INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2014, VOL. 60, NO. 4, PP. 299–307
Manuscript received September 3, 2014; revised December 2014. DOI: 10.2478/eletel-2014-0039

# SDN Controller Mechanisms for Flexible and Customized Networking

Jacek Wytrębowicz, Thorsten Ries, Khoa Truong Dinh, and Sławomir Kukliński

***Abstract*—Software-Defined Networking (SDN) is seen as the most promising networking technology today. The spread of a new technology depends on the acceptance of the engineers implementing the networks. Typically, when engineers start the conceptualization of new network devices that work with a new paradigm, and that should provide expected business values, they must identify and utilize technical enablers for the defined business use cases. This paper tries to summarize essential SDN applications and defines the technical enablers for advanced and efficient SDN networking. To this end, we identify the core technical mechanisms, expecting to provide a useful analysis for the design of new SDN networks.**

***Keywords*—Software Defined Networking, Network Controller, SDN Management**

## I. Introduction

SINCE its appearance, the SDN paradigm has taken a lot of interest by the community of network researchers and designers. The main idea of SDN is to decouple the control plane from the data plane and to centralize control and management functions of a network. As a result the control plane can be implemented purely in software, and the data plane can be realized through relatively simple and inexpensive hardware. Such a network architecture promises better flexibility in network services provisioning, better accessibility to low-level network functions towards new innovative networks. One of the main drivers for SDN deployments is the significant spread of cloud computing and the resulting need for cooperation between new applications and network control functions.

At present the most popular SDN approach is based on OpenFlow [1]. It is defined and promoted by the Open Networking Foundation (ONF), in which major manufacturers and Internet related companies are involved.

ONF specifies the internals of the OpenFlow switch as well as the interactions between the network controller and the switches. The switches operate on data flows, not on individual IP packets. The SDN control software works on a central server and defines predefined matching rules for the OpenFlow switches in a dynamic way. This approach allows direct access and manipulation of the forwarding plane on both physical and virtual (hypervisor-based) switches. Most commercial SDN switches available today (e.g. from BigSwitch, Hewlett-Packard, Brocade, IBM, NEC, Pronto, Juniper, Pica8) support OpenFlow in addition to traditional operation and protocols. The controllers (e.g. NOX, POX, Beacon) are based on generic high performance server platforms and typically are offered as Open Source software. The principles of their creation and evolution were: to have a framework for network application development, and to provide an experimentation environment that will be verified by community working with the open source.

There are other technologies proposed or considered as applicable to SDN: ForCES [2], Path Computation Element [3], BGP-TE [4], Application Layer Traffic Optimization (ALTO) [5]. These technologies are rather complementary but in some cases they are also mutually exclusive. For example, the goal of the Software Driven Network Protocol (SDNP) IETF working group is to enable existing control planes to become more adaptable to application requirements. This leads to rapid and reliable configuration changes between applications and network control planes. In the IETF draft, "Software Driven Networks: Use Cases and Framework" [6], the authors claim that SDNP is also useful to OpenFlow type networks, since SDNP provides the interface between applications and control planes implemented in OpenFlow controllers. Another IETF draft "Use Cases for ALTO with Software Defined Networks" [7] proposes two (vertical and horizontal) architectures for integration with SDN infrastructure. The Vertical Architecture assumes placement of ALTO server over SDN server, the Horizontal Architecture assumes integration of those servers, in order to simplify their interactions. The Vertical Architecture allows better division, management, flexibility, privacy control and long-term evolution of the network. ALTO is the application layer traffic optimization, which is defined by IETF ALTO Working Group. The IETF drafts mentioned above are working documents, which present ideas, not matured solutions. We cite them to point these ideas and related use cases for SDN. A comprehensive review on SDN definition, and the proposed technologies for SDN, is given in [8].

Variances in understanding the SDN concept result from different approaches to the complexity of forwarding network elements. One approach is to build a layer over existing, and typically complex network devices in order to offer centralized control of network services or to enable interactions between the applications and the network, e.g. using the Interface to the Routing System Framework (I2RS) [9]. One position is that SDN should be built over traditional switches [10] and the authors outline how SDN networks can use I2RS. An opposite approach is to use simple network switches, which are

This work has been conducted as part of the Cognitive Software Defined Networks (CoSDN) project, funded by FNR Luxembourg and NCBiR Poland.

J. Wytrębowicz, Khoa Truong Dinh, and S. Kukliński are with Institute of Computer Science, Warsaw University of Technology, Plac Politechniki 1, 00-661 Warsaw, Poland (e-mails: J.Wytrebowicz@ii.pw.edu.pl; t.dinhkhoa@gmail.com; kuklinski@tele.pw.edu.pl).

Thorsten Ries is with Post Technologies, 2, rue Emile Bian, L-2999 Luxembourg, Luxembourg (e-mail: thorsten.ries@post.lu).

capable of effectively forwarding incoming packets or flows, and to control them through cooperation with a central server, which contains all network control logic. Other approaches vary between these two extremes, e.g. by adding fast reroute control logic to the new simple switches, or handling "short-lived" flows in switches to mitigate flow setup delay and controller overhead. In the book [11], the authors explore the emerging concepts and protocols for SDN. They show their variety and assess them from a pragmatic perspective, opting for building SDN over traditional switches.

