

IMPROVING TRAFFIC CONTROL USING SOFTWARE DEFINED NETWORKING FOR INTER-DOMAIN ROUTING

Shpëtim LATIFI, Betim ÇIÇO

South East European University, Tetovo, Macedonia

Arjan DURRËSI

Department of Computer and Information Science, Purdue University
School of Science, Indiana University Purdue University Indianapolis,
Indianapolis, United States of America

ABSTRACT

Software Defined Networking (SDN) is becoming a de facto paradigm in data networks. While smaller campuses and enterprises are not yet widely deploying SDN, larger campuses and data centers are already utilizing the advantages of SDN, primarily taking advantage of easier network management, better resource utilization, better load and traffic balance on network devices and routes. Denial of service has been another concern for the last decades, and in the current internet it has not been fully addressed. The present paper analyzes and tests possible applications of Software Defined Internet Exchange Point (SDX) for inter-domain routing. The results from simulated implementation showed that SDX as a platform offers a greater flexibility and allows for better traffic engineering than conventional techniques. In addition, it gives a possibility to fight denial of service attacks.

Keywords: Software Defined Networking, SDN Controller, IXP, SDX, SDX switch

1. INTRODUCTION

Monsanto *et al.*, (2013) said that SDN was created in response to demands from large data centers, which faced problems coping with very unpredictable traffic patterns. These patterns would cause high demands for particular resources that could not be met with existing network infrastructures. Practically, two solutions were viable in such scenarios: to either scale the infrastructure to meet the peaks, which is expensive and the resources would stay underutilized most of the time, or build the network so it can reconfigure itself to meet the appropriate demands.

SDN is a programmatic (Foster *et al.*, 2011) method where the customer can alter the network to meet the peaks and demands when rapid changes happen, so that social networking sites and large data centers (Stephens *et al.*, 2012) that are dispersed in different locations are able to meet the specific demands.

It offers many benefits, among which better network control and management, a possibility to build logical network overlays and thus manage all the network switches (from a centralized controller rather than distributed (Yazici, 2012; Yaganeh *et al.*, 2013) following the end-to-end path, and not dealing with individual switches at a time. This allows for easier traffic engineering, management and security. Techniques for easier prevention of Denial of Service attacks can be proposed in SDN, as it is explained in the section below.

SDN has also opened a way to the research community and third party vendors to offer enterprise specific solutions, based on consumer needs, thus bringing the prices of switches down (Erickson, 2011; Casado *et al.*, 2012).

For the research community the biggest benefit might be the possibility to rapidly prototype simulations/emulations (Koponen, 2010; Handigol *et al.*, 2012; Gupta *et al.*, 2013) of SDN network environments using real world parameters (data rates, size and number of nodes, means for analysis and troubleshooting) before they are applied on test beds or real networks, and compare its advantages to conventional non-SDN networks. Also, open-source platforms like POX (Gude *et al.*, 2008), and NOX (Gude, 2008), offer the possibility to write control applications for Software-Defined Networking controllers, such as the OpenFlow controllers. OpenFlow has gained a huge deal of use in Virtual Machines, but also in mission critical networks. Google's internal network now runs completely on OpenFlow (B4, Google's SDN-powered WAN, provides connectivity among data centers) (Greenberg *et al.*, 2009; Foster *et al.*, 2013). Traffic includes asynchronous data copies, index pushes for interactive serving systems, and end user data replication for availability. Well over 90% of internal application traffic runs across this network.

As a reminder, Google's global user based services (Open Networking Foundation, 2011), (Google Web Search, Google+, Gmail, YouTube, Google Maps, etc.) require significant amount of data to be moved from one region to another, making these applications and services very WAN-intensive. Google concluded that the delivery of such services would not be scalable with the current technologies due to their non-linear complexity in management and configuration. As a result, Google has decided to use SDN for managing WAN as a fabric as opposed to a collection of boxes. Hence, the future of SDN is already here, and the transition for most data center and large networks has begun. Figure 1 depicts the transition (starting and target network outlooks) as described by the Open Networking Foundation.

