Issues in Information Systems
Volume 22, Issue 1, pp. 320-334, 2021

DOI: https://doi.org/10.48009/1_iis_2021_320-334

OpenFlow switch controller as a policy-based system

Abdur Rahim Choudhary, *Choudhary Associates, rahimchoudhary@gmail.com*

Abstract

In this paper we present the OpenFlow Switch that enables the software designed networks (SDN) as a policy-based system in which the Switch in the data plane is treated like the managed object and the Controller plays the role of a PDP (Policy Decision Point). Viewed as a PDP the Controller itself incorporates digital policies and can use some of the IETF protocols originally intended for the policy-based systems. This approach allows addressing some nagging problems in the SDN technology. Therefore, we also present a policy based SDN Controller architecture, and analyze it to illustrate how to address these problems.

Keywords: OpenFlow, Controller, Hierarchical, Distributed, Architecture, Policy Based Systems, Software Defined Networks, Security, Digital Policies.

Introduction

Much has been written on the promises of the software defined networks (SDN) (Sood et.al, 2019), (Paliwal et.al, 2018), (Goransson and Chuck, 2014). There are, however, challenges (Horvath et. al, 2015), (Sezer et.al, 2014) , (IETF, 2012) in achieving these promises. Some of these challenges include implementation, topology, security, performance, and scalability. Any one of these challenges has the potential to significantly diminish the promise of software defined networks; it is, therefore, important to reexamine the SDN in a new light to open up new ways to meet the challenges. In this paper we reexamine the OpenFlow Switch that enables the SDN technology as a policy-based system (PBS). This is done with the expectation that new approaches will become available to resolve the above-mentioned challenges. We will illustrate the new approach in the context of an SDN security use case. In this context, we will explicitly demonstrate how the challenges are addressed for network latency, size of the backend database, the problem of scalability, the automation of operations, and the evolvability of the system with evolving business intent and requirements.

There are related technologies such as network functions virtualization (NFV) (ETSI, 2021), and business intent-based networking (BIBN) (Riftadi and Kuipers, 2019) though it is not yet standardized. The adoption of policy-based management (PBM) technology not only accommodates these related technologies, the approach taken in this paper prompts them and helps deploy them. We will not dwell on these auxiliary technologies though we will surgically point out where they are incorporated in the proposed architecture for the SDN controller.

The architecture proposed in this research is a brand-new deployment topology. It embeds multiple existing and emerging concepts, namely the OpenFlow switch, the software defined networks, and policy-based management as well as network functions virtualization and the incorporation of business intent into network operations.

The purpose of this paper is to present an approach that resolves the above-mentioned problems, and thereby makes the benefits of SDN better achievable because of the advantages embedded in the underlying architecture.

Section 2 presents the motivational background for this research; Section 3 discusses how to combine OpenFlow and PBM technologies; section 4 presents an architecture for OpenFlow Switch Controller using policy-based system approach; section 5 exemplifies and analyzes the digital policies that are deployed in this controller; section 6 shows how the approach helps in meeting the challenges; and section 7 presents the main conclusions.

Motivational Background

The switches and routers in conventional networks are largely proprietary. The users have no control over their behavior except through configuration parameters that the vendors expose to the users through protocols like the simple network management protocol (SNMP) (Stallings, 1998) . The vendors improve upon the situation a little by deploying private management information bases (IBM, 2020) that offer more configuration choices, but also help the vendors to lock the customers in. The choices offered by these configuration settings reflect the technology state back in the nineties when SNMP capabilities were deemed adequate. Their usefulness for today's networks is limited, inflexible, and un-evolvable for the needs of today's applications and services.

Further, for a large enterprise network containing a large number and types of network elements, it is slow and expensive to use human administrators to manage the configurations of devices within its network infrastructure. This problem requires automation, which has not been adequately possible with the current network management devices. The solution requires more user control over the configurations of the network elements than the vendors currently allow. There is thus a clear disconnect between the network management capabilities that the users need to manage and automate an enterprise network infrastructure, and the capabilities that the vendors currently make available to the users to accomplish the task.

To improve this situation, one needs to note that a typical network switch or router has two important aspects; a data plane and a control plane. The data plane switches the packets in a flow. The control plane instructs the data plane how it should treat the packets in a particular flow. There are many complex considerations in exercising this control. Examples include the quality of service that a packet receives, priority and preemption criteria among the packets, security classification of the packets, security functions each classification should support, origin and destination of the packets, and the network privileges of the user who originates the packets.

