ARCHITECTURAL APPROACHES TO A SCIENCE NETWORK SOFTWARE-DEFINED EXCHANGE

A Thesis
Presented to
The Academic Faculty

by

Joaquin Chung Miranda

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2017

ARCHITECTURAL APPROACHES TO A SCIENCE NETWORK SOFTWARE-DEFINED EXCHANGE

Approved by:

Professor Henry L. Owen
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Russell Clark
School of Computer Science
Georgia Institute of Technology

Professor Raheem A. Beyah
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Mustaque Ahamad
School of Computer Science
Georgia Institute of Technology

Professor George Riley
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: November 7, 2017

To my parents Fernando and Elsa, and my brothers Javier and Isaac,

for their unconditional and endless love and support.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisors, Dr. Henry Owen and Dr. Russell Clark, for their invaluable guidance and support along my PhD journey. Since the beginning, Professor Henry Owen understood my needs as a Fulbright scholar, and he provided the essential guidance to successfully navigate the requirements of my PhD program. Likewise, Dr. Russell Clark exposed me to research projects and academic conferences that positively influenced my PhD thesis, and helped me expand my professional network.

Special thanks to the National Secretariat of Science, Technology and Innovation of Panama (SENACYT for its name in Spanish) and the Fulbright Program, for providing the funding for my doctoral studies. I would also like to thank all the professors and administrative staffs of the School of Electrical and Computer Engineering at the Georgia Institute of Technology.

I would like to give a special recognition to Dr. Raj Kettimuthu from Argonne National Laboratory, as well as my thesis committee members Dr. Raheem Beyah, Dr. George Riley, and Dr. Mustaque Ahamad for their insightful comments and suggestions. I would also like to thank my labmates and colleagues from the Network Security and Architecture lab, GT-RNOC, and the AtlanticWave/SDX project (funded by the National Science Foundation), who contributed valuable ideas, comments, and experience in carrying out this work.

Contents

List of Tables

List of Figures

SUMMARY

To interconnect research facilities across wide geographic areas, network operators deploy science networks, also referred to as Research and Education (R&E) networks. These networks allow experimenters to establish dedicated circuits between research facilities for transferring large amounts of data, by using advanced reservation systems. Intercontinental dedicated circuits typically require coordination between multiple administrative domains, which need to reach an agreement on a suitable advance reservation. To enhance provisioning capabilities of multi-domain advance reservations, we propose an architecture for end-to-end service orchestration in multi-domain science networks that leverages software-defined networking (SDN) and software-defined exchanges (SDX) for providing multi-path, multi-domain advance reservations. Our simulations show our orchestration architecture increases the reservation success rate. We evaluate our solution using GridFTP, one of the most popular tools for data transfers in the scientific community. Additionally, we propose an interface that domain scientists can use to request science network services from our orchestration framework. Furthermore, we propose a federated auditing framework (FAS) that allows an SDX to verify whether the configurations requested by a user are correctly enforced by participating SDN domains, whether the configurations requested are correctly removed after their expiration time, and whether configurations exist that are performing non-requested actions. We also propose an architecture for advance reservation access control using SDN and tokens.

Chapter I

INTRODUCTION

1.1 Problem Definition

Modern scientific instruments (e.g., particle accelerators, large telescopes, and genome sequencers) generate large datasets that are analyzed at supercomputing centers, typically hundreds of kilometers away from the original research facility. To interconnect research facilities with supercomputing centers across long distances, network operators deploy science networks or Research and Education (R&E) networks. These networks allow experimenters to establish dedicated circuits between research facilities by using advance reservation systems [18]. These systems are deployed on top of science networks and manage network resources in a coarse grained fashion (i.e., source and destination endpoints, required bandwidth, and duration of the reservation). Examples of advance reservation systems are the advanced layer 2 service (AL2S) of Internet2 [58] and the on-demand secure circuits and advance reservation system (OSCARS) of the Energy Science Network (ESNet) [77].

In the case of intercontinental dedicated circuits, network operators may take from days to several weeks for planning and provisioning a circuit over multiple science networks, because these tasks are typically done manually [56]. The use of advance reservation systems for requesting international or intercontinental dedicated circuits, combined with novel approaches to networking, such as software-defined networking (SDN) will significantly reduce provisioning times of science network services [23, 56]. However, as these reservations are defined by endpoints, duration, and bandwidth, the scheduling of resources is not flexible; that is, a reservation request will fail if the exact amount of bandwidth between two endpoints is not available within the

specified time frame [6]. This problem is dramatically amplified for intercontinental dedicated circuits, because the reservation spans multiple administrative domains, and participant domains have to reach an agreement on a suitable advance reservation that fulfills the requirements of the original request. Furthermore, despite a majority of domains having available resources for the reservation, if only one domain is not able to provide the requested resources, a multi-domain advance reservation will fail. Moreover, the success rate of finding an agreement decreases as the number of participants with limited bandwidth resources increases. This problem is analogous to trying to reserve a multi-legged flight with airlines that are not part of the same consortium and do not share flight schedules.

Another challenge is that advance reservations terminate at the WAN border router of each domain, and participant domains are interconnected at single junction points [98, 109]. As a result, multi-domain advance reservations are generally provisioned over single paths, adding more complexity to the problem of finding an agreement on the advance reservation. Furthermore, a data transfer has to compete with campus LAN traffic to reach the advanced reservation at the WAN border router of the research facility.

A secondary effect of non flexible scheduling of resources and high failure rates on reservations is the impact in user's productivity. Every time a reservation fails, the systems forces a user (a scientist) into a cycle of trial and error until a suitable time frame is found. Furthermore, the interface for requesting these types of reservations is very complex for domain-expert scientists who are not network operators. Arguably, many of these interfaces were developed *by* network operators, *for* network operators. Furthermore, manual provisioning of these connections, which could take from several days to several weeks [56], is sometimes limited by configuration overhead, poor scalability, and poor testing interfaces [98].

The federated nature of R&E exchange points is based on trust between participant domains. However, an old adage says "trust, but verify", so a responsible network operator wants to verify if his or her request have been enforced by domains participating on a multi-domain advance reservation. Moreover, some participants of the multi-domain advance reservation do not want to reveal internal topology information while still proving that they correctly deployed the requested reservation.

Recently, software-defined exchanges (SDX) have emerged as a new kind of cyberinfrastructure that allows independent administrative domains to share computing, storage, and networking resources by leveraging SDN [20, 49]. By using an SDX controller, network operators can program the fabric of an SDX in an agile way. We posit that by inserting an SDX in the junction point between participant domains in an intercontinental advance reservation, we will increase the success rate of finding a multi-domain advance reservation. The initial benefit of adding SDXs to the advance reservation process is that we overcome the limitation of single-path advance reservation (i.e., SDXs enable multi-domain, multi-path advance reservations). For instance, we may have two SDXs connected through two different advance reservation providers (e.g., WAN1 and WAN2) as shown in Figure 1, providing two independent paths between a telescope and a supercomputer. As a result, an experimenter may request half of the required bandwidth in each domain instead of requesting all the bandwidth in a single domain and not taking advantage of the secondary path. Another benefit of the SDX approach is that we provide alternatives to multi-domain, end-to-end advance reservation such as making reservations only at critical points or combining advance reservation and DiffServ QoS. Moreover, an SDX infrastructure will enable novel scheduling strategies that take advantage of the new infrastructure, as well as novel science network services (e.g., multipath bandwidth splitting across WANs, path migration, and multipoint-to-multipoint advance reservations). Additionally, an SDX will enable a federated auditing framework that allows verification of

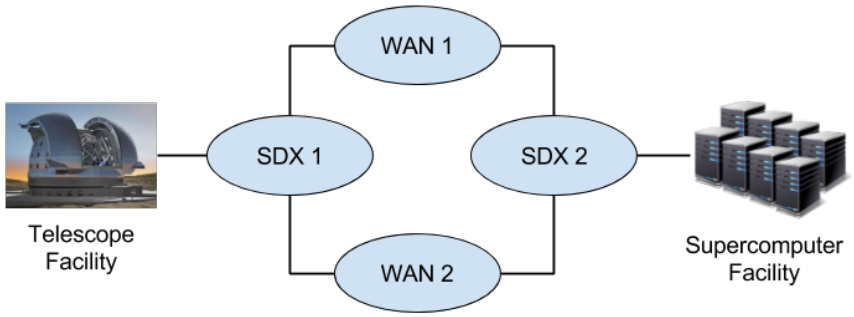


Figure 1: SDX-enabled multi-domain, multipath advance reservations scenario, with two SDXs connected through two different WANs, providing two independent paths between a telescope and a supercomputer.

the following conditions: (1) whether configurations requested by users are correctly enforced by participant SDN domains, (2) whether these configurations are correctly removed after their expiration time, and (3) whether configurations exist that are performing non requested actions. Furthermore, these verifications are conducted without revealing internal information about the participating domains.

1.2 Architectural Approaches to a Science Network Software-Defined Exchange

Thesis Statement: Given that current advance reservation systems present several challenges for deploying multi-domain intercontinental circuits, this work posits that by introducing SDXs in the reservation and provisioning process of intercontinental circuits, we are able to create multi-path, multi-domain advance reservations, which enhances the performance of science data transfers over traditional methods reported in the literature, while increasing the success rate of reservations, providing more intuitive interfaces to end users, and enabling auditing capabilities to network operators. To take advantage of an SDX-enabled advance reservation system for science networks, we develop an orchestration framework for advance reservation systems and SDXs that can provide access control to circuit reservations, flexible reservation requests, and automated provisioning over multiple domains, by leveraging SDN, SDX,

and APIs.

1.3 Contributions

The contributions of this dissertation are the following:

1. **An architecture for intercontinental multi-domain, multi-path advance reservations in science networks** that leverages SDN and SDX. The architecture is composed of an orchestrator that request services from participant domains and SDXs. We evaluated our proposed architecture using single-path vs. multi-path advance reservations over multiple domains, and the data transfer tools that the scientific community currently uses. Our architecture allows us to evaluate the impact of incorporating SDXs in science networks through:

   (a) **A negotiation protocol for multi-domain, multi-path advance reservations**, that allows an orchestrator to compose end-to-end services that take advantage of alternative paths provided by the enriched connectivity of SDXs. Our simulations using this negotiation protocol indicate that the reservation success increases in multi-path systems by splitting the bandwidth reservation over independent participant domains.

   (b) **Architectural approaches at the SDX level** that enable novel science network services, while enhancing the performance of science data transfers over traditional approaches. We evaluated SDX-rule provisioning options and bandwidth splitting strategies that allow data transfer protocols to take advantage of multipath bandwidth splitting.

2. **We propose an interface** that users and other systems can use to request science network services from our orchestration framework.

3. **We propose a federated auditing framework** for configuration verification in an SDX, a communication protocol that allows an SDX to query participant domains without exposing internal information, and include an initial proof-of-concept deployment of our federated auditing framework. In this context, we also propose **a system that uses SDN and tokens** to strongly bind an end-to-end flow to the user or application that requested an advance reservation.

1.4 Literature Survey

1.4.1 Advance Reservation Systems

Experimenters request and manage connections over a high-speed wide area network (WAN) using advance reservation systems [18]. Advance reservation connections are defined by the endpoints they connect, the requested bandwidth, the start time, and the end time. Generally, an advance reservation ends at the border router that connects a site to the WAN, and it is identified by a VLAN ID within the WAN. If a site does not have a high-speed dedicated network (e.g., Science DMZ [28]), scientific flows have to compete with campus LAN traffic before reaching the advance reservation in the border router. Currently, the operator can extend an advance reservation to the end host by manually extending VLANs on each site. This manual provisioning of VLANs on site, however, can take several days. For instance, the coordination of the provisioning process without automation may take between five and 45 days [56]. Additionally, as reservations are defined by the start time, the end time, and bandwidth, the scheduling of resources is not flexible; that is, a reservation request will fail if the exact amount of bandwidth is not available within the specified time frame [6, 98].

Ibarra et al. [56] described the deployment of SDN and OpenFlow on the AmLight international research and education network, which promotes collaboration research

between the United States and Latin America, with the goals of improving operations efficiency and providing network programmability, which were provided by the FlowSpace Firewall (FSF) [43] and the Open Exchange Software Suite (OESS) SDN controller [44]. With the new SDN AmLight, the provisioning time for a Layer 2 circuit that involves up to three domains decreased from five days and ten emails to less than two minutes and zero emails. Although SDN Amlight also automates provisioning of multidomain network reservations, its definition of a domain is a nationwide network (e.g., Internet2 and ESNet in the United Stated and RNP in Brazil). In contrast, our focus includes smaller domains such as national laboratories and university campuses and end-to-end reservations, as we are more concerned with automating provisioning for the last mile between the border router and the endpoint.

Tepsuporn et al. [98] tested the use of end-to-end Layer 2 paths for large dataset transfers over an existing deployment called DYNES (Dynamic Network System) [109]. The DYNES system uses OESS and OSCARS in multiple domains to establish dedicated Layer 2 circuits. While OESS is an intra-domain SDN controller that controls switches using OpenFlow [75], OSCARS supports inter-domain services. The authors identified limitations with configuration overhead, scalability, path provisioning, and testing. For instance, a failed path setup attempt in OSCARS forces a user to wait 15 minutes before issuing a new request [98].

To overcome the rigidity of advance reservation systems when a reservation request fails, Balman et al. [6] developed a novel approach for path-finding in time-dependent networks by taking advantage of user-provided parameters of the total volume (in bytes) and time constraints. Their algorithm finds alternate allocation possibilities, including the earliest time for completion, or the shortest transfer duration, with a quadratic complexity that depends on the number of nodes and existing reservations. They implemented their algorithm and tested it in the OSCARS reservation system as a flexible reservation service, the results of their test confirmed their theoretical

predictions about transfer performance.

1.4.2 Software-defined Networking (SDN)

Under the software-defined networking (SDN) paradigm [80, 64, 55], the control and data planes of network devices are decoupled. This separation enables global network programmability, rapid innovation, and independent evolution of control and data planes. The SDN architecture is divided into three layers: the infrastructure layer, which represents the data plane; the control layer, which represents the control plane; and the application layer, which represents network applications (e.g., switching, routing, or load balancing). The data plane is composed of many forwarding devices or SDN-enabled switches. The control plane is a logically centralized entity, generally known as an SDN controller that could be composed of a single server or several distributed SDN controllers. The SDN controller communicates with SDN switches through the southbound interface and with network applications through the northbound interface. A west-east interface that enables communication between several SDN controllers within the same administrative domain may be added. Furthermore, if these controllers belong to independent administrative domains, a multidomain SDN [98] that can be used to automate the provisioning of advance reservations is also possible [56].

An industry standard for the SDN southbound interface is the OpenFlow protocol [75, 81] that is also the most widely deployed. It works by installing match-action policies in the flow table of OpenFlow-enabled switches. SDN switches that support the OpenFlow protocol are implemented in both hardware and software. An example of a software SDN switch or a virtual switch is Open vSwitch (OVS) [85], a widely used switch in datacenters for network virtualization [63] that was designed for optimal operation in hypervisors [85]. Although the network industry, however, has not yet agreed on a standard for the northbound interface, its efforts are aligned with the

development of intent-based networking interfaces [14, 66]. Intent-based networking uses a prescriptive rather than descriptive approach to network configuration; that is, network operators and applications describe a goal, and the SDN controller decides how to implement it.

When SDN was proposed in 2008 by McKewon et al. [75], the authors assumed a network composed of only switches and routers. This first assumption omitted middleboxes and other common functions in computer networks. In a 2012 paper about SDN shortcomings, Casado et al. [17] reflected on how these shortcomings could be improved by using some ideas from multiprotocol label switching (MPLS). The authors argued that an ideal network design should be simple, vendor neutral, and future proof and thus proposed a network design composed of an edge and a core fabric, each controlled by independent SDN controllers. In this approach, the edge is responsible for complex network services and the core fabric handles basic packet transport. One of the benefits is that edge and core control planes continue to evolve separately.

