High Level Policies in SDN

Libor Polčák[1(✉)], Leo Caldarola[2], Amine Choukir[2], Davide Cuda[2], Marco Dondero[2], Domenico Ficara[2], Barbora Franková[1], Martin Holkovič[1], Roberto Muccifora[2], and Antonio Trifilo[2]

[1] Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno, Czech Republic
{ipolcak,ifrankova,iholkovic}@fit.vutbr.cz
[2] Cisco Systems Sarl, Rolle, Switzerland
{lcaldaro,amchouki,dcuda,mdondero,dficara,rmuccifo,antrifil}@cisco.com

Abstract. Policies for network traffic handling define packet routes through networks, enforce required quality of service, and protect networks from security threats. When expressing a policy, one needs to characterise the traffic to which the policy applies by traffic identifiers. Low level traffic identifiers, such as IP addresses and port numbers, are available in each packet. Indeed, low level traffic identifiers are perfect for data plane routing and switching. However, high level traffic identifiers, such as user name and application name, are better for the readability and clarity of a policy. In this paper, we extend software defined networks with high level traffic identifiers. We propose to add additional interface to SDN controllers for collecting traffic meta data and high level traffic identifiers. The controller maintains a database that maps high level traffic identifiers to a set of flows defined by low level traffic identifiers. SDN applications can apply policies based on both high level and low level traffic identifiers. We leave the southbound protocols intact. This paper provides two examples of High Level SDN paradigms – Application-Aware Networks and Identity-Aware Networks. The first paradigm enables policies depending on application names and characteristics. The latter allows policies based on user names and their roles.

1 Introduction

Recently, Software Defined Network (SDN) emerged [21] as a new paradigm based on the separation of the network control logic (control plane) from the forwarding fabric (data plane). The network control plane is orchestrated by a central SDN controller or a set of cooperating SDN controllers. The data plane is independently implemented in each network device and it allows to forward data as fast as possible, preferably on the line speed. Current SDN controllers are often modular, open source, extensible, and provide so called northbound interface that allows network operators and service providers to simplify network management operations with custom SDN applications that controls the policies followed by the controller, and consequently, the network.

SDN handles traffic [21] with respect to low level *traffic identifiers*, such as header field values and physical identifiers, e.g., an interface identifier.

M.S. Obaidat and P. Lorenz (Eds.): ICETE 2015, CCIS 585, pp. 39–57, 2016.
DOI: 10.1007/978-3-319-30222-5_2

SDN has no insight on the relations between flows. Due to the lack of high level traffic identifiers, it is not possible to directly and consistently specify policies for all flows belonging to a specific application or define policies for specific persons that are a part of a communication.

This paper extends SDN with high level traffic identifiers that are used to specify rules to control network traffic. The suggested high level traffic identifiers include the name of the application that generated a flow, the user name of the person that is a part of the communication, the required bandwidth for a specific flow and others. The network administrator can define specific policies for specific persons or applications, hence, the administrator does not need to specify policies for each flow separately. The extended SDN is called High Level SDN.

Currently, the predominant way of managing and configuring network is on a per-network-node and per-endpoint-type basis, irrespective of the commonalities that traffic patterns exhibit. Another painful point is the decentralised policy configuration. This configuration and management model is error prone, not flexible and cumbersome. High Level SDN unifies and centralizes the configuration. Our major contribution is in the abstraction of network traffic handling. High Level SDN recognizes relations between different flows originating from the same application, the same machine or the same user. Therefore, it is possible to handle traffic of the same application, the same machine, or the same user in a consistent manner. High Level SDN introduces high level traffic identifiers, such as application name or a user name. A network administrator specifies policies based on the high level traffic identifiers. High level policies allow for instance:

- To develop application-aware quality of service capabilities matching business-critical applications traffic and to set its priority higher. In addition, the quality of service can be further tweaked according to priorities associeted with the person that generates or produces the traffic (e.g., calls of a manager have higher priority compared to lower-level staff);
- To precisely reserve network bandwidth according to the expected traffic rate of the flows generated by specific applications;
- To deploy application-dependent or identity-related dynamic service chaining, i.e., routing of flows of a specific application or a specific person through specific network services (e.g., IPS/IDS, packet filtering, load balancing or caching).

High Level SDN does not require any specific SDN controller. However, High Level SDN requires additional sources of information — *traffic metadata*, it does require a modular and extensible SDN controller or a specialized High Level SDN controller. The differences between an SDN controller and a High Level SDN controller are following:

- High Level SDN controller provides additional interface for traffic metadata.
- High Level SDN provides traffic metadata via northbound interface to SDN applications.
- High Level SDN converts rules with high level traffic identifiers to standard SDN rules.

This paper extends the paper *Towards a real application-aware network* [3] presented at the 6th International Conference on Data Communication Networking (DCNET-2015), in which we presented the idea of an Application-Aware Network (AAN) — an SDN extension that process traffic for each application in a consistent manner. In this paper, we generalise the idea and introduce Identity-Aware Networks (IAN) as another example of high level extensions for SDN.

We evaluated the feasibility of the High Level SDN in several testbeds. We employed several SDN controllers: Opendaylight, POX, and Pyretic. One testbed was composed of real routers. High Level SDN allowed to simplify the network policies by reducing the number of rules in the policies enforced by SDN applications.

