AtlanticWave-SDX: An International SDX to Support Science Data Applications

Joaquín Chung*, Jacob Cox*, Julio Ibarra^, Jerônimo Bezerra^, Heidi Morgan^, Russell Clark*, Henry Owen*
* Georgia Institute of Technology
* Atlanta, Georgia
^ Florida International University
^ Miami, Florida

***Abstract*—New scientific instruments that are being designed and deployed in the coming years will dramatically increase the need for large, real-time data transfers among scientists throughout the world. One such instrument is the Large Synoptic Survey Telescope being built in Chile that will produce 6.4 GB images every 17 seconds. This paper describes an ongoing effort to meet the demands of these large data scientific instruments through the development of an international software defined exchange point (SDX) that will meet the provisioning needs for the scientific users. The specific planned and ongoing work in SDX architecture is described with specific consideration for policy specification and security.**

Keywords—Software Defined Exchange; Science Data Applications; SDN

I. INTRODUCTION

The Large Synoptic Survey Telescope (LSST) is a proposed large-aperture, wide-field, ground-based telescope that will scan half the sky continuously for 10 years. The 8.4-meter telescope will be located in the Andes mountains in Chile, taking a 6.4 GB image every 17 seconds. Each of those images needs to be transferred to NCSA within approximately five seconds so that processing can be completed in time to distribute transient alert notifications to the worldwide astronomical community within 60 seconds.

The international long-haul network interconnecting Chile to the U.S. is a critical component of the LSST system design because of the significance of LSST's operational requirements (e.g., the short interval between big data set transfers). The LSST operation will be composed of two channels: a control channel and a data channel. The control channel handles the remote operation of the telescopes in Chile by NOAO in Tucson, AZ.

These channels must be secure (i.e., encrypted), and they require low latency, high priority, and low bandwidth. In every 17 second interval, the data channel is also responsible for transmitting 6.4 GB images within 5 seconds. Such intervals and timing constraints also require high bandwidth availability, besides low latency and high priority. While the control channel requires a few Mbps, the data channel burst is estimated to be close to 90 Gbps.

To achieve the aforementioned requirements, the end-to-end path must provide high resilience, low delay, multiple paths, high bandwidth and an efficient control plane to act in all status changes (i.e., port status, devices outages, etc). The operational complexity of managing different administrative domains, topologies, link technologies, devices, and requirements is challenging when using traditional network operations methodologies for provisioning, monitoring and operating networks.

The AtlanticWave-SDX project aims to develop a capability to support applications, such as the LSST, that have intensive network resource requirements. In response to LSST's requirements for multiple diverse channels for control and data transport, AtlanticWave-SDX will leverage the SDN resources of the AmLight-ExP [1] network between the U.S. and South America.

The end-to-end path for LSST will be composed of different academic networks, some of them supporting SDN and network programmability. Having information about network resources and control for programmability will enable LSST applications to react to network conditions in a more efficient way, sometimes even anticipating issues. For example, a link that will flap might be detected when a CRC/loss number increases. With network programmability, LSST applications will be able to provision multiple paths dynamically and on demand, apply QoS and prioritization policies, and manipulate flows at multiple levels. Using information made available by all network devices on the path, LSST applications will be able to select the preferred paths from among several choices for sending its traffic from Chile to the U.S, vice versa, and also control the telescope remotely.

As mentioned above, the LSST end-to-end path will be provided by multiple networks, and in most cases, these networks are interconnected by academic exchange points. To achieve end-to-end programmability and control, all academic exchange points along the path must support network-aware applications. Fortunately, exposing network control capabilities to applications within a single SDN domain is now feasible and many academic networks (e.g., AmLight, Internet2, and EsNet) provide this capability today. This is not the case for applications that must span multiple domains. Most of the current Academic Exchange Points are still using traditional

methodologies for forwarding (e.g., IP or MAC-based forwarding) and control (e.g., a NOC team controlling network devices through SSH and/or SNMP).