SDN and Scientific Applications

Researchers have already proposed the use of SDN for enhancing scientific application resource management and performance over a WAN connection. For instance, the Lark project [108] proposed a flexible and fine-grained mechanism to manage network resources in high-throughput computing (HTC) systems [7]. Using Linux containers [70], virtual Ethernet devices, and SDN [108], Lark enables network resource management with per-job granularity for HTC systems such as HTCondor. In this architecture, each job is assigned to a separate network namespace [69], and each HTCondor node has a virtual switch (e.g., Open vSwitch or Linux bridge) that interconnects network namespaces to physical interfaces. The work considers only jobs running on the same node and allows users to actively change the network layer when

they submit batch jobs. To demonstrate these capabilities, the authors developed a bandwidth management system and a job aware OpenFlow controller and measured the performance overhead for both implementations. Because of the creation and the configuration of network namespaces, the authors reported a job startup overhead of one second. However, as a typical HTC job duration is measured in hours, this delay is negligible. Furthermore, the authors recognized that their SDN controller adds an additional level of complexity to the system, reducing overall stability.

Researchers also use SDN to enhance scientific application resource management and performance of a WAN connection by developing applications with networking capabilities via end-to-end SDN (DANCES project) [52]. DANCES investigates and develops the ability to add network bandwidth scheduling via SDN programmability to selected cyber-infrastructure services and applications in extreme science. DANCES [52] seeks to enhance the performance of cyberinfrastructure applications (e.g., GridFTP [2] data transfers, SLASH2 [86] distributed file system data transfers, and SCP) by adding network bandwidth scheduling via SDN. The project developed a bandwidth management component called centralized OpenFlow and network governing authority (CONGA), whose main function is to receive bandwidth requests from a resource manager or scheduling system and determine if the request can be fulfilled. To determine if a request is accepted or rejected, CONGA utilizes two criteria: (1) if resources are still available on the network, and (2) if the user is authorized to request this amount of bandwidth.

Network Access Control and SDN

Network access control (NAC), standardized as IEEE 802.1X [57], is a common computer security approach that authenticates endpoints and grants them access to a computer network. NAC, whose main focus is policy enforcement, was one of the first

applications developed for SDN [16, 78, 74, 106]. Casado et al. [16] proposed a secure architecture for the networked enterprise (SANE) that defines a single protection layer that governs all routing and access control decisions in the network. Similarly, Nayak et al. [78] proposed Resonance, a system for securing enterprise networks using dynamic access control policies and network devices as enforcement points. Also using SDN principles, FlowNAC [74] and FlowIdentity [106] adapt the IEEE 802.1X protocol. While FlowNAC performs authorization by a set of pre-defined flow rules per network service, FlowIndentity enforces a policy through a stateful role-based firewall updated dynamically in the SDN controller.

To access network resources, NAC requires the authentication of users/devices and further authorization following the AAA (authentication, authorization, and accounting) framework [29]. A user obtains authentication by many means such as passwords, certificates, and tokens, and authorization by three methods: an agent sequence, in which a user/device contacts a AAA server; a pull sequence, in which a user/device contacts a resource that then contacts the AAA server; and a push sequence, in which a user/device contacts a AAA server, receives a ticket, and then presents the ticket to a resource that also validates the token [102].

Gommans et al. [46] proposed a token-based access control mechanism for multi-domain lightpath (i.e., a fiber optics path) reservations in research and education networks. The authors identified and demonstrated three ways of enforcing access control policies: using a token-based switch at the IP packet layer; including a token in a specific field of RSVP-TE signaling protocol for GMPLS-based network at the control plane; and implementing an authentication, authorization, and accounting server, a token enforcement point, and a lightpath resource allocation system at service layer signaling. However, while this work extended to multiple domains, it did not use SDN because it had not been widely adopted.

1.4.3 Software-defined Exchange Points (SDX)

A novel internetworking paradigm, software-defined exchange (SDX), allows multiple independent administrative domains to share computing, storage, and networking resources. Therefore, an SDX can be regarded as a next-generation advance reservation system. This effort is promoted mainly by users and operators of research and education networks. Currently, networking researchers use SDX to incorporate SDN technologies into the networking infrastructure of Internet exchange points (IXPs) [49] and academic exchange points [73, 72]. Taking into account the exchanged networking resources, we can classify SDX solutions as follows: (1) layer 3 SDXs, which exchange BGP updates in Internet exchange points [49, 96]; (2) layer 2 SDXs [73, 72], which exchange multi-domain Ethernet circuits in research and education networks; and (3) SDN SDXs [15, 67], which interconnect SDN islands. In the following sub-sections we provide more details about these three types of SDXs. For a more extensive review of SDX architectures refer to [20], from which we also discuss the ideas pertaining to the exchange of computing and storage resources.

Layer 3 SDX

A layer 3 SDX provides SDN capabilities to the switching fabric of an IXP. The main characteristics of this kind of SDX is its handling of exchanges of BGP routes between IXP participants, the layer 3 SDX requires a BGP process. The minimum additional requirements are an SDN-enabled fabric, an SDN controller that installs flows between the participants, and a BGP process that listens to the BGP announcements of participants. To enrich policies that can be defined by BGP, a policy manager is recommended. Some examples of layer 3 SDXs are Cardigan [96], a distributed router based on RouteFlow and a mesh of OpenFlow switches represented as a single logical switch, and SDX [49], an SDN framework for improving the network management capabilities of BGP participants in an IXP.

Layer 2 SDX

According to definitions provided at the 23rd GENI Engineering Conference (GEC23), a layer 2 SDX is mainly used in RENs to interconnect research facilities by using layer 2 technologies such as Ethernet VLANs over optical circuits. Arguably, a layer 2 SDX is a redefinition of an advance reservation system. Following this assumption, we could classify SDN-enabled advance reservation systems (e.g., AL2S and OSCARS) as layer 2 SDXs.

SDN SDX

The design objective of the SDN SDX is to interconnect SDN islands managed by independent administrative domains. Again, this definition shares many characteristics of the advance reservation systems we presented in Section 1.4.1. However, while an SDN SDX provides a broader view of the deployment of SDN flow rules within multiple SDN domains, the SDX for scientific applications, the focus of this study, provides a narrower scope of the deployment of SDN flow rules within collaborating scientific facilities. Two ways of building an SDN SDX are utilizing either a centralized or peer-to-peer architecture. A centralized SDN SDX architecture could be implemented in one of three ways: (1) implementing a single logically centralized controller; (2) using an intermediate slice manager such as FlowVisor[93] or FlowSpace Firewall[43] that allows an external controller (i.e., the SDX controller) to manage a portion of each participant's network; or (3) creating a hierarchy of controllers with a local controller at each exchange being managed by a separate higher-level controller. The second way of building an SDN SDX, the peer-to-peer architecture, uses a west-east protocol between controllers in the SDX [67].

SDX Taxonomy

We initiated our investigation by conducting an extensive review of SDX architectures [20]. Taking into account the exchange of computing, storage, and networking

resources the spectrum of definitions for an SDX ranges from networking exchanges to cloud-service exchanges. We defined a taxonomy for SDX [21, 20], based on the resources exchanged as shown in Figure 2. It is important to note that most of the work in inter-cloud precedes the definition of the term SDX. An observation of our study is that infrastructures that do not classify themselves as SDX can be organized under our taxonomy.

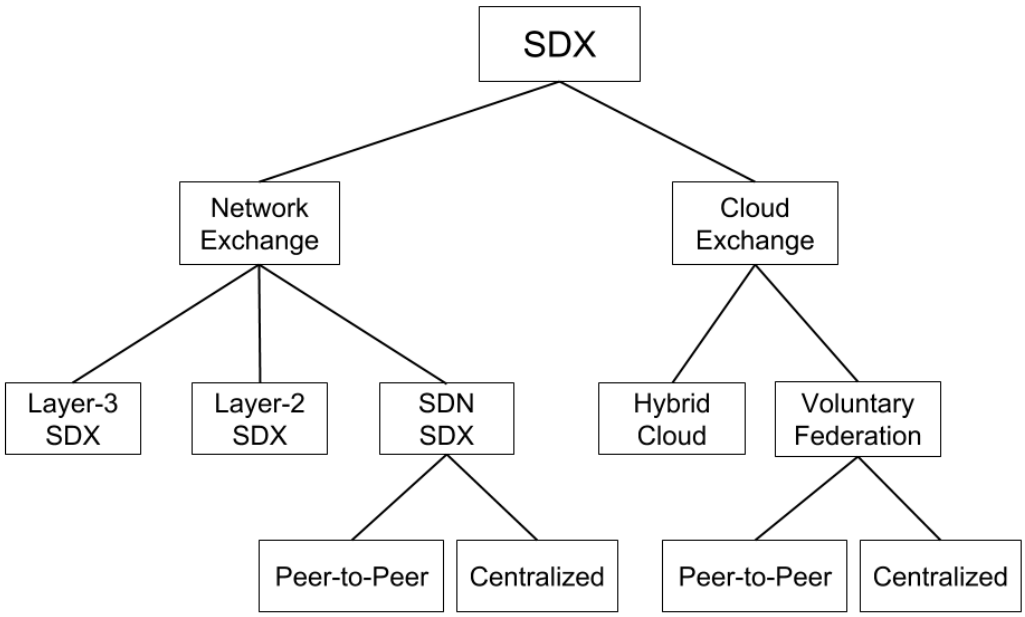


Figure 2: SDX Taxonomy

SDX Use Cases

Our SDX taxonomy classifies SDX architecture infrastructures, which influences the type of application or use case for an SDX. For instance, on an Internet exchange point (IXP), we would like to define richer policies than those allowed by BGP. To augment the capabilities of BGP policies in an IXP, Gupta et al. [49] proposed four SDX applications: application-specific peering, inbound traffic engineering, wide-area load balancing, and redirection through middle boxes. In the context of federated testbeds, the FELIX project[15] defines six applications classified into two main groups: the

data domain and the infrastructure domain. Under the data domain, the authors defined three applications: data-on-demand, data preprocessing, and high-quality media transmission over long-distance networks. The applications for the infrastructure domain are data mobility for inter-cloud use, follow-the-sun (or -moon) principles, and disaster recovery by IaaS migration. Table 1 summarizes SDX applications that we found in the literature.

Table 1: SDX Uses Cases

Use Case	**Description**	**Current Solution**
Application-specific peering[49]	Two neighboring AS exchange traffic only for certain applications. SDX could instead install custom rules for groups of flows corresponding to specific parts of flow space.	ISP could configure packet classifiers, Virtual Routing and Forwarding (VRF) and policy based routing
Inbound traffic engineering[49]	By installing forwarding rules in SDN-enabled switches at an exchange point, an AS can directly control inbound traffic according to source IP addresses or port numbers.	Destination based routing. Need to use AS prepending, communities and selective advertisement
Wide-area server load balancing[49]	A participant could announce anycast prefixes and the SDX controller would rewrite the destination IP address to match the chosen hosting location based on any fields in the packet header.	DNS global load balancing, IPv6 anycast

Table 1: SDX Uses Cases

Use Case	**Description**	**Current Solution**
Redirection through middleboxes[49]	An SDN-enabled exchange point can redirect targeted subsets of traffic through one or more middleboxes.	Manipulate the routing protocols to "steer" traffic through a fixed set of middleboxes.
Data-on-demand[15]	An SDX controller could automatically establish links between different end points (e.g. data storage and applications or users), guaranteeing the reliability of the end-to-end communication (e.g. minimum delay and jitter).	Content Centric Networks / Information Centric Networks
Data preprocessing for minimizing network latency effect for live data[15]	In these situations, a dedicated platform would be placed near the receiver station and perform a suitable preprocessing of the data, reducing the size of the data to be transferred.	WAN compression / WAN acceleration
High-quality media transmission over long-distance networks[15]	A higher quality of the media playback, imposes higher bandwidth and lower delay constraints on the network. This will require inter-domain QoS policies to achieve satisfactory QoE.	Compression and buffering techniques at the client and server ends

Table 1: SDX Uses Cases

Use Case	**Description**	**Current Solution**
DDoS Mitigation[34]	Add DDoS detection and mitigation capabilities in the IXP	BGP FlowSpec
Prefix hijacking prevention[5]	Deploy RPKI in an Software Defined IXP	RPKI
Inter-Cloud use case[15]	Data mobility service by SDN technologies. A SDX controller would be able to transfer user data (such as credentials, applications and services) to a cloud system closer to his/her visiting place.	Mobility Management
Follow the sun (or moon) principles[15]	An SDX controller could shift the load of one federated cloud to another one depending on the availability of resources and time of the day.	Cloud orchestration / scheduling
Disaster recovery by IaaS migrating[15]	An SDX controller could coordinate the configuration of the hypervisor resources with the network bandwidth constraints to allow a fast and efficient migration of the IaaS instance from one site to the other.	Replication to contingency sites

Table 1: SDX Uses Cases

Use Case	**Description**	**Current Solution**
Source-Address Based Multi-Path Routing[67]	Supports multipath inter-domain routing to the same destination based on different source IP addresses.	Traffic Engineering in private WAN
Inter-Domain Path Computation (IDPC)[67]	The purpose of IDPC is to achieve end-to-end QoS routing. After WE-Bridge modules in all the domains exchange local virtual network views including the bandwidth information, each domain can construct a relative global network view.	None

SDX Architectures Comparison

We conducted a comparison of the SDX architectures according to metrics such as scalability, resiliency, and peering boundaries (see Table 2).

Scalability - In the literature only SDX [49], Cardigan [96] and SDN-IP [68] present a performance evaluation of their solutions while FELIX [15], WEBridge [67], and SP-SDN[61]) do not provide any quantitative evaluation of their work. The main scalability metric for SDX, Cardigan, and SDN-IP is the number of RIB entries or flow rules, since they are examples of Layer-3 SDXs. Table 3 shows the performance evaluation results presented in each paper. For SDX, we considered the maximum values of 1,000 prefix groups with 300 participants in the IXP [49]. For SDN-IP, we

Table 2: SDX Architecture Comparison

	Layer-3 SDX	**Layer-2 SDX**	**Centralized SDN SDX**	**Peer-to-Peer SDN SDX**
Goal	To allow BGP peering between SDN and non-SDN networks, enabling richer policy descriptions	To interconnect independent networks using L2 technologies (VLAN) and OpenFlow	To interconnect SDN islands and to extend SDNs across multiple domains using a centralized, hierarchical controller	To interconnect SDN islands and to extend SDNs across multiple domains using a West-East interface, protocol
Architecture components	SDX controller, SDX fabric, Route Server, BGP Speaker, Policy Server	SDX controller, SDN fabric, Policy Server	SDX controller, SDN switches or slices, inter-domain link, Policy Server	SDX controller, West-East Interface/Protocol, West-East Link
SDX Type	Centralized	Centralized	Centralized	Peer-to-Peer
Topology Design	Centralized Single Fabric (Edge and Core), Single centralized controller	Centralized Single Fabric, Centralized controller	Inter-domain L2 links between switches/slices, Centralized controller	Inter-domain L2 links between switches/slices, Peer-to-Peer control plane
Peering Protocol	BGP	Static VLAN, Q-in-Q	OpenFlow, NSI	WE-Bridge + Modified LLDP
Isolation	Virtual switch per participant	VLAN	Slices	Virtual network view

considered the most recent value shown in ONOS SDN-IP Wiki[1] as opposed to the value presented in the paper [68].

Table 3: Layer-3 SDX Scalability comparison

Name	Max. Flow Rules	Compilation or Convergence Time
SDX	30000	700 sec.
Cardigan	1135	1 sec.
SDN-IP	15000 RIB entries	100 RIB updates/sec

Resilience - Resilience is the ability to maintain acceptable levels of service in the face of faults in the network infrastructure. In this regard, since the SDX controller becomes a single point of failure in the centralized approach, the peer-to-peer SDX architecture is inherently more resilient. However, resilience measures can be built into the centralized SDX architecture by the addition of distributed-computing techniques. For instance, SDN-IP is an application that runs on top of ONOS[8], an SDN-distributed network operating system.

Peering Technologies - To enable inter-domain networking, the boundaries between domains need to be set. BGP is used for Layer-3 routing while VLAN mapping and Q-in-Q tunneling are preferred in Layer-2 deployments. More recent architectures such as peer-to-peer SDN SDX require modifications to the LLDP protocol that delineate the boundaries between two SDN domains [67] in an automated fashion.