This paper is organized as follows. Section 2 introduces High Level SDN architecture and basic building block of a High Level SDN controller. Section 3 describes two examples of a High Level SDN – AAN and IAN. Section 4 lists use cases for High Level SDN along with examples of High Level SDN policies. Testbeds are described in Sect. 5. Section 6 reviews related literature. Section 7 discusses the contribution of this paper. The paper is concluded in Sect. 8.

2 High Level SDN Architecture

Controllers manage routers and switches in a plain SDN network via a southbound interface, whereas SDN applications instruct controllers via a northbound interface. Unfortunately, all current SDN southbound protocols, such as OpenFlow or OnePK, limit the amount of information exposed to a controller and consequently to SDN applications: only low level network traffic identifiers and metrics are available. The controller does not have information about the applications that produce the flows, about the relations between flows (e.g., the relation between a SIP control channel and associated RTP streams), or about the users that produce the flows.

Network endpoints do know the meta information about flows they produce or consume. An SDN application that can utilize additional high level traffic identifiers can achieve better traffic handling than a plain SDN application leveraging only low level traffic identifiers.

Figure 1 depicts building blocks of a High Level SDN network, described in detail in the following subsections.

2.1 Network and Hosts

Network is any SDN network controlled by a controller. There are no limits on the SDN network. The network connects hosts and servers. The hosts run applications that can communicate with server applications, e.g., a call manager – SIP server. In this paper we call server applications *application managers*.

2.2 SDN Control Plane

We do not assume any specific SDN controller. However, we assume that the controller provides at least basic OpenFlow functionalities, i.e., it is able to

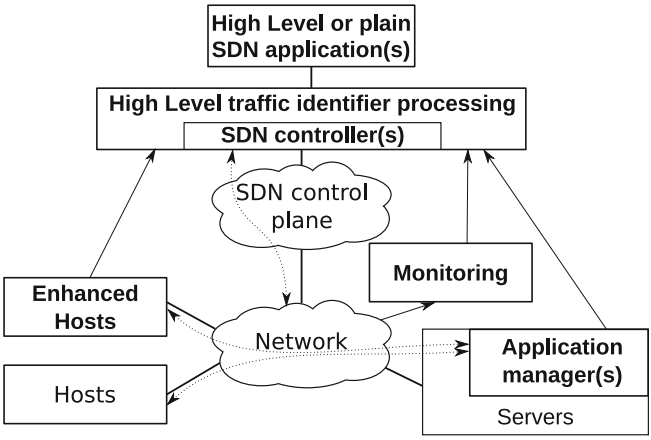

Fig. 1. Architecture of a High Level SDN network.

provide visibility and statistics about the flows crossing the SDN nodes and support basic actions to set quality of service or select the output interface.

2.3 Sources of High Level Information

Besides information provided by the control plane, we consider three additional information sources:

Enhanced Hosts: are equipped with a middle layer that passes locally known information to the controller, e.g., the name of the application that produces a specific flow or the bandwidth requirements of the flow, memory or CPU utilization.

Application Managers: (e.g., the Microsoft Exchange Server, Microsoft Lync or Cisco Unified Communications Manager for VoIP, VMWare's VSphere and OpenStack) are another source of information about application flows. A general trend is to move application managers toward cloud solutions coupled with provisioning of APIs to advertise and control flows.

Monitoring Tools: monitor network and its flows and extract high level information. Monitoring tools include DPI engines, IPFIX probes and collectors, or custom monitoring applications.

2.4 High Level SDN Controller and SDN Applications

A High Level SDN controller provides a consolidated and coherent view on the network traffic, merging flow information gathered from several sources. Any extensible modular SDN controller can be extended to handle high level traffic identifiers. See Fig. 2 for details.

Firstly, data incoming from high level information sources has to be evaluated and processed. As different sources of flow information may provide different

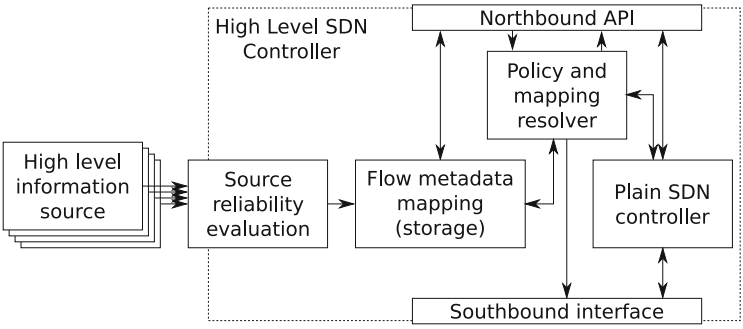

Fig. 2. High Level SDN controller internals.

information, the controller needs to merge the information consistently. One possibility is to rate the reliability of the source, e.g., information learnt from a DPI engine are less reliable than information provided by an application manager.

The controller stores gathered high level information as a mapping that maps high level traffic identifiers (such as application name, user identity) to a set of low-level traffic identifiers. The mapping is internally stored by the controller. The northbound APIs are extended to provide the high level information to SDN applications.

Finally, a High Level SDN controller incorporates a policy and mapping resolver module that converts high level policies (installed by SDN applications) to plain SDN flows by evaluating the stored mapping of the high level traffic identifiers and known traffic flows. Another task of the module is to compute statistics for high level traffic identifiers, e.g., bytes sent and received. Policy and mapping resolver can access the southbound interface either directly or delegate the network control to the plain SDN controller via its northbound API.