For applications that span multiple network domains, all programmable network features mentioned before have to be supported by all academic network exchanges along the path, and with LSST as a use case, a few items are very explicit:

- Control Channel: due to its high priority profile, the LSST application has to be able to apply QoS policies to select the lowest latency port/link of the academic exchange point. If multiple links are available, inbound and outbound traffic engineering techniques will be applied to guarantee redundancy and prioritization, even over the Data Channel. As security is a must for the control channel, if available, encryption has to be enabled in the lowest level possible, and network isolation must be guaranteed to avoid hijacks.
- Data Channel: the main characteristic of the Data Channel is the bandwidth required, which is associated with delay, jitter and tolerance to packet loss. Traffic engineering and prioritization are also mandatory features that all academic exchange points must support. The LSST application will need to define what links to use, create multiple paths and apply other load balance approaches to guarantee its requirements.

An academic exchange point supporting these features is called a Software-Defined Exchange, and it is considered the next step in the network evolution following the SDN line of thinking. This SDX must be open, programmable and resilient. All its external interfaces must also be secure and support standard interfaces to support different kinds of network-aware applications.

II. BACKGROUND

Currently, there is no single, agreed upon definition of what Software Defined Exchange (SDX) means. The spectrum of definitions ranges from Networking Exchanges to Cloud Service Exchanges. Moreover, below the networking SDX definition, we can have: (1) Layer-3 SDX's that provide connectivity and routing between Autonomous Systems (AS) as in the case of an Internet Exchange Point (IXP); (2) Layer-2 SDX's for multi-domain Ethernet circuits; and (3) SDN SDX's to interconnect SDN islands. Likewise, the Cloud Service SDX provides access to compute and storage resources from different administrative domains. In the next sections, we provide examples of recent Layer-3, Layer-2 and SDN SDX's as those are more relevant to the AtlanticWave-SDX project; Cloud Service Exchanges could be seen as Federated Clouds. In Figure 1, we show a taxonomy for the Network Exchanges we consider.

A. *Layer-3 SDX*

As mentioned before, a Layer-3 SDX provides connectivity between different Autonomous Systems. The main characteristic of this kind of SDX is that a BGP process is required to handle the exchange of BGP routes. The minimum additional requirements are a SDN fabric and a SDN controller to install flows between the participants. It is desirable that the SDX has a Policy Manager to enrich the policies that can be defined with BGP. Some examples of Layer-3 SDX's are SDN-IP [2], Cardigan [3,4] and SDX [5], which are described in more detail next.

Lin et al. [2] proposed a solution to enable BGP peering between SDN and non-SDN Autonomous Systems. To achieve BGP peering, the centralized SDN control plane integrates a BGP process; turning the entire SDN AS into a single BGP router from the point of view of its peers. The solution was developed as an application in the ONOS controller, and tested using an emulated Mininet topology. Their experiments tested how the number of Routing Information Base (RIB) entries affects the memory incremental cost. The authors concluded that SDN-IP can scale up to 10,000 RIB entries, processing 100 RIB updates per second.

Cardigan [3,4] is a distributed router based on RouteFlow and a mesh of OpenFlow switches that are represented as a single logical switch. The goal is to implement a SDN-based distributed routing fabric. Cardigan's datapath works in a full-mesh, like router's line cards and fabric cross-connects using proactive flow installation. Cardigan was deployed connecting the Research and Education Advanced Network of New Zealand (REANNZ) to the Wellington Internet Exchange (WIX), handling 1134 flows with a TCP performance of 800Mbps approximately.

Gupta et al. [5] proposed the design, implementation and evaluation of SDX, to improve the network management capabilities of BGP participants in an Internet Exchange Point (IXP). The main idea behind SDX is to present a virtual SDX switch to each BGP participant, so they can realize high level tasks such as: application-specific peering, inbound traffic engineering, wide-area load balancing, and redirection through middle boxes all while ensuring isolation between the policies. For this solution, each participant sends its policies to the SDX controller; then the SDX engine compiles the individual policies and installs a single set of policies on the SDX switch. The authors claim that just adding a SDN switch and controller to an IXP, as in the previous examples, is not enough to realize a SDX. The first version of this SDX was implemented using Pyretic [6] running on a POX controller, an enhanced version is being implemented using Pyretic and a Ryu controller [7].

B. *Layer-2 SDX*

A Layer-2 SDX allows operators to create multi-domain circuits; typically using Layer-2 technologies like Ethernet VLANs. This scenario is mainly used in Research & Education Networks such as Internet2 and ESnet. For instance, Internet2's Advanced Layer 2 Service (AL2S) [8] allows network operators to create their own Layer 2 circuits in the Internet2 AL2S backbone connection two or more endpoints. Similarly, the On-demand Secure Circuits and Advance Reservation System (OSCARS) [9] accomplishes the same goal in the Department of Energy's high-performance science network ESnet.