Deployment Considerations - The deployment of an SDX requires several considerations with regard to downtime, network topology changes, and disruption of services. Although Cardigan was deployed in a small IXP in New Zealand, the study does not discuss deployment issues or experiences [96]. By contrast, the remainder of the proposed solutions have been tested in emulation environments (i.e., Mininet) or in research and education networks. To deploy a large-scale Layer-3 SDX in an IXP, network operators might take advantage of hybrid SDN approaches in which SDN and conventional switches coexist in the same infrastructure. For instance, it is

[1]https://wiki.onosproject.org/display/ONOS/SDN-IP+Architecture

preferable that an IXP maintain its high-speed core fabric while adding SDX switches in the edge, which allow richer BGP policies.

Regarding centralized SDN SDX architectures, a sliced approach will increase the cost of deployment. For instance, if a certain SDN domain does not participate in an exchange using a slice manager, the deployment of the slice manager should be carefully planned so that it does not interrupt service in a production network. Since the resources allocated to the SDX are isolated and restricted, the sliced approach provides the most security. In the case of the subcontroller architecture, one should consider the communication protocol between controllers in the hierarchy (i.e., the customer controller, the ISP controller, and the ISP sub-controller) and have a rule conflict resolution mechanism for security.

Finally, control messaging should be deployed over secure protocols. In the case of a centralized SDX controller over slices, the use of TLS should be mandatory, and mutual authentication between the controller and slices is recommended. Likewise, peer-to-peer approaches should include security mechanisms in their protocols such as encryption of the payload and authentication of peer SDX controllers.

Security Considerations for SDXs

The NSF report on the workshop on software-defined infrastructure (SDI) and software-defined exchanges [88] identified network slicing as a promising way of securing SDI and SDX. A network slice is a logical instantiation of a physical network that provides the isolation of user traffic. The authors emphasize that to achieve secure slicing, SDX slices should incorporate admission control, secure slice provisioning, unforgeable slice identifiers across domains, and verification and auditing mechanisms. Although network slicing provides security guarantees for an SDX, the SDN paradigm introduces a new attack vector on a network infrastructure. For instance, if we assume a compromised SDN controller that allows an attacker to gain control over the whole network

or compromised switches that install malicious flow rules on the data plane [65], such problems could be extrapolated to an SDX as a malicious SDX controller or a participant domain. In the following paragraphs, we present studies that have tackled the problem of malicious SDN controllers and switches on a single domain.

One architecture that has addressed the problem of compromised switches is FlowMon. This architecture detects compromised switches through real-time analysis of network traffic statistics collected by OpenFlow in an SDN controller. Their main objective is to detect packet droppers (i.e., switches that purposely drop packets) and packet swappers (i.e., switches that forward packets to port that they were not intended for). To achieve their goal, the authors extended an SDN controller with two extra functional blocks: a malicious switch detection and prevention (MSDP) block and a policy block. While the MSDP continually and transparently analyzes the communication between the controller and switches, the policy block contains a set of rules enforced whenever a malicious switch is detected. The authors then proposed algorithms for detecting packet droppers by using information collected from port statistics from switches, and packet swappers by investigating the reports of unknown flows and comparing their expected output interfaces to their actually observed ones. Their results indicated that both algorithms had the ability to detect malicious switches in a mixed environment.

In another study that mitigate problems with malicious controllers, Schiff and Schmid [89] analyzed distributed control planes that are resilient to malicious controllers, represented by a malicious network administrator, a compromised controller software, or unintentional misconfigurations. The authors argued that a control plane that is resilient to malicious controllers requires a basic notion of memory and awareness of history. Their solution introduced a model in which a majority of benign controllers is responsible for accurately updating data plane switches despite the presence of malicious controllers by using a light-weight inband mechanism analogous

to consensus protocols. Their model assumes that data plane switches are trusted and each switch maintains a summary of the controller state and history. After verifying that a majority of controllers agree on the change, the switches implement the requested update.

Betge-Brezetz et al. [10] proposed a solution for alleviating a possible lack of trust between an SDN controller and its applications. Their proposal was similar, in essence, to that in [89] because it relied on several redundant controllers that may also be running separate executing environments. However, instead of a consensus protocol, the authors introduced an intermediary layer, a trusted-oriented controller proxy (ToCP), between the control plane and the data plane. ToPC was responsible for collecting and analyzing configuration requests from all redundant controllers and evaluating if they were consistent and trustworthy before determining whether to deploy them in the data plane or not. Their solution, implemented in Java, used the OpenVirtex [1] network hypervisor. Their results showed that ToPC introduced performance costs because of the degradation of service and the addition of computing resources.

1.5 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 introduces our reference architecture for orchestrating intercontinental multi-domain, multi-path advance reservations in science networks leveraging SDN and SDX, and Chapter 3 shows the design, implementation, and evaluation of a system for multi-domain, multi-path intercontinental advance reservations based on our architecture. Chapter 4 present the design of a proposed interface that allows domain-expert scientists to request novel network services for supporting big data science research. Chapter 5 presents FAS, a federated auditing system for SDXs that allows network operators to verify whether their configurations have been correctly deployed on an SDX without revealing internal

topology detail of participant SDXs. In chapter 5 we also present an architecture for advance reservation access control using SDN and tokens. Finally, Chapter 6 presents the conclusions of this thesis, and defines future research necessary to further evaluate the new architecture, protocols, and interfaces impact on actual science data network infrastructures.

Chapter II

ARCHITECTURE OVERVIEW

To support multi-path, multi-domain advance reservations for intercontinental dedicated circuits, we require an architecture that takes advantage of the agile programmability of SDN and the enriched connectivity provided by SDX to compose functional multi-path, multi-domain advance reservations while improving the success rate of user's requests and the performance of science data transfers. Our initial investigation reveals that no existing architecture is capable of providing multi-path, multi-domain advance reservations. Hence, we propose a new architecture [24], which is composed of the following components (see Figure 3):

1. Site controllers residing at sites research facilities that generate or process data.
2. WAN and SDX controllers that interconnect participating sites.
3. Orchestrators that consume services from site, WAN, and SDX controllers, while exposing end-to-end services to end users.
4. Users (e.g., domain-expert scientists) or applications (e.g., data workflow management systems) that consume end-to-end services composed by an orchestrator.

2.1 Site, WAN, and SDX Controllers

The site, WAN, and SDX components of our architecture follow the same SDN abstraction proposed by the ONF (i.e., infrastructure layer, control layer, and application layer). In our architecture, the application layer of SDN represents the science

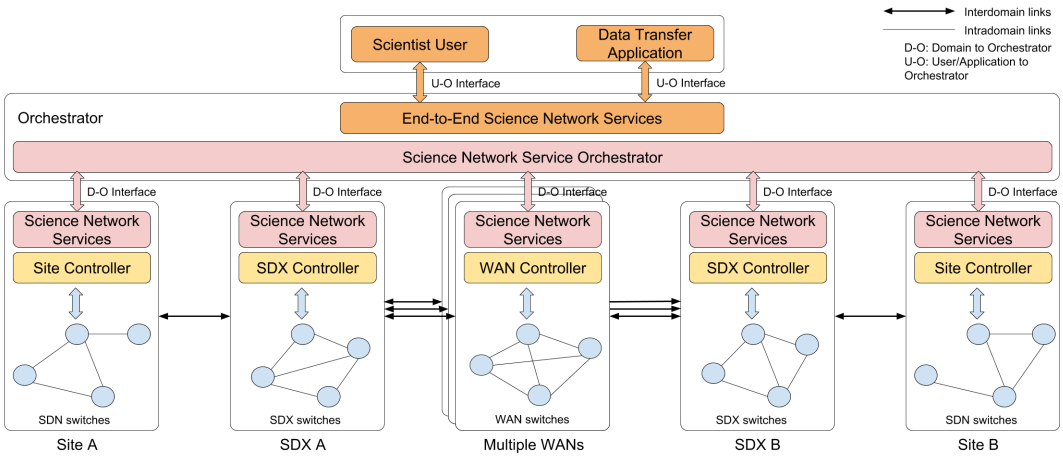


Figure 3: Reference architecture for end-to-end service orchestration in multi-domain science networks. Several independent administrative domains are connected by inter-domain links, and expose science network services to a centralized orchestrator through the domain to orchestrator (D-O) interface. The orchestrator then composes end-to-end science network services and exposes them to domain-expert scientists and data transfer applications through the user to orchestrator (U-O) interface.

network services exposed by each type of controller (i.e., site, WAN, and SDX controller). In this context, a site, WAN, or SDX controller may be any type of existing SDN controller, advanced reservation system, or SDX controller. The main requirement is that the northbound interface of these controllers should abstract the details of the network infrastructure and expose relevant science network services. More details about this type of interface is provided in Section 2.3.1. The infrastructure layer is composed of the data plane switches of each participant domain.

2.2 Orchestrator

The orchestrator is in charge of consuming services exposed by participant domains (e.g., Site, WAN, and SDX controllers), and composing end-to-end scientific services. For instance, in order to connect site A to site B in Figure 3, the orchestrator needs to know if all domains in between can provide this connectivity. To successfully compose end-to-end services, an orchestrator requires resource management, scheduling, and path computation functionalities. Our orchestrator maintains a minimal set of tables

or "databases": a table of participant domains and the services they provide, and a global topology view. Two approaches exist for managing and scheduling resources. The orchestrator may maintain a centralized resource manager and scheduler, or the orchestrator may allow each domain to manage their resources, and query for their network status whenever is needed. Currently, some [40] advocate for a loosely coupled infrastructure, leaning which matches the second approach.

An example of end-to-end service composition is the following: a user or application wants to connect a telescope in site A to a supercomputer in site B with a maximum latency of 100 ms. After verifying the domains involved in this end-to-end service, the orchestrator contacts each domain, querying if a path with the requested maximum delay is possible. Each queried domain does not have global knowledge about the end-to-end path, but they can commit to latency of their portion of the end-to-end path. The orchestrator then has to evaluate if a path that meets the end-to-end latency requirement can be formed. Otherwise, the orchestrator will try to find an alternative path, or will try to negotiate a maximum delay higher than 100 ms. In the presence of multiple paths and SDXs interconnecting them, the chances of composing a successful end-to-end service rises.

2.3 Interfaces and Services

Users in our system are domain-expert scientists whom in most of the cases do not have expertise in network operations, but still need to request reservations to expedite their data transfers. Additionally, scientists use data workflow management systems (e.g., Globus [45]) to automate the process of moving and sharing data across research facilities. In our reference architecture, both scientists and applications request end-to-end science network services to the orchestrator by using interfaces that abstract network infrastructure details in our reference architecture. The following subsections provide more details about the interfaces that allow communication between site,

WAN, or SDX controllers and an orchestrator, and between users or applications and orchestrators.

2.3.1 Domain to Orchestrator (D-O) Interface

The domain to orchestrator interface, depicted as *D-O interface* in Figure 3 allows a science network orchestrator to consume services from a site, WAN, or SDX controller. To understand the services that should be exposed by a site controller, we studied the Energy Science Network (ESNet) requirement review reports from 2013 to 2015 [26], and synthesized the most common scientific data transfers as follows: bulk data transfer, real-time data transfer, and management network traffic. Bulk data transfer is used to move large data sets between research facilities. GridFTP [2], an enhanced version of FTP for scientific applications, is one of the most used protocols for performing this kind of data transfer. Real-time data transfers are used for data streaming applications. For instance, a sensor network installed in an agricultural field transfers real-time sensor data to a remote server, or a scientist may access a remote visualization of a simulation running on a supercomputer. Management traffic allows the monitoring of the network by conducting active network performance tests, or managing scientific workflows, such as scheduling a backup or changing the orientation of a remote telescope.

2.3.2 User/Application to Orchestrator (U-O) Interface

The user/application to orchestrator interface, depicted as *U-O interface* in Figure 3 allows a scientist or a scientific application to request services from a science network orchestrator. To overcome rigid interfaces that only allow users to request a certain amount of bandwidth during a limited amount of time, we propose to describe a request based on time and bandwidth constraints. For instance, continuing with the example presented in Section 2.2, the user requires that her connection has strict time constraints (i.e., the transfer starts at 6:00AM local time, after the telescope has

taken images of the night sky), and the connection requires maximum bandwidth to finish the job as soon as possible. Details of bandwidth and delay should be left for network operators. A domain-expert scientist, for example a physicist, only wants to transfer X amount of gigabytes before certain deadline. The *U-O interface* includes flexible parameters that allow the orchestrator to negotiate an optimal solution to a user request, given the user constraints and the network state. Although, the *U-O interface* is an important component of the overall architecture, we will focus on the *D-O interface* and the components pertaining the network infrastructure for this study.

2.4 Authentication and Authorization

Authentication and authorization are important topics in this architecture for multi-domain science networks. Authentication is required for the orchestrator to authenticate participant entities and users or applications coming from several independent administrative domains. Then, authorization is required before the orchestrator issues any provisioning request to participant domains. Digital certificates may be used for mutual authentication between the orchestrator and participant entities, and a federated identity management system (e.g., Shibboleth [94]) may be used for the orchestrator to authenticate users and applications. Under this model, users are not required to have an account at each location, but they are authenticated using their institutional credentials. We propose to enforce authorization by installing policies on each entity (i.e., Site, WAN, and SDX controllers) that define what are users allowed to do, depending on their roles or affiliation (e.g., institution, project, or individual).

2.5 Consensus and Negotiation Protocol

Consensus among participant domains is vital for ensuring consistency of end-to-end services across multiple domains. For example, if one of the domains involved in an end-to-end service is not able to provide the requested service, the orchestrator

should be able to resolve this issue. Another example is that two users request the same resources at the same time, leading to a potential race condition. We propose to incorporate a two-phase commit protocol in the *D-O interface* to ensure consensus among participant domains and the orchestrator.

A negotiation protocol is a vital component of our architecture that allows the orchestrator to compose multi-domain, multi-path advance reservations. In traditional settings, an orchestrator will only be allowed to make reservations after a path computation system determines the domains on a single path. In our approach, a negotiation protocol will allow the orchestrator to explore multiple paths, and distribute the bandwidth reservation among several advance reservation systems. Then, negotiate with several SDXs to interconnect these advance reservations and compose a final end-to-end service.

Chapter III

ORCHESTRATING INTERNATIONAL ADVANCE RESERVATIONS WITH SOFTWARE-DEFINED EXCHANGES

3.1 Motivation

3.1.1 Bandwidth Request Splitting in Advance Reservation Systems

Traditionally, advance reservations requests are defined by source and destination endpoints, required bandwidth, start time, and end time. An advance reservation system performs path computation and scheduling operations to verify if resources are available to fulfill a request. Current implementations try to find an exact match for constraints provided in the request, and they fail if a suitable advance reservation is not found. Researchers have proposed scheduling algorithms for flexible (or malleable) advance reservation that increase the success rate of a reservation request on single domain scenarios [6, 101, 105]. However, for intercontinental advance reservations, in which the circuit spans multiple domains and follows a single path, flexible/malleable techniques lose their benefits because participant domains have to agree on the rigid constraints of the original request to compose the end-to-end service.

Let us consider a multi-domain, single-path advance reservation, with N participant domains and a success probability p for each individual domain. Then, the success probability of the entire reservation request (i.e., all domains succeed) is p^N. For instance, according to Xiao et al. [105] a conventional reservation system has a 66% success rate. If we consider an international advance reservation that spans two independent administrative domains, the success rate decreases to 43.56%.