A High Level SDN controller provides northbound API that allows to handle traffic according to high level identifiers. A network operator can install specific SDN applications that handle traffic according to high level policies incorporating high level traffic identifiers. Hence, the High Level SDN controller abstracts the complexity of traffic handling. Moreover, the High Level SDN controller design ensures rule consistency across the SDN network.

In addition to reliability, each metadata source depends on a specific trust model that prevents the possibility of false metadata injection. Therefore, the interface from enhanced hosts, application managers, and monitoring tools to the enhanced SDN controller should employ a two-way authentication mechanisms. In our experiments, we did not focus on authentication security, or, we used Medianet authentication model [6], which provides a two-way authentication.

3 High Level SDN Examples

During our research, we examined two High Level SDN paradigms. Firstly, we focused on AAN [3]. The main aim is to provide consistent handling of

application traffic. Secondly, in this paper, we analyze traffic handling based on a person (or alternatively a machine) that takes a part in a specific communication. We denote such a paradigm as IAN.

The main feature of both paradigms is the abstraction from the complexity of dealing directly with network flows. This section gives examples of metadata processed by a High Level SDN controller. However, rather than specifying a fixed set of metadata that has to be processed by any High Level SDN controller, we suggest metadata that can be available — the list is not exhaustive. The set of available metadata also depends on the high level information sources and their capabilities.

3.1 Application-Aware Networks

The number of network services and applications is constantly growing. As each application shows different requirements in terms of bandwidth and latency, it becomes critical for network operators to be able to map traffic to applications.

Getting the right insight into the different applications allows network operators to plan their network capacity and policies to deliver the right quality of service satisfying the application requirements. For example, applications such as VoIP and video conferencing need low jitter and latency whereas peer-to-peer file transfer require high throughput to minimize the download time.

An AAN needs to provide enough high level traffic identifiers to differentiate the traffic according to its application, type, and business priority. The traffic identifiers should include at least:

Application Name – a string that uniquely identify the application that produces the traffic. The application name can be accompanied by additional information describing the specific application, such as application version, its vendor etc.

Application Category – the nature of the application, e.g., Voice over IP (VoIP), video streaming service, web pages, etc.

Device Class – type of the device that generated the traffic, e.g., a general purpose computer, surveillance camera, VoIP phone, etc.

Media Type – characteristics of the transferred data, such as audio, video, text, image, etc.

For each traffic flow, it is desirable to gather additional metadata, such as required bandwidth, latency, and traffic identification.

A High Level SDN application can create policies referring to high level traffic identifiers. For example, it can restrict access to specific links for a specific application, category or a device class to specific links: `Prefer links with bandwidth of 1 Gbps and higher for traffic of (application VLC or category Business-Critical).` A High Level SDN application can also reserve bandwidth, etc.: `Reserve bandwidth of 10 Mbps for traffic of type PhoneCall.`

3.2 Identity-Aware Networks

A typical business network is accessed by employees of the company, contractors, visitors, customers, etc. Typically, each role is associated with specific privileges. The categories can be further subdivided, e.g., the employees can be distinguished based on their department, importance for the company, etc.

The assignment of correct privileges to each network user is important to maintain the security of the network or to asses a specific quality of service to each user group.

An IAN has to provide high level traffic identifiers identifying users in the network, their importance and role. The supported traffic identifiers should include at least:

Personal ID – unique identification of a person.
Name – the name of the person.
Groups – the role of each user, e.g., employee, contractor, etc. Users and roles can be grouped together and form groups and subgroups, e.g., management, accounting department, etc.

Beside personalised services, IAN can profit from machine identification. For example, one can prepare specific policies for specific servers, e.g., `Drop traffic from device DatabaseServer leaving the network`.

The SDN application can create policies referring to the above mentioned traffic identifiers, for example, it can prioritize communication of specific persons or machines, e.g., traffic of the management or business-critical servers. An example of an identity-based policy is: `Reserve bandwidth of 1 Gbps at egress links for management`.

3.3 Other Traffic Identifiers and Combinations

Naturally, there are more than the two above-mentioned paradigms of High Level SDN. For example, the network operator might want to fine tune traffic handling in the network according to the geographical location of the communicating parties.

Moreover, AAN and IAN does not need to be deployed separately. On the contrary, their co-deployment allows to define even more precise policies, for example: `Drop traffic from device DatabaseServer leaving the network except traffic of application called System Updater`.

4 Use Cases

High Level SDN can be applied in a number of use cases. This section highlights the advantages of high level policies. Each use case is accompanied with examples of policies benefiting from high level traffic identifiers.

4.1 Remote Intrusion Detection System

An intrusion detection system (IDS) analyzes a copy of network traffic for anomalies. A remote IDS can analyse traffic gathered from many network links from different geographical locations. In addition, an IDS can be specialized for a certain type of traffic, or, the network operator is interested only in scanning the traffic of certain users (e.g., guests) or a specific type of devices (e.g., devices not managed by the company). This use case deals with the remote IDS scanning a specific type of traffic. Consider the following policy examples:

- `Duplicate data from applications called (Jabber or Thunderbird) to device IDS.`
- `Duplicate data from users in groups (Guest or Unknown) to device IDS.`
- `Duplicate data from devices (SmartPhone or SmartWatch) to device IDS.`

The SDN application processing, in cooperation with the High Level SDN controller, has to ensure the following:

- Select SDN switches that duplicates the traffic, for example, based on traffic load, path of the original traffic, or the location of the IDS.
- The network has to distinguish between the original traffic and its copy, for example by a VLAN ID or an MPLS tag.
- Populate the switching table of devices on the path with rules handling original traffic and the copy.