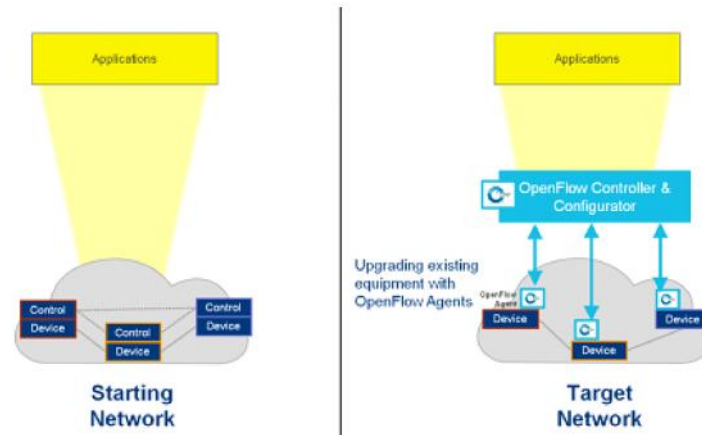


Fig. 1. Starting network outlook (left) and target network outlook (right) of major networks of data centers and campuses. The picture is courtesy of ONF.

The following section of this paper outlines the practical applications related to implementing an SDN in inter-domain routing and advantages, compared to conventional inter-domain routing.

II. Analysis of SDN application in inter-domain routing

SDN changes the way we manage networks, datacenters, backbones, enterprises, etc. Most of these are applications of SDN are used in internal networks, enabling software technologies such as network virtualization, traffic engineering, etc. It is interesting to view how SDN relates and how it fundamentally changes how traffic is delivered in between independently operated networks. BGP (border gateway protocol) lacks sufficient flexibility for performing various traffic engineering and traffic management operations. It's generally possible to route only on destination IP address blocks, and it is only possible to exert indirect control over how switches and routers forward traffic, through local preference, etc. But generally it is difficult to introduce new network services, such as ability to arbitrarily route traffic flows through middleboxes.

There are a variety of wide-area services on the other hand, and it would be extremely valuable to network operators had there been a way to implement them. One possibility is application specific peering, or providing the ability to autonomous systems to exchange traffic only for specific applications, such as video streaming. Another application that could be significantly enhanced is the ability to block denial-of-service traffic, such as the ability to block traffic in upstream autonomous systems, much closer to the source. Other applications

include direct client request to different data centers, depending on where they are originating, steering through network functions, and various inbound traffic engineering options, such as splitting incoming traffic over multiple incoming peering links.

SDN allows routers and switches to match packets on multiple header fields rather than just their destination IP address. It also allows a network controller to control entire networks with a single program, possibly even including remote autonomous systems, as opposed to just immediate neighbors, and provides mechanism for direct control over packet handling rather than indirect control, via various routing protocols. It also allows and gives the ability to perform many different actions on packets, beyond simply just forwarding them.

Given the previously given and known SDN capabilities, SDN deployment in inter-domain routing can be applied at internet exchanges (IXP). An SDN deployment even at a single IXP can benefit tens of hundreds of providers, and this is possible without requiring any of the providers to deploy any new equipment. ISPs have also tremendous interest in innovation. Practically, the number of IXPs is growing globally, with some 400 IXPs worldwide nowadays (Lantz *et al.*, 2010; Canini *et al.*, 2012).

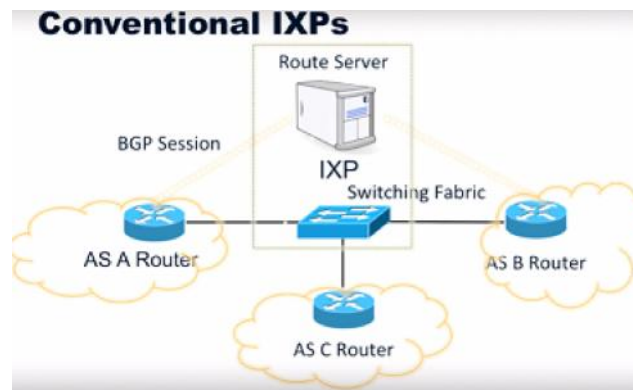


Fig. 2. Conventional IXP layout.

In conventional IXPs (Figure 2), participant autonomous systems exchange BGP routes with a route server. The Autonomous Systems (AS) then make their own forwarding decisions and forward traffic to each other over a common layer 2 switching fabric. The primary function of a route server in an IXP is to avoid having each of these autonomous system make pairwise connections with one another. Instead, all of them can communicate directly with the route server. So, the idea of Software Defined Inter-domain exchange point (SDX), (Gupta *et al.*, 2014) is to turn a route server into a smart SDN controller, and make the layer L2 switch an SDN capable switch that can take commands

from the SDN controller (Erickson, 2011), concerning the forwarding table entries that one can have installed for each participant.

