Proposed Method for Partial Node Replacement by Software Defined Network

Sawsan Youssef
Brno University of Technology
Faculty of Information Technology
Bozetechova 2, Brno, Czech republic
Email: iyoussef@fit.vutbr.cz

Ondrej Rysavy
Brno University of Technology
Faculty of Information Technology
Bozetechova 2, Brno, Czech republic
Email: rysavy@fit.vutbr.cz

***Abstract*—Since it is impractical to replace the entire traditional network by the SDN network due to some constraints i.e. financial budget, limited skills to SDN, in addition to the need to have the benefits and flexibility of the traditional network, the partial replacement implemented by deploying or replacing some legacy nodes by the SDN switches have emerged. Such replacement requires routing and security addressing coordination issues. In this research, we present our proposed solution for automatic replacement of a segment of the legacy network by SDN nodes, and generation of a set of OpenFlow rules and switches configuration that meets the traditional network behavior requirements. The rules are identified based on the analysis of the network traffic acquired from the legacy segment.**

I. Introduction

The main concept of the Software-Defined Network is the separation of the control plane from the data plane; such a separation allows the operator to insert new functions in the network, increasing the flexibility and the programmability of the network. In the traditional systems opposed to the SDN, the forwarding devices run control functions such as the forwarding decision and the path discovering algorithms, maintain the network state, etc. The function of individual network devices is reprogrammed, and the devices together run distributed algorithms for routing and security policy enforcement. On the other hand, the SDN concept considers that the network devices have only basic functionality necessary for packet forwarding, and the network functionality is composed of the set of network applications executed mainly by network controllers.

Due to several constraints of the full replacement of the traditional network by the SDN nodes, the incremental deployment of SDN is often considered leading to the hybrid network containing the conventional IP network and SDN network. Such a gradual installation of the SDN nodes, smooth the migration toward the SDN networks, and take benefits from the two kinds of systems. Still, on another side, the partial deployment of SDN switches faces several challenges; one of them is the consistency between the protocols and policies in the whole network. Because of different devices and rules control packet forwarding, the hybrid SDN must be configured to provide consistent routing and security policies for different network segments.

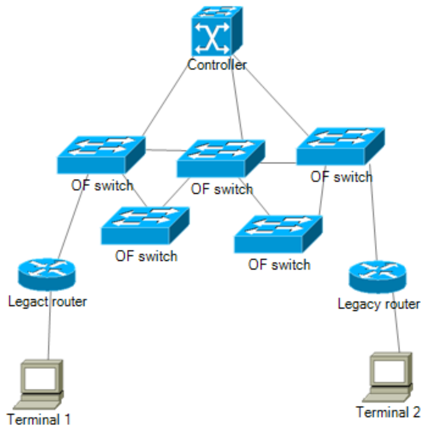

Fig. 1. Hybrid SDN network topology

OpenFlow preinstalled rules play an essential role in mitigating the flooding of the undefined packets and reducing the time needed to make the rule decision about the traffic, in addition to minimize the signaling overhead with the controller. This paper is organized as follows: section II presents the related work followed by proposed framework in section III, then conclusion and future work are presented in section IV.

II. Related Work

The replacement of the IP infrastructure by OpenFlow switches strategy is a different subject on the objective to be achieved, for example; the SDN nodes could be deployed among traditional switches and behave like virtual IP nodes, e.g., Cardigan [1]. Conversely, the grouping of the IP nodes in a VLAN could be achieved to make IP VLAN controlled by the SDN controller, e.g., HybNet [2]. The third type is to create two island solutions, and every island is controlled by its functions, but such a solution requires a translation between the two types of nodes, e.g., B4 [3].

Such a migration toward Hybrid SDN should preserve the original policies of the network like Routing and End-To-End policy. One of the researches targeted the problem of creating

SDN rules during migration from IP to HSDN networks is project Exodus [5], where Cisco router configurations are used to produce the corresponding rules for the HSDN network. Another approach for SDN migration is presented in B4 [3], where a logic replacement of the BGP border router by the SDN switches is presented and proxy is added for communication between the two segments .

Such frameworks did not ensure the routing or End-To-End policy of the original IP segment, in addition to the need to properly analyze and parse the configuration of each type of network device possibly from different vendors.