4.2 Packet Filter

Packet filters are a special type of a firewall, they drop malicious or unwanted traffic based on a set of rules. For example, a packet filter can be configured to drop traffic to a server from departments that should not access the server. Consider the following examples:

- `Drop data from users not in group IT department to application device SQLServer.`
- `Drop data from users in group Guest to any other user.`
- `Drop data from user Unknown to device Printer.`

The packet filtering SDN application has to instruct the controller to configure SDN switches to drop the traffic. The controller usually configure devices as close to the traffic source as possible. Another option is to optimize the number of SDN rules offloaded to SDN switches and drop the traffic close to the destination.

4.3 Path Load Balancing

Redundant links in network topologies often offer alternative paths from a source to a destination. Nevertheless, the available bandwidth or latency can differ on each path. Path load balancing aims at distributing traffic to available paths to improve quality of experience (QoE). For example, consider the following policies:

- `Route data from application FTP Client along path with most available bandwidth.`
- `Route data of users from department Hotline along path with lowest delay.`
- `Route data from devices VoIP Telephone along best QoE path.`

The implementation should monitor load, latency, and other characteristics of traffic links and paths. In case of a congestion, the High Level SDN application should reroute the traffic to another path meeting the policy, if available.

4.4 Traffic Monitoring

Network administrators have to monitor user behavior [12], for example, for conformance with legislation [8,26], for security incident tracking, potential future planning, and service access billing. In all cases, the controller should provide flow and derived statistics to High Level SDN applications, either gathered using southbound protocol or from external source, such as a IPFIX probe. High Level SDN controller should derive statistics for high level traffic identifiers by linking low-level-traffic-identifier statistics. Consider the following examples:

- `Monitor traffic statistics of users accessing device ProtectedServer.`
- `Monitor traffic statistics of application Backup for data exchanged with device BackupServer.`
- `Monitor traffic statistics of group Servers for data exchanged with device DatabaseServer.`

After a High Level SDN application subscribes for statistics, the Policy and Mapping Resolver converts high level traffic identifiers to a set of low level traffic identifiers. Then, the Policy and Mapping Resolver instructs the SDN controller to monitor the set of low level traffic identifiers. Every few seconds, the SDN controller polls statistics from SDN switches. In the case of IPFIX statistics, probes continuously send statistics to the SDN controller. After the SDN controller receives fresh statistics, it passes them to the Policy and Mapping Resolver, which combines statistics from low level traffic identifiers to statistics of high level traffic identifiers. The High Level SDN controller sends the result to subscribed High Level SDN applications.

5 Testbeds

During our research we have created application-aware testbed and identity-aware testbed to validate our high level SDN concept. We implemented the use-cases from Sect. 4 in our testbeds using several SDN controllers: Opendaylight[1], POX[2] and Pyretic[3].

[1] https://www.opendaylight.org/.
[2] http://www.noxrepo.org/pox/about-pox/.
[3] http://frenetic-lang.org/pyretic/.

5.1 Application-Aware Testbed

The application-aware testbed (depicted in the Fig. 3) is composed of different types of endpoint devices and applications, including both proprietary and open sources devices. Two Cisco phones and two Jabber clients are registered with a Cisco Unified Communication Manager and adopted SIP to establish audio/video sessions. Phones and Jabber clients make video calls, thus each producing two media network flows for each call: audio and video. In addition, a video and audio stream were multiplexed in a single network flow from a server to a VLC[4] client. Jabber and VLC have been enhanced with the adoption of the Media Service Interface (MSI) library [6], thus becoming application-aware endpoints.

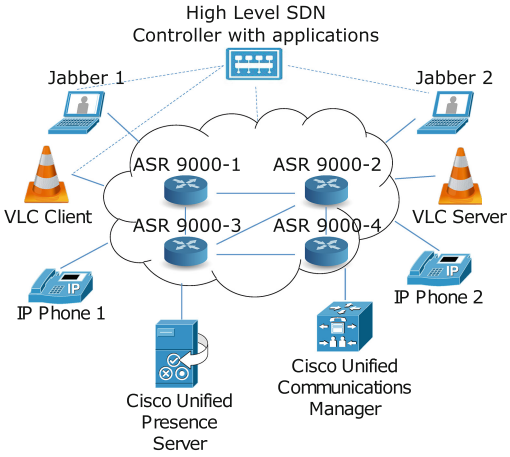

Fig. 3. Topology used in our application-aware testbed.

The ASR 9000 routers punt SIP packets produced by the Cisco IP phones and the Jabber clients to the SDN controller. Those SIP packets are then passed to a DPI module. Because Cisco phones are configured not to encrypt signaling traffic, their SIP packets are then parsed by the DPI module. The DPI module, in turn, collects SIP session details and pushes them to the common database thus mapping network flow details to application.