Recently just proposed technology, named Protocol Oblivious Forwarding (POF) [12], is similar to OpenFlow. POF offers more flexibility for switch programming than OpenFlow, and in the future can compete with it in SDN deployments. OpenFlow and POF specify the internals of the switch and its interface to the network controller. Other above-mentioned technologies define interfaces for network applications and interactions between networks. None of these technologies deals with control mechanisms of the network, which are supposed to work on the central network controller.

A detailed presentation and analysis of the SDN technologies mentioned above could be a subject of a stand-alone paper, but they are out of the scope of this article. In this work, we try to highlight essential network control mechanisms, perceiving them as enablers of new SDN functionalities. Before we outline the known SDN use cases, and we highlight relating functionalities.

## II. SDN Use Cases

SDN comes with the promise of enabling new network services only through the way in which the network is operated. From the technical point of view two main properties should be emphasized: firstly, the already mentioned flow forwarding ability, which can be managed for each individual flow; and secondly the centralized approach to network control, which means that in a single place all information about all network nodes, links, and traffic is available. Consequently the basic networking operations can and should be optimized in a centralized manner, what is difficult to achieve in today networks, because of multiple limitations with distributed control. Most importantly, SDN enables the creation of highly customized networks, which are tailored to meet specific needs.

Network providers and devices vendors expecting to benefit from SDN published many SDN use cases. The proposed ideas are mostly about creating business profits when applying SDN to their networks, rather than about technical issues. These ideas are typically discussed in the context of the type of SDN owner, i.e.: an enterprise, a content provider, a cloud services provider and a network provider. It happens that the same technical idea is presented as several use cases, but differs in the ownership of the network and its business perspective.

Common SDN use cases point to where this new network concept can be beneficial. Analyzing these, we can distinguish five domains of evident benefits: internal networks for big data centers, transmission between big data centers, access provisioning for data centers, network administration, and network security enforcement.

While building big multi-tenant data centers, SDN facilitates network orchestration with respect to workload changes, allowing a higher level of personalization services, traffic steering and service insertion. Inside these data centers, the load should be balanced for both the servers and their communication paths. The administrator should be able to quickly locate a network failure and to protect other traffic in such situations. Furthermore, for the ease of network management, data centers typically contain virtual switches and firewalls. Therefore, it may be desirable to have virtual patch panels for manipulation of virtual links, which can be managed remotely in a centralized manner, without the need of manual cross-swiching of physical links.

Creating an infrastructure for cloud computing, administrators have to create and to connect virtual servers through secured access, requiring both physical and virtual firewalls. Like in data centers, contents of huge data repositories need to be balanced and moved between remote localizations. Combining SDN and virtualization, it is possible to provide network infrastructure as a service, which can be beneficial for research, experimentation or test purposes and consequently to provide full network virtualization that allows customers to connect their own Network Operating System to the logical switches that are running on top of the physical infrastructure. In fact, SDN enables different network abstraction levels to the customers. The IETF draft "Software Defined Networks Use Case for Virtual Connection and Network on Demand" [13] points to another important idea: to build application services over heterogeneous infrastructures of physical and virtual networks. The SDN approach supports the provision of the two services (i.e. Virtual Connectivity and Network on Demand) in an automated way without delays related to human interactions with network devices.

Providing access to data center services is desirable to connect with a source of requested content or requested service, regardless of its IP address. An SDN controller (SC) can easily redirect incoming content/service requests on the fly. Also the controller can provide rendezvous services [14], which benefit from central information about their location to effectively deliver services such as Content Delivery Networks (CDN), p2p systems, and data center applications. In this context, the "Content Distribution Network Interconnection (CDNI) Problem" is introduced in RFC 6707 [15]. CDNI defines, among others, the Request Routing Interface between Upstream CDN and Downstream SDN. An integration of OpenFlow into CDNI is discussed in [16].

Due to the centralized control of network resources, access can be offered with individual QoS parameters. SDN has the ability to guarantee end-to-end bandwidth requirements between different data centers [17]. The same counts for an enterprise deploying a private WAN by using SDN. When provisioning intra-data center interconnectivity, SDN can provide dynamic traffic steering with QoS guarantees.

Another common need is calendaring of huge data transfers between data centers, like remote backups, multimedia content distributions or virtual hosts displacement (from one to other data center). Consequently, demand placement, bandwidth cal-

endaring, and bandwidth probing are also expected functions of SDNs.

Network administration is still a complex, resource intensive and error prone work. Hence, network operators are looking for more reliable solutions at reduced administration cost, while users interests are contrary: they expect fast reconfiguration of network resources on their demand/needs. For example, a high mobility of users and frequent changes of computation services are observed in campus networks.

SDN promises simpler administration with more automatic functionalities. First of all, the controller provides this ability to automatically redirect and distribute traffic, based on network owner policies. Next based on extensive network statistics, SDN enables efficient energy functions. For instance, exploiting information of port and switch use, a university could dynamically turn down segments of the network across the campus during the night or in the period when students leave for vacation or faculty for holidays. Moreover the centralized control enables more efficient flow distribution, leading to higher customer satisfaction and better utilization of network resources.

