Scientific Bulletin of the "Petru Maior" University of Tîrgu Mureş
Vol. 13 (XXX) no. 1, 2016
ISSN-L 1841-9267 (Print), ISSN 2285-438X (Online), ISSN 2286-3184 (CD-ROM)

# Optimal Switch Configuration in Software-Defined Networks

**Béla GENGE**[1], **János SZTRIK**[2]
[1]*"Petru Maior" University of Tîrgu Mureş*
[1]*Nicolae Iorga Street, No. 1, 540088, Tîrgu Mureş, Romania*
[1]e-mail: bela.genge@ing.upm.ro
[2]*University of Debrecen*
[2]*Egyetem tér, No. 1, 4032, Debrecen, Hungary*
[2]e-mail: sztrik.janos@inf.unideb.hu

## Abstract

*The emerging Software-Defined Networks (SDN) paradigm facilitates innovative applications and enables the seamless provisioning of resilient communications. Nevertheless, the installation of communication flows in SDN requires careful planning in order to avoid configuration errors and to fulfill communication requirements. In this paper we propose an approach that installs automatically and optimally static flows in SDN switches. The approach aims to select high capacity links and shortest path routing, and enforces communication link and switch capacity limitations. Experimental results demonstrate the effectiveness and scalability of the developed methodology.*

**Keywords**: Software-Defined Networks, OpenFlow, integer linear programming

## 1 Introduction

An emerging paradigm in traditional IP networks is the replacement of local router-based decisions with global routing decision software. A prominent enabler of this trend is OpenFlow, a protocol designed to ensure remote access to the forwarding pane of a network switch [1]. OpenFlow separates control from forwarding, enabling the implementation of more complex traffic management techniques. Besides OpenFlow, a key advancement in the field is the Software-Defined Networks (SDN) paradigm. SDN provides the means to create virtual networking services and to implement global networking decisions. SDN relies on OpenFlow to enable communications with remote devices and it is considered to revolutionize the way decisions are implemented in switches and routers.

Despite its indisputable advantages, it should be noted that SDN is an emerging paradigm and its understanding, benefits, but most importantly its disadvantages require careful examination. While several studies have revealed that SDN may indeed enhance the resilience of communication networks, especially in the industrial sector [2, 3], the provisioning of communication flows in SDN switches requires careful planning to avoid configuration errors and to fulfill communication requirements. To this end, communication flows, hereinafter called simply *flows*, may be subject to various requirements including Quality of Service (QoS) pertaining to real-time packet delivery, security requirements, e.g., the presence of intrusion detection systems (IDS), reliability of communication paths. Besides these aspects, the provisioning of communication flows needs to account for the hardware limitations of communication lines and SDN switches. These may significantly limit the capacity of links, and the maximum number of rules that may be installed in SDN switches.

This paper alleviates the aforementioned issues by developing an approach that automatically and optimally install flows in SDN switches. The approach leverages an optimization problem that aims to install flows on the highest capacity links and using the shortest path routing. Subsequently, the problem encapsulates several constraints that ensure that capacity limitations are satisfied.

Besides these aspects, the methodology embraces

Python software modules that call the Solving Constraint Integer Programs tool to derive an optimal solution, and to configure the SDN switches by means of the Floodlight SDN controller.

The remaining of this work is structured as follows. Section 2 provides a brief overview of related studies. Section 3 presents the developed SDN switch configuration methodology, while Section 4 provides the experimental results. The paper concludes in Section 5.

## 2 Related work

Several recent studies demonstrated the benefits of SDN-enabled communication infrastructures and identified key challenges in adopting this emerging technology. Yonghong Fu *et al.* [7] developed Orion, a hybrid hierarchical control plane for large-scale networks. Orion defines three planes: the domain physical network, the tier 0 control plane consisting of area controllers, and the tier 1 control plane consisting of a distributed set of domain controllers. Tuncer *et al.* [8] developed an SDN-based management and control framework for backbone networks. The approach followed a hierarchical and modular structure to support large-scale topologies and the simple integration of various management applications. The work of Tuncer also proposed a network planning algorithm based on the uncapacitated facility location problem. In [9] the authors developed Dionysus, a system for consistent network updates in SDN. Dionysus builds the graph of network update dependencies and schedules these updates by taking into account the performances of network switches. To eliminate packet losses [10] proposed zUpdate, a solution that uses packet labeling for zero packet losses during network updates. In comparison to these works, this work places an emphasis on the optimal configuration of SDN networks. It is aimed at delivering a starting point, which may then be adopted in the development of more complex network design problems.