Jabber flows are instead announced by MSI to the appliaction-aware endpoint manager that, similarly to DPI module, stores the Jabber application to network flows mapping into the common database. The same mechanism is adopted for the VLC client that announces RTP flows carrying the video and audio through MSI to the High Level SDN controller.

Normalisation is then applied on all the acquired data, according to the degree of reliability and completeness of the sources. For example, SIP packets generated by Jabber clients are not only advertised by MSI but also by the

[4] http://www.videolan.org/vlc/index.html.

DPI module (if they are not encrypted). This happens because the SIP packets are punted to the SDN controller and handled to the DPI module. Therefore normalisation enables for merging of the information coming from both sources, giving higher priority to the information provided by MSI as it is more reliable. The testbed treats the information sources according to the following priorities:

- Application managers are operated by network administrators and are the most reliable source. Information about both encrypted and unencrypted traffic is visible. They cover only the applications provided inside the testbed.
- Application-aware enhanced hosts have a little bit lower reliability as the endpoint can provide false information as a result of a virus infection or malicious tampering with the MSI library. Nevertheless, MSI signals information about relations between flows. Application-aware enhanced hosts also provide information about both encrypted and unencrypted traffic.
- DPI is the less reliable source as it cannot decrypt encrypted flows and there is a risk of false positive or false negative identification of a protocol.

We created two presentations [4,25] from the experiments with the testbed. The first one [4] focuses on quality of service management. The network operator is able to assign priorities to each running application and consequently provide desired quality of experience during congestion.

The latter presentation [25] focuses on path load balancing. The presented application-aware High Level SDN applications distributes traffic flows of selected applications to available paths. The goal of the High Level SDN application was to show that it is possible to operate the AAN even if not all flows are recognized by the High Level SDN controller.

Two sets of applications were operated in the testbed: (1) those running on enhanced hosts and (2) those that initiated flows that were not advertised to the High Level SDN, i.e. those that created background traffic.

The aim of the High Level SDN application [25] was to load balance the traffic of the business-critical applications across all possible paths between the source and destination. In case the High Level SDN application detected that a link carrying priority traffic was congested, it automatically rerouted the traffic to links with available capacity. Hence, even if the link was congested because of traffic from legacy endpoints (that were not application-aware), the AAN architecture allowed to ensure that the priority traffic reaches its destination without unnecessary retransmits.

5.2 Identity-Aware Testbed

The identity-aware testbed (see Fig. 4) consists of SDN switches, hosts, servers, an SMTP DPI probe and an IDS. From the hosts, several users access multiple services on the servers. Servers provide HTTP web service for user authentication, SMTP mail service for sending e-mails and FTP file service for data storage. The web service is configured to send user login name to controller after each successful authentication. DPI probe analyzes the SMTP traffic and sends user authentication login names from each SMTP session to the SDN controller.

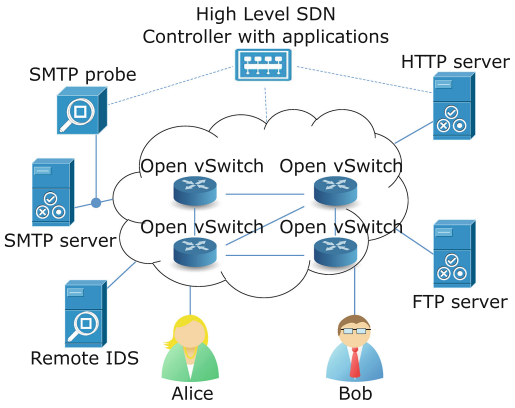

Fig. 4. Topology used in our identity-aware testbed.

The controller configures switches proactively to minimize the delay of packet processing. The controller detects mapping between IP addresses, MAC addresses and attachment points of each device from ARP packets. By default, all devices belong to user *Unknown*. When a user of a device authenticates to the web server or sends an e-mail, the controller learns the user login name and lower traffic identifiers from the high level information sources (HTTP authentication module and SMTP DPI probe).

We tested [14,16] several use cases in this testbed. The implementation of the remote IDS use case [14] dynamically selects the SDN switch that copies data to be monitored according to (1) the distance from the monitored user or device and (2) the number of rules installed to the SDN switch. The goal is to spread the rules across multiple SDN switches.

6 Related Work

It is already possible to control network traffic by high level identifiers. Network vendors offer solutions providing network application meta data (e.g., Medianet [27]) or custom identity-aware network control (e.g., Identity Driven Management [15]). This section focuses on the current state of research and on existing solutions analysing high level traffic identifiers.

Historically high level processing used heuristics [1,10], which inspect and infer flow characteristics. Heuristics may be based on port ranges [13], IP subnetting (special network subnets for specific applications, e.g., phones), or deep packet inspection (DPI) [22], e.g., application level gateway. Port based solutions suffer [22] from port overloading and inconsistent port usage. IP subnetting solutions are error prone and result in network management hassle. DPI is computationally expensive and becomes a challenge with the wider adoption of encrypted signaling and secured traffic. In addition, DPI-provided insights are hardly shared between network nodes on the path of the flow.

Network vendors proposed different solutions to make the network application-aware, such as Medianet [27], Application Visibility and Control [7] or Junos Application Aware [19]. A common characteristic of these solutions is the decoupling of flow identification from flow policies (prioritization, routing and others) through application specific tags. These tags can be either explicitly signaled as in the case of Medianet or locally produced by deep packet inspection as in the case of the Junos Application Aware solution. Based on the information about the requirements of the flows gathered from these tags, network nodes can decide the policies to be applied; in principle, this allows for a smooth collaboration between the applications and the network. However, today these solutions are mainly vendor specific. The lack of standards leads to a poor or a non-existing interoperability.