Now, let us consider a multi-domain advance reservation that allows path diversity, and the requested bandwidth can be split over multiple paths. For simplicity, let us consider two research facilities connected across two possible advance reservation systems. We are allowed to request advance reservations to both systems, and we obtain a successful multi-domain, multi-path advance reservation if the sum of multiple path bandwidth requests is greater than or equal to the original required bandwidth. For N possible bandwidth offers from each domain, we obtain N^2 overall possible combinations. The success probability of the system is given by the following expression:

$$\frac{\sum_{x=1}^{N} x}{N^2} = \frac{N(N+1)}{2N^2} = \frac{1}{2} + \frac{1}{2N}$$

As N tends to infinity, the probability of success of such a system is approximately 50%. Fortunately, these types of scenarios are possible in the real world. The average degree of both ESNet's and Internet2's routed network topologies is approximately three (i.e., on average we could find three mutually exclusive paths between a source and a destination within any of these domains). Furthermore, we also find this path diversity on intercontinental links originated from the United States, as shown in Figure 4. The topology maps of ESNet [35] and Internet2 [42] report at least three links to Asia Pacific, three links to Latin America, and four links to Europe. However, this improvement is not possible without architectural changes to the system that supports end-to-end science network services. What architectural approaches will enable multi-path, multi-domain advance reservations for intercontinental dedicated circuits, while enhancing the performance of science data transfers? We posit that *by inserting an SDX in the junction point between participant domains enabling advance reservations to multiple domains in the path between two research facilities, we can compose functional multi-path, multi-domain advance reservations that enhance the performance of science data transfers.*

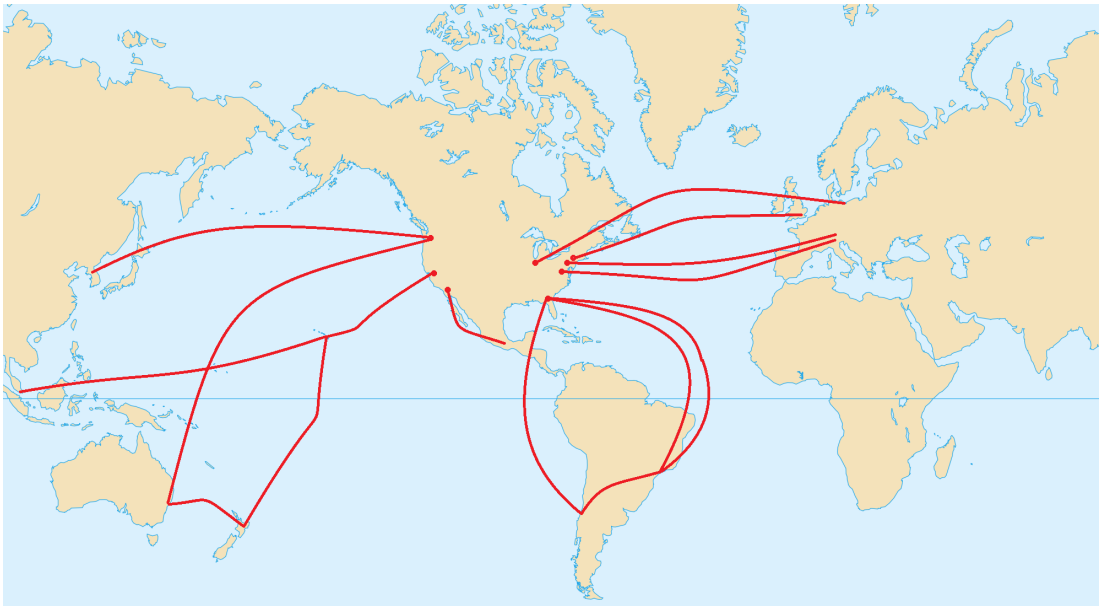


Figure 4: Intercontinental R&E links originated from the United States.

3.1.2 Software-defined Networking (SDN)

By taking advantage of the agile programmability of SDN, Ibarra et al. [56] have improved the provisioning time of international advance reservations in R&E networks from several days to a few minutes. Furthermore, in our work [23] we proposed the use of SDN and tokens to protect access to advance reservations at the research facility end, while keeping the same improvements achieved in [56]. Although SDN effectively reduces provisioning times of advance reservations, international or intercontinental advance reservations will require WAN-optimized protocols for the coordination and composition of science network services. What architectural approaches will allow multiple independent administrative domains to cooperate in the composition of multi-path, multi-domain advance reservations while maintaining the improvements attained by SDN in terms of provisioning times? We postulate that *an orchestration layer on top of domain, SDN controllers or advance reservation systems, combined with a WAN-optimized negotiation protocol will maintain the composition and provisioning of multi-path, multi-domain advance reservations in the order of seconds.*

3.1.3 Software-defined Exchange (SDX)

SDXs are another architectural innovation that will enable multi-path, multi-domain advance reservations. Moreover, SDXs will also enable novel science network services such as multi-path bandwidth splitting across independent WAN providers, scheduled path migrations that are transparent to data transfer applications, and multipoint-to-multipoint advance reservations. As SDX is a nascent technology, we need to know the advantages and disadvantages of using SDX as an interconnection point for multi-path, multi-domain advance reservations. For instance, it is well known that when using hashing for load balancing, all traffic corresponding to the same hash will be sent to the same interface. Nevertheless, we can take advantage of data transfer protocols (e.g., BBCP [82] and GridFTP [2]) that create multiple parallel TCP streams [54], and distribute these streams over a multi-path, multi-domain advance reservation. Furthermore, data transfer protocols have been designed for resilience. Another example is the case when networking resources are not available on a continuous time window in all domains. We propose to use SDXs to transparently migrate advance reservations from one path to another in the middle of an advance reservation. What are the architectural approaches at the SDX level that will enable the aforementioned novel science network services while enhancing the performance of end-to-end data transfers? We hypothesize that *by designing SDX services that take advantage of the network traffic characteristics of data transfer protocols, we will improve the performance of science data transfers, while enabling novel services not offered before by science and R&E networks.*

3.2 Design

In this section we present the design challenges for a system that provides multi-domain, multi-path advance reservation for intercontinental circuits in science networks [24]. We focus on the orchestrator and its interfaces that are used to communicate with end users and participant domains, the negotiation protocol, and the SDX services required to compose this kind of circuits. Additional components of the orchestrator such as path computation, scheduling, and resource management are outside the scope of this work, but algorithmic approaches to these standard components are readily available in the literature.

So far we have presented the orchestrator as an entity that oversees the composition of intercontinental advance reservations. However, many questions emerge in terms of real-world deployment and management: is the orchestrator centralized or distributed? Who runs and manages the orchestrator? We propose that a single entity deploys several instances of the orchestrator for load balancing and resilience. The orchestrator then, is physically distributed and logically centralized. The orchestrator may be run by a consortium of network providers. For more flexibility, we propose that each scientific community runs its own orchestrator that exposes services to orchestrators in higher levels, creating a hierarchy of end-to-end service orchestrators.

3.2.1 General Workflow

This section describes the general workflow for requesting and composing multi-path, multi-domain intercontinental advance reservations. We assume that multiple paths exist between two research facilities, and these paths traverse multiple administrative domains that provide connectivity and guaranteed bandwidth by using advanced reservation systems. We also assume that SDXs serve as interconnection points for

these administrative domains, enabling richer connectivity. An orchestrator (see Section 2.2) then is in charge of receiving user requests, requesting science network resources from the participant domains, and composing end-to-end services. We assume that advance reservation systems provide network service offers the same way airlines allow us to consult flight availability. For example, the newest code base for OSCARS provides a *"what if?"* function that will allow scientists to plan their network reservations on a Web interface [37].

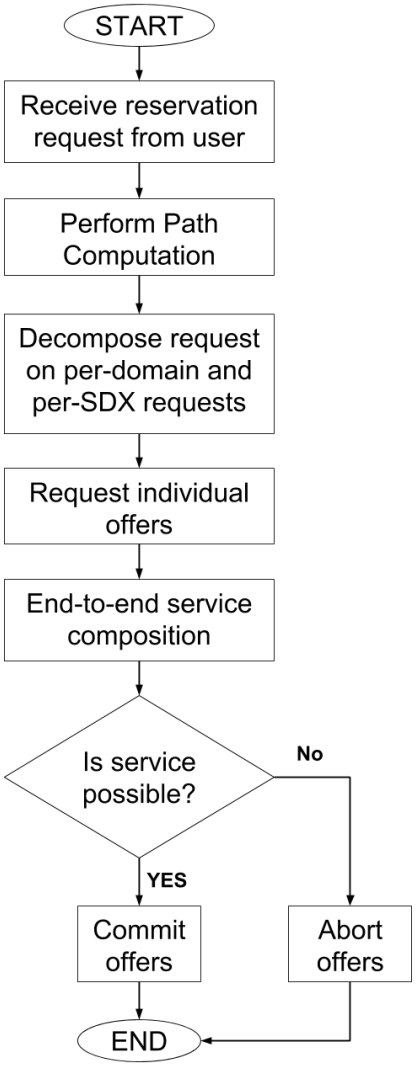


Figure 5: General workflow for requesting multi-domain, multipath advance reservations.

Figure 5 depicts the general worflow for requesting multi-domain, multipath advance reservations. The workflow starts with a user requesting an international or intercontinental advance reservation to the orchestrator, which performs path computation to determine the domains and SDXs on the path. From the orchestrator's point of view, participant domains are seen as links, while SDXs are seen as interconnection points. Then, the orchestrator decomposes the end-to-end request into individual requests for each domain and SDX on the path, and the orchestrator requests reservation offers to participant domains. Finally, the orchestrator uses these offers to compose an end-to-end service, commit offers and contact SDXs to make interconnections if an end-to-end service is possible, and aborts unused offers. Otherwise the orchestrator aborts all offers.

3.2.2 Negotiation Protocol

In this section we take a deeper look into the negotiation protocol that allows the orchestrator to compose multi-path, multi-domain advance reservations. By allowing multiple paths, and using SDXs as interconnection points, we increase the chances of obtaining a successful reservation as demonstrated in Section 3.1.1. The negotiation protocol is divided in two phases: phase 1 requests offers from participant domains and composes an end-to-end service, and phase 2 commits the successful offers, aborts unused offers, and request interconnection at SDXs. It is important to note that not all participants are willing to provide reservation offers, either because they have legacy systems, or because they have privacy concerns. We identify those domains that provide reservation offers as visible domains, and those that do not provide offers as blind domains. Visible domains are considered as the initial option to compose the end-to-end service. Blind domains are only considered if visible domains do not have enough resources. The rationale behind this strategy is that by considering blind domains for remaining resources, we increase the chances of success because it is easier

to allocate smaller amounts of bandwidth. Our negotiation protocol is composed of seven types of messages: *Reservation*, *ReqOffers*, *SendOffers*, *ReservationPrep*, *Commit*, *Abort*, and *ReservationResp*, that we describe in Table 4.

Table 4: Negotiation Protocol Messages

Message Type	**Description**
Reservation	Message from the user to the orchestrator requesting an intercontinental advance reservation
ReqOffers	Message from the orchestrator to visible domains requesting advance reservation offers
SendOffers	Message from visible domains to the orchestrator replaying with a list of advance reservation offers
ReservationPrep	Message from the orchestrator to all participant domains and SDXs requesting the preparation of an advance reservation
Commit	Message from the orchestrator to all participant domains and SDXs committing an advance reservation already prepared
Abort	Message from the orchestrator to all participant domains and SDXs aborting an advance reservation already prepared
ReservationResp	Message from all participant domains and SDXs to the orchestrator replaying if a preparation, commit, or abort request was a success or a failure. This message is also used by the orchestrator to report the results of the request to the user

Figure 6 shows the detailed negotiation protocol considering N participant domains, with M visible domains and $N - M$ blind domains. We consider three scenarios:

1. **No visibility ($M = 0$):** All participant domains are blind domains (i.e., traditional advance reservation systems).

2. **Full visibility ($M = N$):** All participant domains are visible domains (i.e., provide bandwidth offers).

3. **Partial visibility ($M \neq N$):** blind domains and visible domains participate in the orchestration process.

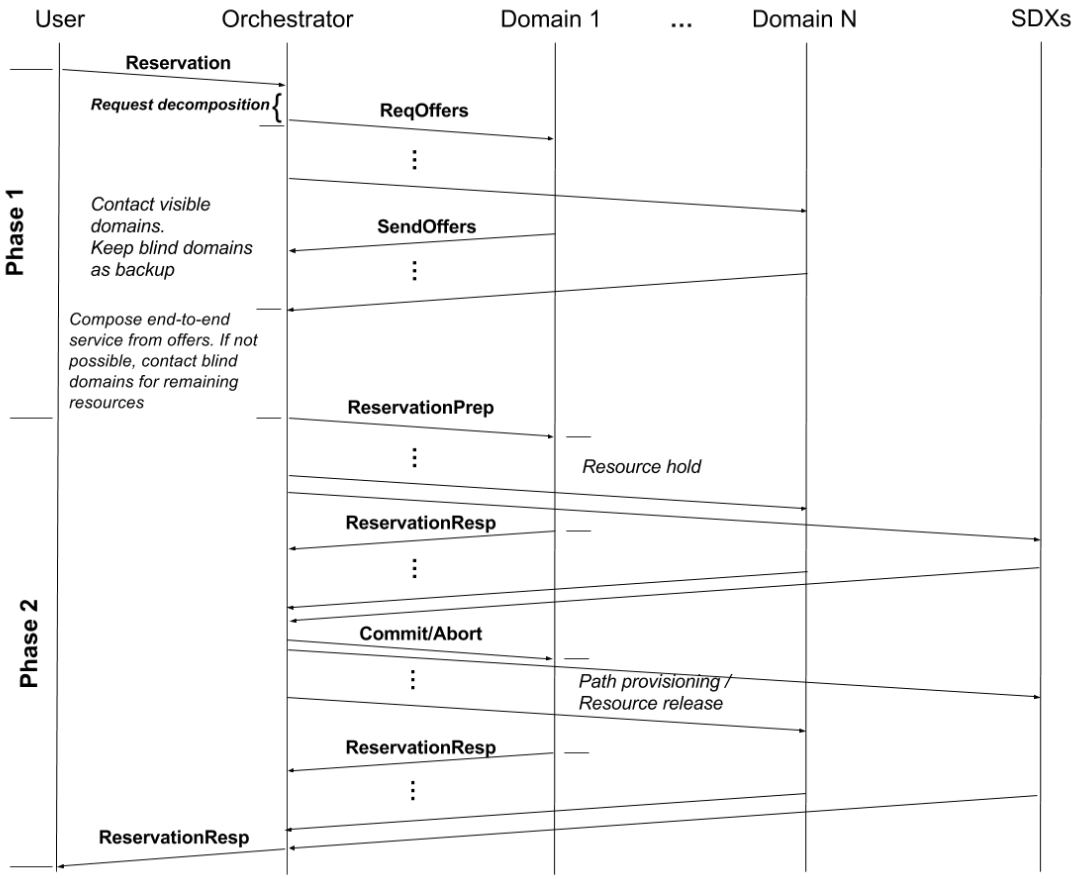


Figure 6: Negotiation protocol for multi-path, multi-domain advance reservation with M visible domains and $N - M$ blind domains.

The negotiation starts with a user requesting a *Reservation*. This reservation is decomposed by the orchestrator into individual reservation requests. How the orchestrator divides the original bandwidth request depends on the number of visible and blind domains participating in the process. The orchestrator sends *ReqOffers* messages to the M visible domains. These domains respond with *SendOffers* messages to the orchestrator, which uses these offers to compose an end-to-end service. Each *SendOffers* message contains a token ID [23] to identify the reservation request, because a domain controller may handle several requests from other individual users or orchestrators at a time. If the orchestrator is able to compose an end-to-end

service, the orchestrator transitions to phase 2 of the negotiation protocol by initiating a two-phase commit process with the participant domains and the SDXs (using *ReservationPrep*, *Commit*, *Abort*, and *ReservationResp* messages). Otherwise, the orchestrator requests the remaining resources to the blind domains and tries to compose a new end-to-end service. If the service composition succeeds, the orchestrator transitions to phase 2, otherwise the reservation request fails.

3.2.3 SDX Rules

As mentioned in section 3.2.1, SDXs are considered interconnection points in our design. For simplicity, we consider that SDXs have sufficient bandwidth to allocate user requests, so advance reservations are not needed inside the SDX itself. Additionally, we assume that SDXs in a given domain are in a single location (i.e., SDXs are not geographically distributed systems inside a single domain). We also assume that advance reservation systems provision layer 2 dedicated circuits or L2 tunnels over VLANs at each interconnection point. As a result, an SDX allows rules that bridge a VLAN in an inbound port to another VLAN in an outbound port, split traffic among several outbound ports, and create the corresponding mirror policies for bidirectional traffic.