Traditional network security concepts typically lack the global view of network activities. This view is very beneficial for effective attack or misbehavior detection and the implementation of proper counter measurements. SDN can fill this gap and enable a comprehensive security environment with individual, fine-grained policy definition and security measures, the implementation of effective countermeasures plus pro-active fault and attack prevention schemes. For instance, firewall and intrusion detection systems (IDS) can be easily placed at strategic positions within the network, i.e. on ingress switches, even their location being dynamically adapted in case of security incidents or network re-configurations. The detailed statistics of the whole network support also forensics and attack mitigation actions.

However, it has to be noted that SDN itself creates additional attack vectors, which are non-existent in traditional networks. For instance, the data exchange between switches and the centralized controller could potentially cause a denial of service attack simply by a huge number of distinct and previously unknown flows.

Table I summarizes the key requirements to enable the described use cases. Expected advantages of SDN, that are pointed in the analyzed use cases, are:

- faster provisioning of network access and services,
- possibility of cooperation between cloud applications and network control applications,
- improved security,
- lower investment and operational costs for network owners,
- better load balancing,
- possibility of automatic energy savings control.

TABLE I
SUMMARY OF SDN USE CASES DOMAIN AND RELATED NEEDS

| **Domains of SDN use cases** | **Key needs from the SDN controller** |
|---|---|
| Internal networks for data centers | Traffic steering and service insertion, e.g. via virtual switches & Load balancing |
| Transmission between data centers | Load balancing & Bandwidth probing and calendaring |
| Access provisioning | Virtual connections and networks on demand & Support for content delivery and distribution & Dynamic QoS reservations |
| Network administration | Traffic distribution according to network owner policies & Energy saving |
| Security | Global view of network activities & Support for attack prevention schemes |

## III. SDN FROM AN IMPLEMENTATION PERSPECTIVE

The analysis of SDN use cases and applications leads to the conclusion that the benefits from SDN are based on inexpensive flow switching that is combined together with flexible, centralized control and management of simplified switches. The OpenFlow technology, which supports well these concepts, is a good base for SDN projects and deployments. An OpenFlow cloud of switches with its controller belongs to a network owner. The owner can have many such networks, which are cross connected to form a bigger infrastructure. On the top of the OpenFlow controllers, other technologies like ALTO could be developed.

Even though the SDN principle is to centralize control and management functionality, the switches should embed a simple control element to process operations that cannot be performed remotely or that should be processed locally to achieve reasonable response times. The switch should have a local intelligence that support auto-configuration or may offer Plug&Play functionality. The local intelligence should process neighbor discovery and keep-alive/failure detection autonomously (without stimulations of a network controller). Moreover, if there are signal amplifiers or retransmitters on connection lines, a simple Hello protocol may be required for this purpose. Such a protocol is also needed, if wireless links are in use. To achieve carrier-grade fast (sub 50 ms) recovery, this mechanism needs to be implemented locally on the switch too. There are also other functionalities that would benefit from local intelligence, e.g. data aggregation for remote monitoring or data analysis for distributed intrusion detection. It needs to be noted that all possible local extensions need to be implemented with low complexity. Whereas, high complexity tasks should be performed on the central controller, they are: network topology tracking and mapping, routing, traffic engineering and computing forwarding tables for the switches, calculating parameters for achievable QoS, admission control, administration and management of the network, or interoperation with legacy protocols and applications (Business Support Systems and Operation Support Systems).

To this end, the SC has to maintain information about the controlled network, to offer services for external applications, and to perform a set of tasks in order to operate the network efficiently. Therefore, SC databases store information gathered from network devices and information assembled from external management requests. We distinguish between four principal databases and present their current implementation status in current distributions of popular open-source SCs in

TABLE II
PRINCIPAL DATABASES AND THEIR PRESENCE IN SELECTED SCS

| Principal DB | NOX | POX | Beacon | F & H[a] | Trema |
|---|---|---|---|---|---|
| Topology | X | X | X | X | X |
| Usage History | - | - | X | X | X |
| ACLs | X | X | - | X | - |
| Usage Plan | - | X | X | X | - |

[a] Floodlight and Hydrogen

TABLE III
SC PRINCIPAL TASKS AND THEIR SUPPORT IN SELECTED SCS

| Proposed SC tasks | Hydrogen | Trema | Others[a] |
|---|---|---|---|
| Network discovery | X | X | X |
| Statistics collection | X | X | X |
| Path evaluation for flows | X | - | - |
| Access control | X | - | X |
| QoS path attribution | X | - | - |
| Fault detection & Rerouting | X | - | X |
| Energy savings control | - | - | - |
| Network sanity checking | X | - | - |

[a] NOX, POX, Beacon, Floodlight

Table II. For the comparison, we selected: NOX [18], POX [19], Beacon [20], Floodlight [21], Trema [22] controllers and the Hydrogen platform proposed by the Opendaylight project [23].

The Topology database contains a description of all network links, both active and inactive, and characteristics of all external and internal interfaces. The Usage History database collects data of link and queue usage of every switch port. The ACL database contains access control lists for every ingress network interface. Finally, the Usage Plan registers the planned link usage based on queue priorities.

In addition to maintaining the mentioned databases, SC has to calculate and store network control data, i.e.: unicast routing data, multicast routing data, QoS parameters for every link and every offered path. In small networks, the set of offered paths may be equal to the set of all possible paths, while in large networks, the offered paths may be selected from the set of all possible paths in a way of satisfying optimal routing needs.

