Implementing Geo-Blocking and Spoofing Protection in Multi-Domain Software Defined Interconnects

Himal Kumar†, Anu Mercian∗, Sujata Banerjee⋆,
Craig Russell⋄, Vijay Sivaraman†
†UNSW, Australia, ∗Hewlett-Packard Enterprise, US, ⋆VMware, US, ⋄CSIRO Data 61, Australia
himal.kumar@unsw.edu.au,anu.mercian@hpe.com,sujata@banerjee.net
Craig.Russell@data61.csiro.au,vijay@unsw.edu.au

Abstract

Motivated by recent attacks like the Australian census website meltdown in 2016, this paper proposes a system for high-level specification and synthesis of intents for Geo-Blocking and IP Spoofing protection at a Software Defined Interconnect. In contrast to todays methods that use expensive custom hardware and/or manual configuration, our solution allows the operator to specify high-level intents, which are automatically compiled to flow-level rules and pushed into the interconnect fabric. We define a grammar for specifying the security policies, and a compiler for converting these to connectivity rules. We prototype our system on the open-source ONOS Controller platform, demonstrate its functionality in a multi-domain SDN fabric interconnecting legacy border routers, and evaluate its performance and scalability in blocking DDoS attacks.

Keywords Geo-Blocking, Security Intents, Internet Exchange Point, Software Defined Networking

ACM Reference format:
Himal Kumar†, Anu Mercian∗, Sujata Banerjee⋆,
Craig Russell⋄, Vijay Sivaraman†. 2017. Implementing Geo-Blocking and Spoofing Protection
in Multi-Domain Software Defined Interconnects . In *Proceedings of XDOMO'17, Belgrade, Serbia, April 23, 2017,* 6 pages.
DOI: http://dx.doi.org/10.1145/3071064.3071065

1 Introduction

Distributed Denial-of-Service (DDoS) attacks on web-based services are escalating — our motivation in this work comes from the recent meltdown of the Australian Census web-site in August 2016 [2], which is believed to have been caused by large-scale off-shore DDoS attacks, leaving millions of Australians unable to access the web-site on census night, and becoming a national embarrassment for the Australian Bureau of Statistics (ABS). When the ABS in response sought to geo-block access to the census site to limit it to Australians, the complexities of manual configuration (white-listing of allowed IP blocks) became apparent, with necessary support services (like DNS) failing and the site being down for several more days. This paper therefore explores the feasibility of automating the process of configuring security, by requiring the user to specify only high-level policies that are dynamically compiled down to enforceable forwarding rules in the data-plane.

In this paper we focus specifically on geo-blocking and IP address spoofing. Geo-blocking is an effective way of restricting access to the service (loosely) to a geographical boundary, thereby limiting the ability of attackers outside this boundary to bring down the service. The boundary can be specified in terms of the country code or AS numbers, which then need to be translated into a set of IP prefixes. This set can be large, and manual configuration of these prefixes into border routers (or special security appliances like PacketViper [10]) can be expensive and error-prone. Further, attackers can spoof IP addresses, making it difficult to identify legitimate from illegitimate traffic. In this paper therefore we also tackle IP address spoofing, using a minimalistic approach that only accepts traffic originating from a domain if that domain advertises that prefix in its BGP control plane. Domains have an incentive to participate in this so as to reduce their liability for (reflection) attacks originating from their network. We believe that this combination of geo-blocking and source IP address spoofing is an effective (though by no means comprehensive) way to protect critical web-services.

The skills required to implement, operate, and maintain security-related access control lists (ACLs) can be prohibitive for small entities (including government agencies), and therefore we advocate that they only specify their security requirements in terms of high-level policies, such as the region they want to geo-block their service to. The natural place to implement the security data filtering is the (public or private) Internet exchange point (IXP), which inter-connects the domain to other domains - the IXP carries both the data-plane and control-plane traffic for all domains, and therefore has both visibility and expertise to execute security intents. Further, the IXP can "scrub" the traffic in its fabric so the malicious traffic does not even need to enter the domain under attack, protecting its access link resources. Recent advances in software defined networking (SDN) technology enable data-plane forwarding rules to be consistent with the control plane updates, enabling the realization of new security capabilities as envisaged in SDX [14]. Our own previous work in [18] has built and deployed an SDN-based multi-domain inter-connect that is operational across six research institutions in Australia – our work in this paper extends the capability of this Australia-wide SDN testbed to include the security use-cases developed in this paper. Our specific contributions are:

- We develop a geo-blocking and spoof-protection solution comprising a grammar for specifying high-level intents and compilation to low-level flow-rules;
- We architect and implement our solution as an SDN application on the ONOS Controller platform, coupled with live BGP information and geo-mapping services;
- We evaluate our system using real hardware and traffic in terms of its correctness, responsiveness, and scalability to a large number of flow-rules.

The rest of the paper is organized as follows: §2 presents relevant background on SDXs, intent frameworks and security at modern IXPs. §3 and §4 talks about the policy grammar and system architecture respectively. Our compiler algorithm is described in §5 and §6 presents IP Spoofing prevention. Our system is evaluated in §7 and paper is concluded in §8.

2 Related Work

SDXs [14] [13] have been used for doing traffic engineering and load balancing. Geo-Blocking is a relatively new topic with few expensive solutions such as Packet Viper [10]. Very few solutions [14] [20] have attempted to leverage SDXs, but have not considered Geo-Blocking intents holistically, which is the novelty of our proposed solution. Most of the IP spoofing solutions like "Network Ingress Filtering" [7] are implemented in a distributed network and can also be reused in a centralized network. Also, black-holing at an exchange point [19] has also been talked about as a way to prevent DDoS attack and implement fine grained policies using Openflow matches in the fabric. Our previous work CASToR [18] have talked about SDX policies to switch traffic between private and public clouds. A recent work Propane [21] talks about automating BGP configurations in the network devices using high level abstract language and BGP security has also been talked about recently in [15]. Also, intent based networking is not new and many intent-frameworks [11] [16] and compilers [12] [17] have been proposed in the past.

3 Geo-Block Policies and Grammar

We develop a high level abstract language specification to easily express Geo-Blocking intents or policies in a human readable language. We develop the following structure of a single Geo-Block policy:

def geoblock (**Policy** Id) {

Source = [**Country** (Codes), **AS** (AS nos.),

IP (IP Prefixes)]

Destination = [**Domain** (www.domain.com),

AS (AS nos.), **IP** (IP Prefixes)]

Classifier = [**Port** (Ports), **Protocol** (tcp)]

Exceptions = [DNS, Amazon, Google]

Time = []

Action = ALLOW / BLOCK }

The **def** keyword is used to define the policy with a unique associated ID. The policy contains following attributes as shown above:

- *Source*: Sources can be a combination of standard country codes (e.g AU, US), AS numbers and IP prefixes, all separated by commas. They represent the places from where the traffic is originating.
- *Destination*: It refers to the domain, AS number or IP prefixes of the services to which the traffic is destined to. This is typically whole or a part of the network of the entity who is passing the Geo-Block policy. E.g. An IXP customer.
- *Classifier*: These are the network classifiers to restrict the destination space to specific port numbers or protocols. This field is not mandatory.
- *Exceptions*: These refer to the essential services, domains or areas which should not be blocked. It is a crucial part of the policy as some essential services like DNS which lies within the source Geo-area domain may not be blocked. (E.g. Global DNS servers like Google). Other exceptions may include trusted entities like Amazon or Microsoft Cloud as they may be in use for the functioning of services of the customer passing Geo-Block policies. It may also include specific IP prefixes which are being used to manage or control any of the destination services.
- *Time*: This is the time for which the Geo-Block policy should remain active and can be infinite which will require to withdraw the policy manually using the ID provided.

- *Action*: This ENUM action refers to the creation of white-listed or black-listed model. This is the default action for the rest of the traffic which is not mentioned in the Geo-Block policy. If the default action is ALLOW, the sources listed in the policy will be blocked and the rest of the traffic will not be affected whereas if the default action is BLOCK, all traffic will be blocked except the sources mentioned in the policy. Exceptions are always allowed.