Figure 7 illustrate the bandwidth splitting service block diagram. Our architecture takes advantage of the multi-streaming capability of existing science network data transfer protocols (e.g., BBCP and GridFTP). For instance, the Globus implementation of GridFTP [3] uses both TCP striping [54] and parallel TCP to achieve multi-streaming. We propose that an SDN switch and an SDN controller create flow rules that assign a new VLAN ID to every new TCP flow. Ideally, these switches and SDN controllers will be provisioned on demand for each new multi-domain, multi-path advance reservation, and may reside at the edge of the SDX or at the end sites. The orchestrator provides a pool of VLANs that are mapped to each independent

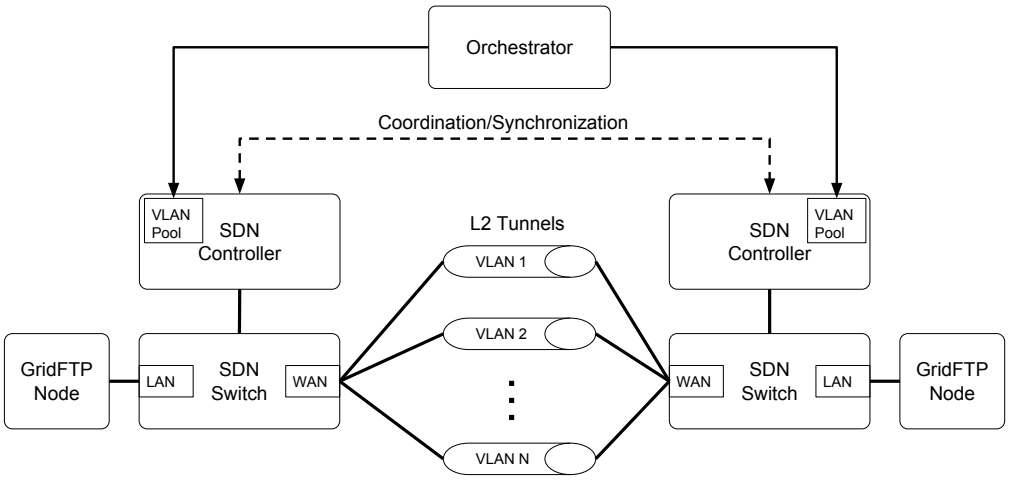


Figure 7: Block diagram of bandwidth splitting service components for SDX rule provisioning.

path at the SDX.

The SDN switches in Figure 7 have two ports: a WAN port that receives all the VLAN IDs representing the L2 tunnels, and a LAN port that connects the end site. The SDN controllers receive a pool of VLANs from the orchestrator, and tags each new packet from a specific flow that appears on the LAN port with a new VLAN ID from the pool before sending the packet to the WAN port. For every new packet that arrives on the WAN port, the SDN controller removes the VLAN tag and forwards the packet to the LAN port. The SDN controller selects VLAN IDs from the pool in a round robin fashion. To ensure that all the traffic belonging to a single flow traverses the same circuit, a synchronization or coordination between the SDN controllers assigning the VLANs might exist. Otherwise, we might have the forward traffic of a TCP flow traversing one tunnel, and all the ACKs returning over another tunnel. The use of multiple parallel and disjoint TCP flows is not new, but how to accomplish this in SDX-enabled science data networks is novel.

We consider three approaches for provisioning bandwidth splitting rules in an SDX: (1) synchronized, (2) unsynchronized, and (3) coordinated VLAN provisioning. The synchronized approach relies on both SDXs iterating over the VLAN pool

synchronously, effectively mapping each TCP stream to an end-to-end path. The unsynchronized approach does not care about mapping TCP streams to an end-to-end path, as long as all path converge to the same SDX and the VLAN ID is stripped or changed to a single VLAN ID before a packet is sent to the end site. The coordinated approach guarantees that each TCP stream is mapped to an end-to-end path by proactively installing a return traffic rule on the receiver side, for each new forwarding rule that appears on the sender side. A fourth approach is to proactively install all possible combinations of TCP source and destination ports mapped to the pool of VLANs. This approach is too expensive in terms flow table entries, and requires prior knowledge of the TCP port ranges used in both data transfer nodes. Dynamic configuration of data transfer nodes is outside of the scope of this work. For that reason, we do not consider the proactive approach in this study.

3.2.4 Interconnecting the Last Mile

An important component of our design is the last mile interconnection between the WAN border router and the scientific instrument or supercomputer at the research facility. Ideally, an SDN controller at the research facility will provision this last mile interconnection, or a science DMZ [28], a dedicated network for scientific data transfers, may be used to protect the science network traffic. In the absence of any of those mechanisms, traditional QoS techniques may be used to provide prioritized access to the network for scientific data transfers.

3.3 Implementation

In this section we present the implementation of an orchestrator for multi-path, multi-domain advance reservations, and the implementation of an SDX to support these services.

3.3.1 Orchestrator Implementation

We implemented the orchestrator in Python using an agent-based approach. In this approach, each participant domain hosts an agent that receives offer requests from an orchestrator, process those requests internally, and send offers back to the orchestrator. We selected an agent-based approach as opposed to simply consuming the APIs provided by each participant domain because that allows us to control the WAN communication channel between orchestrator and participant domains, while allowing us to customize interfaces for each domain controller. The orchestrator communicates with the agents using the general remote procedure call (gRPC) protocol [48], a high-performance RPC framework optimized for distributed computing and mobile environments. gRPC uses HTTP/2, a binary protocol that multiplexes multiple streams over a single TCP connection, for establishing communication channels between servers and stubs. On the other hand, HTTP/1.1 uses multiple TCP connections to issue parallel requests. Another advantage of gRPC is that it uses protocol buffers [47] for defining services and message types, and serializing data.

3.3.2 Negotiation Protocol Implementation

Considering the three scenarios described in section 3.2.2 (i.e., no visibility, full visibility, and partial visibility), we define three variants of the negotiation protocol for bandwidth splitting:

1. **Equally Splitting:** This strategy could be applied to any scenario. However, it is more suitable for the *no visibility* scenario, because it does not require the ability to request offers. In this approach the orchestrator divides the original bandwidth request in equal parts among the participant domains (see Algorithm 1).

2. **Partial Offers:** This approach is mainly applicable to the *partial visibility* scenario. Here the orchestrator contacts the visible domains for bandwidth

offers. If the orchestrator is able to compose an end-to-end service with these offers only, the orchestrator proceeds with Phase 2 of our negotiation protocol (i.e., provisioning). Otherwise, the orchestrator tries to request the remaining bandwidth from blind domains (see Algorithm 2).

3. **Full Offers:** This approach is only applicable to the *full visibility* scenario. In this approach the orchestrator contacts all participant domains for bandwidth offers. If the orchestrator is able to compose an end-to-end service with these offers, the orchestrator proceeds with Phase 2, otherwise the reservation request fails (see Algorithm 3).

Algorithm 1 Equally Splitting

```
BD ← Set of blind domain
VD ← Set of visible domains
D ← BD ⋃ VD
N ← Total number of participant domains
M ← Number of visible domains
EqSplitReq ← BwReq/N
for domain ∈ D do
  if EqSplitReq > AvailableBw then
    GoToPhase2()
  else
    ReservationFail()
  end if
end for
```

3.3.3 SDX Implementation

Our SDX implementation is based on AtlanticWave/SDX [33], an SDX controller written in Python that uses the Ryu SDN Framework [84] as an OpenFlow [75] speaker, and has a REST API and Web application for management. Currently, AtlanticWave/SDX supports advance reservation of L2 tunnels using the Web interface or the REST API. We added the bandwidth query functionality through a REST API in AtlanticWave/SDX to support our negotiation protocol. We verified that OSCARS, supports a similar functionality through their Web interface, but it does not

Algorithm 2 Partial Offers

$BD \leftarrow$ *Set of blind domain*
$VD \leftarrow$ *Set of visible domains*
$D \leftarrow BD \bigcup VD$
$N \leftarrow$ *Total number of participant domains*
$M \leftarrow$ *Number of visible domains*
for $domain \in VD$ **do**
  $offers \leftarrow ReqOffers()$
end for
if $\sum offers \geq BwReq$ **then**
  $GoToPhase2()$
else
  $EqSplitReq \leftarrow \frac{BwReq - \sum offers}{N-M}$
  for $domain \in BD$ **do**
    if $EqSplitReq > AvailableBw$ **then**
      $GoToPhase2()$
    else
      $ReservationFail()$
    end if
  end for
end if

Algorithm 3 Full Offers

$BD \leftarrow$ *Set of blind domain*
$VD \leftarrow$ *Set of visible domains*
$D \leftarrow BD \bigcup VD$
$N \leftarrow$ *Total number of participant domains*
$M \leftarrow$ *Number of visible domains*
for $domain \in D$ **do**
  $offers \leftarrow ReqOffers()$
end for
if $\sum offers \geq BwReq$ **then**
  $GoToPhase2()$
else
  $ReservationFail()$
end if

have a REST API for bandwidth queries.

The AtlanticWave/SDX controller provisions L2 tunnels using VLAN IDs, in the same way OSCARS and AL2S provision their circuits. We take advantage of this property to create our bandwidth splitting service using an Open vSwitch (OVS) [85] switch in OpenFlow mode and a Ryu SDN controller to aggregate two or more L2 tunnels into a single network service as described in section 3.2.3.

We implemented the three rule provisioning strategies (i.e., synchronized, unsynchronized, and coordinated) as Ryu apps. All approaches iterate over a pool of VLANs assigned to each L2 tunnel in a round robin fashion. The synchronized approach relies on traffic isolation to maintain synchronization between both iterators. In other words, a pair of OVS-Ryu on each end controls all the traffic of a single multi-domain, multi-path advance reservation. Both controllers start at the beginning of the list and advance synchronously with every new flow. For the unsynchronized approach, we intentionally forced one of the SDN controller to start iterating its VLAN pool list from a greater index. For the coordinated approach, we used a single Ryu controller on the Orchestrator controlling both OVS switches at the edge of the SDXs. We chose this approach for simplicity, but the same goal could be achieved with two separate controllers controlling each other through a REST API or another communication channel.

3.4 Evaluation

In this section we evaluate the success rate of the three variations of our negotiation protocol, and the performance of several provisioning strategies for a multi-path, multi-domain advance reservation service.

3.4.1 Orchestrator Microbenchmark

Figure 8 compares the system latency of our orchestrator for requesting resources from eight participant domains, while variating the RTT between the orchestrator and

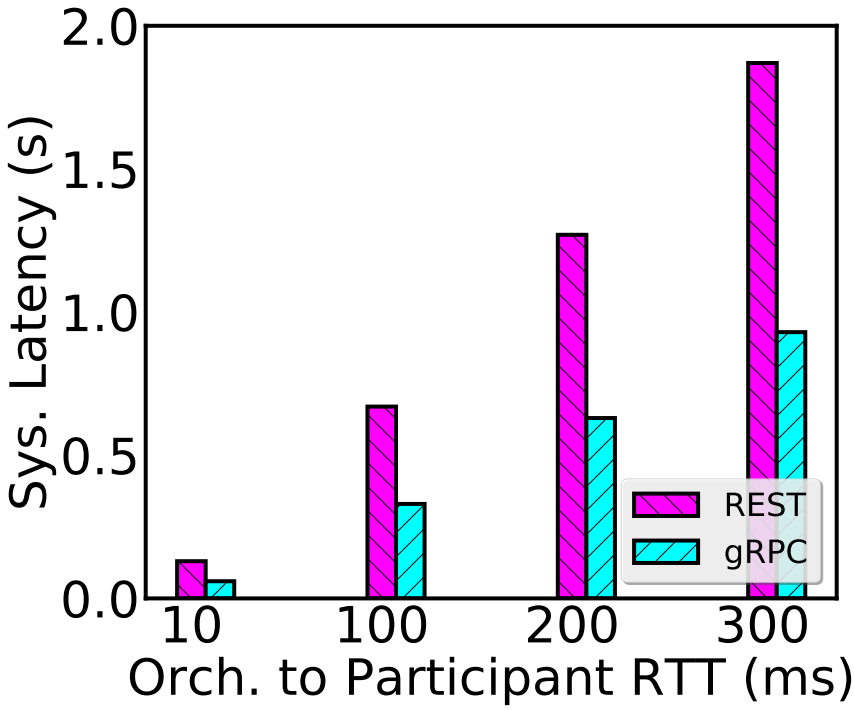


Figure 8: System latency microbenchmark for an orchestrator requesting resources from eight participant domains using REST and gRPC, and variating the RTT between participants and the orchestrator.

participants. We compare REST and gRPC for the communication channel between the orchestrator and participant domains. We used the GENI (Global Environment for Network Innovations) platform [9] to conduct our experiment. Our results shows that even in the worst case scenario (i.e., using REST when the RTT is 300 ms), the system latency remains below two seconds. We show that gRPC provides better WAN performance as the system latency remains under one second all the values of RTT tested.

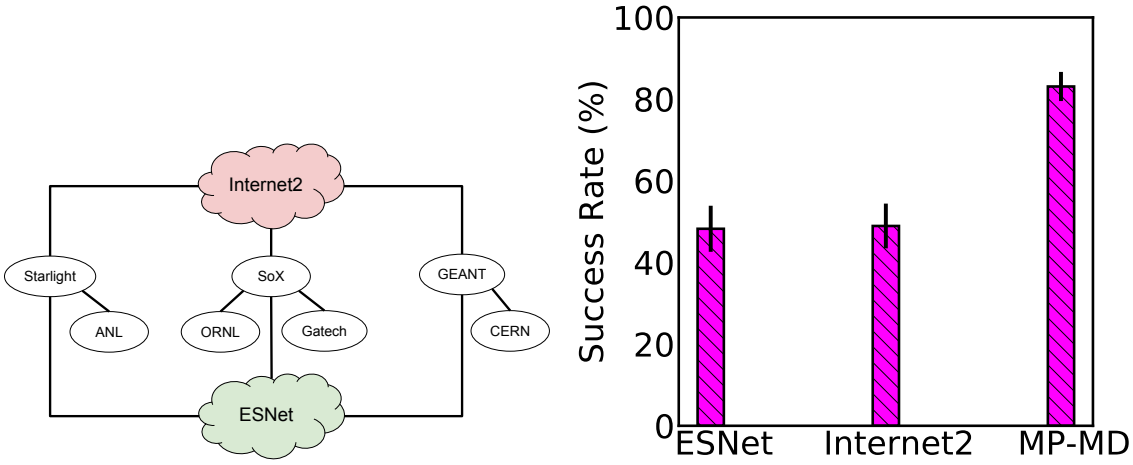
3.4.2 Multi-path, Multi-domain Advance Reservations

To evaluate our multi-path, multi-domain advance reservation we consider the topology depicted in figure 9(a). This topology is composed of four end sites (ANL, CERN, Gatech, and ORNL), connected to three regional networks (GEANT, Starlight, and SoX) where an SDX might reside. These three regional networks are further connected

to two R&E networks: ESNet and Internet2. To simulate our proposed concepts in this real network, we created a registry of advance reservations for both ESNet and Internet2. Each record on the registry represents a time window, and contains the available bandwidth (randomly generated) for every possible point-to-point connection. For our simulation we generate a random request composed of a time window, a required bandwidth, a source, and a destination. We send this request to both domains individually, and evaluate whether the domains have enough available resources. For our multi-path, multi-domain advance reservation service we evaluate whether the sum of the available bandwidth in both domains satisfies the request.

We assume a maximum bandwidth of 1 Gbps on both R&E networks. For simplicity, we represent the registry of advance reservations as a time series, and for each point in the time series we assign an available bandwidth value from a uniform distribution. The limitation of this approach is that the uniform distribution provides a lower bound for the success rate of a single domain advance reservation system (e.g., OSCARS [77], AL2S [58], and DANCES [52]). As already discussed in section 3.1.1, the reservation success rate can be improved with flexible reservation techniques [6, 101, 105], but these benefits are lost once we try to deploy single-path, multi-domain advance reservations. For this reason, we consider the parameters used in this simulation a fair representation of the single-path, multi-domain advance reservation.