Further services like: availability checking for a flow of demanded QoS parameters, reporting on flows' activities, flow calendaring with QoS parameters, and provisioning of virtual network elements are expected to be provided by the SC. Therefore, the SC has to perform at least the tasks listed in Table III to effectively control dependent switches. As the table shows, most SCs show a similar set of supported tasks. However it should be noticed, that there are functional differences between their implementations on different SCs.

The network operator needs to define configuration policies for every SC task and for every service offered by the controller. Such policies define constrains, warnings and alarms related to the business prerequisites of the owner. New applications supporting management of such policies (i.e. administration of SDN networks) are needed.

Though some of the mentioned services and tasks are offered and performed in traditional IP networks too, the SDN paradigm has the potential to enable better functionality and effectiveness. From the technical point of view we can distinguish the mechanisms that are crucial to enable SC services, and tasks which are essential to vivify the SDN use cases presented above. We discuss these mechanisms in the subsequent chapter.

## IV. CORE SDN CONTROLLER MECHANISMS

In the previous section we described use cases, which are indispensable to today's network operations. Based on these, we derive the core SC mechanisms, which form the base for autonomous network operations. These can be used by dedicated network applications to offer further advanced functionalities for the network owner, for providers of network services and for users. Technically it is possible to deploy network applications (or their elements) directly on the SC. However, for security and reliability reasons, the network owner will limit such deployment, allowing only for applications supporting network management and provisioning of network services.

Even though the expected SDN advantages are commonly repeated, and SDN use cases are known, there is still a lack of understanding of the core operations of a SC: which mechanisms should be implemented in the SC? Which mechanisms should be implemented on application level of the controller or even outside the controller, i.e. on the switches or on specialized application controllers? For our review, we take only SDN related functionality into consideration, which are independent from the implementation platform, i.e. the controller software. All issues related to such a platform, like software updates, internal data representation and communication security, are intentionally omitted.

Routing is neither single nor simple task, and it needs a comment here. Routing means collecting and processing data, to find a path to a given destination (unicast) or to a given source (multicast). In the case of routing via internal links of an SDN cloud, the SC can relatively easy do this job, as it has always all required information and even one thread or process can do the calculations. Hence, internal routing can be considered as SC duty. However, in the case of routing via border interfaces of an SDN cloud, the job is much more complex. Therefore, support of existing routing protocols, like OSPF, IS-IS or BGP, is required on every border interface. A network application can then be hosted on a separate server to provide external routing for an SDN cloud. Such an approach has been successfully deployed by Google in their B4 SDN network, which is a private WAN connecting data centers across the planet [24].

The subsequent description is an attempt to define essential core controller functionalities and their placement/relations to each other:

1) Access Control for Flows (AC4F),
2) Multipath Unicast and Anycast Routing for QoS and Load Balancing Control (MPR),
3) Multicast Routing for QoS Control (MCR),