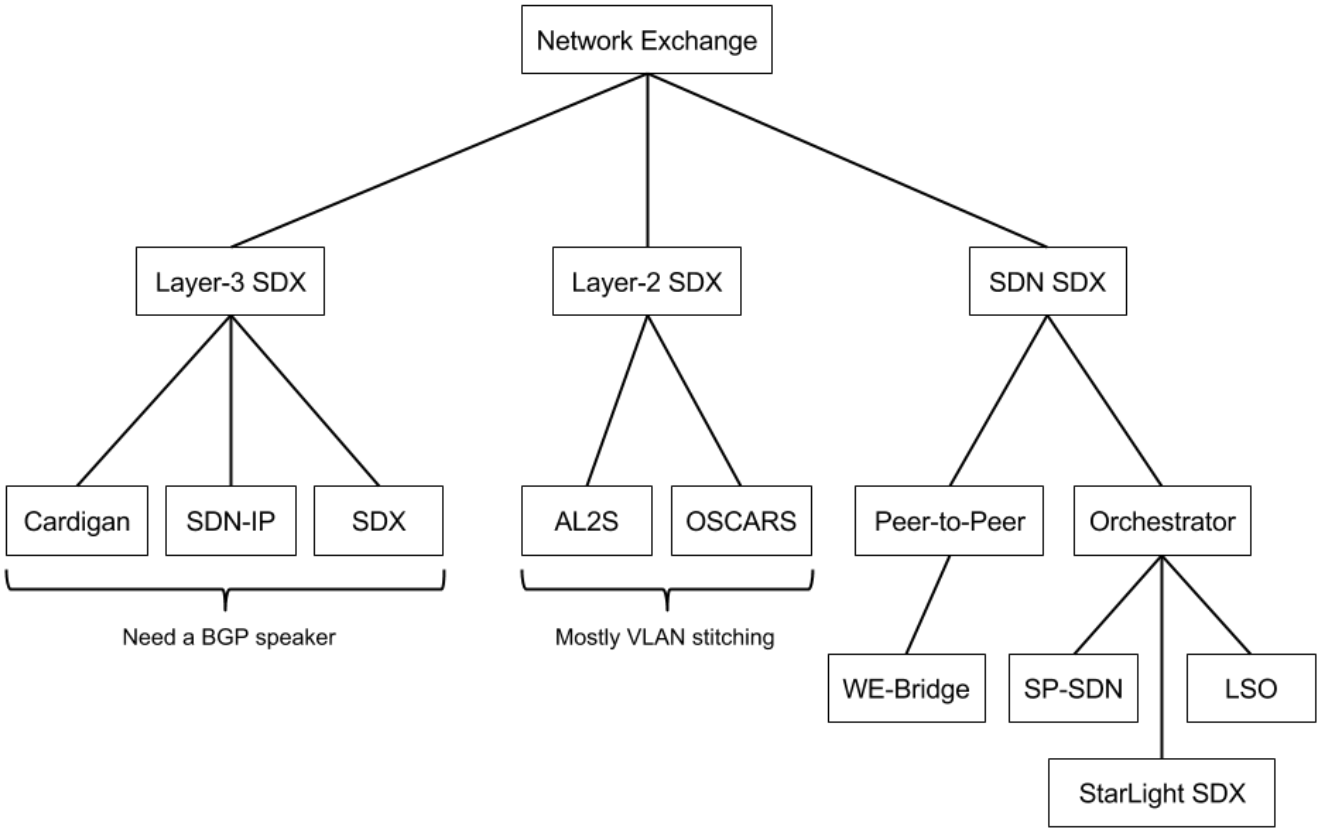

Fig. 1 - Network SDX Taxonomy

C. SDN SDX

The design objective of the SDN SDX is to interconnect SDN islands managed by different domains. The WE-Bridge [10] is a mechanism to enable different SDN administrative domains to peer and cooperate. WE-Bridge itself is not an inter-domain routing protocol, but a platform to exchange basic network information between different domains. The main goal is to improve inter-domain routing by announcing domain-views containing rich/fine-granularity information/policies, to enable various inter-domain innovations based on network information. This solution includes a network view virtualization, and a virtual network format and distribution using JSON. The peer relationships are established through a peer-to-peer control plane and a modified version of Link Layer Discovery Protocol (LLDP) to connect domain border switches. Contrary to the peer-to-peer approach used by the WE-Bridge, Mambretti et al. [11,12] proposed a centralized Path Controller to manage the resources of federated controller in order to interconnect federated SDN islands.