An essential method for ensuring the SDN End-To-End policy is One-Big switch that is presented in [4], such an approach was implemented by using the equivalence classes (set of packets that are manipulated in same manner) of the SDN switched forwarding rules to create non-overlapping rules of One-Big switch, the main limitation is there is no insuring of routing policy inside the SDN segment. The generation of SDN rules by a packet trace example was implemented in NetEgg tool [6]. Still, such an approach is used for creating only forwarding rules in pure SDN networks without a discussion about the filter or rules replacement. All these approaches analyzed the existing SDN forwarding rules by taking a static snapshot of the network rules or data plan.

We found a lack of upgrading strategies toward Hybrid SDN. Previous solutions for migration from IP to SDN, such as Exodus [5] and Telekinesis [7], consider translating the static cisco configuration files into SDN rules involving a lot of complexity because of different vendors and configuration languages, and these approaches do not provide verification of the produced rules concerning the original configuration, and because of the limited capacity of OpenFlow table; it is not practical to translate the configuration without rule minimizing or rules scheduling. In this paper we will address the migration of traditional network to SDN network by analyzing the traffic data acquired from normal training of network.

In the proposed research, the objective is to analyze and assess the transformation toward Hybrid SDN networks (an example in Fig. 1) considering routing and security policy. The main goal is to develop a method for creating a Hybrid SDN configuration based on profiling the network behavior of the traditional network and creating routing and security policy models.

III. PROPOSED FRAMEWORK:

In this section, we present the network model to generate the SDN rules (output) from IP network communication (input); the intermediate traffic analysis model that captures IP network behavior is created first. After that, the model, which is the source of the exploration procedure that makes SDN rules will be presented.

A. Problem Formalization

The entire network is represented as a directed graph: $B\left(N,L\right)$ where $N \subseteq IP$ is a set of the nodes in the network topology represented, and $L \subseteq NXN$ is a set of edges in the network that refer to the connections between nodes.

The specific network segment that will be replaced by SDN nodes is represented as: $G\left(V,E\right)$ where $V \subseteq N$ and $E \subseteq L$. Routers are represented as a non-intersected subset of IP. The router is thus defined by IP addresses assigned to its interfaces.

For all routers R_i and R_j, the following must hold: R_i, R_j $\subseteq IP$ and $R_i \cap R_j$ = $\emptyset$ or R_i = R_j. Let IP represents the set of all Internet addresses, this set thus also contains addresses assigned to router interfaces. (Table I presents the main items of the model.)

TABLE I
FORMAL MODEL SYMBOLS.

Symbol	Description
F	Flows that are traversed in the network
S	Switches in the network
En	The endpoints of the network
C_s	Capacity of the SDN switch
$P_{i,j}$	$s_x, s_y, ..$ path form i to j.
$r_{i,j}$	Single rule.
$m_{i,j}$	Match fields of the rule $r_{i,j}$.
$d_{i,j}$	Decision field of the rule.
$v_{i,j,k}$	Test if the $r_{i,j}$ is placed on the switch s_k .
RI	Router local interfaces
EP	Endpoints of network (Hosts/Networks)
RN	Router immediate neighbors IP addresses
RL	Router links with neighbor interfaces

Forwarding Table FT of the switches in the resulting SDN segment consists of the records describing the observed traffic of the sector before replacement.

For every router R, the observed traffic is represented as: OT_R = $< InIface, SrcIP, DstIP, Proto, OutIface >$ where $InIface, OutIface \in IfaceR$, $Proto \in ProtoType$, $SrcIP$ and $DstIP \in IP$, and $IfaceR = if_1$, if_2, $...if_n$ is the set of router interfaces. Each interface has assigned IP address, thus $if_i \in IP$.
The set of the protocols are represented as $ProtoType = Tcp, Icmp, Udp, Igmp$.

B. Proposed framework

The proposed solution is divided into several logical steps:

1) Traffic Collection: In order to collect traffic in the legacy network, all routers were NetFlow enabled (ingress and egress monitoring on all interfaces), we chooses NetFlow because of its feature to present the ingress and egress port of the incoming flow.
2) Feature Extraction: From the collected network traffic, we need to select such features that are substantial for routing and security models. The main features for extracting the paths, topology, filters, and forwarding rules are: source IP, Destination IP, protocol type (in case of taking QoS under consideration), in addition to input interface and output interface. This information will be used to extract the paths and forwarding rules of the flows.

Algorithm 1 Broadcast Records Isolation

```
    Input : $OT$
    Output :$RN$, $D_1$, $RI$
1: For all records from $D$:
2: if $DstsIP \in BroadcastIP$ then
3:     Add $SrcIP$ to $RN$
4:     Add $InIface$ to $RI$;
5: else
6:     Add $record$ to $D_1$;
7: end if
   Add $RN$ to $V$;
```

3) Topology Extraction: In this work, the proposed network topology discovery method, depends on the existence of flows of control and routing protocols.