## 3 Developed approach

This section presents the developed SDN configuration methodology. It starts with an overview on the developed methodology and it continues with a description of the network configuration problem and of the developed software.

### 3.1 Overview

The architectural overview of the methodology developed in this paper is depicted in Figure 1. As shown in this figure, the methodology comprises of three main components. First, the SDN configuration software, i.e., the main contribution of this work, which glues together the remaining components. This tool

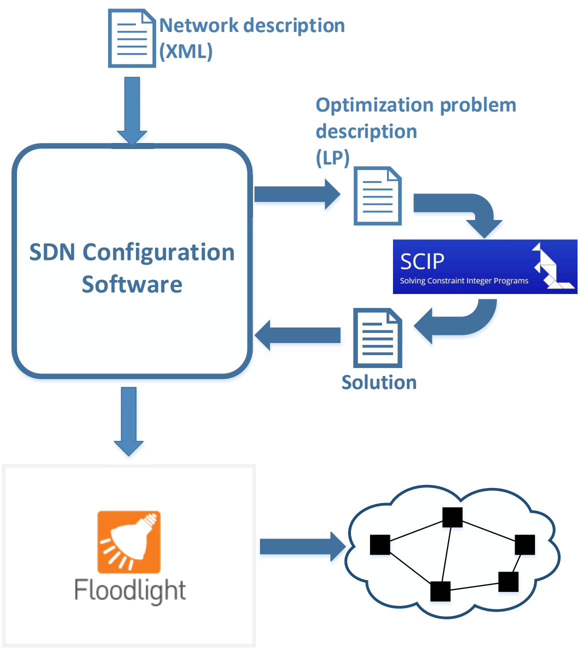
Network description (XML)
Optimization problem description (LP)
SDN Configuration Software
SCIP
Solving Constraint Integer Programs
Solution
Floodlight

**Figure 1. Architectural overview of the developed SDN configuration methodology.**

takes a network description given in XML format and automatically generates a Linear Programming (LP) description of the network design problem. Once a solution is provided by the external Solving Constraint Integer Programs (SCIP) tool [6], the SDN configuration software takes the solution and sends the new network topology to the Floodlight controller [5]. In turn, the Floodlight controller uses the OpenFlow protocol to install static flows on SDN switches.

### 3.2 Network model

We assume a demand matrix of flows routed between access and egress switches. The routing needs to be performed in such a way to reduce communication delays by means of selecting the shortest paths and the largest capacity links. Flows are assumed to be non-bifurcated multicommodity flows such that each flow can only be routed on one path.

The undertaken approach installs flows across an SDN pool of switches by means of an SDN controller, e.g., Floodlight. Flows are installed on each switch as *static flows*, which means that the routing rule for each flow does not change dynamically. Accordingly, it is important to identify the level of granularity for the definition of a communication flow, in order to deliver an effective strategy against malicious traffic. For example, by defining a flow between IP addresses `10.1.1.0/24` and `10.2.1.0/24` all the IP-based protocols, e.g., UDP, TCP, will be permitted and routed across the SDN between all the hosts located in the two networks. However, in the case that an attacker initiates a new flow between two hosts located in the aforementioned networks, this malicious traffic will also be permitted. Consequently, a more appropriate configuration would define a communication flow incorporating the source and destination IP addresses, the source and destination MAC addresses, and the protocol type. This would reduce the set of hosts from which the attacker could generate a new communication flow.

Obviously, in the case that the attacker has sufficient knowledge on the network, he/she could exploit the permitted protocols and their vulnerabilities for achieving his/her goals. Therefore, it needs to be noted that the developed methodology represents a first level of defense against attackers. Nevertheless, previous work demonstrated that flow whitelisting can be an effective countermeasure for blocking specific malware features. More specifically in [11] an experiment was conducted with the Stuxnet malware [12, 13]. A network with four hosts running Windows XP SP2 was configured and one of the hosts was deliberately infected with the Stuxnet malware. Stuxnet exploited vulnerability MS08-067 in the SMB protocol (used in file sharing) to copy itself onto another station. In the case that this SMB traffic is whitelisted, then Stuxnet can successfully replicate itself. However, [11] also showed that once it infected a host, Stuxnet also started to initiate connections towards `www.windowsupdate.com` and `www.msn.com` (to test Internet connectivity). In the case that this traffic is not whitelisted, it is blocked and consequently, Stuxnet (together with other similar malware) will not be able to contact its Command and Control servers.