Similar approaches are the Service Provider SDN (SP-SDN) [13] and MEF's Lifecycle Service Orchestration (LSO) [14]. Both proposals envision a service orchestration layer on top on the SDN control layer, which span different administrative domains. Some application examples presented in these projects are: elastic WAN, network slices on-demand, VPN circuits on-demand, and end-to-end Network-as-a-Service.

D. SDX Characteristics

As we have seen, a SDX could exchange BGP routes, Layer-2 circuits, computing and storage capacity. More generally, an important characteristic of an SDX is its ability to exchange networking, computing or storage resources in a common point, between different administrative domains. Furthermore, the capability to apply richer policies to the exchange of these resources is another important characteristic of the SDX. Finally, in terms of security, strong isolation of constituent data and control interfaces is a desirable characteristic of a SDX.

III. ACHITECTURE

The AtlanticWave-SDX project is working to extend the SDX concept to a production deployment of a multi-domain SDX involving three academic exchange points, which include Southern Light; AMPATH and SoX, using the AtlanticWave 100G network. When fully deployed, AtlanticWave-SDX will provide application users with an end-to-end service that supports the traffic policy requirements of the application across multiple Autonomous Systems and physical exchanges.

There are several alternatives for how such an end-to-end capability can be provided. Figure 2 shows the proposed topology with three options of deployment. Option 1 assumes a single SDX controller that manages the entire IXP switch fabric. While this approach is the simplest technical option it is not ultimately viable in a distributed, multi-party environment.