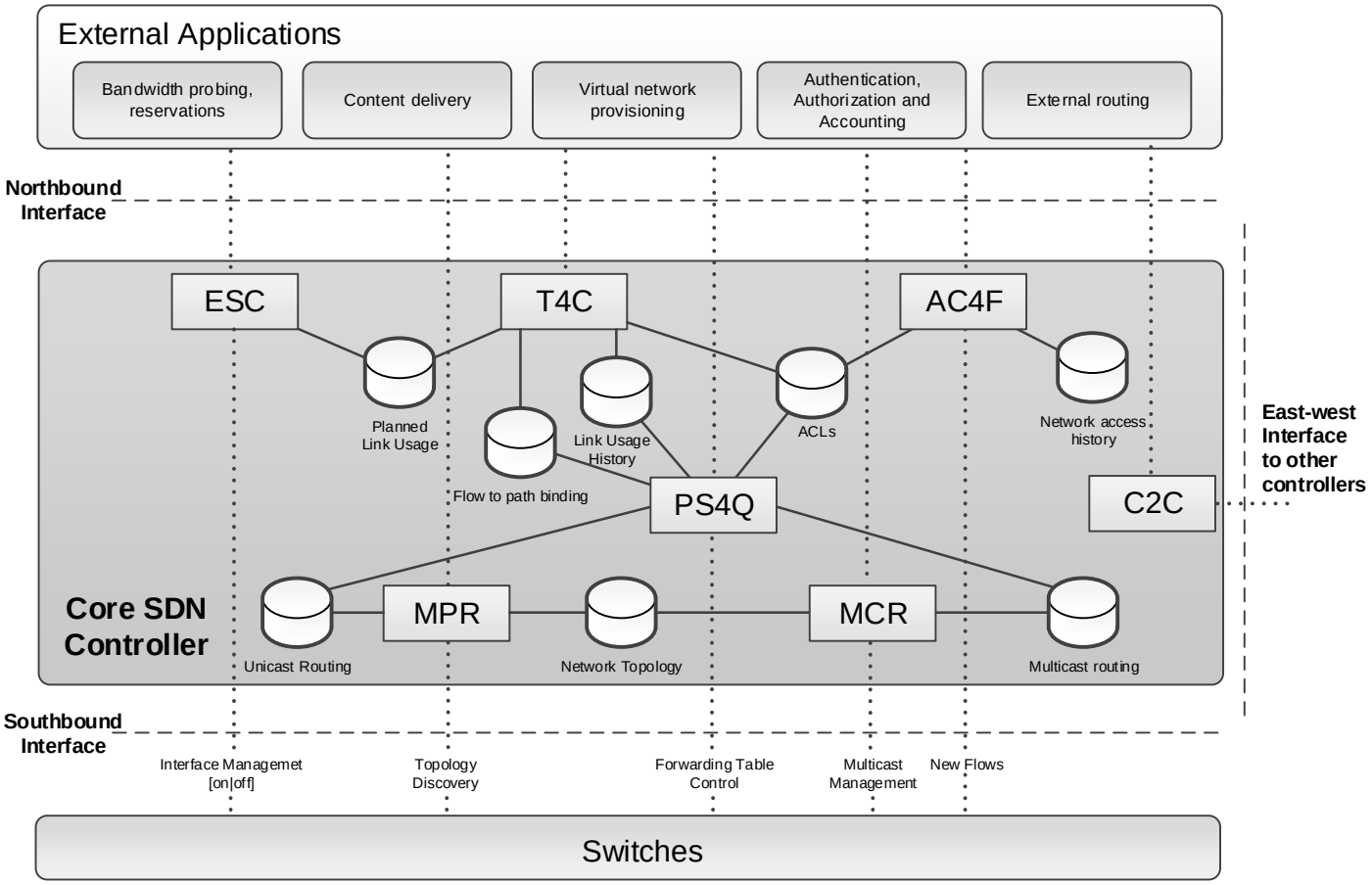


Fig. 1. Interactions between SC mechanisms and the controlled network

4) Path Selection for End-to-End QoS Flows (PS4Q),
5) Transactions for Flow calendaring (T4C),
6) Energy Savings Control (ESC),
7) Cooperation with other SDN controllers (C2C).

For each mechanism we provide its aim and related functions. To this end, we have analyzed the above-mentioned controller platforms: NOX, POX, Beacon, Floodlight, Trema and Hydrogen, and we present their implementation status in the individual mechanism's descriptions. Fig. 1 depicts interactions between SC mechanisms and the controlled network; it points also usage of the main databases of the controller.

### *A. Access Control for Flows*

Especially in critical infrastructures, there is a need of dynamically controlling access rights of appearing flows and to direct them according to policies but also according to observed features and flow behavior. Access Control for Flows mechanism (AC4F) cooperates with SDN switches, AAA servers, firewalls, and with the network administrator via configuration files. AC4F is able to evaluate access requests and to recognize users of regular flows, thus to apply predefined rules for them, e.g. to direct a flow via a path of requested bandwidth or via pre-selected middleboxes (like firewalls, intrusion detection systems, WAN optimizers). AC4F queries the path selection mechanism (PS4Q, see below), to find a path for every non-rejected new flow. AC4F directly operates on two SC databases. It uses and can update the access control lists (ACLs) for every ingress network interface. It writes to the history of link usage and queue usage of every switch port.

In traditional IP networks, switches have access control lists, and routers have packet filters. The central AC4F mechanism can provide better functionality due to on-line cooperation with AAA servers, firewalls, IDSs, WAN optimizers, which can work as local or remote applications. AC4F can deliver aggregated and statistical data about flows in the network. It can also process admission rules related to events observed on remote points of the network (e.g. load incoming on all border interfaces and originated from, or designated to, a given network can have a defined limit). AC4F can support also automatic (i.e. fast) provisioning of network services, according to predefined management rules.

AC4F provides essential services for the SDN use cases that are related to access provisioning.

The use cases should be implemented as control applications. However, AC4F has to support the binding between new flows and their preprogrammed (on demand) virtual connections and networks. Similarly, AC4F has to support services for content delivery and distribution, by recognizing and binding appearing connection request with preprogrammed destination addresses.

As a basic mechanism, admission control is already provided in most of the existing controllers. The Floodlight controller has a firewall module, which is used to apply ACL rules to allow/deny traffic based on a specified match. In the NOX and POX controllers, the access control mechanism is also provided to check whether a new host/user or flow is added to the network. However, we could not find this mechanism in Beacon and Trema controllers, which has only the ability to set priorities for new flows by setting VLAN tag, and to define different paths for the flows. Hydrogen controller provides user authorization mechanism, and gives access for user flows. Moreover, BGP protocol has also been implemented in this controller to provide an enhanced mechanism for access control.

### *B. Multipath Unicast and Anycast Routing for QoS and Load Balancing Control*

Multipath Unicast and Anycast Routing for QoS and Load Balancing Control (MPR) is supposed to build and update a routing database for unicast and anycast flows. The multipath approach gives failover paths and supports load balancing. For every pair of border interfaces, possible paths are analyzed and relevant paths are selected to satisfy QoS and load policies. MPR is activated by every topology change in the network, i.e. activation or deactivation of an interface or a switch (due to a failure or to an energy saving action). Moreover, it checks periodically the network load, and if need be, it recalculates paths' parameters that are stored in the routing database.

MPR maintains the network topology database, thus it performs network topology discovery. Depending on the implementation, topology discovery can be considered as a separate SC submodule. In that case, such submodule should fire the processing of MPR every time a topology change is discovered.

