Apoorv Shukla, André Schütze, Arne Ludwig, Szymon Dudycz, Stefan Schmid, Anja Feldmann

# Towards Transiently Secure Updates in Asynchronous SDNs



# Towards Transiently Secure Updates in Asynchronous SDNs

Apoorv Shukla
TU Berlin, Germany
apoorv@inet.tu-berlin.com

André Schütze
TU Berlin, Germany

Arne Ludwig
TU Berlin, Germany
arne@inet.tu-berlin.de

Szymon Dudycz
Uni Wroclaw, Poland
szymon.dudycz@gmail.com

Stefan Schmid
Aalborg University, Denmark
schmiste@cs.aau.dk

Anja Feldmann
TU Berlin, Germany
anja@inet.tu-berlin.com

## ABSTRACT

Software-Defined Networks (SDNs) promise to overcome the often complex and error-prone operation of traditional computer networks, by enabling programmability, automation and verifiability. Yet, SDNs also introduce new challenges, for example due to the asynchronous communication channel between the logically centralized control platform and the switches in the data plane. In particular, the asynchronous communication of network update commands (e.g., OpenFlow FlowMod messages) may lead to transient inconsistencies, such as loops or bypassed waypoints (e.g., firewalls). One approach to ensure transient consistency even in asynchronous environments is to employ smart scheduling algorithms: algorithms which update subsets of switches in each communication round only, where each subset in itself guarantees consistency. In this demo, we show how to change routing policies in a transiently consistent manner. We demonstrate two algorithms, namely, WAYUP [5] and PEACOCK [4], which partition the network updates sent from SDN controller towards OpenFlow software switches into multiple rounds as per respective algorithms. Later, the barrier messages are utilized to ensure reliable network updates.

## CCS Concepts

•**Networks** → *Network reliability; Programmable networks;*



*SIGCOMM '16 Florianópolis, Brazil*



ACM ISBN 978-1-4503-4424-1..
DOI: 10.475/123_4

## Keywords

SDN; Mininet

## 1. INTRODUCTION

While Software-Defined Network (SDN) outsources and consolidates the control to a logically centralized software controller, SDN still needs to be regarded as an asynchronous distributed system: the installation of new policies as well as the update of existing policies need to be communicated from the control plane to the dataplane elements (switches, universal nodes, appliances, etc.). This transmission is performed over an asynchronous and unreliable network, and hence, updates may actually take effect out of order.

Based on the update scheduling algorithms WAYUP and PEACOCK presented in [5] and [4] respectively, we divide the policy updates from SDN controller to OpenFlow soft switches into rounds. Each round culminates in the SDN controller sending the barrier requests to each OpenFlow switch receiving updates in that round and receiving barrier replies for acknowledgement. Once done with receiving all barrier replies, the SDN controller initiates the next round of policy update. Therefore, synchronicity and consistency is ensured avoiding out of order updates and ensuring waypoint enforcement [5], weak loop freedom [4]. More work on multiple policies can be found on [1][3].

## 2. PROTOTYPE

We have implemented the WAYUP [5] and PEACOCK [4] algorithms[1] successfully in *Mininet* based on a Python-based Ryu SDN controller. and have been running our evaluations with respect to the update time of flow tables in OpenFlow switches (OVS)[2]. Here, our focus is

[1]For details about the algorithm, we refer to reader to the cited papers.
[2]In multi-vendor hardware switches, this demo might