The current developments in cloud computing services, Internet of things, and Big Data analytics further complicate these factors for configuring, managing and automating the network infrastructures. In the case of US department of defense (DoD) the information superiority requirements and the needed

network centric warfare capabilities place very specialized additional requirements. Current network management capabilities are grossly inadequate for such operations, and DoD took some initiatives such as the adoption of Policy Based Network Management (PBNM) (Choudhary, 2007) to help with automation and to allow the desired evolvability. However, such an approach requires that the switch or router open up to loosen the proprietary nature of these devices and to allow more control for the enterprise users to manage their infrastructure in accord with their specific operational requirements.

The OpenFlow Switch opens up a proprietary switch into its switching and controlling functions, giving the user greater control over the controller operations via standardization of the controller. This, however, still does not facilitate automation of the network infrastructure configurations. This is where Policy Based Systems (PBS) technology enters as a crucial enabler. In addition, the PBS technology enables the controller to do its functions more efficiently and more in accord with the user requirements. The two technologies are thus crucially complementary so that both need to be deployed together.

The switching in the data plane becomes a policy managed object; and the controller acts like a policy decision point (PDP) to manage the switching operations. The PBS capabilities within a OpenFlow Controller enable interesting capabilities such as automation, flexibility and evolvability, and adaptability to the future requirements. For real-time critical operations, the PBS allows the use of common operations picture and situation awareness parameters to be encapsulated within the Digital Policies. Below, we present further details.

Combining OpenFlow and PBS Technologies

Figure 1 illustrates the basic functions of a typical switch or a router as consisting of a data plane and a control plane. Opening of the switch introduces a standardized module in an otherwise proprietary control plane. It also applies a standard protocol for this standardized module to communicate with a user customizable Controller. The standardized protocol is the OpenFlow protocol, and a compliant switch is an OpenFlow Switch. A controller that complies with the OpenFlow protocol is an OpenFlow Controller. Standardization allows an enterprise to procure the OpenFlow Controllers from one vendor and the OpenFlow Switches from a different vendor, and even mix and match between Controllers and Switches from multiple vendors. The hold of the vendors over the customers is therefore loosened, there is scope for new vendors to step in effectively leveling the playing field between the vendors, and there is far bigger scope for the enterprise users to innovate.

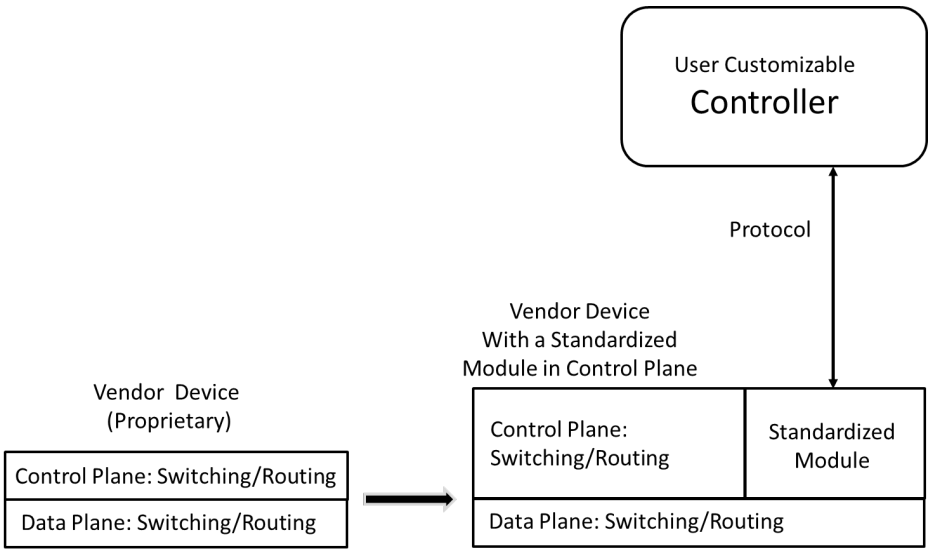

Figure 1 A proprietary vendor device on the left is opened up for OpenFlow behavior on the right.

A network that deploys OpenFlow compliant Switches and Controllers is a Software Defined Network (SDN). Figure 2 shows an OpenFlow Switch architecture (Open Networking Foundation, 2015). OpenFlow protocol for a user-controllable device behavior was specified originally at the Stanford University (McKeown et.al. 2008) and subsequently at the Open Networks Forum (Open Networking Foundation, 2015). The controller in an OpenFlow device is user programmable. The logic programmed in the controller is used to define the routing behavior of an OpenFlow switch. An OpenFlow device comprises of a control channel, a logically centralized controller, a number of ports, and a number of tables that provide the pipeline processing for data path.