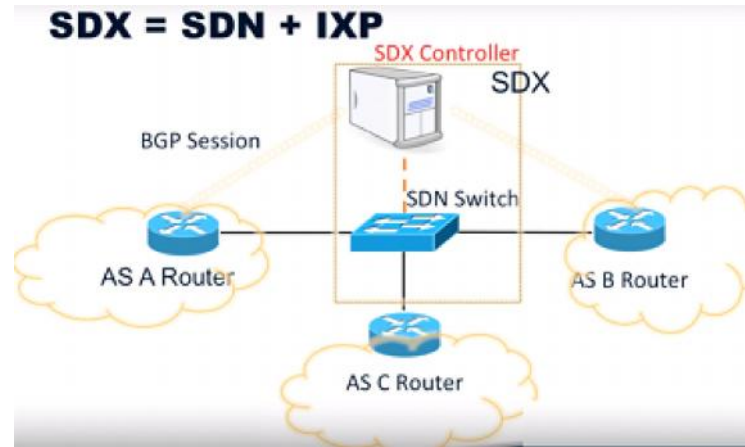


Fig. 3. SDX layout.

This seemingly simple change enables a wide variety of new applications. One can be prevention of Denial of Service attacks upstream. In Figure 3 we have used AS C with two ingress routers at the exchange point. Let's suppose that AS C want to ensure that port 80 traffic enters its network via interface C1. This is very difficult to do with BGP, where you are not able to forward traffic in custom ways, based on specific fields in packet headers. On the other hand, an SDN enabled switch in an IXP makes it possible for a participant to install a rule that pertains only to a specific portion of packet header (this application however is not implemented in this paper).

III. PREVENTING DENIAL OF SERVICE (DOS) ATTACKS IN SDN

Denial of Service attacks are one of the most significant and serious problems currently facing the Internet users. For many users these attacks are an irritation, even if they understand the reason for the poor performance they occasionally observe. But for the Internet to achieve its full potential, it has to be able to offer highly reliable services, even in the face of hostility. The biggest DoS problem is caused by distributed denial of service (DDoS) attacks, where the attacker compromises a large number of systems and then uses these as zombie systems to attack the victims. DDoS attacks of sufficient scale provide the firepower needed to overwhelm almost all victims. They can also be combined with spoofing or reflection to make the attack even more difficult to defend against. Currently most DDoS attacks do not bother to spoof the source addresses

because, as no automatic push-back mechanism is widely deployed, it takes so long to shut down each zombie that there is no need to hide their identity (Handley and Greenhalgh, 2004; Durrësi *et al.*, 2006).

DDoS is principally an issue due to widespread exploitation of software vulnerabilities, which permit the control of large numbers of compromised systems. To gain sufficient scale, such exploitation is typically automated using worms, viruses, or automated scanning from already-compromised hosts (so called “bots”). Fast-spreading worms are extremely hard to combat in the current Internet Architecture. Although viruses and bots are a serious issue, their spread rate is slower, which permits a wider range of defense options (Handley and Greenhalgh, 2004). A number of solutions have been proposed in the last over 10 years to fight DDos, and the fact that this is still an interesting and live topic, helps conclude that DDoS is still an issue in the current Internet, and therefore not fully addressed.

In SDN there is the ability to block denial-of-service in a significantly enhanced way, such as to block traffic in upstream autonomous systems, much closer to the source. For example if AS 1 is receiving Denial of Service attack traffic from AS 3 (Figure 4), the victim AS 1 can install drop rules potentially at multiple SDN enabled exchange points, thus preventing this traffic much closer to the source (section III gives the code we used to install a drop rule at an exchange point).

In comparing to conventional defenses, SDX-based DDoS defense allows for an AS to remotely influence the traffic by remotely installing drop rules further upstream, and these rules can be more specific than in standard Access Control rules. For example these drop rules can be based on multiple header fields, such as source address, destination address, port number, etc. Furthermore, these drop rules can be coordinated across multiple IXPs therefore making it difficult for attack traffic to enter via any path.

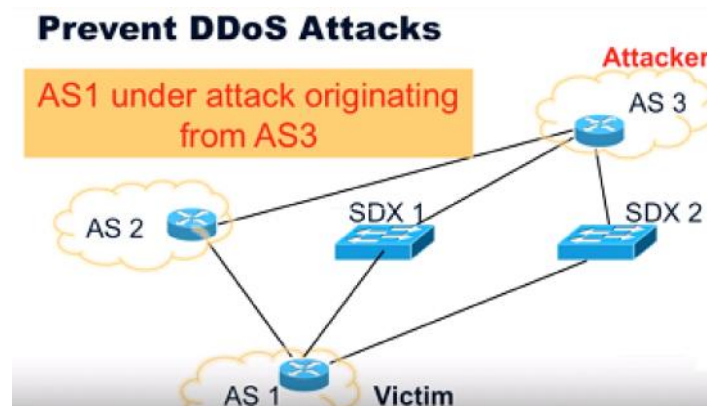


Fig. 4. SDX preventing attack.