Figure 9(b) shows that our multi-path, multi-domain approach has an 85% success rate when two independent paths are available, compared to approximately 50% success rate for the state-of-the-art (single path) approach. This 50% success rate of the state-of-the-art is a result of assuming a uniform distribution for the advance reservation registry. The multi-path, multi-domain approach evaluated in this simulation considers a full visibility scenario (section 3.2.2) with full offers (section 3.3.2) from our negotiation protocol.



(a) (b)

Figure 9: Simulation topology and results: (a) topology for multi-path, multi-domain advance reservation evaluation simulation; and (b) success rate for multi-path, multi-domain advance reservation evaluation compared to the state-of-the-art methods.

3.4.3 Negotiation Protocol Success Rate

To evaluate the success rate of the three variants of our negotiation protocol (i.e., equally splitting, partial offers, and full offers), we simulated a scenario in which an orchestrator can request advance reservations from up to four participant domains to compose a multi-path, multi-domain advance reservation. We chose four domains, because this is a reasonable number of multiple intercontinental paths between two sites as mentioned in section 3.1. For each participant domain, we generated a bandwidth schedule of 1000 entries that provide the available bandwidth at a given point in time. A user generates 100 random bandwidth requests within the time window defined by the aforementioned 1000 entries. We ran the simulation 32 times and took the averages for each scenario.

Figure 10 shows the results of our simulations. The horizontal line represents the success rate for a single domain, which is 49.56% under our assumptions. Any of our strategies outperform the baseline. In the worst case scenario (i.e., equally splitting under no visibility), the success rate of our orchestrator is approximately 58%. Under

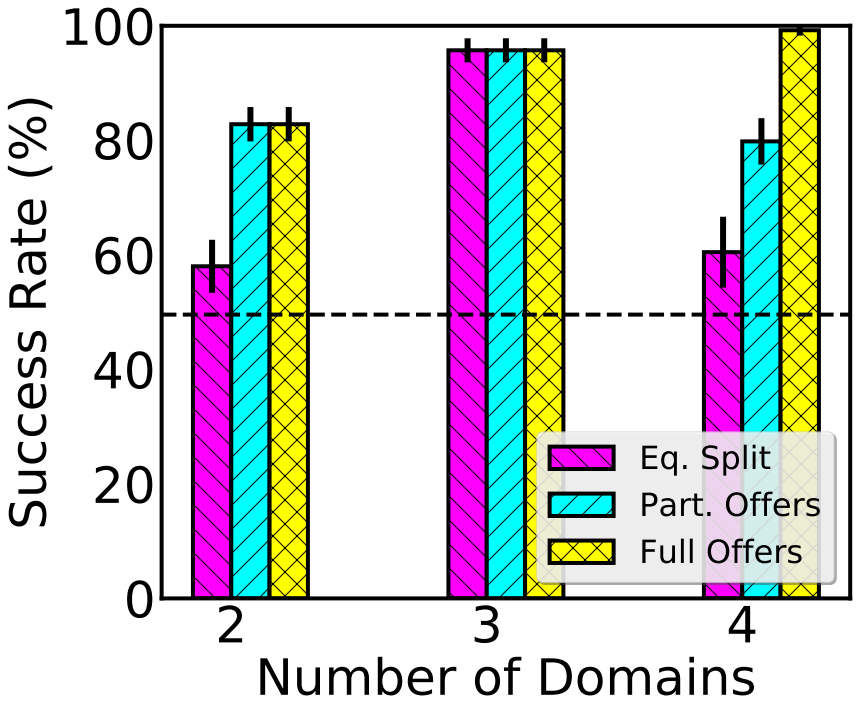


Figure 10: Negotiation protocol success rate for three bandwidth splitting strategies and up to four participant domains.

the best conditions (four visible domains), our orchestrator (with full offers) achieves approximately 99% success rate, or $2X$ improvement.

3.4.4 SDX Experimental Setup

Figure 11 shows the topology of our experimental setup, and table 5 shows the specifications of the equipment we used to build the testbed. Our testbed is composed of four virtual switch instances or bridges (bridge1, bridge2, bridge3, and bridge4) hosted by a Corsa DP2100 OpenFlow dataplane. Each bridge is connected to an instance of the AtlanticWave/SDX controller [33] (SDX1, SDX2, SDX3, and SDX4) running on a Docker container inside a Dell PowerEdge R220 server. This server also hosts our orchestration system: four instances of our orchestration agents (agent1, agent2, agent3, and agent4), and one orchestrator. Each orchestrator agent runs on a Docker container, and each one is paired with an SDX instance, while the orchestrator runs on another Docker container and communicates with the agents using gRPC.

We used two customized Supermicro server as the GridFTP endpoints. Each server runs a docker container with either a GridFTP server or a GridFTP client (globus-url-copy), an Open vSwitch (OVS) [85] virtual switch, and a Ryu SDN controller [84]. We added a delay of 45 ms on each server's network interface for a 90 ms RTT to emulate an intercontinental link. We tuned the TCP configuration of both endpoint servers for 1 Gbps link speed, 90 ms RTT, and parallel streams as recommended by ESNet's Linux Tuning guideline [36].

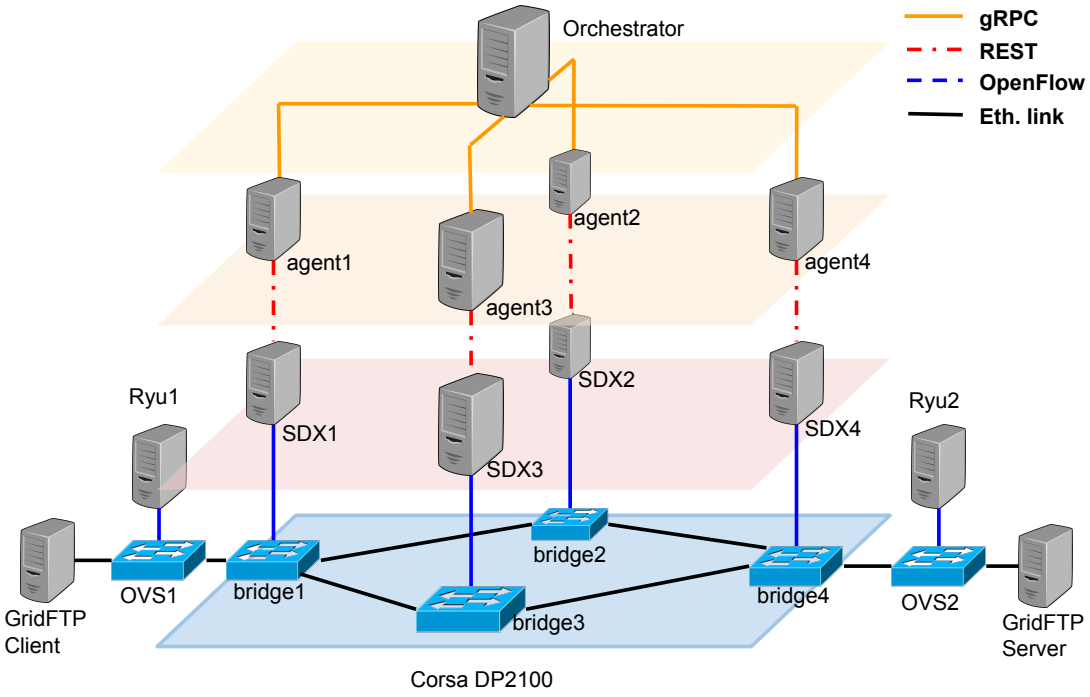


Figure 11: Experimental setup topology.

3.4.5 Throughput Baseline

Data Transfer Methods

We measured the throughput baseline of a data transfer over a single-path, multi-domain advance reservation versus a data transfer over a multi-path, multi-domain advance reservation on our testbed using two data transfer methods: GridFTP memory-to-memory (m2m), and GridFTP disk-to-disk (d2d) data transfers. We used iperf3,

Table 5: Experimental setup, equipment specifications

Equipment	**Specifications**
Corsa DP2100	OpenFlow 1.5, multiple flow tables, multi-context virtualization, 48 Gb packet buffer, 10 Gbps line-rate
Dell PowerEdge R220	Ubuntu Server 16.04, 16 GB RAM, four Intel(R) Xeon(R) CPU E3-1220 v3 @ 3.10GHz processors, four port Gigabit Ethernet card
Customized Supermicro	Ubuntu Server 16.04, 8 GB RAM, four Intel(R) Xeon(R) CPU X3430 @ 2.40GHz, two Gigabit Ethernet interfaces

a well-known bandwidth measuring tool as a reference. Figure 12 shows the results of performing the aforementioned data transfer over a 1 Gbps link with 90 ms RTT. For iperf3 and GridFTP memory-to-memory, we sustained the data transfer for five minutes, or the equivalent of transferring a 37.5 GB of data at line-rate over a 1 Gbps link, while for GridFTP disk-to-disk we actually transferred a 20 GB file, which is a reasonable size for a scientific dataset [26].

Figure 12(a) shows that iperf3 only reaches 514 Mbps of throughput for a single L2 tunnel of 1 Gbps of bandwidth, while GridFTP only reaches 488.56 Mbps and 426.72 Mbps of throughput for memory-to-memory and disk-to-disk, respectively. The reason for this low performance is that our endpoints are optimized for parallel TCP streams. As we see for two and four parallel TCP streams, iperf3 utilized 93.6% of the link (936 Mbps of throughput on average), and GridFTP memory-to-memory used 88.92% (or 889.24 Mbps on average). However, GridFTP disk-to-disk is only able to use approximately 67% (670.36 Mbps on average) of the link with parallel streams.

Figure 12(b) shows the throughput baseline after splitting the bandwidth reservation among two 500 Mbps L2 tunnels. For one and two TCP streams per tunnel,

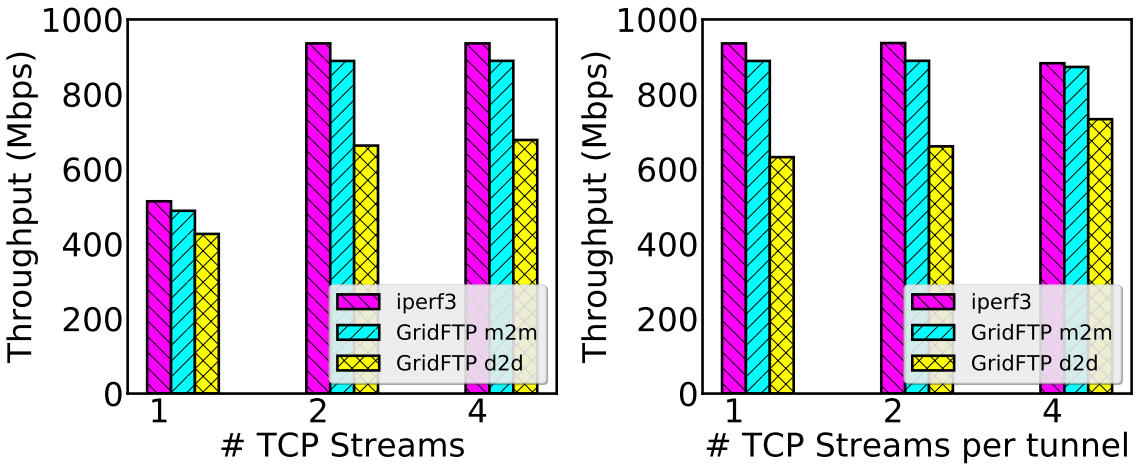


Figure 12: Throughput measurements while performing data transfers using iperf3, GridFTP memory-to-memory (m2m) and GridFTP disk-to-disk (d2d) over a 1 Gbps link with 90 ms RTT: (a) shows the baseline for a single L2 tunnel of 1 Gbps, and (b) shows the baseline for two L2 tunnels of 500 Mbps each.

iperf3 achieves 936 Mbps of throughput. However, it is only able to achieve 883 Mbps with four parallel TCP streams per tunnel. GridFTP memory-to-memory shows more consistent results, with 889.12 Mbps with one and two TCP streams, and 873.04 Mbps with four streams. On the contrary, GridFTP disk-to-disk obtains a slight improvement after using four TCP streams, achieving 733.44 Mbps of throughput, compared to the 632 Mbps and 660.8 Mbps obtained with one and two parallel streams respectively.

Number of TCP Streams

ESNet recommends the use of two or four parallel TCP streams for GridFTP data transfers. We verified that this recommendation holds true for our bandwidth splitting service by measuring throughput for a GridFTP memory-to-memory data transfer. We considered five bandwidth splitting approaches described in table 6. The main goal of the orchestrator in this scenario is to split a bandwidth reservation among two L2 tunnels, obtaining an aggregate bandwidth of 1 Gbps. For instance, one strategy is to split the bandwidth into two 500 Mbps tunnels. Another strategy is to split the

request into one tunnel of 100 Mbps and another tunnel of 900 Mbps.

Table 6: Splitting Strategies

Code	**Description**
SS1	Tunnel 1: 100 Mbps, Tunnel 2: 900 Mbps
SS2	Tunnel 1: 200 Mbps, Tunnel 2: 800 Mbps
SS3	Tunnel 1: 300 Mbps, Tunnel 2: 700 Mbps
SS4	Tunnel 1: 400 Mbps, Tunnel 2: 600 Mbps
SS5	Tunnel 1: 500 Mbps, Tunnel 2: 500 Mbps

Figure 13 shows that for two and four parallel TCP streams, the throughput of a data transfer stays very close to the no-splitting baseline of 889.24 Mbps. For one stream per tunnel, the throughput increases as the bandwidth splitting strategy is more balanced. This behavior can be explained from our observation in figure 12(a). The TCP stream using a tunnel with a larger bandwidth reservation cannot fill the pipe, because the endpoints are optimized for parallel streams. Meanwhile, the stream using the smaller reservation is limited, resulting in a poor overall performance. In the case of eight streams per tunnel, the throughput results are not optimal, as many TCP streams are competing for the same resources. These results are important because the orchestrator has to return meaningful recommendations to the end user for their data transfers to run optimally. For instance, given that two streams per tunnel provides optimal performance, our orchestrator should recommend the end user to four parallel TCP on her application, because the reservation was split among two tunnels. In the case of splitting the bandwidth among three tunnels, the orchestrator's recommendation should be six parallel streams.

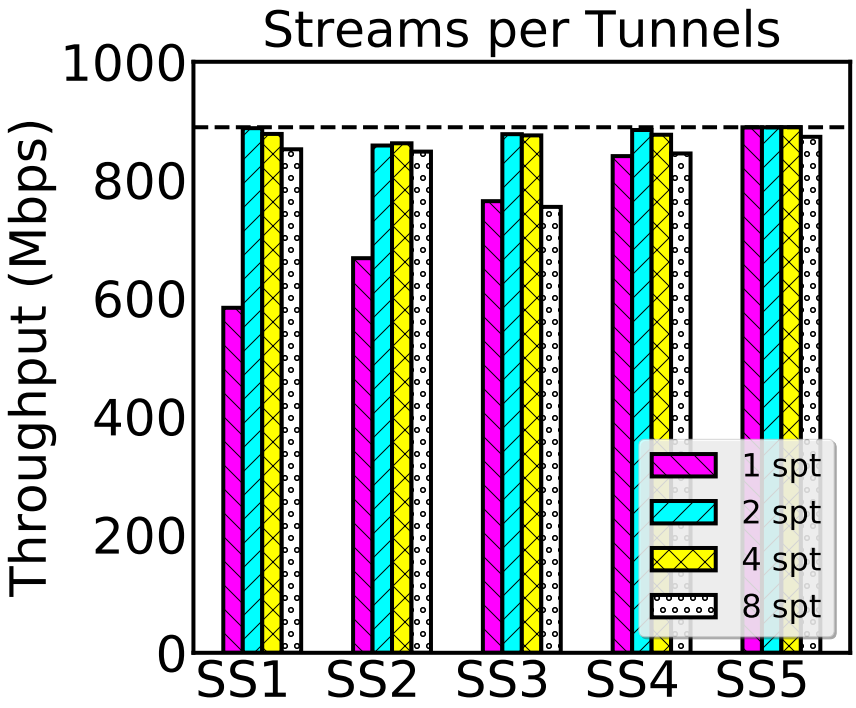


Figure 13: Effect of number of parallel TCP streams and bandwidth splitting strategies on throughput for a GridFTP memory-to-memory data transfer over a 1 Gbps link with 90 ms RTT.