TABLE II
BROADCASTING FLOW EXAMPLE.

SrcIP	DstIP	InIface	OutIface	Proto
192.168.60.1	255.255.255.255	2	0	Udp

For instance, routing protocols such as RIP, EIGRP, and OSPF allow a router to discover other adjacent routers on its local links and networks (see Table II). The vital information used in the detection of neighbor devices is the presence of broadcast or local multicast packets.
To extract routers interfaces IP addresses and directed neighbors; the proposed algorithm is represented:

- Extracting Broadcast/Multicast records as a starting point to determine the immediate neighbor IP addresses (see algorithm 1 and 2) . Connections sourced from the neighbors will be analyzed to derive the Router local interface IP address and Interfaces number and the direct links. If the endpoints do not send periodic packets to prove its presence, then they will not appear in the router discovery topology and step 2 will manipulate with such case.
- The router endpoints will be concluded from records that contain interface numbers that are not discovered by step 1. To extract such connections, we perform the following algorithm (See algorithm 2): The records that contain new values of $InIface$ or $OutIface$ not listed in the existing router local interfaces RI; will be analyzed and new interface number will be added to RI, and the $SrcIP$ or $DstIP$ will identifies the endpoint address.
- The filters to drop specific flows will be explored from the NetFlow traffic records which have the field $OutIface$ is 0, and it is not targeted a local interface of the router.

4) OpenFlow Rules Extraction: In order to replace the IP network area with an equivalent SDN segment, the forwarding rules need to be generated for the SDN switches.
The SDN forwarding table contains OpenFlow rules $r_{i,j}$ represent the rules between the source i and destination j that consist of a match condition $m_{i,j}$ and an action $D_{i,j}$: forward to output port (if 0 it mean: drop the packet), as depicted on the example in Table III.

Algorithm 2 Extract the Neighbor Connections

```
    Input : $RN$, $V$, $D_1$, $RI$ ;
    Output: $D_2$, $V$, $EP$, $RL$, $Router-Interface-Table$,
$Router-Link-Table$.
    For all $record$ from $D_1$:
2:  if $OutIface = 0$ and $srcIp \in RN$ then
        Add $DstIP$ to $V$;
4:      Add <$SrcIP, DstIP$> to $RL$;
        Add <$DstIP, InIface$> to $Router - Interface - Table$;
6:      Add <$srcIP$, $InIface$> to $Router - Link - Table$
    else
8:      Add $records$ to $D_2$ ;
    end if
10: For all the records from $D_2$:
    if $Inface \notin RI$ then
12:     Add $SrcIp$ to $EP$;
        Add $InIface$ to $RI$;
14:     Add <$SrcIP$, $InIface$> to $Router - Link - Table$;
    else { IF $record$ where $OutIface \notin RI$}
16:     Add $DstIp$ to $EP$;
        Add $OutIface$ to $RI$;
18:     Add <$DstIP$, $OutIface$> to $Router - Link - Table$;
    end if
20: Add $EP$ to $V$;
```

TABLE III
OPENFLOW RULE FIELDS.

Router/switch	**Match $m_{i,j}$**	**Action $D_{i,j}$**
R	INPORT=InIface, SourceIP= SrcIP DestinationIP=DstIP	Forward to OutIface

The direct approach for OpenFlow rules generation from NetFlow records is to set the Match expression as: <$SrcIP, DstIP, InIface$>, and the action set to forward to the $OutIface$. Thus, the traffic that matches some existing NetFlow record is forwarded, and the other traffic is dropped. The filters at the edge routers of the SDN block will be determined during the traffic analysis. We assume that the SDN switch has one FlowTable conforming to OpenFlow specification 1.0, the first stage of the rule extraction supposed to be exact match (exact rule for every flow).

5) Rule Optimization: Because of the OpenFlow memory limitation, it is not possible to keep similar number of forwarding rules in each switch as the traditional switch,

or even to store all access policy on one edge switch or even One-Big switch, the rules should be optimized and distributed after compression without violation of the policy, and the number of rules in every switch should be less than the switch capacity:

$$\forall s \in S : \sum_{i,j} v_{i,j,s} \leq c_s \tag{1}$$

Where the $v_{i,j,s}$ is a Boolean test if the $r_{i,j}$ is installed in the switch s, and C_s is the capacity of switch FlowTable.