Recently Cisco introduced Medianet [27]: a solution that makes the network aware of the application traffic that it is carrying. In more details, Medianet endpoints leverage a proprietary middle layer (namely MSI) to advertise flows Metadata tags, e.g., application name, traffic type, media type and business importance. Network devices such as routers and switches take local decisions to handle the advertised flows in the desired way.

In identity controlled networks, vendors provide Identity Driven Management [15], Identity Service Engine [5], Identity and Policy Control [18]. End users submit their credentials to an autentification server (RADIUS, web), which forwards the credentials to centralized control of network configuration. Identity Service Engine [5] focuses on controlling access to network resources, while Identity and Policy Control [18] also implements accounting and tracking users activity. Identity Driven Management [15] can limit bandwidth and assign QoS based on user identities. Similarly to application-aware networks, these solutions are proprietary and lack interoperability.

6.1 Application Awareness in SDN

The usage of Software Defined Network technology to enable Application-Aware Networks is a fairly novel concept in both industry and academia. Jarschel et al. [17] showed a QoS boosting application that monitor the Youtube streaming experience. If certain quality thresholds are crossed, the application requests the BigSwitch SDN controller to enforce a routing change for the affected Youtube network flows.

Solutions such as that of Insieme (Application Visibility and Control [7]) also combine Application Awareness and SDN: DataCenter deployment and management of applications, databases and their traffic is unified through a unique controller that is drivable with a set of APIs. Similarly, PlumGrid [24] provides network abstraction for data-centers with comparable promises to those of Insieme. Curtis et al. [9] propose new ways to provide flow scheduling in Data Centers through OpenFlow. Das et al. [11] adopt OpenFlow to dynamically aggregate flows according to flow properties such as bandwidth, latency, jitter; thus providing different levels of QoS to applications. In the same spirit of using OpenFlow to aggregate traffic according to application knowledge,

Zhang et al. [28] propose an extension to the OpenFlow protocol to enable application optimization in Optical Burst Switching networks. Finally, Bredel et al. [2] show a viable way to coordinate Traffic-Engineering for traffic generated by scientific collaborations like the CERN's LHC.

Application awareness we present in this paper is similar to that of Jarschel et al. [17]. However, our approach is suited for a broader range of applications. In addition, we adopt carrier-grade IOS-XR hardware and Medianet, application-aware feature developed by Cisco.

6.2 Identity Awareness in SDN

Several recent studies investigated network control based on user identities. Resonance [23] uses OpenFlow switches to enforce high level dynamic acces control policies based on flow-level information and real-time alerts. Users authenticate through a credential based web site. The research, however, focused only on security, specifically on detecting potentially infected computers and taking appropriate action if one was found (e.g., dropping its traffic).

AuthFlow [20] also combines authetication and an access control mechanism. AuthFlow links user credentials from RADIUS server to a set of flows belonging to the authenticated host. AuthFlow denies access to network resources from unauthenticated users. Moreover, the application is able to allow or deny traffic based on different privilege level of each authenticated host.

The main goal of current identity-based software defined networks is (1) to find the most secure way to authenticate users, (2) to secure network resources and enable access to specified authenticated users or groups of users only. In this paper, we generalize the latter by using the knowledge of user identity in universal applications. Besides access control, our use cases include routing (e.g., data of a manager goes though a priority path), load balancing and duplicated packet delivery to IDS.

7 Discussion

In this section, we compare the High Level SDN to a classical distributed network and to an SDN network without high level traffic identifiers. Moreover, we describe a possible behaviour of a High Level SDN in typical network scenarios.

7.1 Evaluation

Being built on top of an SDN network, the High Level SDN maintains main advantages of the SDN architecture while reducing the complexity of applying the correct policies to related flows (typically belonging to a single application, a class of applications, or specific groups of users).

When expressing a policy, one needs to characterise the traffic to which the policy applies by certain traffic identifiers. The traffic identifiers define a class of traffic (a.k.a. class-map). Low level traffic identifiers, such as IP addresses and

port numbers, are available in each packet. Hence traffic handling on nodes is traditionally defined by low level traffic identifiers. However, network administrators tend to define traffic in more abstract form, e.g., in relation to the person that generated the traffic or for specific applications.

We define the policy complexity as the number of network nodes where a class-map must be explicitly declared multiplied by the number of entries in the policy itself. Let us consider a network of N nodes with the number of applications denoted as A, each application carrying F_A different flows in average.

An alternative view on network traffic handling is based on user identities. Imagine the same network with N nodes with the number of attached users denotes as U, each user opens F_U active flows in average.
The policy complexity can be evaluated in the following way:

- In a conventional network, a rule has to be distributed for each flow of each application or each user on every network node. Therefore, the total complexity can be evaluated as $\mathcal{O}(A \times F_A \times N + U \times F_U \times N)$.
- In high-level-traffic-identifiers-aware distributed networks (HL-distributed networks), for instance Metadata-aware network, the complexity reduces to $\mathcal{O}(A \times N + U \times N)$ since the network can deal directly with applications and specific user traffic instead of flows.
- In SDN, the controller (or the set of cooperating controllers) abstracts the complexity of programming each node separately. However, a rule for each flow must be explicitly declared; thus, the total complexity in this case becomes $\mathcal{O}(A \times F_A + U \times F_U)$.
- In High Level SDN, application policies are handled in a straightforward manner since the complexity with flows is abstracted by the high level SDN controller; thus, the number of class-maps that need to be specified reduces to $\mathcal{O}(A + U)$.