IV. SDX program

In order to run experiment on testbed, or simulate it, users will need a set of tools. We have used a software router called Quagga (both Quagga and Mininext are SDX dependencies) in our case. Previously users need to have installed Mininet '(Mininet)' on Linux. To get Quagga, type:

```
$ sudo apt-get install quagga
```

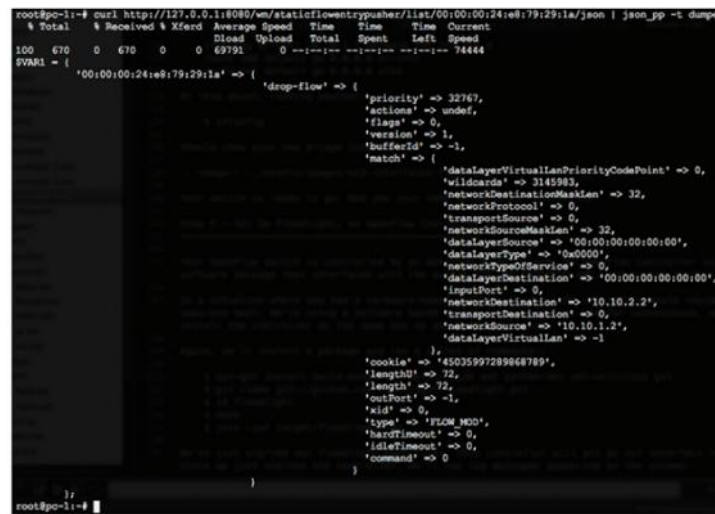
Then users need to install Mininext (which is an extension of Mininet), using:

```
$ sudo apt-get install help2man python-setuptools  
$ git clone https://github.com/sdn-ixp/sdx.git
```

The drop rule installation at the exchange point, given that the topology has been previously configured (Latifi *et al.*, 2014) is done through the following code:

```
$ curl -d '{"switch": "[DPID]", "name": "drop-flow",  
"src-ip": "IP1", "dst-ip": "IP2", "active": "true"}'  
http://127.0.0.1:8080/wm/staticflowentrypusher/json  
  
$ curl http://127.0.0.1:8080/wm/staticflowen  
trypusher/list/[DPID]/json | json_pp -t dumper
```

After applying the rule, we can see that it is applied and effective at the exchange point. This specific rule is given based on destination and source IP address. Note that actually the drop rule can be adjusted and modified based on port number or as a combination of multiple header fields (IP version, etc.).



```

root@pc-1:~# curl http://127.0.0.1:8080/wm/staticflowentrypusher/ist/00:00:00:24:e8:79:29:1a/json | json_pp -t dumper
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           % Done    0     0    0 69791 0 --:--:-- --:--:-- --:--:-- 74444
SVAR1 = {
  "00:00:00:24:e8:79:29:1a" => {
    "drop-flow" => {
      "priority" => 32767,
      "actions" => undef,
      "flags" => 0,
      "version" => 1,
      "bufferId" => -1,
      "match" => {
        "dataLayerVirtualianPriorityCodePoint" => 0,
        "wildcards" => 2145983,
        "networkDestinationMaskLen" => 32,
        "networkProtocol" => 0,
        "transportSource" => 0,
        "networkSourceMaskLen" => 32,
        "dataLayerSource" => "00:00:00:00:00:00",
        "dataLayerType" => "0x0000",
        "networkTypeOfService" => 0,
        "dataLayerDestination" => "00:00:00:00:00:00",
        "inputPort" => 0,
        "networkDestination" => "10.10.2.2",
        "transportDestination" => 0,
        "networkSource" => "10.10.1.2",
        "dataLayerVirtualian" => -1
      },
      "cookie" => "4035997289668789",
      "length" => 72,
      "length" => 72,
      "outPort" => -1,
      "xid" => 0,
      "type" => "FLOW_MOD",
      "hardTimeout" => 0,
      "idleTimeout" => 0,
      "command" => 0
    }
  }
}
root@pc-1:~#

```

Fig. 5. Drop rule installed at exchange point.

V. CONCLUSIONS AND FUTURE WORK

The Internet is changing, and there are new challenges for content delivery and IXPs are playing an increasingly dominant role. SDN lets providers innovate and add new capabilities to the system.