MPR calculates paths and estimates their QoS parameters. Given by the global topology view, the controller has the ability to determine the best path for incoming flows. The controller is able to consider relative priorities of all paths and allocate network resources efficiently. This mechanism ensures that switches don't compete for the best path after a link failure. In traditional IP networks, routers build appropriate data structures using routing protocols, and layer 2 switches operate Spanning Tree Protocols. The multipath processing is limited. Most deployments do not allow for load balancing, and secondary paths are calculated after a failure.

While MPR can do routing calculations itself for internal flows, flows terminating outside the network require information distributed by legacy routing protocols. For that reason, cooperation with related control applications (which maintain the legacy routing protocols on boarding interfaces of the network) is required. Moreover, the applications have to distribute information on destinations reachability, for which they require MPR to provide relevant information.

All analyzed controllers perform topology discovery. Floodlight and POX have a simple load balancer for TCP and UDP flows. Incoming flows for a given service can be randomly redirected to one of the known servers – the destination IP address is substituted appropriately. Trema doesn't provide full load balancing functionality. Provided load balancing can be used only in the simulation tests, and cannot work in the real network environment. Hydrogen platform has the Affinity Metadata Service (still under development) to support load balancing. This service operates on variables: Affinity Identifier, Group, Link, and Attribute. The Affinity Attribute describes the network properties that are assigned to Affinity Link to meet workload performance. The service has capability to isolate traffic to be forwarded using separated physical links or paths not shared by other traffic. We haven't seen this mechanism implemented in other controllers.

### *C. Multicast Routing for QoS Control*

Especially multimedia content distribution requires multicast data transmission based on given QoS requirements. Hence, the aim of the Multicast Routing for QoS Control (MCR) mechanism is to build and update the routing database for multicast streams. The database is updated for every multicast stream and for every change in the set of the stream receivers. Based on this information, MCR calculates the distribution paths individually per stream trying to satisfy policy-defined QoS parameters. The multicast routing policies are supposed to be altered by the network administrator and by authorized network applications.

For SDN-internal multicast streams, MCR can do routing calculations itself. To transport the streams via borders of the network, information distributed by legacy multicast routing protocols is required. Therefore, a cooperation with related control applications, that maintain the legacy multicast routing protocols on boarding interfaces of the network, is required. Because of the need to distribute information on sources reachability, as well as on possible transits for external sources, MCR has to offer the required information to the applications.

Most OpenFlow switches already support multicast mechanisms by pushing packets/flows out of multiple ports. However the analyzed SDN controllers do not have the capability to define individual paths for a multicast stream satisfying QoS requirements. The only exception is Hydrogen, which allows binding QoS parameters to multicast streams using Affinity Attributes.

### *D. Path Selection for End-to-End QoS Flows*

Based on given policies, the aim of the path selection mechanism (PS4Q) is to satisfy a requested set of QoS parameters and to select end-to-end paths. The selection is performed using the data calculated by either the multipath routing mechanism or the multicast routing mechanism. PS4Q is demand triggered by the following mechanisms: Access Control for New Flows (AC4F), Transactions for Flow calendaring (T4C), and Energy Savings Control (ESC).

Moreover it supports dynamic flow rerouting in case of link failures or in case of network application request. PS4Q keeps track of the bindings between flows and paths and cooperates with AC4F, T4C, and with SDN switches. It updates their forwarding tables and also processes failure alerts from them.

Processing data for switch forwarding tables is the indispensable function for every IP router and Ethernet switch. The centralized PS4Q mechanism simplifies the cooperation with other mechanisms (T4C, ESC), and cooperation with management applications. The clue for enabling the SDN use cases, as discussed earlier, is the interface for network control and management applications. The applications can support: traffic steering and service insertion in big multi-tenant data centers, virtual patch panels for virtual switches and firewalls, Content Delivery Network services, load balance for both the servers and their communication paths, and so on.

As we mentioned above, the Beacon controller has the ability to set priorities to incoming flows. The POX controller can additionally add a VLAN tag to the packet header to set the priority, thus enabling QoS for end-to-end paths. The PS4Q mechanism is not implemented in NOX, Trema, and Floodlight controllers.

The Affinity Metadata Service (Hydrogen controller) gives the ability to select/isolate a path for a flow with QoS parameters. Moreover, the Defense4All Service proposed for Hydrogen is also able to redirect the traffic, what can be applied in case of anomaly detection or attack. This service is responsible for configuring the network, in order to prevent system against suspicious traffic, and is also responsible for restoring the network to original configuration.

### *E. Transactions for Flow Calendaring*

The aim of flow calendaring (T4C) is to enable planning of big data transfers and to enable throughput reservation for these. The supposed throughput availability can be checked for the required time period. Calendaring adjustments should be performed by the network administrator and authorized network applications only. In traditional IP networks, administrators process such services manually.

T4C has to process information from two SC databases: the ACLs for every ingress network interface, and the network topology and characteristics of all external and internal interfaces. T4C can also consider data from the link usage history database, and signalize possible future conflicts of links usage. Next it alters the planned link usage database.

There is no bandwidth reservation mechanism in the analyzed controller distributions.