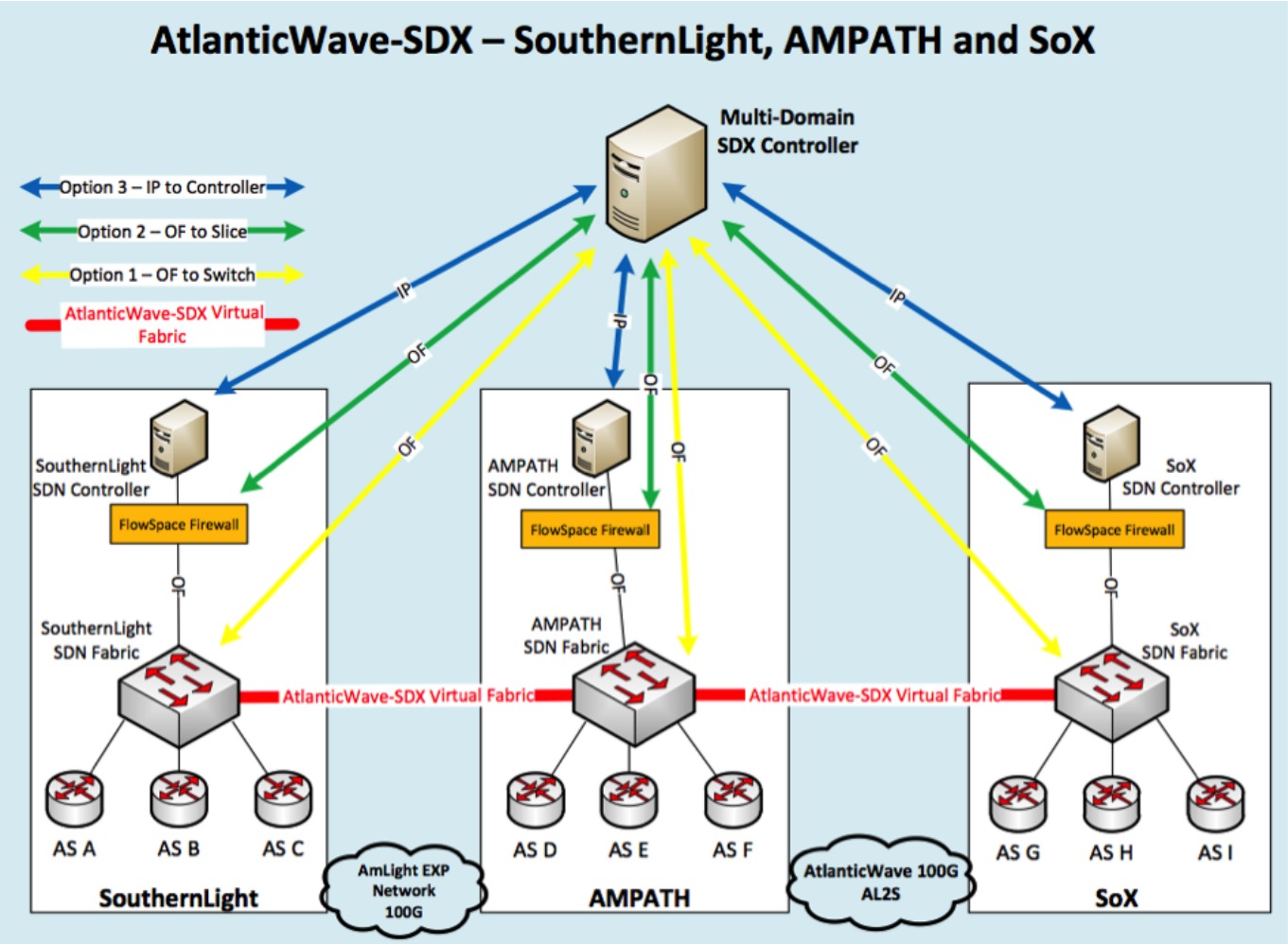

Fig. 2 - AtlanticWave-SDX

Option 2 introduces an intermediate slice manager, such as FlowVisor [15] or Flowspace Firewall [16], that allows individual controllers to be handed a slice of the network resources to be managed while isolating those resources from others. Option 3 creates a hierarchy of controllers with a local controller at each exchange being managed by a separate higher level controller. We expect Option 2 to be the most practical approach for the near term and intend to focus here for the initial implementation and deployment. In this work, we are extending our previous work [5] in SDX design to include both lower layer concepts (e.g. VLAN stitching) and upper layer concepts (e.g., application-based routing, load balancing, QoS, etc). We are designing and implement a software toolkit with APIs for application developers to tell the controller what demand they will introduce, at what times, and with what performance requirements, so the controller can plan/schedule the use of resources with prior knowledge of "when" and "what". The software developed in this project will be based on the SDX controller presented in [5] and available from GitHub [7].

This software is being actively used and extended, including ongoing work to deploy it on GENI [17]. The AtlanticWave-SDX project includes significant effort in "hardening" this software to make it production-ready and in extending it beyond the current Pyretic-based policy language to include programmable APIs for developers that support the specific application use cases identified here.

IV. Towards a Policy API for SDX

Before talking about SDX Policies, it is necessary to know what kind of applications can be deployed in an SDX. In [5] the authors proposed four applications: application-specific peering, inbound traffic engineering, wide-area load balancing and redirection through middle boxes. In general, the four applications match fields of the TCP/IP header and apply actions accordingly. However, in the LSST scenario the application needs to comply with certain latency and bandwidth requirements. These requirements cannot be defined using only fields of the TCP/IP header or the network topology status; the SDX controller requires external information such as SNMP, sFlow or perfSONAR [18] measurements.

Taking into account the considerations stated previously, there are several candidates for a Policy API for SDX. In [5], the authors opted for Pyretic, a high level programming

language for SDN. Similarly, the ONOS controller introduced the concept of intents for network policy specification [19]. On the other hand, WE-Bridge [10] proposed JSON as its policy API. Other valid contenders for a Policy API are RESTful and XML interfaces. To illustrate what SDX policies would look like, we present three examples: application specific peering, on-demand circuit provisioning and bandwidth calendaring.

A. *Application Specific Peering*

Consider three Autonomous Systems (A, B and C) connected to an SDX. Both B and C are advertising the same IP prefix to A (See Figure 3). SDX's Route Server decides best BGP path for these prefixes and advertises to A. In this example [20], routes advertised by B are preferred over C, for instance because of the AS-path length. For example, A might want its traffic destined for port 80 (dstport 80) to go to B, while traffic destined for port 4321 or port 4322 to go to C. This policy could be implemented as follows:

```
if (dstport == 80)
    forward to B
else if (dstport == 4321 || dstport == 4322)
    forward to C
```

This may be implemented in Pyretic as follows:

```
match(dstport = 80) >> fwd(B) + match(dstport=4321/4322) >> fwd(C)
```