OpenFlow protocol is used for communications between the switch and the controller. The protocol version 1.5.1 defines only one SDN controller. We have shown two controllers in Figure 2 because we will latter discuss an architecture with multiple controllers, though they present themselves together as a logically centralized controller. The communications between the switch and the controller take place over an OpenFlow channel using transport layer security (TLS) or a simple transport control protocol connection (Open Networking Foundation, 2015). The connection enables the controller to send control messages to the switch, it enables the switch to send asynchronous messages to the controller to inform about the device state; and it also enables symmetric messages that either the switch or the controller can send.

It is instructive to compare this architecture with that of a policy-based system (PBS) (Choudhary, 2004). We will not go into details of a PBS but according to reference (Choudhary, 2004) PBS consists of a policy enforcement point (PEP) which communicates with the device on one hand and the policy decision point (PDP) on the other. The PEP and the PDP communicate using the common open policy service (COPS) protocol (Durham et. Al, 2000).

Comparing the Policy-Based Management architecture as discussed in (Choudhary, 2004) with the OpenFlow architecture shown in Figure 2, the following mappings are obvious:

PBS PEP → SDN Switch
PBS PDP → SDN Controller
PBS COPS protocol → SDN OpenFlow protocol

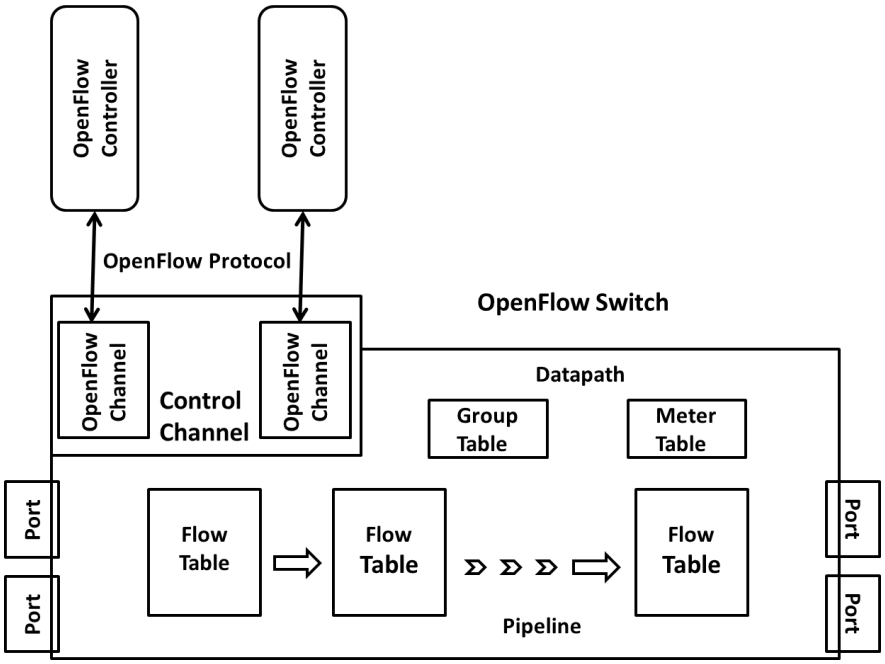

Figure 2 An OpenFlow Switch Instance

In addition, PBS has two other capabilities which can also be usefully leveraged for SDN architecture.

1. Backend database for use by the PDP, namely the policy repository, not shown in Figure 2 but needed for the policy-based implementation of the controller.
2. Digital policy management (DPM) functions are needed to manage the policies that populate the controller. The capability is useful for all policy-based controllers in the SDN infrastructure of an enterprise and can be a shared capability among them as well as the management capability for these functions. It enables to present a unified and logically centralized view of the controllers in the SDN infrastructure as is indeed required by the current OpenFlow specification (Open Networking Foundation, 2015).
   The DPM capability has been well researched (Lymberopoulos, 2004) (Damianou, 2002) (Strassner, 2004) (Lymberopoulos et. Al 2003)), tools have been developed (Damianou et. al, 2002) (Damianou et. al, 2001), and standards have been specified (Jun Bi et. al, 2015) (IETF, Policy Working Group, 2021). All this can be potentially leveraged for SDN controllers when the policy-based approach is adopted.

There are multiple PDPs in a system just as there are multiple controllers in an SDN infrastructure, though they logically represent themselves as a single PDP/Controller. A PDP can manage multiple devices just as an SDN controller can manage multiple switches. Inversely, a device can be managed by multiple PDPs; for example, a PDP that manages network traffic, another PDP that manages Security, and yet another PDP that manages quality of service. Each specialized PDP would be populated with correspondingly specialized digital policies. Similarly, there can be multiple sub controllers, within a

logically centralized SDN controller, that will specialize in managing traffic, security, and quality of service, etc.