### *F. Energy Savings Control*

Not only in wireless ad-hoc or sensor networks, efficient energy saving plays an important role for a reliable and durable operation. Saving energy is basically an important aspect for all network deployments, in order to reduce operational costs. Typically in wired networks, the network administrator defines energy savings policies and executes them by changing configuration of network devices. The energy savings control mechanism (ESC) may help the administrator by controlling execution of a defined policy in an automatic way.

ESC needs to be embedded seamlessly into the general picture. By cooperating with the PS4Q mechanism, it changes paths for on-going flows by sending requests to PS4Q, preserving calendared flows that were scheduled by T4C. Furthermore ESC needs the ability to switch on/off interfaces or devices through direct communication with the network switches.

Though the OpenFlow protocol provides the ability manage ports administratively (i.e., activate/deactivate), there are no means to control the power of physical interfaces. The way of doing this based on an energy saving policy is not available in the above-mentioned controllers. Such a realization would require extensions to the existing protocols in order to monitor and control energy consumption.

Certainly, energy-saving does not stop at this point: topology and routing optimisation have an impact on energy consumption as well. For instance, parts of a redundant topology could be turned of, as long as it is not required and SDN may provide the required capabilities for such an intelligent traffic engineering. In [25], Heller et al. argue that even critical network components may be turned of or put in standby if they are idle while still maintaining robustness and performance of the network.

Today's deployments widely use custom-made software to provide remote power control on network devices and interfaces. Therefore typically SNMP (Simple Network Management Protocol) or out of band communication channels (e.g. via telephone modems to power supplies) is utilized. As the available hardware and software for remote power control can vary, communication with them would be better managed by a control application and not directly by an SC module. ESC having full access to the controller data bases can efficiently compute desired power controls and switching communication paths in advance.

### *G. Cooperation with Other SC*

Until now, SDN has been mainly deployed as single network domains, managed by one controller per network. Though there are failover implementations based on multiple controllers, it is basically only the master controller managing one single domain. Such an approach may be acceptable for small private networks, but big commercial networks, require higher scalability and reliability. Therefore multi-controller-multi-domain architectures are the logical consequence. A well-engineered multi-controller deployment has also a strong impact on the security of the network. The SDN controller in a single-controller network is not only a single point of failure but also the most interesting target for attackers. A faulty or malicious controller can easily compromise the entire network. Therefore, a multi-controller architecture, even only for single domain networks, is essential to provide a higher level of security of the SDN network.

The multi-controller requirements for reliability and for scalability show many differences. In fact, high reliability can be achieved by well-known solutions that are applied for fault-tolerant computer system in data centers, i.e. redundant hardware, software, and mechanisms solving consensus in a network of unreliable processors, e.g. Raft [26]. These solutions can be applied for SDN controllers and servers hosting control applications. However, it is desired that redundant processes set out stable states for mirroring and synchronization.

In the case of providing scalability, the processes should also set out stable states for full or partial synchronization of their data structures and states. Even thought the aim of synchronization is different, the way of providing it can be the same. For that reason we propose a generic mechanism for cooperation with other SCs.

For the above-mentioned reasons, SC should contain a mechanism for cooperation with other controllers, so called Controller-to-Controller (C2C) communication. Two kinds of cooperation ought to be considered: between adjacent network controllers, and between controllers that provide redundancy for a given domain.

In single-domain networks, elastic distributed controller architectures can be used to dynamically adapt to traffic conditions by distributing the load across several controllers [27]. However, as mentioned above, large-scale multi-domain networks demand advanced intra-domain network management. In the case of SDN, this means that the controllers need to communicate with each other to improve the overall network security and to allow end-to-end management [28]. In fact, all proposed mechanisms would benefit from C2C communication and cooperation e.g. by extending path selection mechanisms over multiple network domains. As these operations require a high degree of information exchange, such that the existing SC capabilities need to be extended in a way that internal databases should be ready (entirely or partially) for synchronization between controllers and enable storing and processing data even for switches from other domains. Here, known solutions for synchronization of distributed databases can be applied. For these operations, dedicated policies need to be in place or may be negotiated as part of the cooperation. This is still the matter of future research.

To our knowledge, there is no mechanism for cooperation with SCs in the distributions of the analyzed SDN controllers.

## V. Conclusions

Most published SDN use cases highlight the multifaceted areas of applications that exist for these kind of networks. Many vendors already provide SDN switches and network controllers and there are ambitious efforts to bring the required technologies as RFC to IETF. However, SDN is still in the early stage of its evolution, and at present a lot of concurrent works are carried out to reach the maturity of existing SDN technologies.

Indeed, Software Defined Networking shows huge potential in development of innovative network services and applications. Flow forwarding together with the central control of a cloud of simplified switches, promises lower CAPEX and OPEX for network owners. For that reason we can expect many developments on SDN controllers to support the growing number of SDN deployments. Yet, several distinct controllers have been developed, most of them with different features, aligned to different needs. We hope that our work will help in the definition of a common base of SDN controllers.

The paper summarizes the possible applications of SDN, and exploits the advantages of this new network paradigm. We have analyzed common needs and looked on their requirements from a technical implementation point of view, trying to define the core mechanisms of an SDN controller and to define the functional structure of the controller.

The proposed set of controller functionalities is essential for both basic and advanced network operations for flexible and customized networking. Depending on individual needs, these mechanisms could be designed to be included on demand in the network operation system. For that reason, the SC platform should support loadable modules with standardized APIs. We believe that the described mechanisms further enlarge the great potential of Software Defined Networking and that they should be included into all SCs.