Feature velocity in distributed environments depends on the number of different operating systems and different hardware platforms to be supported. In the case of SDN, the decoupling of the capabilities exposed by the network platforms and the flow characteristics allows for an increased velocity in delivering high-level-traffic-identifiers-aware features. This is achieved through writing software (an SDN application) once rather than integrating software into different network platforms. Hence, the feature velocity of SDN is fast compared to conventional or HL-distributed networks.

The control plane abstraction ensures that there is a single set of policies applied to all SDN switches in the network. In comparison, the cumbersome management of consistent rules on each node might result in inconsistent configuration, unpredictable behaviour of the network, and complicated troubleshooting. The network administrator has to maintain N configurations consistently.

In a conventional deployment, the policy complexity often results to a high number of class-map rules. The class-map rules has to be managed directly on the network nodes in a consistent manner. The maintenance of the consistency incorporates high operational costs accompanied with slow feature velocity. Despite lower complexity of HL-distributed deployment, it still requires medium

operational costs as it requires big effort to maintain consistent behaviour. The decoupling of control and data plane and the abstracted complexity of the network lowers the operation costs of SDN-based paradigms. Moreover, High Level SDN treats traffic of specific high level characteristics in a consistent manner.

The comparison is summarized in Table 1.

Table 1. Comparison of available solutions to enforce policies for specific flows related to user identity and applications.

	Conventional	HL-Distributed	SDN	High Level SDN
Policy complexity $\mathcal{O}(\dots)$	$A \times F_A \times N + +U \times F_U \times N$	$A \times N + U \times N$	$A \times F_A + +U \times F_U$	$A + U$
Feature Velocity	Slow	Slow	Fast	
Consistent behavior	Cumbersome	Cumbersome	Automatic	
Operational cost	High	Medium	Low	Lowest

7.2 Typical Scenarios

Let us consider some specific scenarios to illustrate the benefits deriving from the deployment of a High Level SDN.

Network Congestion. Although congestion of a link is easy to detect in a distributed network, none or few network technologies exist to react and reroute flows in a timely way.

With SDN, the controller can immediately re-route traffic depending on queue occupancy or interface load increase. However, the information available to an SDN controller are limited to low level identifier (up to the transport layer).

Compared to the SDN, the High Level SDN offers two main advantages. Firstly, the routing operations can be based on high level requirements; thus all flows of a single application or a single user can be handled in a consistent way without the complexity of mapping flows by the SDN application. The second, and more significant advantage, is that the High Level SDN controller can potentially access high level metrics and eventually, it can detect the looming resource insufficiency for specific traffic in advance.

Topology Change. In the event of a topology change that impacts the path of flows with specific requirements, a regular Medianet solution requires a new distribution of Medianet tagging in the new path. If the change reoccurs with some frequency because of physical impairments, it can result in a flapping behavior and degraded quality. High Level SDN controller with complete network and application view can instead choose a stable path that avoids the prevents periodic rerouting.

Application Policy Change. In a Medianet network, whenever application requirements change, the new requirements are pushed by administrator to the network devices through configuration. Administrator has to foresee possible problems in advance so that the change targets only affected devices; and avoids changes to unrelated traffic so that a single change does not trigger an avalanche of changes. In High Level SDN, the new application requirements are enforced on all affected devices by the High Level SDN controller, which has the complete view of all flows and their paths.

8 Conclusion

Network policies are complex and, typically, they are based on abstract identifiers, such as application name, application class, user name, or user role. Users demand consistent application behavior and high quality of multimedia streams. Business environment requires traffic handling based on user role, e.g., access to some resources is limited for unpriviliged users. Conventional policy management requires to consistenly maintain configuration of many network nodes, however, on top of routing and switching according to the information carried in packet header fields, several solutions aims to simplify the configuration based on abstract traffic characteristics. SDN does not offer high level characteristcs and traffic identifiers.

This paper introduced High Level SDN architecture achieving several goals:

- The architecture is generic, extensible, and compatible with current SDN controllers. We demonstrated its suitability on an application-centric and user-centric paradigm.
- High level policies simplify traffic management.
- High level SDN is applicable to many use cases.

The testbed evaluation of the implemented High Level SDN applications confirmed that the idea of High Level SDN applications is easily applicable in real networks. During the project, we brought application-awareness Cisco IOS-XR routers, specifically, ASR 9000. Additionally, tesbeds proved fast application velocity. We were able to develop custom SDN applications in several days or weeks without the need to wait for a new version of a network operating system.

Acknowledgements. This work was supported by Cisco Systems Switzerland where the idea of AAN emerged, was implemented, tested and evaluated. The work focusing on IAN and generic High Level SDN is a part of the project VG20102015022 supported by Ministry of the Interior of the Czech Republic and it was also supported by the BUT project FIT-S-14–2299.