## 3.3 Configuration problem

The network configuration problem's parameters are depicted in Figure 2. We define $I$ to be the set of flows and $J$ to be the set of switches. Let $d_i$ denote the demand of flow $i$ and $u_{jl}$ the capacity of link $(j,l)$, where $j,l \in J$. We assume that if switches $j$ and $l$ are not connected, then $u_{jl} = 0$. Then, let $x_{ij}^A$ be a binary parameter with value 1 if the access end-point of flow $i$ is connected to switch $j$, and $x_{ji}^E$ a binary parameter with value 1 if the egress end-point of flow $i$ is connected to switch $j$. Then, we define $s_j$ to be the capacity of switch $j$ in terms of the maximum number of supported forwarding rules that may be installed. Lastly, we define $t_{jl}^i$, a binary variable with value 1 if flow $i$ is routed on link $(j,l)$. The solution of the problem will set the values of $t_{jl}^i$ in the case that flow $i$ is selected for routing between switches $j$ and $l$ (see Figure 3).

The problem's objective is to select the shortest routing path for each flow, while selecting the links

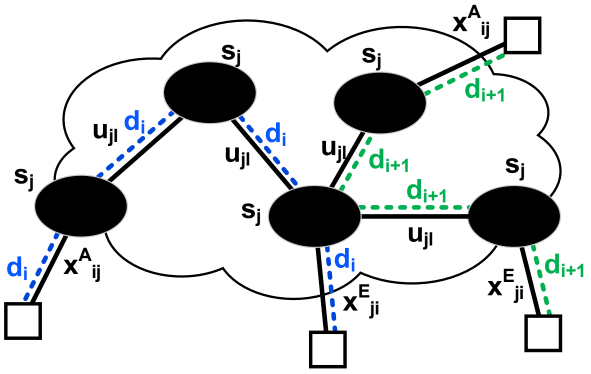

**Figure 2. The optimization problem's parameters.**

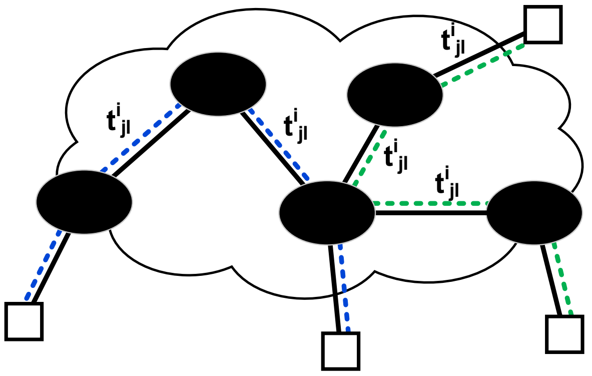

**Figure 3. The optimization problem's variables.**

with the largest capacities:

$$\min \sum_{j,l \in J} \left( \frac{1}{u_{jl}} \sum_{i \in I} d_i t_{jl}^i \right). \quad (1)$$

The optimization is subject to the following constraints. Constraints (2) denote classical multicommodity flow conservation constraints:

$$x_{ij}^A - x_{ji}^E - \sum_{l \in J} \left( t_{jl}^i - t_{lj}^i \right) = 0, \quad \forall j \in J, i \in I. \quad (2)$$

Equations (3) are capacity constraints used to ensure that the link capacity is not exceeded:

$$\sum_{i \in I} d_i t_{jl}^i \le u_{jl}, \quad \forall j,l \in J. \quad (3)$$

Lastly, we define the switch capacity constraints to ensure that the maximum number of forwarding rules installed in switch $j$ does not exceed the switch capacity:

$$\sum_{i \in I, l \in J} t_{jl}^i + \sum_{i \in I} x_{ji}^E \le s_j, \quad \forall j \in J. \quad (4)$$

**Table 1. Network design problem solve time.**

| Switches | Flows | Time[ms] |
|---|---|---|
| 5 | 20 | 12 |
| 10 | 20 | 21 |
| 20 | 20 | 35 |
| 20 | 50 | 84 |
| 20 | 100 | 231 |

**Table 2. Communication flow installation time.**

| Flows | Time[ms] |
|---|---|
| 5 | 16 |
| 20 | 51 |
| 80 | 176 |

### 3.4 SDN configuration software

The SDN configuration software is written in the Python language and comprises of three main modules. The *NetworkTopology* module processes the network description provided as an XML file and it builds an internal representation of the network topology. The *ModelSolver* generates an LP description of the network design problem, it calls the external SCIP tool, and it processes the solution. Lastly, the *Controller* implements Floodlight's REST API and sends to the external controller the flows that need to be installed on the SDN switches.