6) Evaluation: To compare the end-to-end behavior of the original network with the hybrid SDN, the same traffic patterns that are used in the traditional network will be sent again in the hybrid network and will be stored in a matrix, the matrix contains the result of applying function $Reachability(i, j)$ to test the reachability between the source i and destination j, and the difference between the matrix-es before and after replacement will be checked. The network behavior should not be violated in the resulting network. The reachability will be identified as:

$$Reachability(i,j) = \begin{cases} 0 & \text{Flow was dropped} \\ 1 & \text{Flow was delivered} \end{cases} \tag{2}$$

The set of the reachable switches that ingress flow can reach should be the same as before the replacement.

C. Current Status

Several examples in the virtual environment on top of eve-ng tool were implemented, containing routers and endpoints (Virtual PCs). The traffic was generated by using an Ostinato traffic generator. All routers were NetFlow enabled (ingress and egress monitoring on all interfaces). For collecting the NetFlow traffic, the following components should exist:

- NetFlow collector on dedicated server(s). I used NfDump tools installed on Ubuntu server 16.4 (8 GB RAM, HDD 500 GB).
- NFSEN: NFSEN is a graphical web based front end for the Nfdump NetFlow tools.

So far, the experiments were done for routing protocols RIP and OSPF, and for different kinds of network configurations (with/without ACLs).

In this paper, we have discussed the safe migration from traditional IP network to HSDN architecture, it is necessary to ensure that newly introduced SDN blocks will interoperate with the rest of the system. One of the most fundamental interoperability problems is to provide coherent routing and security. In One-Big-Switch [4] the destination packet header is analyzed to extract the equivalence class (to obtain the forwarding graph and the one-big switch forwarding rules). The input port and the protocol type as well are not considered in their solution which could minimize the network provision; our model will use the (input port of the packet) to distinguish the paths and the filters will be detected at the edge switches in addition to create hop-by-hop configuration.

IV. Conclusion and Future work

In this paper, the partial replacement of traditional networks by SDN network was addressed, the network behavior represented by the forwarding rules and the end-to-end policy were discussed. Our proposed framework relied on traffic analysis to describe the network behavior (topology and the forwarding rules), such forwarding rules will be used to create the SDN switches rules, this will be the base for checking the violation and illegitimate access to the hybrid SDN. The main future work is complete demonstration of the model, and propose a method for optimizing the rules (compression and distribution) inside the SDN segment.

Acknowledgment

This work has been supported by project FIT-S-20-6293: Application of AI methods to cyber security and control systems.

References

[1] J. Stringer, D. Pemberton, Q. Fu and C. Lorier , R. Nelson and J. Bailey et al., "Cardigan: SDN distributed routing fabric going live at an Internet exchange", *IEEE Symposium on Computers and Communications (ISCC)*, 2014, pp. 1-7.

[2] L. Hui and A. Nipun and Z. Hui and L. Cristian and R. Junghwan and J. Guofei, "HybNET: Network Manager for a Hybrid Network Infrastructure", *Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference*, USA, 2013, pp 1-6.

[3] J. Sushant and K. Alok and M. Subhasree and O. Joon and L. Poutievski, "B4: Experience with a Globally-Deployed Software Defined Wan", *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, 2013, pp 3-14.

[4] Y. Jiaqi and L. Xin and J. Dong, "Simulation of a Software-Defined Network as One Big Switch", *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 2017, pp 149-159.

[5] T. Nelson, A. Ferguson, D. YuU, R.Fonseca , S. Krishnamurthi, "Exodus: toward automatic migration of enterprise network configurations to SDNs", *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, 2015, pp. 1-7.

[6] Y. Yuan, D. Lin and S. Anil and H. Verma and A. Chelluri and R. Alur and B. T. Loo,"NetEgg: A Scenario-Based Programming Toolkit for SDN Policies", *IEEE/ACM Transactions on Networking*, vol.26, n.5, 2018, pp 2104-2117.

[7] C. Jin, C. Lumezanu, Q.Xu, Z-L.Zhang, and G.Jiang,"Telekinesis: Controlling legacy switch routing with OpenFlow in hybrid networks", *(Symposium on Software Defined Networking (SDN) Research, SOSR* , 2015.

[8] X. Nguyen, D. Saucez, C. Barakat and T. Turletti, "Rules Placement Problem in OpenFlow Networks: A Survey", *in IEEE Communications Surveys and Tutorials*, Second quarter, vol. 18, no. 2, 2016, pp. 1273-1286.