on the implementation challenges. The existing app "ofctl_rest.py" is used as a basis for the implementation of a new Ryu app called "ofctl_rest_own.py". The original SDN controller app provides functionality to do network updates consisting of a single round OpenFlow message sent from SDN controller to OpenFlow switch. In this demo, we will use multiple round updates to show waypoint enforcement to ensure the networks are transiently secure without security being compromised. We implement the app "ofctl_rest_own.py", which provides the ability to create a message queue at the SDN controller side to enqueue the REST messages in a message queue for each round of network update as per the WayUp algorithm. All messages save the update schedule and the OpenFlow messages in the message object and therefore, every round of the update schedule is processed in the same way. If the SDN controller starts to process a message, it begins with the first round, which is set to be the current round. In the current round, there are a set of switches which have to be updated. The SDN controller retrieves the corresponding OpenFlow message for every switch (OVS) in the set and sends them out to the switches. Later, the SDN controller sends a barrier request to every switch of the set and waits for barrier replies. For every barrier reply received by the SDN controller, it determines the source switch. This switch is removed from the set of switches of the current round of the first message in the message queue. If the set is empty, the current round finishes and the SDN controller goes on to process the next round of the messages. If the message object does not have a next round, the SDN controller deletes the message from the queue and starts processing the next message. Here is an example of a REST message:

{
*"oldpath":[<dp-num>,<dp-num>,<dp-num>],*
*"newpath":[<dp-num>,<dp-num>,<dp-num>],*
*"wp":<dp-num>,*
*"interval":<time in ms>,*
*<type>:[<OpenFlow message information>],*
*<type>:[<OpenFlow message information>],*
*<type>:[<OpenFlow message information>],*
*<type>:[<OpenFlow message information>],*
}

The WayUp REST request consists of a header part and a body part. The header part consists of the input parameters of WayUp. These are the old route, the new route, the waypoint, and the time interval. In Ryu, the switches which are connected to the controller, are identified by integer values called *datapaths*. The waypoint is a string, which can be converted to an integer value and the old and new route are strings, which can be converted to a list of integer values. The integer values are ordered in the list in the way they are passed by the network packets along the route. For example, if in the route at first switch 2 is passed, then switch 1 and then switch 3, so the path would look like: $\langle 2, 1, 3 \rangle$. In the body part of the REST message, we can find information about the OpenFlow messages that the SDN controller has to send to the switches. In the common case, a WayUp REST request consists of several OpenFlow messages. For example, an OpenFlow message in the body part could be sent to the URL: `http://<controller-address><controller-port>/stats/flowentry/add` of the SDN controller and the controller would add the flow entry to the OpenFlow switch specified in the flow entry. The type field is used to specify type of FlowMod.

face problems as mentioned in [2]. Therefore, this demo is just about the asynchronicity of the control channel.

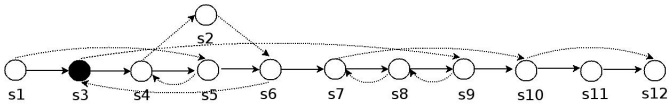


**Figure 1: Example Topology. Solid lines show the old routing policy and the dotted lines show the new routing policy. Black Node s3 is the waypoint**

As shown in Figure 1, the test setup for transiently secure network updates tool consists of 12 nodes or OpenFlow (OVS) switches with host h1 connected to switch 1 and host h2 connected to switch 12 in mininet. Node/switch 3 is the waypoint, e.g., Firewall or IDS. The edges having a solid line, build the old route through the network. The edges having a dashed line, build the new route through the network.[3] The source code can be found at : `https://bitbucket.org/Apoorv1986/transiently_secure_code`

**Acknowledgements** — The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement No. 619609 (project UNIFY).

## 3. REFERENCES

[1] S. Dudycz, A. Ludwig, and S. Schmid. Can't touch this: Consistent network updates for multiple policies. In *IEEE/IFIP DSN*, 2016.

[2] M. Kuzniar, P. Peresini, and D. Kostic. What you need to know about sdn flow tables. In *PAM*, 2015.

[3] A. Ludwig, S. Dudycz, M. Rost, and S. Schmid. Transiently secure network updates. In *ACM SIGMETRICS*, 2016.

[4] A. Ludwig, J. Marcinkowski, and S. Schmid. Scheduling loop-free network updates: It's good to relax! In *ACM PODC*, 2015.

[5] A. Ludwig, M. Rost, D. Foucard, and S. Schmid. Good network updates for bad packets. In *ACM HotNets*, 2014.

[3]The link to the video of the prototype is at http://tinyurl.com/zf4v7qo