The PDPs are organized in a hierarchy (Choudhary, 2005) to represent the progressively higher levels of information abstractions. Similarly, the SDN controllers can be placed in a hierarchical topology.

We draw attention to Cisco OpenFlex protocol (Smith et. al, 2015) . It uses policies which seems a sound approach. However, it also is potentially in competition with the OpenFlow protocol upon which SDN is based, and this could dilute the momentum for the SDN.

Using the ideas presented in this section, the next section develops a policy-based architecture for SDN controller.

Policy Based Architecture for SDN Controller

The hierarchy of PDPs in a policy-based system (PBS) (Choudhary, 2005), as discussed in the previous section, is used to formulate a distributed and hierarchical architecture for SDN controller, shown in Figure 3.

A similar research is also the subject of a Master's Thesis (Koshibe, 2013). Figure 3 shows five hierarchical layers. The five hierarchical levels correspond to the five general levels of information detail that the controller might support.

At the highest level is the information needed to provide a common operating picture (COP) for the entire network; for examples, a theatre of war operations, or the state of national critical infrastructure. The controller at this hierarchical level supports the type of digital policies that are commensurate with the common operating picture requirements of an enterprise. Policy rule sets will be formulated (Choudhary, 2011) to construct this information and display it for human managers.

The next lower level corresponds to the regional or segmental situation awareness (SA). The information at this level is like the COP layer but it is more granular and detailed. Corresponding digital policies incorporate the SA requirements, and generate the results for the human managers.

The next lower level addresses the information needs specific to the service that the controller supports. For instance, the security management requires a different type of data and rules compared with the needs of a Differential Services controller for Quality of Service. Similarly, a controller for the cloud computing services requires a different type of digital policies information than the network management controller for a specific device configuration. Results from this layer, and the lower layers, are used in detailed operations in machine-to-machine context. They are also aggregated to produce the SA and COP information for human managers.

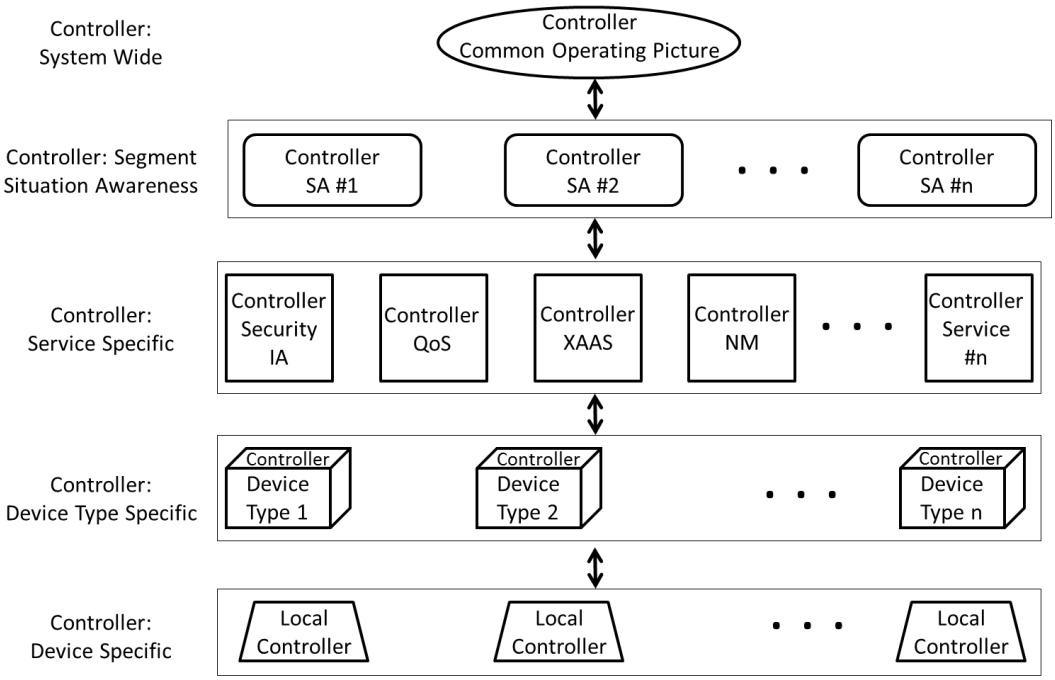

Figure 3 SDN Controller Architecture

The next lower hierarchy corresponds to the device type that the controller manages. For instance, the edge switch requires a different type of information than an interior switch, and a Juniper router has different configuration settings than a Cisco router. The controller needs to incorporate the correspondingly detailed datasets, device information models, and the policy rulesets.