In this paper, we have analyzed results gained from testing new policies for traffic engineering for inter-domain routing using Princeton's SDX platform, which is available. It can be seen that using SDN for interdomain routing proves not only useful, but much more practical for applications such as attack prevention and inbound traffic engineering, with more specific rules applying to specific traffic. In this paper, we tested the suggested robustness of the proposed SDX, as a SDN Internet Exchange Point, by installing drop rules at one exchange in based on source and destination IP address. This proves to be easier, and more straightforward compared to traditional IXPs utilizing BGP.

Based on our conclusions on one hand, and the positive trend in networks towards SDN, we can anticipate that SDN will be around in the future and that as new control and data applications become available, SDN will start to become part of more and more corporate, enterprise and university campuses.

Our future work will rely on current research work to test inter-domain routing applications in the SDN realm, as well as proposed specific solutions based on user needs. We are strongly confident that SDN as an organizing principle of data networks will have a future in major networks with hundreds or thousands (or even more) nodes, therefore based on this, our research will continue by running more exhaustive and different tests on various scenarios, topologies

and data traffic, until we have analyzed thoroughly all aspects of our implementations (Tootoonchian, 2011).

REFERENCES:

- Canini M, Venzano D, Peresini P, Kostic D, Rexford J. 2012.** A nice way to test OpenFlow applications. *Proceedings of USENIX NSDI '12*, April.
- Casado M, Koponen T, Shenker S, Tootoonchian A. 2012.** Fabric: A Retrospective on Evolving SDN. In *HotSDN*.
- Durrësi M, Barolli L, Paruchuri V, Durrësi A. 2006.** Scalable traceback against distributed denial of service. *IJWGS*.
- Erickson D. 2011.** Optimizing a virtualized data center. In *SIGCOMM*.
- Foster N, Guha A, Reitblatt M, Story A, Freedman M, Katta N. 2013.** Languages for software-defined networks. *Communications Magazine, IEEE*, **51(2)**:128–134.
- Foster N, Harrison R, Freedman MJ, Monsanto C, Rexford J, Story A, Walker D. 2011.** Frenetic: A Network Programming Language. In *ICFP*.
- Greenberg A, Hamilton JR, Jain N, Kandula S, Kim C, Lahiri P, Maltz DA, Patel P, Sengupta S. 2009.** VL2: a scalable and flexible data center network. *Proceedings of ACM SIGCOMM '09*, August.
- Gude N, Koponen T, Pettit J, Pfaff B, Casado M, McKeown N, Shenker S. 2008.** NOX: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, **38(3)**:105–110.
- Gude N. 2008.** NOX: towards an operating system for networks, *SIGCOMM CCR*, **38(3)**.
- Gupta A, Vanbever L, Shahbaz M, Donovan SP, Schlinker B, Feamster N, Rexford J, Shenker S, Clark R, Katz-Bassett E. 2014.** SDX: A Software Defined Internet Exchange. *SIGCOMM*.
- Gupta M, Sommers J, Barford P. 2013.** Fast, Accurate Simulation for SDN Prototyping, *HotSDN '13*, August 16, Hong Kong, China.
- Handigol N, Heller B, Jeyakumar V, Lantz B, McKeown N. 2012.** Reproducible network experiments using container-based emulation. *Proceedings of ACM CoNeXT*, 253–264.
- Handley M, Greenhalgh A. 2004.** Steps towards a DoS-resistant Internet Architecture, *SIGCOMM*.
- Koponen T. 2010.** Onix: A distributed control platform for large-scale production networks, *OSDI*.
- Lantz B, Heller B, McKeown N. 2010.** A network in a laptop: rapid prototyping for software-defined networks. *Proceedings of ACM HotNets .10*, page 19.

Latifi Sh, Durrësi A, Çiço B. 2014. Emulating enterprise network environments for fast transition to Software-Defined Networking, MECO.

Mininet, <http://mininet.org/>

Monsanto C, Reich J, Foster N, Rexford J, Walker D. 2013. Composing Software Defined Networks. *Proceedings of USENIX NSDI '13*, April.

Open Networking Foundation. 2012. OpenFlow Switch Specification (Version 1.3.0), June.

POX. <http://www.noxrepo.org/pox/about-pox/>.

SDX, <http://sdx.cs.princeton.edu/>

Stephens B, Cox A, Felter W, Dixon C, Carter J. 2012. PAST: scalable ethernet for data centers, *Proceedings of ACM CoNeXT '12*, 49–60.

Tootoonchian A. 2012. On controller performance in software-defined networks”, in *HotICE*,.

Yaganeh SH, Tootoonchian A, Ganjali Y. 2013. On the Scalability of Software-Defined Networking. *IEEE Communications Magazine*. Feb.

Yazici V. 2012. Controlling a Software-Defined Network via Distributed Controllers. *NEM Summit*.