3.4.6 Rule Provisioning Strategies

In this section we study the effects of several provisioning and bandwidth splitting strategies on the throughput of a GridFTP memory-to-memory data transfer over a 1 Gbps link with 90 ms RTT. We consider three provisioning strategies (as described in section 3.3.3): synchronized VLANs, unsynchronized VLANs, and coordinated VLAN. We also consider the same bandwidth splitting strategies described in table 6. Figure 14 shows the throughput measurement results for this experiment.

Figures 14(a), 14(b), 14(c) and 14(d) shows the results for one, two, four, and eight TCP streams per tunnel, respectively. Regardless of the provisioning or bandwidth splitting strategy, the *two streams per tunnels* approach provides the optimal performance, with throughput results close to the single path baseline (889.24 Mbps). In the worst case scenario, the maximum performance loss 3.47%. In the best case

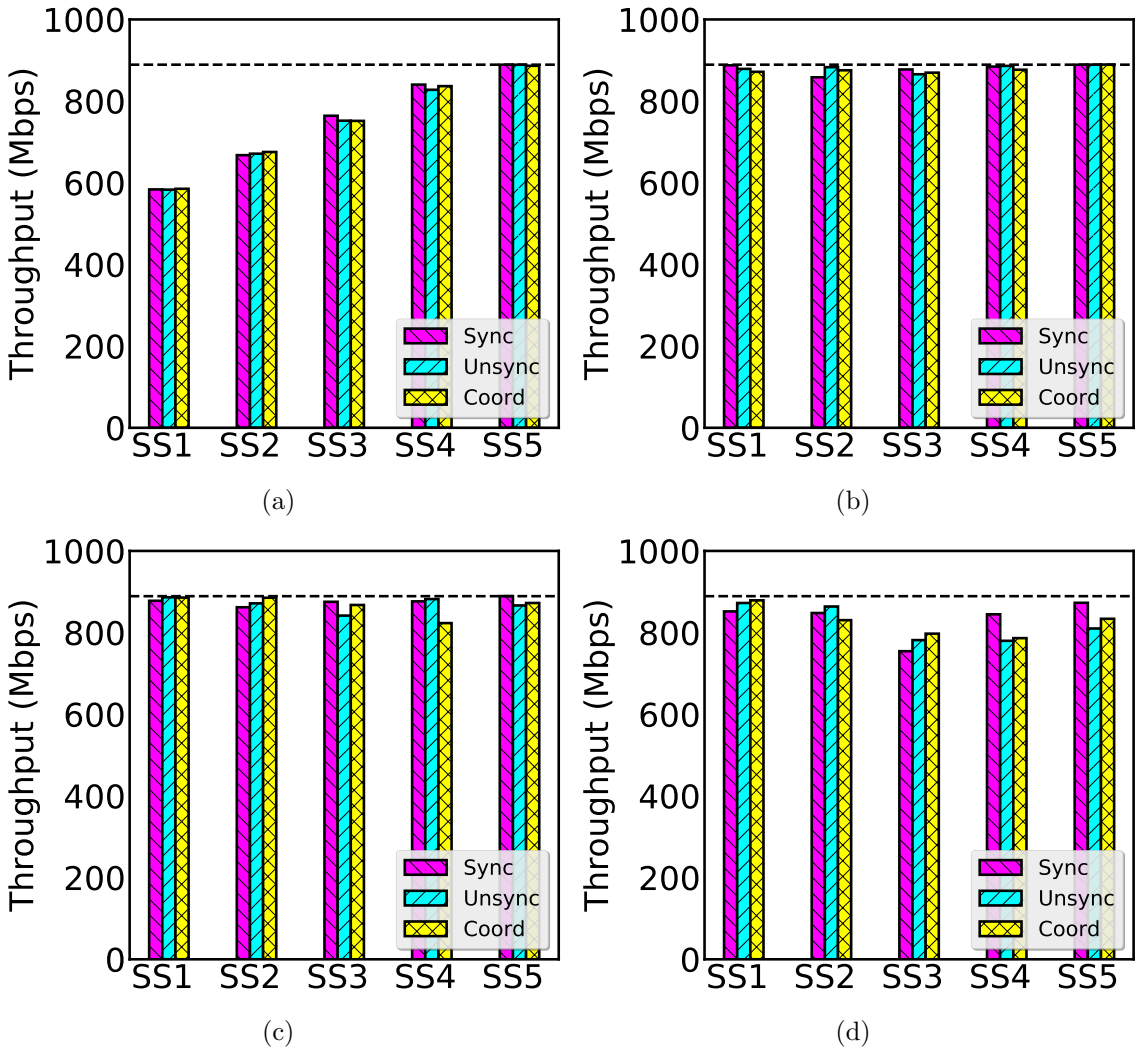


Figure 14: Effect of provisioning and bandwidth splitting strategies on throughput while sending a 20 GB file using GridFTP disk-to-disk over a 1 Gbps link with 90 ms RTT: (a), (b), (c), and (d) shows the results for one, two, four, and eight TCP streams per tunnel, respectively. We observe that two streams per tunnel is the recommended setting to achieve the optimal performance. The baseline for each scenario is represented as a horizontal dashed line.

scenario, we measured 280 kbps above the baseline. This might look insignificant, but in the context of a large data transfer that might last 24 hours, this means an extra 3 MB file can be transferred. Likewise, the *four streams per tunnel* approach provides close to optimal throughput results, regardless of the provisioning strategy. On the contrary, *one and eight streams per tunnel* strategies provide non-optimal results for the reasons already explained in section 3.4.5, and should not be considered

for production environments.

It is a well-known fact that OpenFlow rule provisioning add an extra delay to the transmission of the first packet of a flow. Nevertheless, we do not observe a significant overhead on throughput, because OpenFlow's delay is in the order of milliseconds, and the total transmission time for this experiment is five minutes. Furthermore, real-world data transfers might last hours, making OpenFlow's provisioning delays even more negligible. Although a significant difference between the three provisioning methods does not exist for the optimal configuration, we recommend the use of the *coordinated VLANs* approach. As mentioned in section 3.3.3, the *coordinated VLANs* provisioning strategy guarantees that all traffic of a single TCP flow traverses a single L2 tunnel. This is beneficial for troubleshooting and auditing purposes. On the other hand, the *synchronized VLANs* and *unsynchronized VLANs* are completely reactive, and do not introduce as much delay, because each OVS reacts the packets arriving to their interfaces. However, in the *unsynchronized* approach, or in the event synchronization is lost in the *synchronized* approach, the forward and return traffic of a single TCP flow might traverse two separate L2 tunnels. This situation complicates troubleshooting and auditing for multi-path, multi-domain advance reservations.

3.4.7 Oversubscription

In this experiment we measured the improvement factor for a GridFTP memory-to-memory data transfer, and a 20 GB GridFTP disk-to-disk data transfer over a 1 Gbps link with 90 ms RTT. From our baseline measurements, GridFTP disk-to-disk achieves at most 733.44 Mbps of throughput using four parallel TCP streams. Considering this observation, we hypothesize that by oversubscribing the aggregate reservation, we will obtain a higher throughput. For instance, we oversubscribed a 1 Gbps link by requesting two L2 tunnels of 600Mbps for an aggregate of 1.2 Gbps. Figure 15 shows the improvement factor for several percentages of oversubscription.

We observe that with 40% to 50% oversubscription, we obtain $1.12X$ improvement for GridFTP disk-to-disk, but anything below or above it produces lower improvement factors. Furthermore, there is no significant improvement for GridFTP memory-to-memory. For these reasons we do not recommend that the orchestrator oversubscribes physical links in the last mile, although additional resources are available in the WAN providers.

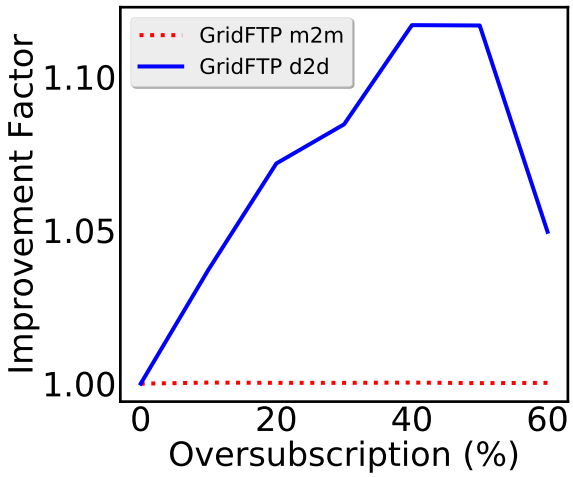


Figure 15: Improvement factor in GridFTP's average throughput for oversubscription of the physical, while maintaining multi-path, multi-domain reservations within limits. For instance, requesting two 600 Mbps L2 tunnels for an aggregate of 1.2 Gbps gives us 20% oversubscription on a 1 Gbps link.

3.5 Conclusions

In this chapter we presented an architecture for end-to-end service orchestration in multi-domain science networks that leverages SDXs for providing multi-path, multi-domain advance reservations. We implemented an orchestrator for multi-path, multi-domain advance reservations and an SDX to support these services. Our implementation uses an agent-based approach in which site agents communicate with a centralized orchestrator that serves as a single point of contact for end users. We developed a negotiation protocol that improves the success rate of intercontinental multi-domain reservation from approximately 50% when using single-path circuits

to up to 99% when four paths are available under the conditions presented in section 3.4.3. We evaluated our solution using GridFTP, which has multiple TCP flow splitting capability and is one of the most popular tools for data transfers in the scientific community. In our experiments, we tested our system under several conditions of bandwidth splitting ratios, SDN rule provisioning strategies, and number of GridFTP streams, and generated recommendations for the optimal performance of reservations. To confirm our simulation results and recommendations, we need to deploy our architecture in an actual science data network (e.g., ESNet 100GB SDN testbed, AtlanticWave/SDX or GENI) so as to collect real system results to confirm our results presented here.

Chapter IV

NOVEL NETWORK SERVICES FOR SUPPORTING BIG DATA SCIENCE RESEARCH

4.1 Introduction

Modern scientific instruments (e.g., particle accelerators, large telescopes, and genome sequencers) generate enormous amounts of data. These large datasets are analyzed at supercomputing centers, typically hundreds of kilometers away from the original research facility. For instance, a large telescope located in the Andes mountains in Chile, taking multiple gigabyte images to be transferred to the United States so that processing can be completed in time to distribute transient alert notifications will use a dedicated network connection between the two facilities. Figure 16 shows a map of the interconnection between the Large Synoptic Survey Telescope (LSST) in Chile and a supercomputer in the United States.

Network operators may take from days to several weeks to plan and provision a circuit over an R&E network, because these tasks are typically done manually [56]. To interconnect research facilities such as LSST with supercomputing centers across long distances, network operators deploy scientific networks or Research and Education (R&E) networks. However, the interface for requesting these types of reservations is very complex for domain-expert scientists who are not network operators. Arguably, many of these interfaces were developed *by* network operators, *for* network operators. Additionally, as reservations are defined by duration and bandwidth, the scheduling of resources is not flexible; that is, a reservation request will fail if the exact amount of bandwidth is not available within the specified time frame, which forces the scientist into a cycle of trial and error until a suitable time frame is found. Furthermore,

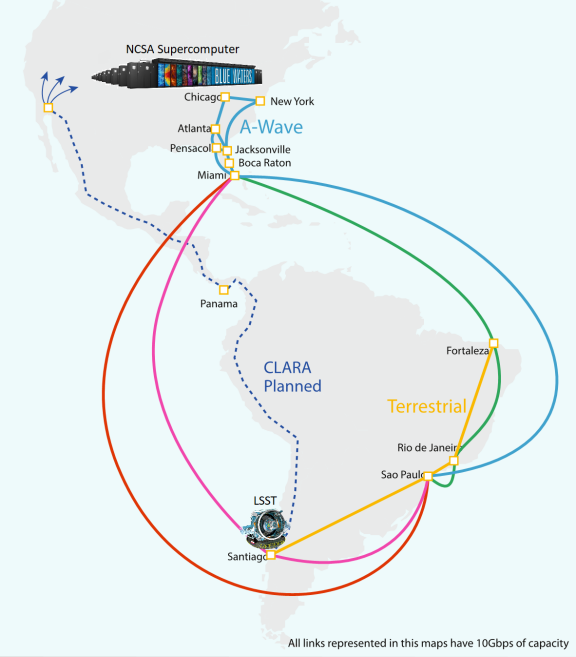


Figure 16: Map of the interconnection between the LSST in Chile and the NCSA supercomputer in the United States.

manual provisioning of these connections, which could take from several days to several weeks [56], is sometimes limited by configuration overhead, poor scalability, and poor testing interfaces [98].

To alleviate the shortcomings of current advanced reservation systems, we require an architecture for R&E networks that allows agile programmability of end-to-end network services over multiple domains, protects access to reservations from end-to-end, and provides flexible reservation capabilities. In this chapter we mapped our reference architecture (see Chapter 2) to AtlanticWave/SDX [21], an SDX controller currently being developed by Georgia Tech and Florida International University, as part of a multi-year project to create a distributed SDX over Sao Paulo in Brazil, and Miami and Atlanta in the United States. The main goal of the AtlanticWave project is to interconnect the aforementioned LSST telescope in Chile with supercomputers in the US. We investigate how our *U-O interface* will allow domain-expert scientists

and data workflow management systems to reserve network resources from a multi-domain SDX. This work was presented at Gateways 2017 (12th Gateway Computing Environments Conference) organized by the Science Gateways Community Institute (SGCI) [22].

4.2 Infrastructure Assessment of a Regional Science Network

In this section we present an assessment of Southern Crossroads (SoX) [95], an organization that provides high-speed, global connectivity, and other commodity services to the Southeastern U.S. Research and Education community. We surveyed nine network operators of participating institutions at SoX to understand their cyberinfrastructure at a high level, and also to understand their interactions with domain scientists. The following subsections show the responses to our questions and our analysis of these responses.

4.2.1 What kind of science research do you support, or are you planning to support in the future?

With this question, our goal was to understand which are the main research areas supported by SoX participants. Bioinformatics, genomics, and systems/networking research were mentioned at least twice. After that, we found a wide variety of topics such as neuro imaging, particle research from the Compact Muon Solenoid (CMS) at CERN, medical research, cloud and distributed systems, and engineering modeling.

4.2.2 What applications those scientists use?

The vast majority of respondents (seven out of nine) are unaware of what applications their scientists use. Two of them mentioned Globus as one of the applications used by scientists. These two respondents mentioned more applications such as FTP, SCP, GridFTP, Matlab, Galaxy Genomic pipelines, custom applications, Aspera Connect, among others. Furthermore, one of the respondents mentioned that many of these

scientists have their own IT staff. Nevertheless, we were able to confirm that many of the applications used by scientists are for data movement (e.g., FTP, SCP, GridFTP, Aspera Connect, and Globus).

4.2.3 Where are they connecting (inside and outside your network)?

Figure 17 show the answers of the participants to the question: *where are they connecting (inside and outside your network)?*, in reference to domain scientists. Interestingly, most of the responses indicate that scientists connect with external academic collaborators, then internal collaborators, and other SoX participants. Very few connect with external corporate collaborators.