Finally, we have the lowest level of hierarchy, namely the local controller. This corresponds to the concept of a local PDP in a policy-based management system. Even though it may be collocated with the device, it is part of a distributed controller that specializes in handling the most frequently needed device queries, such as the device configuration settings and mapping the policy actions on to device behavior configuration parameters. Its use can overcome the communication delay issue

(Zhang Wen-bo et. al, 2010).

Controllers at all levels of hierarchy are populated with appropriate digital policy rulesets. The needed policies are created, deconflicted, distributed, deployed and activated by the DPM capability. Once activated, they are evaluated and executed only as needed by the nature of the real-time traffic that flows through the SDN.

The current OpenFlow Switch specification (Open Networking Foundation, 2015) uses only one Controller and therefore does not address the organization among the controllers, or the interfaces between them, though such organization and interface are needed (Jun Bi et. al, 2015)

(Benamrane et. al, 2017).

This problem of missing protocol for the North-South interfaces is potentially resolved when the controller is implemented as a PBS. There are south and northbound interfaces within the hierarchy. Using southbound interfaces, the higher-level controllers manage the lower-level controllers, and the lower-level controllers use the northbound interfaces to query the higher-level controllers for guidance. In the language of PBS, the lower-level PDP is managed by the upper-level PDP, and therefore, in this context, the lower-level PDP actually behaves like a PEP with respect to the higher-level PDP. It is a subtle point. The communications between the upper-level PDP and the lower-level PDP therefore can potentially use the COPS protocol which is designed for communications between the PDP and the PEP.

This is a special circumstance whereby a PDP behaves like a PEP with respect to another PDP which is at a higher hierarchical level within the architecture. The more conventional PEPs exist in our architecture at the interface between the SDN controller and the switch. At this interface, the switch is virtualized and implemented in the software for the PEP that populates the interface. This is where the NFV technology is potentially used and deployed in our approach. The reader should appreciate the extent to which the NFV technology is called for. It is called for at every interface where a switch is controlled by the SDN controller. That means we must virtualize every type of switch that the controller manages. Examples are the switches from different vendors, switches at the border, switches in the interior, and special purpose switches that incorporate specialized requirements for security, quality, priority, and preemption etc. When used for the North-South interfaces within the SDN controller, this virtualization means that NFV technology is used to virtualize security, quality of service, etc., as well as all kinds of protocol governed behaviors within the PDPs.

Controller Digital Policies

A digital policy is IF (Condition) THEN (Action) type rule. It consists of three parts: (a) the condition part that specifies the event or the set of events of interest that are used to trigger the policy action; (b) the logic used to evaluate the IF condition and THEN action rules; (c) and the execution of the policy action. Details of the architecture are contained in the digital policies that populate the controllers. The controllers at higher hierarchical level are populated with policies that embody higher levels of abstraction for the operational goals. For example, the common operations picture and the situation awareness controllers use policy rules that specify the business goals; while the local controllers contain the nitty gritty of the device operations and corresponding configuration parameter settings.

The user programs in the OpenFlow controller serve as the policy logic to deduce policy actions from policy events. For instance, the logic can be an algorithm, a pattern matching scheme, and a community of distributed autonomous artificial intelligence agents.

We will discuss below the events that the OpenFlow switch can report to the controller for the purpose of evaluating the policy conditions; we will also discuss the policy actions that the controller can command to the switch.

For specificity, we will do the illustrations in the context of a scenario that detects a security anomaly and responds to it.

What the opening of the switch does for service applications is simply to provide read and write access to specified switch parameters. The switch provides this access to the controller using the OpenFlow

protocol. For the anomalous behavior security scenario that we will use in this section, the following parameters in an OpenFlow switch are of particular interest. These are counters that the switch maintains.

1. *32-bit reference counter in the per flow table*
2. *32-bit duration counter in the per flow entry*
3. *64-bit received packets counter per port*
4. *64-bit transmitted packets counter per port*
5. *32-bit duration counter per port*
6. *64-bit transmitted packets counter per queue*
7. *32-bit duration counter per queue*
8. *32-bit duration counter per group*
9. *32-bit duration counter per meter*

The above 9 counters are required by OpenFlow protocol (Open Networking Foundation, 2015). There are additional 31 counters that are left optional in the current version 1.5.1 but some of them may become required in future versions. The counters can be used both in the policy condition evaluation as well as in the policy action decision logic. The switch provides these parameters to the controllers to which it is directly connected. All controllers can use this information, including the local controllers.

Information is also gathered from other controllers in our distributed and hierarchical architecture. Internal interfaces and East-West interfaces (not shown in Figure 3) are used to gather this information which is used in the following ways.