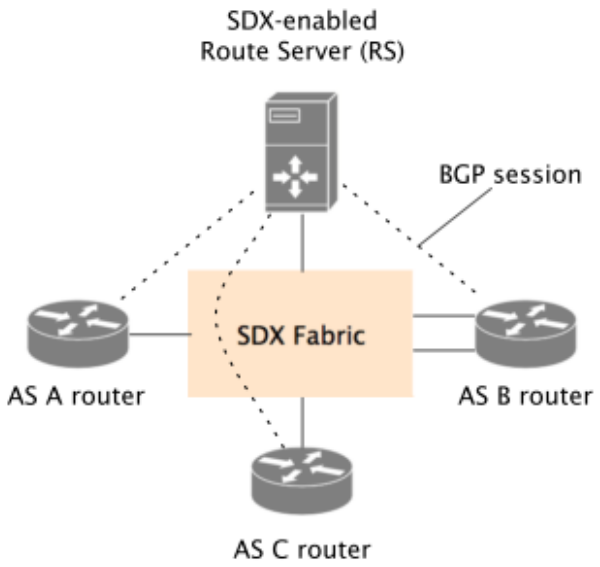

Fig. 3 - SDX network topology [5]

B. *On-Demand Virtual Circuit Provisioning*

This application provides the capability of provisioning virtual circuits on demand like Internet2's AL2S and ESnet's OSCARS. However, the SDX controller could take advantage of Network Monitoring Systems (NMS) such as perfSONAR to define Service Level Agreement (SLA) compliance and elastic WAN services, enriching the policies. In this scenario the SDX policy might look like:

```
if (current_latency > SLA_latency)
    secondary = findSecondaryPath()
while (current_latency > SLA_latency)
    LoadBalance(primary, secondary)
```

The while loop represents a dynamic policy, this behavior could be represented using Pyretic Dynamic Policies, ONOS Intents, an active polling mechanism, or a reactive triggered signal coming from the NMS (e.g. SNMP Traps). Another option is to use state machines as proposed by Kim et al. in Kinetic [21].

C. *Bandwidth Calendaring*

As proposed in [13], bandwidth calendaring will allow the SDX to reserve bandwidth for particular times. This is particularly relevant for the LSST because images are going to be sent each night. However, the circuits used could be in different timezones, making the reservation a more interesting problem. A possible representation of the policy is:

```
scheduled_time = 21:00:00 GMT -5
if (current_time == scheduled_time) {
    BW = 90 // Bandwidth in Mbps
    t = 60 // Reservation time
    OnDemandVC(BW, t)
}
```

Once again, Pyretic Dynamic Policies, ONOS Intents, or Kinetic style state machines are the candidates for implementation.

V. Security Concerns for SDX

Whenever we introduce new components in a network architecture, we also introduce new vulnerabilities; SDX is no exception. Considering the three types of SDX, we could say that the Layer-3 SDX will inherit all BGP vulnerabilities, the Layer-2 SDX will carry the same vulnerabilities of a shared Ethernet domain, and finally the SDN SDX will also introduce controller vulnerabilities. Such threats include DDOS attacks, attack inflation, exploitation of logically centralized controllers, compromised controllers (affecting the entire network), malicious controller applications, and negative impacts on recovery speeds [22]. Moreover, SDX introduces its own vulnerabilities as the SDX controller is a middle-man that every participant has to trust, and there is a possibility that some participants will declare policies that interfere with the proper function of other participants as a result, a trust relationship must be established between the applications loaded on the controller and the devices the controller manages [23,24].

The security issues with BGP are: prefix hijacking, TCP specific attacks, and manipulation of BGP attributes. Prefix hijacking occurs when an AS mistakenly or maliciously announces a prefix that has not been assigned to it. Some common TCP attacks are eavesdrop, man-in-the-middle, and DDoS (which can cause route flapping). Controllers are even more susceptible to TCP-based attacks since few controllers

actually use secure TCP connections [22]. Surprisingly, we observe that this issue occurs in spite of the OpenFlow protocol [25] allowing for an SSL secure channel between controller and switch. Already, several solutions (i.e., Resource Public Key Infrastructure or RPKI [26] and Secure BGP or S-BGP [27]) have been proposed to make BGP more secure and eliminate prefix hijacking. In consideration of these security requirements, Bailey et al. [26] combined RPKI and CARDIGAN to enforce the consistency of BGP announcements with its forwarding rules. Subsequently, mechanisms must also be developed to establish trust between controllers in order to ensure proper forwarding or detect malicious elements before a misconfiguration can occur and damage the network [16]. Equally important is the need for fast recovery after a link failure to mitigate packet loss and maintain the 17 second intervals required for the LSST project. This requires that mechanisms be incorporated throughout the network to notify the SDX controller of failures, so it can flush its flow entries and select new routes [28].