We can see how easily a Geo-Block policy could be expressed using an abstract high level grammar. To realize the same level of details and to implement it in the network is a complex process. If we were to block a country of source, it will require putting rules corresponding to all the IP prefixes geographically contained within that country. The IP prefixes can range from few hundreds to thousands and maintaining them in the network devices manually is tedious and close to impossible. This process is highly manual and is prone to human errors as some configuration or exceptions may be neglected or missed by the administrator. The challenge is even higher when an operator detects a DDoS attack on their network and needs to implement these policies using low level rules in the network elements in a very short period. Clearly, implementing a Geo-Block security policy manually in the network can be a nightmare and is the reason why current solutions are not that effective or are very costly. Automating the whole process and expressing the policies and requirements at an abstract level reduces manual complexity, rules out errors and substantially decreases response time while increasing the accurateness.

4 System Design and Architecture

Our Geo-Blocking System is built on our CASToR platform, described in the next subsection, and make use of the ONOS controller. Fig. 1 shows the high level architecture of the Geo-Block system and related components. Figure consists of an IXP switching fabric being dynamically programmed by the ONOS controller using Openflow as per the intent and flows passed by CASToR and Geo-Block application. There is a route server as in a traditional exchange point for receiving BGP control plane information and advertising it back to all the IXP customers. We make use of BGPMon [3] standard application which peers with the route server and outputs BGP updates in easy to parse XML format which is read by the Geo-Block application. This information is used by the Geo-Block application to prevent IP spoofing as described in the next section. The SQLite database is used by the application to maintain the mapping of Geo areas (E.g. Countries), Domains, AS numbers to IP prefixes.

4.1 The CASToR-ONOS Platform

The Geo-Block application is built as an add-on on top of CASToR[18] which is our interconnect application written on ONOS [9] controller. ONOS provides a high availability,

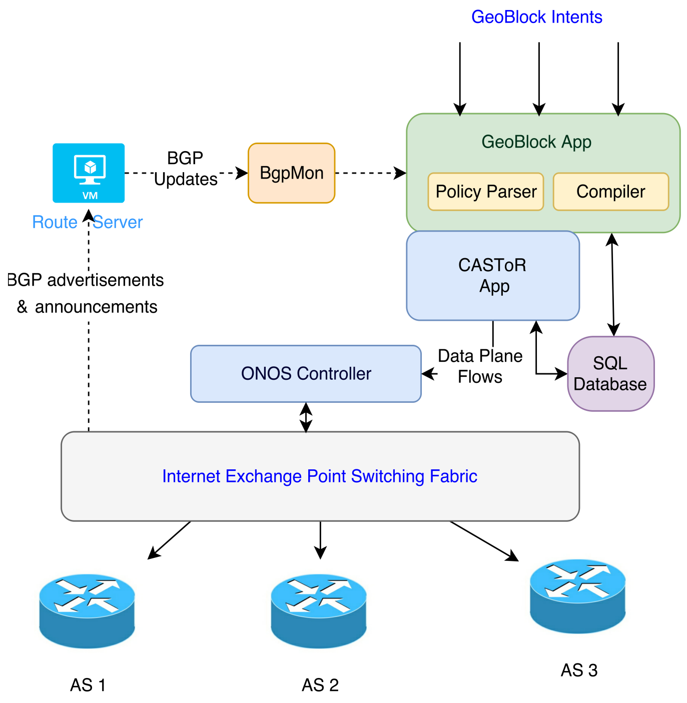

Figure 1. Geo-Block Architecture

carrier grade SDN controller platform and its Intent framework [8] provides many low level connectivity intents to easily program the data plane. CASToR application written over ONOS provides the facility of configuring the IXP customers and provisioning of the control plane as well as the data plane connectivity amongst all IXP members through a simple web interface. CASToR is now a part of the official release of ONOS.