## References

[1] *OpenFlow Switch Specification*, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf, Open Network Foundation, October 2013, Version 1.4.0 (Wire Protocol 0x05).

[2] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification," Internet Requests for Comments, RFC Editor, RFC 5810, March 2010. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5810.txt

[3] A. Farrel, J. P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-Based Architecture," Internet Requests for Comments, RFC Editor, RFC 4655, March 2010. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4655.txt

[4] H. Gredler and J. Medved, "Advertising Traffic Engineering Information in BGP," Working Draft, IETF Secretariat, Internet-Draft draft-gredler-bgp-te-01.txt, July 2011.

[5] J. Seedorf and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement," Internet Requests for Comments, RFC Editor, RFC 5693, October 2009. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5693.txt

[6] D. Stiliadis, F. Balus, W. Henderickx, N. Bitar, and M. Pisica, "Software Driven Networks: Use Cases and Framework," Working Draft, IETF Secretariat, Internet-Draft draft-stiliadis-sdnp-framework-use-cases-01.txt, October 2011.

[7] H. Xie, T. Tsou, D. Lopez, and H. Yin, "Use Cases for ALTO with Software Defined Networks," Working Draft, IETF Secretariat, Internet-Draft draft-xie-alto-sdn-extension-use-cases-01.txt, June 2012.

[8] S. Sezer et al., "Are we ready for SDN? Implementation challenges for Software-Defined Networks," *Commun. Mag.*, vol. 51, no. 7, pp. 36–43, July 2013.

[9] A. Atlas, T. Nadeau, and D. Ward, "Interface to the Routing System Framework," Working Draft, IETF Secretariat, Internet-Draft draft-ward-i2rs-framework-00.txt, February 2013.

[10] S. Hares and M. Chen, "Use Cases for Virtual Connections on Demand (VCoD) and Virtual Network on Demand (VNoD) using Interface to Routing System," Working Draft, IETF Secretariat, Internet-Draft draft-hares-use-case-vn-vc-01.txt, February 2014.

[11] T. D. Nadeau and K. Gray, *SDN: Software Defined Networks*. O'Reilly Media, Inc., 2013.

[12] H. Song, "Protocol oblivious forwarding: Unleash the power of SDN through a future proof forwarding plan," in *Proc. 2nd ACM SIGCOMM workshop HotSDN*, 2013, pp. 127–132.

[13] M. Chen, M. McBride, and L. Ou, "Software Defined Networks Use Case for Virtual Connection and Network on Demand," Working Draft, IETF Secretariat, Internet-Draft draft-mm-sdn-vc-vn-on-demand-use-case-02.txt, January 2013.

[14] K. Gurbani, M. Scharf, T. V. Lakshman, and V. Hilt, "Abstracting network state in Software Defined Networks (SDN) for rendezvous services," in *IEEE Int. Conf. Communications*, 2012, pp. 6627–6632.

[15] B. Niven-Jenkins, F. L. Faucheur, and N. Bitar, "Content distribution network interconnection (cdni) problem statement," Internet Requests for Comments, RFC Editor, RFC 6707, September 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6707.txt

[16] M.-K. Shin, S. Lee, D. Chang, and T. Kwon, "CDNI Request Routing with SDN," Working Draft, IETF Secretariat, Internet-Draft draft-shin-cdni-request-routing-sdn-01.txt, February 2013.

[17] G. Bernstein and Y. Lee, "Use cases for High bandwidth Query and Control of Core Networks," Working Draft, IETF Secretariat, Internet-Draft draft-bernstein-alto-large-bandwidth-cases-02.txt, July 2012.
[18] "Nox project documentation," http://www.noxrepo.org/_/nox-doxygen/index.html, last accessed: February 19, 2014.
[19] "Pox project documentation," https://openflow.stanford.edu/display/ONL/POX+Wiki, last accessed: February 19, 2014.
[20] "Beacon project documentation," https://openflow.stanford.edu/static/beacon/releases/1.0.4/apidocs/, last accessed: February 19, 2014.
[21] "Floodlight project documentation," http://docs.projectfloodlight.org/display/floodlightcontroller/Floodlight+Documentation, last accessed: February 19, 2014.
[22] "Trema project documentation," http://trema.github.io/trema/, last accessed: February 19, 2014.
[23] "Opendaylight project documentation," http://www.opendaylight.org/news/2013/09/converge-network-digest-opendaylight-hydrogen-open-source-sdn, last accessed: February 19, 2014.
[24] Sushant Jain et al., "B4- Experience with a Globally-Deployed Software Defined WAN," in *Proc. ACM SIGCOMM conference*, 2013, pp. 3–14.
[25] Brandon Heller et al., "ElasticTree: Saving Energy in Data Center Networks," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 17–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855711.1855728
[26] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annual Technical Conference*, 2014, pp. 305–320.
[27] A. Dixit, F. Hao, S. Mukherjee, T.V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," in *Proc. 2nd ACM SIGCOMM workshop HotSDN*, 2013, pp. 7–12.
[28] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain sdn controllers," The Computing Research Repository (CORR), August 2013. [Online]. Available: http://arxiv.org/abs/1308.6138