Concerning Layer-2 SDXs, LAN switches must be securely configured since switches in a shared Ethernet network are more vulnerable to malicious packets. A few examples of layer-2 attacks include MAC flooding, VLAN hopping, man-in-the-middle (via MAC address spoofing), and hijacking [29]. Unfortunately, with SDN, detecting and mitigating these attacks now becomes the responsibility of the network controller. While we are working on methods for detecting rogue DHCP servers and spoofed MAC addresses within the SDN framework, such methods require additional compute resources from SDN controllers and may raise scalability concerns [30].

Finally, from the Policy perspective, we would like for the policies of each SDX participant to only affect its own policy space. As a consequence, strong isolation is one of the main security requirements. Furthermore, each SDX controller becomes the middle-man that every participant has to trust. Thus, the controller functionality is a potential point of failure. For these reasons, controller resiliency and policy verification are desirable. Other countermeasures should include access control, attack detection, event filtering, firewall and IDPS, flow aggregation, forensics support, packet dropping, rate limiting, and shorter timeouts [22,23]. Regrettably, most of these countermeasures are not yet fully supported and work is ongoing to implement them [22,24].

VI. Conclusions and Next Steps

While an exact definition for a Software Defined Exchange (SDX) has yet to reach a consensus, the LSST project presents a unique use case for furthering the development of SDX. In this paper, we discussed the AtlanticWave-SDX project's goals, design, policy API, and security concerns. Once complete, the AtlanticWave-SDX will provide for an international long-haul network interconnecting Chile to the U.S. Additionally, with network programmability, LSST applications will be able to provision multiple paths dynamically and on demand, apply QoS, prioritize policies, and manipulate flows at multiple levels. Furthermore, by using information made available by all network devices along the path, LSST applications will be empowered to choose preferred paths from multiple transit options between Chile and the U.S and control the telescope remotely.

References

[1] J. Ibarra, J. Bezerra, H. Alvarez, M. Stanton, I. Machado, E. Grizendi, and L. F. Lopez, "Benefits brought by the use of OpenFlow/SDN in the AmLight intercontinental research and education network," in International Symposium of Integrated Management of Networks of the IEEE, 2015, pp. 1–6.

[2] P. Lin, J. Hart, U. Krishnaswamy, T. Murakami, M. Kobayashi, A. Al-Shabibi, K.-C. Wang, and J. Bi, "Seamless interworking of SDN and IP," in Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM - SIGCOMM '13, 2013, no. Figure 2, p. 475.

[3] J. P. Stringer, C. Lorier, and N. Zealand, "Cardigan : Deploying a Distributed Routing Fabric," Proc. Second ACM SIGCOMM Work. Hot Top. Softw. Defin. Netw. - HotSDN '13, pp. 169–170, 2013.

[4] J. Stringer, D. Pemberton, Q. Fu, C. Lorier, R. Nelson, J. Bailey, C. N. A. Corrêa, and C. E. Rothenberg, "Cardigan: SDN Distributed Routing Fabric Going Live at an Internet Exchange," in Computers and Communication (ISCC), 2014 IEEE Symposium on, 2014, pp. 1–7.

[5] A. Gupta, E. Katz-Bassett, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, and R. Clark, "SDX," ACM SIGCOMM Comput. Commun. Rev., vol. 44, no. 4, pp. 551–562, Aug. 2014.

[6] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, "Modular SDN Programming with Pyretic," USENIX, vol. 38, pp. 40–47, 2013.

[7] "Ryu Based SDX Controller". [Online]. Available: https://github.com/sdn-ixp/sdx-ryu. [Accessed: 09-AUG-2015]

[8] "Advanced Layer 2 System". [Online]. Available: http://www.internet2.edu/products-services/advanced-networking/layer-2-services/. [Accessed: 06-AUG-2015]

[9] "On-demand Secure Circuits and Advance Reservation System". [Online]. Available: http://www.es.net/engineering-services/oscars/. [Accessed: 06-AUG-2015]

[10] P. Lin, J. Bi, S. Wolff, Y. Wang, A. Xu, Z. Chen, H. Hu, and Y. Lin, "A West-East Bridge Based SDN Inter-Domain Testbed," IEEE Commun. Mag., vol. 53, no. February, pp. 190 – 197, 2015.