4.2 Geo-Block Application

The Geo-Block application consists of two main components:

- *Policy Parser*: It is used to parse and break down the high level policies into lower level components of the policy. The application exposes a REST API to accept the high level policies in the form of a string as described in the previous section. Our parser integrates an effective language parsing tool ANTLR [1] which is used to define a grammar and break down the intents into different components such as blocking, connectivity and exception policies. This information is then used by the compiler as stated next.
- *Compiler*: It takes the low level policy and parsed components from the parser and translates them into ONOS intents and finally into flow level rules which can be installed into the switching fabric. This process is detailed in the compiler algorithm section.

4.3 Geo-Block Database

This essential database holds thousands of entries and mappings used for translating the Geo country codes, domains, AS numbers to the corresponding IP prefixes. The database is implemented using SQLite and a Django back-end framework [4] supporting the necessary REST APIs for querying the database. The geographical IP prefix ownership data is provided by a trusted source Maxmind [6] and can be synced on a per weekly or monthly basis. Some of the data was also collected from Hurricane Electric BGP toolkit [5]. The Django Back-end implementation also provides configurable APIs which can be used to add extra data to the database manually by the administrator which is specific to its customers. For E.g.- Exceptions and global DNS servers IP addresses and prefixes can be added manually and updated on a regular basis very easily.

5 Compiler Algorithm

The compiler is responsible for translating the high level policies into low level network flow rules which can be installed into the switching fabric to realize the intent. Fig. 2 shows a detailed flowchart of the compilation process and is explained below:

- The Policy Parser in the Geo-Block application receives the intents and parse them into useful tokens which can be mapped into JAVA classes and objects. The source and exception tokens are queried to get their corresponding IP prefixes from the database using REST APIs. New lower level policies are formulated using the IP prefixes which contains set of sources, destination and network classifiers. From here, exceptions are separated from the policies and are treated separately.
- Exceptions are always allowed. Therefore, they are converted into ONOS connectivity intents and are ultimately translated to network flow rules.
- If the default action was BLOCK, which means everything except the sources provided in the policy should be blocked, connectivity intents and flow rules are created to allow those sources to the destination.
- If the default action was ALLOW, the sources specified needs to be blocked and everything else is allowed. The blocking flow objectives are created at the egress on the device where the customer passing the policy is located. Location info of the customers is provided by CASToR. A suitable destination match is selected - If the destination is a subset of customer's network, IP prefixes supplied in the destination are used as a match. If the destination is whole of the customer, only destination MAC is required and reduces number of flow rules.
- IP spoofing filtering rules are also created using the BGP information received through BGPMon. These IP spoofing filtering rules are regularly updated whenever there is a new BGP update which is shown in the flow chart as 'Live BGP Trigger'. IP spoofing is described in detail in the following section.