1. Local controller can use the information for switch configuration.
2. The higher-level controllers evaluate the information by applying their respective policies and deriving action decisions.
3. The information is aggregated over switches in the SDN infrastructure. The aggregated information is used to evaluate SA and COP policies to understand if the enterprise goals are being met.

The counters are useful information to monitor for tell-tale signs of abnormal traffic behavior, i.e., an anomaly. For instance, if the reference counter in the per flow table rolls over unusually fast, it may indicate unusual flow condition and may deserve a closer look for a possible anomaly. Closer scrutiny is made possible by the detailed level counters at the packet level. The monitoring results shall be correlated at the packet level, flow level, meter level, and group level. The results are used by the digital policies deployed within a controller.

For the security scenario that we are using in this section, the controller logic will define three things: the normal behavior of the network, the expected deviations from the normal behavior, and the anomalous behavior. Digital policies for the three cases will be formulated, stored in policy repository, distributed to the controllers, deployed and activated, and policy actions executed. Most of this can happen automatically by the use of DPM capabilities.

Detection Policies

The anomaly detection policies define what the normal behavior is for the traffic through various switches. In particular, these policies will define the counter states during normal operations. A normal

behavior specifies parameters like the following: (a) most likely behavior for each counter; (b) ranges of expected variations over the likely behavior for each counter; (c) correlative behavior of various counters; (d) co-occurrences of related counter behaviors.

Controller digital policies will use this information. They suspect an anomaly; identify the anomalously behaving flow(s); identify the path(s) taken by the anomalous flow(s); and quantitatively estimate as to how far apart is the suspicious behavior from the normal behavior.

To confirm the attack the policies use estimates like the following: (a) The extent to which the anomaly deviates from the normal behavior and the expected variations over the normal behavior. (b) The contextual location of the anomalously behaving switch may be in the safe or unsafe zone from a security point of view. For instance, it may be located within an enterprise's confidential area or it may be publicly accessible. (c) The alert events reported by one switch may or may not correlate with the alert events reported by other switches. For instance, when the campus switch reports a flow reference counter status alert at the same time a WAN switch may issue meter duration counter alert. (d) The campus switch alerts and the WAN switch alerts are viewed within a larger context of situation awareness for the entire enterprise region; for instance, the context provided by a raised national security threat level; and the context of threat alert conditions in other parts of the network. The decisions are performed in an integrated information setting; for instance, if the threat conditions are already elevated the concern for false positive determination may be lowered in favor of security concerns.

Action Policies

The type of policy action would depend on the type of detected attack, the targets of the attack, the activated security policies, and the SA and COP conditions. For example, the action could be taken at the target machine or at the border switch; it could be at the packet level, flow level, interface level, the switch level, or at the level of a conglomeration of switches. The policy-based implementation of the controller can support a wide range of actions. The only limitation is the type and granularity of action sets supported by the relevant switches, and extent and agility of switch reconfigurability.

The database of relevant switch configurations and the related parameters is known to the controller and its policies.Here we like to caution about the security posture of OpenFlow protocol itself. The protocol does not implement its own security and there are potential vulnerabilities in the specification. For example: (1) OpenFlow recommends to use TLS connection; however, TLS is supported but not mandated. (2), the protocol allows the switch to use self-signed controller certificates or use pre-shared key exchange to authenticate the entities; and (3) the protocol uses IPv6 headers that can have their own vulnerabilities (Choudhary, 2009). While we use a security scenario to demonstrate the operations within our proposed architecture, we do not dwell on the security posture of the OpenFlow protocol because that is not the focus of this paper.

Meeting SDN Challenges

The policy-based controller approach presented in this paper can meet major SDN challenges (Horvath et. al, 2014), (Horvath et. al, 2015). These include: (1) the network latency which can cause timeouts in carrying out the controller directives for the switch and the receipt of the requested guidance by the switch from the controller; (2) the missing protocols for the inter-controller communications that are needed for the North-South as well as the East-West interfaces within the logically centralized but

topologically distributed controllers; (3) the size estimation for the backend database which determines the storage and retrieval performance as well as the timeliness of the deployment of digital policies; (4) Scalability of the system to handle the number of switches, different types of expertise that the controller needs to provide, different types of switches that the controller needs to manage, and different types of user requirements for operations; (5) Automation of operations so that switch communications can be automatically provisioned and reconfigured; (6) Evolution of operations to respond to the changes in the business intent for the operations, substantial changes in the operational rules and scenarios, future proofing with evolvability of the system configurations and operational scenarios, and promoting business intent based networking.