[11] J. Mambretti, J. Chen, and F. Yeh, "Software-Defined Network Exchanges (SDXs) and Infrastructure (SDI): Emerging Innovations In SDN and SDI Interdomain Multi-Layer Services and Capabilities," in Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014 First International, 2014, pp. 1–6.

[12] J. Mambretti, J. Chen, and F. Yeh, "Software-Defined Network Exchanges (SDXs): Architecture, Services, Capabilities, and Foundation Technologies," in Proceedings of the 2014 26th International Teletraffic Congress (ITC), 2014, pp. 0–5.

[13] J. Kempf, M. Körling, S. Baucke, I. Más, and O. Bäckman, "Fostering Rapid, Cross-domain Service Innovation in Operator Networks through Service Provider SDN," in IEEE International Conference on Communications, 2014, pp. 3070–3075.

[14] "The Third Network: Lifecycle Service Orchestration Vision". [Online]. Available: https://www.mef.net/Assets/White_Papers/MEF_Third_Network_LSO_Vision_FINAL.pdf. [Accessed: 09-AUG-2015]

[15] R. Sherwood, G.Gibb, K. K. Yap, G .Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer". OpenFlow Switch Consortium, Tech. Rep. 2009

[16] "FlowSpace Firewall". [Online]. Available: http://globalnoc.iu.edu/sdn/fsfw.html. [Accessed: 13-AUG-2015]

[17] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, I. Seskar, "GENI: A federated testbed for innovative network experiments", Computer Networks, vol. 61, pp. 5-23, ISSN 1389-1286, 2014.

[18] "perfSONAR". [Online]. Available: http://www.perfsonar.net/. [Accessed: 05-AUG-2015]

[19] ONOS Wiki, "Intent Framework". [Online]. Available: https://wiki.onosproject.org/display/ONOS/Intent+Framework. [Accessed: 12-AUG-2015]

[20] Coursera SDN Course, "SDX Assignment". [Online]. Available: https://docs.google.com/document/d/1wLF3RZEwMCRioyvaVl73kjXeNgqIaA3LfUlGsI2mkb4/edit?usp=sharing. [Accessed: 10-AUG-2015].

[21] H. Kim, J. Reich, A. Gupta, M. Shahbaz, N. Feamster, and R. Clark, "Kinetic: Verifiable Dynamic Network Control," pp. 1–11.

[22] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, 2015.

[23] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, "Rosemary: A robust, secure, and high performance network operating system," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014, pp. 78–89.

[24] SDX Central, "SDN Security Challenges in SDN Environments". [Online]. Available: https://www.sdxcentral.com/resources/security/security-challenges-sdn-software-defined-networks/. [Accessed: 10-AUG-2015].

[25] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFow: Enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69-74, Mar. 2008.

[26] J. Bailey, D. Pemberton, A. Linton, and C. Pelsser, "Enforcing RPKI-Based Routing Policy on the Data Plane at an Internet Exchange," HotSDN 2014, pp. 211–212, 2014.

[27] A. Boldyreva and R. Lychev, "Provable Security of S-BGP and other Path Vector Protocols: Model, Analysis and Extensions," ACM Conference on Computer and Communications Security 2012, pages 541–552, 2012.

[28] Sharma, Sachin, Dimitri Staessens, Didier Colle, Mario Pickavet, and Piet Demeester. "Enabling fast failure recovery in OpenFlow networks." In Design of Reliable Communication Networks (DRCN), 2011 8th International Workshop on the, pp. 164-171. IEEE, 2011.

[29] H. Altunbasak, S. Krasser, H. Owen, Grimminger, H. Huth, and J. Sokol. "Securing Layer 2 in Local Area Networks. Networking - ICN 2005. P. Lorenz and P. Dini, Springer Berlin Heidelberg. 3421: 699-706.

[30] Giotis, K., Christos Argyropoulos, Georgios Androulidakis, Dimitrios Kalogeras, and Vasilis Maglaris. "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments."Computer Networks 62 (2014): 122-136.