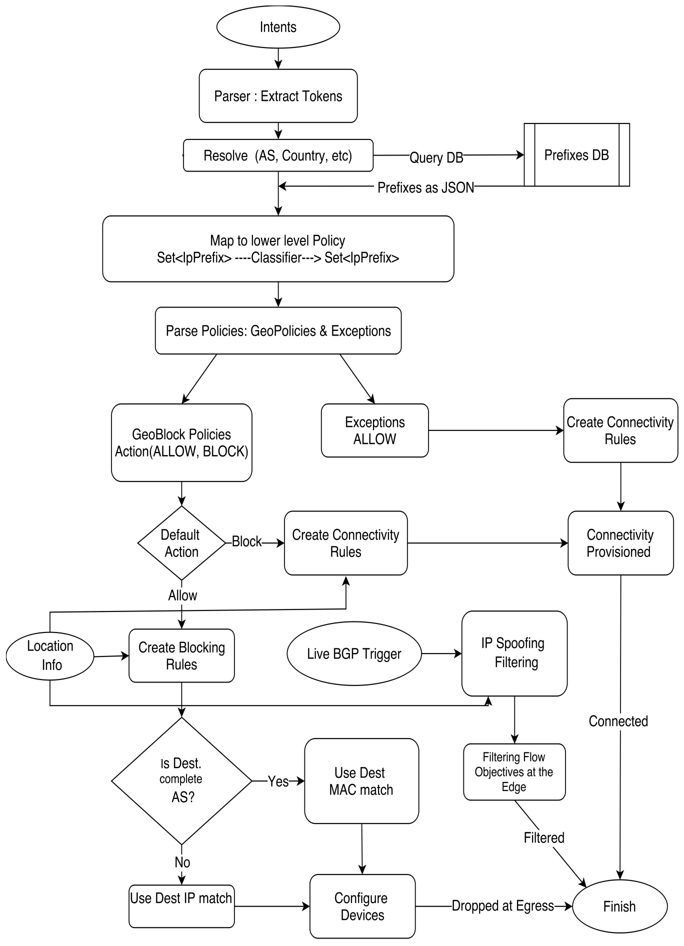

Figure 2. Compiler Algorithm

6 IP Spoofing prevention

IP spoofing can be used to bypass the Geo-Blocking solution to a certain extent. Though, a full proof IP spoofing prevention solution is not realistic, Ingress Filtering is often used as a precaution in many data centers and routers. We implement ingress filtering to prevent IP spoofing and make the Geo-Blocking solution more effective. An IP-Spoofing policy can be written and passed in the following format:

def spoof_protect (**Policy** Id) {

Customer = [IP address of the Border Router] }

Customer requesting the IP spoofing protection service will provide the IP address of its Border Router using which it is connecting to the exchange. After passing this policy, a direct customer of the IXP will only be able to send traffic with source IP address which belongs to it and are advertised at the IXP using BGP. This BGP control plane information is made available by BGPMon who peers with the route servers to get all the BGP updates and pass them to the Geo-Block application in an XML format. This is used to create filtering rules at the ingress of each direct IXP customer who asks for the IP spoofing policy or service perhaps at a cost. This is beneficial, as it saves the customer from the liability of originating an IP spoofing attack. For E.g. consider the scenario in Fig. 3.

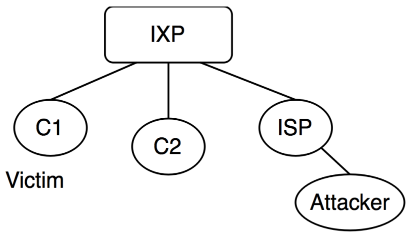

Figure 3. IP Spoofing scenario

C1 and C2 are two direct IXP customers in the same geographical area, say Australia (AU). Attacker is present outside AU and is connecting via an ISP (Internet Service Provider) peering at the exchange. C1 passes a Geo-Block policy blocking all traffic coming to it from outside AU. If the attacker is able to compromise some machines (bots) in C2's network, it can use them to originate malicious traffic with spoofed IP sources belonging to C1. This technique could be used in launching a DNS reflection DDoS attack on C1. However, if the IP spoofing rules are implemented, C2 will not be able to originate any spoofed traffic as it will only be allowed to send traffic with sources being advertised by it using BGP. Hence, all spoofed traffic will be dropped at the ingress point of C2 at the IXP. This could save C2 from being liable of originating an attack. Though this filtering is very effective, it is not full proof as it can be overridden in IP Prefix hijack situations. This technique will also fail if the attacker itself spoofs the addresses of C1 (Victim) as ISP will not have much interest in implementing and paying for the ingress filtering on its side.

7 Prototype Evaluation

We tested our prototype Geo-Block application in a laboratory testbed as shown in Fig. 4. The testbed emulated an SDN enabled exchange point and consisted of a single OpenFlow switch (NoviFlow NS1132), a controller running ONOS and the Geo-Block application, a single route server and three connected routers (peers) to represent autonomous systems that interconnect at the exchange point. The controller and the route server were hosted in virtual machines on commodity servers while the border routers of the three peers used were Cisco 3800 series. For test traffic we used a Spirent TestCenter equipped with a 12-port Hypermetrics CM 10/100/1000 module and firmware version 4.24.1026. The Spirent generator acted as both a traffic source and sink and allowed us to very accurately measure performance metrics such as throughput, loss and latency for multiple concurrent flows and generate traffic up to line rate. The capacity of each link between the border routers and the NoviFlow switch is shown in the Figure.

