP proxies for integration into the BelWue network and reconfigure BSs in case of a failure in the NeIF to reroute traffic through the BelWue core network. The presented testbed consists of four desktop PCs, three SDN-capable switches, and one consumer switch, and virtualization techniques are used to model a multitude of testbed nodes. The testbed helped to develop SDN-NeIF and serves as a first proof-of-concept. SDN-NeIF is currently implemented as a prototype on the real NeIF platform.

[2]This virtualization mechanism runs with Linux Kernel 4.1 on PCs with Intel Core i7 processors. With higher kernel versions it works only on server machines like Intel Xeon e5 or higher.

References

[1] BelWü-Koordination, "BelWü – das Landeshochschulnetz," http://belwue.de.

[2] Open Networking Foundation members, "OpenFlow Switch Specification," The Open Networking Foundation.

[3] Y. Rekhter, T. Li, and S. Hares, "RFC4271: A Border Gateway Protocol 4 (BGP-4)," Jan. 2006.

[4] ESnet, "Science DMZ," https://fasterdata.es.net/science-dmz/science-dmz-architecture/.

[5] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The Science DMZ: A Network Design Pattern for Data-intensive Science," in *International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013.

[6] The Internet2 Community, "Internet2," http://www.internet2.edu/.

[7] Mark McCahill, "SDN, IDM, and Research Computing at Duke," http://meetings.internet2.edu/media/medialibrary/2015/10/19/20151007-McCahill-SDN-IDM-ResearchComputing-Duke.pdf.

[8] GlobalNOC, "SciPass," https://globalnoc.iu.edu/sdn/scipass.html.

[9] GEANT Community, "GEANT Project," http://www.geant.org/Projects/GEANT_Project_GN4-1/Pages/Home.aspx.

[10] *802.1AX: Link Aggregation*, LAN/MAN Standards Committee of the IEEE Computer Society, 2014.

[11] S. Nadas, "RFC5789: Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6," March 2010.

[12] Ryu SDN Framework Community, "Ryu," http://osrg.github.io/ryu/.

[13] Quagga team, "Quagga Routing Suite," http://www.nongnu.org/quagga/.

[14] Intel Corp., "igb Linux Base Driver for Intel Ethernet Network Connection," http://sourceforge.net/projects/e1000/files/igb_stable.

[15] ——, "Intel Virtualization Technology(VT-x)," http://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html, 2006.

[16] ——, "Intel Virtualization Technology for Directed I/O (VT-d) Architecture Specification," http://www.intel.com/content/www/us/en/intelligent-systems/intel-technology/vt-directed-io-spec.html, 2012.

[17] ——, "Intel Virtualization Technology for Connectivity (VT-c)," http://www.intel.com/content/www/us/en/network-adapters/virtualization.html, 2012.

[18] Intel LAN Access Division, "Intel VMDq Technology," Intel Whitepaper, Intel Corp., Whitepaper, 2008.

[19] PCI SIG, "Single Root I/O Virtualization and Sharing Specification 1.1," http://www.pcisig.com/specifications/iov/single_root/, 2010.

[20] A. Kivity *et al.*, "kvm: the Linux Virtual Machine Monitor," in *Linux Symposium*, 2007.

[21] QEMU team, "QEMU 2," http://wiki.qemu.org/ChangeLog/2.0, 2014.

[22] Red Hat, "libvirt: The Virtualization API," http://libvirt.org, 2012.

[23] D. Lezcano, S. Hallyn, S. Graber *et al.*, "Linux Containers," https://linuxcontainers.org/, 2008.

[24] Wikipedia, "cgroups," http://en.wikipedia.org/wiki/Cgroups.

[25] P. Emelyanov and K. Kolyshkin, "PID namespaces in the 2.6.24 kernel," http://lwn.net/Articles/259217/, 2007.

[26] J. Corbet, "Network namespaces," http://lwn.net/Articles/219794/, 2007.