Network Latency

If a flow does not match the Flow Table of a switch, the following actions may be necessary (Open Networking Foundation, 2015).

Switch stores a copy of the packet.
Switch sends a copy to the controller.
Controller modifies the Flow Table in the switch.
Controller modifies the Flow Tables in all the switches downward along the flow.

These actions are performed over the network using OpenFlow channel. The network latency can impact the timely reconfiguration of the switches. To address this, the architecture distributes (Schmid and Suomela, 2013) the component controllers appropriately 'near' the switches to minimize the network latency. The time sensitive decisions can be delegated to the local controller to the extent feasible.

Inter-Controller Communications

Current OpenFlow specification (Open Networking Foundation, 2015) does not address controller to controller interaction (Benamrane et. al, 2017) because the specification assumes just one centralized controller. We have hierarchy among the controllers that classifies the controller by their functions and the level of abstractions of the network information. This clarifies the objectives of communications between the controllers, and the type of information that is communicated.

The south and northbound interfaces communicate between layers below and above in hierarchy. Our implementation as a policy-based system makes each controller like a PDP. Details of inter controller communications correspond to inter PDP communications. Within the PDP hierarchy, the higher-level PDP manages the lower-level PDP which, in this context, behaves rather like a PEP. Therefore, much of the COPS specification (Durham et. al, 2000) can potentially be reused.

Backend Database

How large is the SDN backend database and how does it operate? This is difficult to answer. However, when we implement the controller as a policy-based system, we know that the backend database corresponds to the policy storage and retrieval database within the DPM system. These elements are familiar from the work of the policy working groups at IETF (IETF, Policy Working Group, 2021).

Scalability

Our architecture is distributed and hierarchical and can scale both horizontally and vertically. Horizontally, we can easily introduce new device types and new services, etc. Vertically, we can introduce new levels of information abstraction, etc.

Automation

Most of the operations can be automated because they are policy managed, and policy functions are automated using DPM functions.

Evolution

The policy-based systems can gracefully evolve in terms of new operational requirements and new services to be offered. That is because the controller functions can be changed by tweaking policy parameters, without changing any lines of code in the controller. If the code is also allowed to be changed, the evolution capabilities grow indefinitely.

Conclusions

According to Gartner, like other hypes in technology, SDN has also ebbed. That is largely because of problems in SDN architecture. This paper seeks to realize the SDN potential, combined with newer initiatives like network functions virtualization (NFV) and business intent-based networking (BIBN). For that purpose, this paper has leveraged the technology of policy-based systems to architect a logically centralized but implementation wise distributed and hierarchical SDN controller. The approach allows a reuse of a substantial body of research and IETF specifications already performed for PBS. For example, the Digital Policy Management functions facilitate automation of SDN infrastructure management. This solves one of the nagging challenges posed by large ensembles of cloud computing, Internet of things, and Big Data projects. Other challenges, such as scaling and performance, are addressed via the hierarchical and distributed aspects of the architecture. To assess the extent to which the enterprise goals are met, the SA and COP policies produce valuable metrics. The issue of communication delays is addressed by the use of local controllers and also by a suitable deployment topology.

Acknowledgment

Author acknowledges support from Choudhary Associates who specialize in R&D in policy-based systems and cybersecurity.

References

Benamrane F., BenMamoun, M., Benaini R. (2017, January). New Method for Controller-to-Controller Communication in Distributed SDN Architecture, International Journal of Communication Networks and Distributed Systems January.

Benamrane, F., Ben mamoun, M., Benaini, R. (2017, January). New Method for Controller-to-Controller Communication in Distributed SDN Architecture, International Journal of Communication Networks and Distributed Systems.

Choudhary, A. R. (2004, June). Policy Based Management, Bell Labs Technical Journal, Vol. 9, No. 1.

Choudhary, A. R. (2005, November). Digital Policies as a Tool for Transformational Communications, Proceedings of the IASTED conference on Communications, Internet and Information Technology (CIIT), held at Cambridge MA, during Oct 31-Nov 2.

Choudhary, A. R. (2007). Policy Based Management: Deployment Challenges in the GIG, Proceedings of the Military Communications Conference, October 29-31, Orlando, FL, USA.

Choudhary, A. R. (2009, November). In-depth analysis of IPv6 security posture, 5th Conference on Collaborative Computing: Networking, Applications and Worksharing, held in Washington D.C., USA, November 11-14.

Choudhary, A. Rahim (2011, November). Policy Rule-Sets for Policy Based Systems, Proceedings of the IEEE (UK) 6th International Conference for Internet Technology and Secured Transactions, Abu Dhabi, UAE, November 11-14.