With this setup we were able to replicate, to a reasonable approximation, the Australian Census attack. We assumed that the Census website and associated infrastructure were accessed via AS 1 while attack traffic in the form of a DDoS was sourced from outside Australia via AS 3. The legitimate traffic originating within Australia was sourced from AS 2.

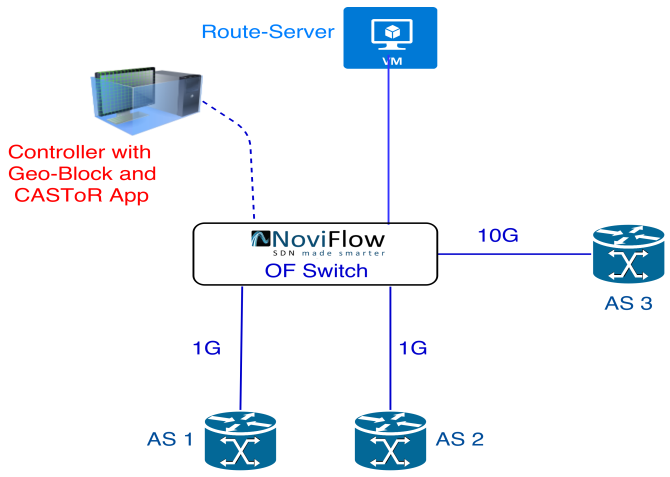

Figure 4. Experimental Setup

The results of our Census experiment are shown in Fig. 5. All traffic originated in AS 2 and AS 3 and was destined for AS 1 where the Census website is located. With the Spirent we used a combination of public source IP addresses to emulate hosts from within Australia and hosts external to Australia. The figure shows the total traffic rates from sources outside and inside Australia together with the combined rate. Initially the outside traffic rate was approximately 80 Mb/s and the inside rate 500 Mb/s. In this case, the link connecting AS 1 is not oversubscribed and there is no packet loss. At some time between 100 and 200s we initiated a DDoS attack by increasing the outside Australia traffic rate to 500 Mb/s thus driving the utilization of the link connecting AS 1 to 100%. There was still no loss of the legitimate traffic so we then increased the DDoS rate to 1 Gbps at approximately 300s. The figure clearly shows that the link to AS 1 became congested and consequently the legitimate traffic streams experienced loss rates of 30-35% on an average, demonstrating that with no mitigation the DDoS attack was succeeding.

We then invoked the Geo-Block policy in which the controller sent flow table updates to the switch to drop all packets with source addresses from outside Australia. This resulted in approximately six thousand rules being installed in the switch pipeline that allowed only packets with source