References

1. Bendrath, R.: Global technology trends and national regulation: explaining variation in the governance of deep packet inspection. Technical report, Delft University of Technology (2009), Paper originally prepared for the International Studies Annual Convention

2. Bredel, M., Barczyk, A., Newman, H.: Application-aware traffic engineering for wide area networks using openflow. In: SuperComputing Conference, Emerging Technologies (2013)
3. Caldarola, L., Choukir, A., Cuda, D., Dondero, M., Ficara, D., Muccifora, R., Polčák, L., Trifilo, A.: Towards a real application-aware network. In: Proceedings of the 6th International Conference on Data Communication Networking (DCNET-2015), pp. 5–12. SciTePress - Science and Technology Publications (2015)
4. Choukir, A., Caldarola, L., Cuda, D., Dondero, M., Ficara, D., Muccifora, R., Polčák, L., Trifilo, A.: Towards a real application aware network (2013). http://youtu.be/QHYPhAhIwVw
5. Cisco: Cisco identity services engine (2015). http://www.cisco.com/c/en/us/products/security/identity-services-engine/
6. Cisco Systems: Cisco MSI Deployment Guide (2013). http://www.cisco.com/web/solutions/medianet/docs/Cisco_MSI_Installation_Guide.pdf
7. Cisco Systems: Application Visibility and Control (2014). http://www.cisco.com/c/en/us/products/routers/avc_control.html
8. Council of Europe: Convention on Cybercrime (2001), ETS No. 185
9. Curtis, A.R., Kim, W., Yalagandula, P.: Mahout: low-overhead datacenter traffic management using end-host-based elephant detection. In: IEEE INFOCOM (2011)
10. Dainotti, A., Pescape, A., Claffy, K.: Issues and future directions in traffic classification. IEEE Network **26**(1), 35–40 (2012)
11. Das, S., Yiakoumis, Y., Parulkar, G., McKeown, N.: Application-aware aggregation and traffic engineering in a converged packet-circuit network. In: Optical Fiber Communication Conference and Exposition (OFC/NFOEC) and the National Fiber Optic Engineers Conference (2011)
12. ETSI: ETSI ES 201 158: Telecommunications security; Lawful Interception (LI); Requirements for network functions. European Telecommunications Standards Institute (2002), version 1.2.1
13. Fraleigh, C., Moon, S., Lyles, B., Cotton, C., Khan, M., Moll, D., Rockell, R., Seely, T., Diot, S.: Packet-level traffic measurements from the sprint IP backbone. IEEE Network **17**(6), 6–16 (2003)
14. Franková, B.: Lawful Interception in Software Defined Networks (2015). Master's thesis (in Czech), Brno University of Technology, CZ
15. Hewlett-Packard: Identity driven management: technical brief (2015). http://www.hp.com/rnd/pdf_html/IDM_technical_brief.htm
16. Holkovič, M.: SDN Controlled According to User Identity (2015). Master's thesis, Brno University of Technology, CZ
17. Jarschel, M., Wamser, F., Hohn, T., Zinner, T., Tran-Gia, P.: SDN-based application-aware networking on the example of youtube video streaming. In: European Workshop on Software Defined Networks (2013)
18. Juniper Networks: Identity and policy control (2015). http://www.juniper.net/us/en/products-services/ipc/
19. Juniper Networks Inc: Junos Application Aware: Deep packet Inspection (2015). http://www.juniper.net/us/en/products-services/network-edge-services/service-control/junos-application-aware/
20. Mattos, D.M.F., Ferraz, L.H.G., Duarte, O.C.M.B.: AuthFlow: Authentication and Access Control Mechanism for Software Defined Networking, Technical Report, Electrical Engineering Program, COPPE/UFRJ, April 2014. http://www.gta.ufrj.br/ftp/gta/TechReports/MFD14.pdf

21. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
22. Moore, A.W., Papagiannaki, K.: Toward the accurate identification of network applications. In: Dovrolis, C. (ed.) PAM 2005. LNCS, vol. 3431, pp. 41–54. Springer, Heidelberg (2005)
23. Nayak, A.K., Reimers, A., Feamster, N., Clark, R.: Resonance: dynamic access control for enterprise networks. In: Proceedings of the 1st ACM workshop on Research on Enterprise Networking, pp. 11–18, ACM (2009)
24. PLUMgrid: PLUMgrid: virtual network infrastructure (2014). http://plumgrid.com
25. Polčák, L.: Integration of SDN and medianet metadata (2014). http://youtu.be/CqDYn4-DKn8
26. The Council of the European Union: COUNCIL RESOLUTION of 17 January 1995 on the lawful interception of telecommunications (96/C 329/01) (1996)
27. Wilkins, S.: Designing for Cisco Internetwork Solutions (DESGN) Foundation Learning Guide (CCDA DESGN 640–864). Pearson Education, Boston (2011)
28. Zhang, D., Mai, S., Guo, H., Tsuritani, T., Wu, J., Morita, I.: Openflow-based control plane for the application-aware lobs network. In: OptoElectronics and Communications Conference (2013)

http://www.springer.com/978-3-319-30221-8

E-Business and Telecommunications
12th International Joint Conference, ICETE 2015, Colmar, France, July 20–22, 2015, Revised Selected Papers
Obaidat, M.S.; Lorenz, P. (Eds.)
2016, XXIII, 534 p. 178 illus. in color., Softcover
ISBN: 978-3-319-30221-8