Damianou M., Dulay N., Lupu E., Sloman M. (2001, January). The Ponder Policy Specification Language, Workshop on Policies for Distributed Systems and Networks, January 29-31, Bristol, UK.

Damianou N. (2002, February). Policy Framework for Management of Distributed Systems, Department of Computing, Imperial College, University of London, UK, Ph.D. Thesis.

Damianou N., Bandara A., Sloman M., Lupo E. (2002). A Survey of Policy Specification Approaches, Department of Computing, Imperial College, University of London, UK.

Durham D. (ed), Boyle J., Cohen R., Herzog S., Rajan R., Sastry A. (2000, January). The COPS (Common Open Policy Service) Protocol", IETF RFC 2748.

ETSI (2021, January). ETSI ISG NFV: Work Program and Releases Overview.

Goransson P., Chuck B. (2014). Software Defined Networks – A Comprehensive Approach, ISBN: 978-0-12-416675-2, Morgan Kaufmann Publishers.

Horvath R., Nedbal D., Stieninger M. (2015). A Literature Review on Challenges and Effects of Software Defined Networking, Procedia Computer Science V.64, p. 552-561.

IBM (2020, January). MIB Types and Objects,

https://www.ibm.com/support/knowledgecenter/en/SSGU8G_14.1.0/com.ibm.snmp.doc/ids_snmp_050.htm (retrieved August 24, 2021).

IETF (Internet Engineering Task Force), Network Working Group (2012, October). Internet Draft draft-mrw-sdnsec-openflow-analysis-00.txt, Security Analysis of the Open Networking Foundation (ONF) OpenFlow Switch Specification.

IETF, Policy Working Group (2021, retrieved August 24). Core policy framework specification documents https://datatracker.ietf.org/wg/policy/documents/

Jun Bi, Rafiee, H., Choudhary, V., Strassner, J., Ramascanu D. (2015, May). Simplified Use of Policy Abstractions (SUPA) Gap Analysis, IETF Internet Draft, https://datatracker.ietf.org/doc/draft-bi-supa-gap-analysis/?include_text=1, May 19.

Koshibe, A. (2013). The Design and Evaluation of a Hierarchical OpenFlow SDN Control Plane, Master Thesis, Rutgers, State University of New Jersey, http://dx.doi.org/doi:10.7282/T3KW5D23

Lymberopoulos L. (2004, October). An Adaptive Policy Based Framework for Network Management, Department of Computing, Imperial College, University of London, UK, Ph.D. Thesis.

Lymberopoulos L., Lupo E., Sloman M. (2003). An Adaptive Policy-Based Framework for Network Services Management, Journal of Network and Systems Management Vol. 11 No. 3.

McKeown, N. et.al. (2008, April). OpenFlow: Enabling Innovation in Campus Networks, ACM SIGCOMM Computer Communications Review, Vol. 38, #2, p. 69-74.

Open Networking Foundation (2015, March). ONF TS-025, OpenFlow Switch Specification, Version 1.5.1 (Protocol version 0x06).

Paliwal M., Shrimankar D., Tembhurne O. (2018, June). Controllers in SDN: A Review Report, IEEE Access.

Riftadi. M. and Kuipers, F. (2019, June). P4I/O: Intent-Based Networking with P4, 2019 IEEE Conference on Network Softwarization (NetSoft) held in Paris, France, during June 24-28.

Schmid, S. and Suomela, J. (2013, August). Exploiting Locality in Distributed SDN Control, Proceedings of the 2nd ACM SIGCOM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, August 12-16.

Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., and Rao, N. (2014, August). Are we ready for SDN? Implementation Challenges for Software-Defined Networks, IEEE Communications Magazine, CTN Issue.

Smith, M., Adams, R., Dvorkin, M., Laribi, Y., Pandey, V., Garg, P., Weidenbacher, N. (2015, October). OpFlex Control Protocol, IETF Draft, draft-smith-opflex-02, October 19.

Sood K., Karmakar K., Varadharajan V., Tupakula U., Yu S. (2019). Analysis of Policy-Based Security Management System in Software-Defined Networks, IEEE Communications Letters V. 23, p. 612-615.

Stallings, W. (1998). SNMP, SNMPv2, SNMPv3, and RMON 1 and 2, 3rd Edition, ISBN:0201485346, Addison-Weseley.

Strassner, J. C. (2004). Policy-Based Network Management: Solutions for the Next Generation, ISBN 1-55860-859-1, Morgan Kauffman Publishers.

Zhang Wen-bo et. al. (2010, April). A Policy Based Network Management Architecture for Satellite Networks, IEEE International Conference on Information Management and Engineering (ICIME), April 16-18.