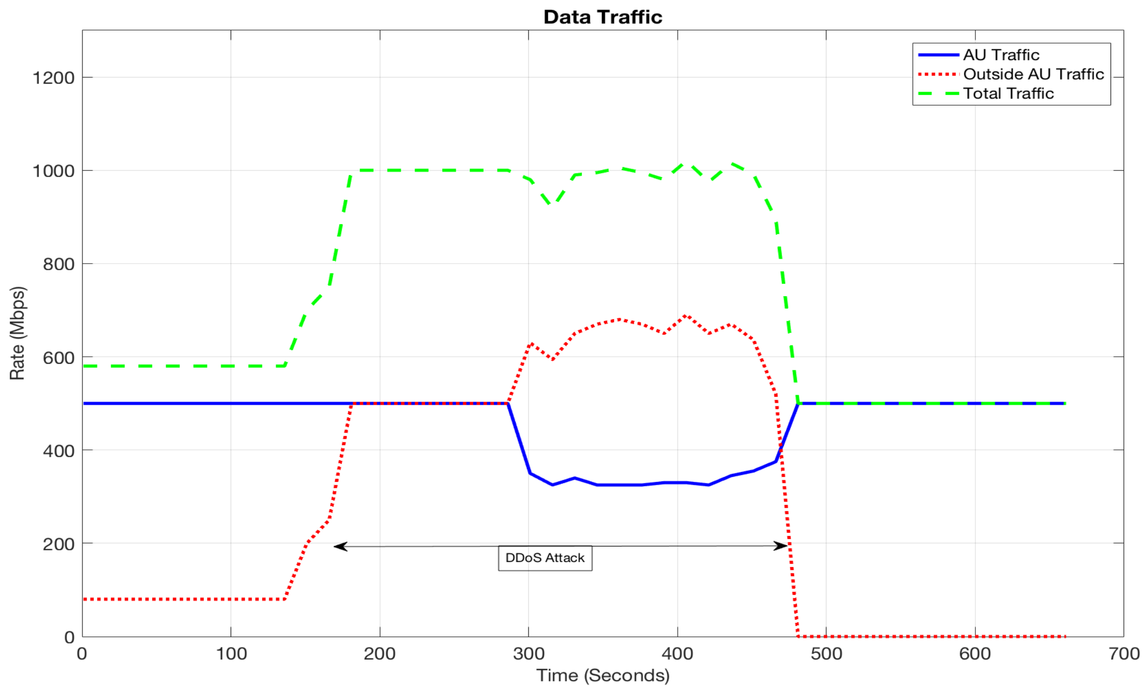

Figure 5. Data Traffic Plot

addresses corresponding to Australian IP prefixes and all other packets were dropped. The result of this policy can be seen in the figure just before the 500s time marker. The DDoS traffic from outside Australia reduced to zero, the legitimate traffic from within Australia recovered back to 500 Mb/s with no loss and the Census service was fully available again. Our system was responsive and scaled to thousands of flows that were pushed to the switch in no time.

8 Conclusion

SDXs offer better programmability and control over the switching fabric and can be used for a wide variety of use-cases. In this paper, we studied two new use cases of Geo-Blocking and IP Spoofing and developed a system on an Open Source platform to implement it. We demonstrated that a simple yet effective automated Geo-Blocking solution can help in prevention of DDoS attacks and eliminate extraneous traffic with a very low response time. We validated our architecture and implementation using experimentation with a practical scenario on real hardware and systems. The developed system will be open-sourced and made available as an add-on to the CASToR application of ONOS in its next official release.

References

[1] Antlr language parsing tool. http://www.antlr.org/.
[2] Australia census : Abs website crashes. http://www.abc.net.au/news/2016-08-09/abs-website-inaccessible-on-census-night/7711652.
[3] Bgpmon : Tool for monitoring bgp updates. http://www.bgpmon.io/.
[4] Django web framework. https://www.djangoproject.com/.
[5] Hurricane electric bgp toolkit. http://bgp.he.net/.
[6] Maxmind : Geoip databases. https://www.maxmind.com/en/home.
[7] Network ingress filtering. https://www.ietf.org/rfc/rfc2827.txt.
[8] Onos intent framework. https://wiki.onosproject.org/display/ONOS/Intent+Framework.
[9] Open network operating system. http://onosproject.org/.
[10] Packet viper : Advanced ip filtering solutions. http://www.packetviper.com.
[11] Project boulder nbi (north bound interface). http://opensourcesdn.org/projects/project-boulder-intent-northbound-interface-nbi.
[12] Pyretic framework. http://frenetic-lang.org/pyretic/.
[13] A. Gupta et al. An industrial-scale software defined internet exchange point. *NSDI*, 2016.
[14] Arpit Gupta et al. Sdx: A software defined internet exchange. *ACM SIGCOMM*, 2014.
[15] Avichai Cohen et al. Jumpstarting bgp security with path end validation. *ACM SIGCOMM*, 2016.
[16] C. Monsanto et al. Composing software defined networks. *USENIX NSDI*, 2013.
[17] C. Prakash et al . Pga: Using graphs to express and automatically reconcile network policies. *SIGCOMM*, 2015.
[18] Himal Kumar et al. A software defined flexible inter-domain interconnect using onos. *EWSDN*, 2016.
[19] Marco Chiesa et al. Inter-domain networking innovation on steroids: Empowering ixps with sdn capabilities. *IEEE Communications Magazine*, 2016.
[20] R. Lapeyrade et al. Openflow-based migration and management of the touix ixp. *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016.
[21] Ryan Beckett et al. Don't mind the gap: Bridging network-wide objectives and device-level configurations. *ACM SIGCOMM*, 2016.