## 4 Results

We performed several tests in order to assess the performance of the developed approach. Accordingly, we measured the time for solving the optimization problem and the time for installing flows. The tests have been performed on an emulated network topology recreated with the Mininet network emulator [4] on Ubuntu LTS 14.04.3 64-bit OS, and a host with Pentium Dual Core 3.00GHz CPU and 4GB of memory.

At first, we measured the time in which SCIP solved the network design problems of various sizes (see Table 1). Accordingly, for a network topology including 5 switches and 20 flows the solver executes in 12ms. This increases up to 21ms for 10 switches and to 35ms for 20 switches. On the other hand, by setting a fixed number of switches of 20 and by increasing the number of flows to 50, we also measure a more increased solve time of 84ms. By further increasing the size of the problem up to 100 flows, we measure a 231ms solve time. Next, we measured the flow installation time (see Table 2). Accordingly, for 5 flows the installation time is completed in 16ms, for 20 flows in 51ms, while for 80 flows in 176ms. These results show that the network problem solve time and the flow installation time exhibit a linear behavior, which is significant evidence of the scalability of the developed methodology. Nevertheless, it should be noted that the approach may be further extended with additional constraints and parameters in order to capture more particular aspects of communication networks from different domains.

## 5 Conclusion

We developed a methodology to automatically and optimally configure SDN networks. The approach accounts for the selection of maximum capacity links, for the selection of shortest routing paths, for the capacity and the limitations of links and SDN switches. As a result, the developed methodology can save time and it can assist the configuration of SDN networks. Nonetheless, the network design problem should be seen as a starting point for other and more complex network configuration problems. Accordingly, the methodology and more specifically, the developed optimization problem, may be extended with additional parameters and constraints in order to embrace the particularities of different scenarios and domains.

## Acknowledgment

This research was supported by the Hungarian Academy of Sciences under grant no. 6611/10/2015/HTMT.

## References

[1] N. McKeown, T. Anderson, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J.Turner, OpenFlow: Enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review, vol. 38(2), pp. 69–74, 2008.

[2] A. Goodney, S. Kumar, A. Ravi and Y.Cho, Efficient PMU networking with software-defined networks, Proceedings of the Fourth IEEE International Conference on Smart Grid Communications, pp. 378–383, 2013.

[3] E. Molina, E. Jacob, J. Matias, N.Moreira and A. Astarloa, Using software-defined networking to manage and control IEC 61850 based systems, Computers and Electrical Engineering, vol. 43, pp. 142-154, 2015.

[4] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz and N. McKeown, Reproducible network experiments using container-based emulation, in Proceedings of the 8th International Conference on

Emerging Networking Experiments and Technologies, (New York, NY, USA), pp. 253-264, ACM, 2012.

[5] Project Floodlight, http://www.projectfloodlight.org/floodlight/ [Online; accessed April 2016].

[6] T. Achterberg, Scip: solving constraint integer programs, Mathematical Programming Computation, vol. 1(1), pp. 1-41, 2009.

[7] Y. Fu, J. Bi, Z. Chen, K. Gao, B. Zhang, G. Chen, and J. Wu, A hybrid hierarchical control plane for flow-based large-scale software-defined networks, IEEE Transactions on Network and Service Management, vol. 12, pp. 117-131, June 2015.

[8] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, Adaptive resource management and control in software defined networks, IEEE Transactions on Network and Service Management, vol. 12, pp. 18-33, March 2015.

[9] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford and R. Wattenhofer, Dynamic scheduling of network updates, SIGCOMM Computer Communications Review, vol. 44, pp. 539-550, August 2014.

[10] H. H. Liu, X. Wu, M. Zhang, L. Yuan, R. Wattenhofer and D. Maltz, zupdate: Updating data center networks with zero loss, SIGCOMM Computer Communications Review, vol. 43, pp. 411-422, August 2013.

[11] B. Genge, D.A. Rusu, and P. Haller: A Connection Pattern-based Approach to Detect Network Traffic Anomalies in Critical Infrastructures. 2014 ACM European Workshop on System Security (EuroSec2014), Amsterdam, The Netherlands, pp. 1-6, 2014.

[12] M. Hagerott: Stuxnet and the vital role of critical infrastructure operators and engineers. International Journal of Critical Infrastructure Protection, vol. 7(4):244-246, 2014.

[13] B. Genge, F. Graur, and P. Haller: Experimental Assessment of Network Design Approaches for Protecting Industrial Control Systems, International Journal of Critical Infrastructure Protection, vol. 11, pp. 24-38, 2015.