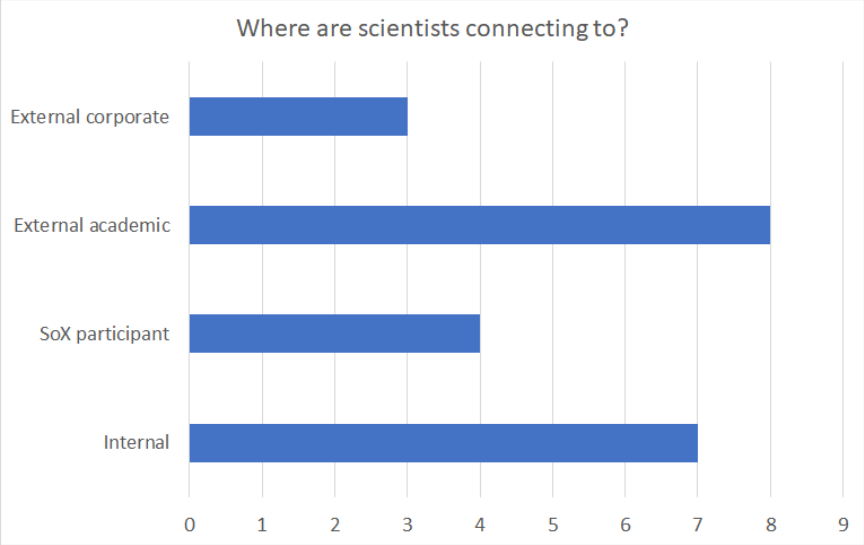


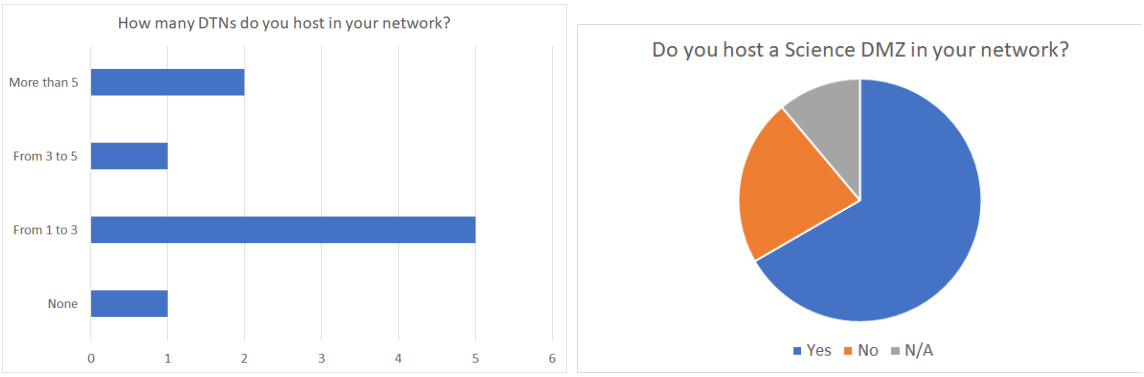
Figure 17: Question 3: Where are they connecting (inside and outside your network)? We asked participants to mark all options that applied.

4.2.4 Infrastructure Questions

To better understand the cyberinfrastructure of SoX participants, we asked the following questions:

1. How many data transfer nodes do you host in your network?

2. Do you host a Science DMZ?



(a) (b)

Figure 18: Infrastructure questions: (a) How many data transfer nodes do you host in your network? And (b) do you host a Science DMZ?

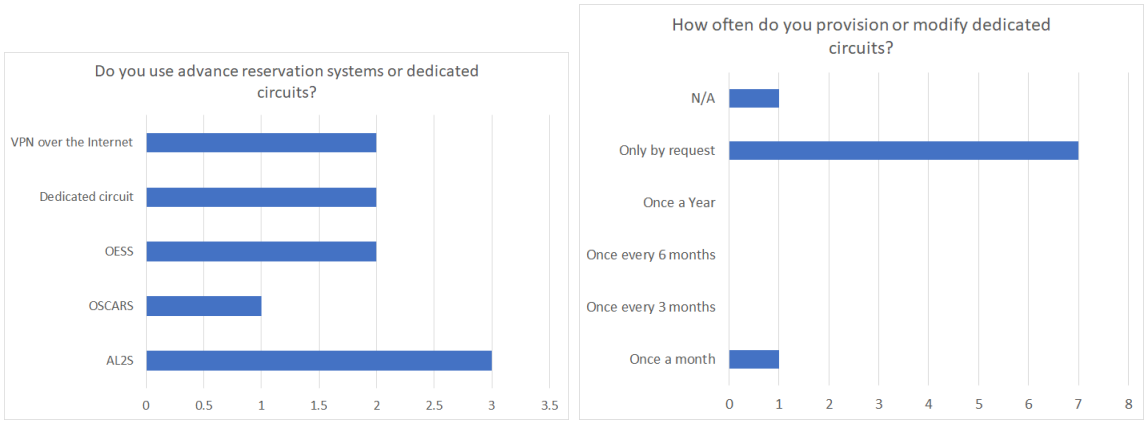
3. Do you use advance reservation system or dedicated circuit?

4. How often do you provision or modify dedicated circuits?

Eight out of nine respondents host a data transfer node (DTN) in their network (see Figure 18(a)), and six out of nine host a Science DMZ (see Figure 18(b)). Most of the respondents use advance reservation systems, with the advanced layer 2 service (AL2S) from Internet2 being the most popular (see Figure 19(a)). In general, network operators only modify dedicated circuits by request (see Figure 19(b)). These results show us that even at a regional level, bid data science research is becoming more relevant. Moreover, as the size of data sets increases and more external academic partnerships are established, there will be an increase demand of science network services.

4.3 AtlanticWave/SDX Architecture

As big data science research becomes more global, the need for science network services increases. In this section we present the high-level architecture of AtlanticWave/SDX, and the science gateway interfaces that allow domain-expert scientists to request network resources. The proposed AtlanticWave/SDX architecture is composed of the following components (see Figure 20):



(a) (b)

Figure 19: Infrastructure questions: (a) Do you use advance reservation system or dedicated circuit? And (b) how often do you provision or modify dedicated circuits?

1. Local controllers and local switches that reside at SDX domains (equivalent to the site controllers described in section 2.1).

2. An SDX controller that interconnects participating sites (equivalent to the orchestrator described in section 2.2).

3. Users (e.g., domain-expert scientists or network operators) and applications (e.g., data workflow management systems) that consume end-to-end services composed by an SDX controller.

4.3.1 SDX User Interface

Some users of AtlanticWave/SDX are domain-expert scientists whom in most of the cases do not have expertise in network operations, but still have to request reservations to expedite their data transfers. Additionally, scientists use data workflow management systems (e.g., Globus [45]) to automate the process of moving and sharing data across research facilities.

In the proposed AtlanticWave/SDX architecture, both scientists and applications request end-to-end science network services from the SDX controller by using the *Network Service Science Gateway* (mapped to our *U-O interface*, section 2.3.2) that

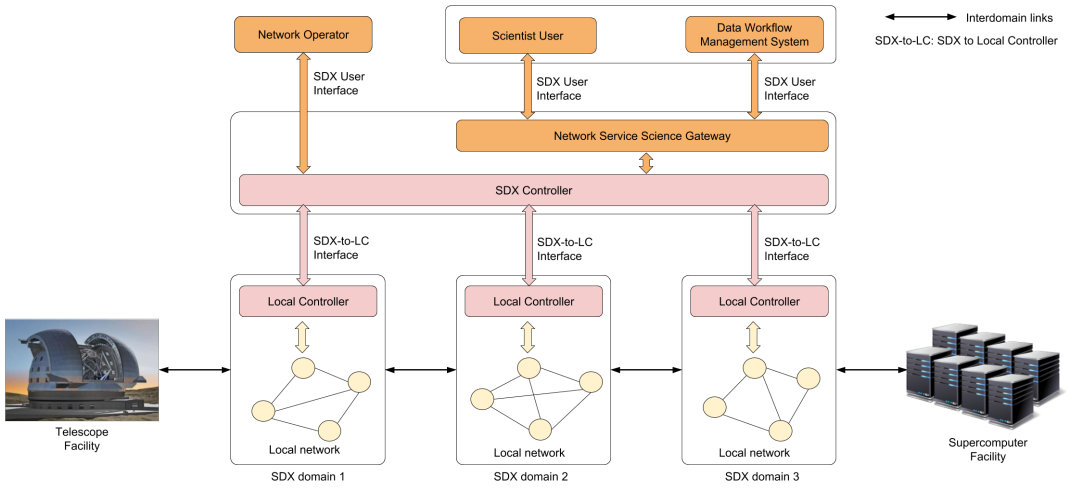


Figure 20: High-level architecture for AtlanticWave/SDX, with local controllers at three domains, and an SDX controller exposing services to users (e.g., domain-expert scientists, network operators, and data workflow management systems).

interfaces with and abstracts network infrastructure details. The SDX user interface allows a scientist or a data workflow management system to request services from AtlanticWave/SDX. To overcome rigid interfaces that only allow users to request a certain amount of bandwidth during a limited amount of time, we propose to describe a request based on data set size and a deadline for finishing the transfer. Details of the network parameters should be left for the SDX controller to decide. A domain-expert scientist, for example an astrophysicist, only wants to transfer X amount of gigabytes before a certain deadline. The *SDX user interface* may include negotiation capabilities that allow the SDX controller to provide an optimal solution to a user request, given the user constraints and the network state. Table 7 illustrates both a reservation request for a network operator of the AtlanticWave/SDX, and a request for a domain-expert scientists request in JSON format, and how the *SDX user interface* abstracts the complexities of the network for the experimenters.

Table 7: SDX user request samples in JSON format

Network Operator Request	Domain-expert Scientist Request

<pre>{"l2tunnel":{
"starttime":"2017-10-23T23:20:50",
"endtime":"2017-10-25T23:20:50",
"srcswitch":"atl-switch",
"dstswitch":"mia-switch",
"srcport":5,
"dstport":7,
"srcvlan":1492,
"dstvlan":1789,
"bandwidth":1000}}</pre>
	<pre>{"dtntunnel":{
	"quantity":"7TB",
	"deadline":"2017-10-25T23:20:50",
	"srcdtn":"gt-dtn",
	"dstdtn":"fiu-dtn"}}</pre>

4.3.2 Authentication and Authorization

Authentication and authorization are important topics in the AtlanticWave/SDX architecture. The SDX controller authenticates participant entities and users or applications coming from several independent administrative domains. Then, the SDX controller requires authorization before it issues any provisioning requests to participant domains. Digital certificates may be used for mutual authentication between the SDX controller and local controllers, and a federated identity management system (e.g., Shibboleth [94]) may be used for the SDX controller to authenticate users and applications. Another approach may be to integrate AtlanticWave/SDX with existing systems such as Globus. Under these models, users are not required to have an account at each location, but they are authenticated using their institutional credentials. We propose to enforce authorization by installing policies on each entity (i.e., participant R&E networks) that define what users are allowed to do, depending on their roles or affiliation (e.g., institution, project, or individual).

4.4 Use Cases

4.4.1 Simplifying Current Science Network Services

Astronomers use instruments that generate sets of data on the scale of gigabytes, and they need to transfer these sets of data in a few seconds to be processed in a remote facility. To guarantee a high-throughput network connection between facilities, an experimenter may reserve network resources through AtlanticWave/SDX. After verifying the domains involved in this end-to-end network service, the SDX controller (see Fig. 20) contacts each local controller (using an equivalent of our *D-O Interface*, section 2.3.1), querying whether a path with the constraints specified by the experimenter is possible. Each queried domain does not have global knowledge about the end-to-end path, but they can commit to guarantees of their portion of the end-to-end path. The SDX controller then evaluates whether a path that meets the end-to-end requirements can be formed. Otherwise, the SDX controller will try to find an alternative path, or will try to negotiate a path with alternative constraints.

4.4.2 Future Generation Science Network Services

The proposed AtlanticWave/SDX architecture will allow science networks to provide more flexible services. For instance, let us consider an experimenter who wants to move telescope data every morning at 6:00 AM. Instead of reserving a dedicated connection for the experiment (that could last years), the scientific network may expose a bandwidth calendaring service in which the experimenter selects at what hours of the day she will need the reservation. We could take this use case one step further by correlating weather data with previous data transfer patterns, and suggest to the experimenter the optimal time frames for upcoming data transfers. For example, on a cloudy day, AtlanticWave/SDX will avoid provisioning a network reservation, as the telescope's view was obscured by clouds.

Another example is to use machine learning (ML) on data transfer patterns and

historical reservation data to create a predictive reservation service. In this case, the SDX will suggest a predefined reservation to an experimenter based on previous reservation and real usage patterns. The experimenter will be able to confirm or decline the reservation. A more aggressive approach is to provision the reservation with a lower priority, and wait for traffic to be sensed on the network. If no traffic is sent before a preset threshold, the reservation is eliminated.

In another scenario, a dataset is hosted in several locations. An experimenter only needs to know the name of the needed dataset. The *SDX user interface* of AtlanticWave/SDX will locate the closest repository and request a dedicated network connection between the repository facility and the experimenter's facility. Moreover, if the closest facility to the experimenter is congested, AtlanticWave/SDX may negotiate a more distant facility with better network conditions to provide the data set. We could take this example one step further and expose an energy efficiency score, and the SDX controller may be able to compose end-to-end green paths as a novel scientific network service.

4.5 AtlanticWave/SDX Prototype

In this section we present the prototype of the AtlanticWave/SDX controller [33], and how this matches with our reference architecture (Chapter 2.) AtlanticWave/SDX is written in Python, using the Ryu SDN Framework [84] as an OpenFlow [75] speaker, and has a REST API and web application for management. The controller is divided into three components, that will run at the initial three locations of the prototype (i.e., Atlanta, Miami, and Sao Paulo). The main three components of the AtlanticWave/SDX controller are the following:

- **Participant Interface(s):** The participant interfaces are where network operators and scientists install rules that dictate how network flows behave. This

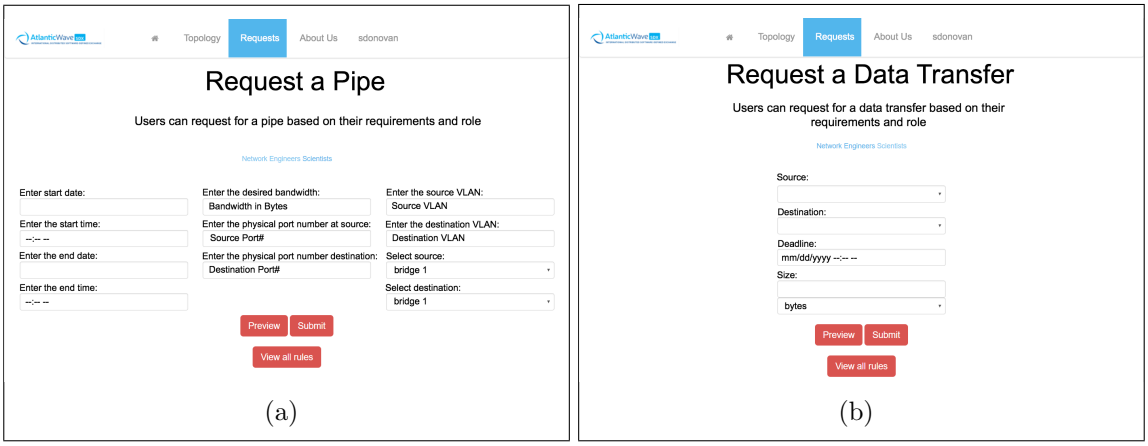


Figure 21: AtlanticWave/SDX Web portal for requesting science network services: (a) network operator view, and (b) domain-expert scientist view.

interface corresponds to the *network science service gateway* described in Section 4.3, and the *U-O Interface* described in Section 2.3.2. For the initial prototype, AtlanticWave/SDX provides a Web portal that allows users to create and install rules (see Figure 21).

- **SDX Controller:** The SDX controller is responsible for authentication and authorization of participants and local controllers, taking rules from the participant interfaces and breaking them down into per-location rules for the local controllers, handling federation challenges from many participants installing rules on a shared network, and providing an interface for the participants. This is the equivalent of the orchestrator in our reference architecture (see Section 2.2)

- **Local Controllers:** Each location in the AtlanticWave/SDX will have a local controller that controls the local switch(es). The local controller has one main job: take the abstract rules from the SDX controller, and translate them to a switch friendly protocol (e.g., OpenFlow). The local controller also bootstraps the configuration of the switch to establish connectivity between the local controller and the SDX controller. This is an instantiation of our Site controllers

in Section 2.1.

We implemented Shibboleth from Internet2 for handling authentication at the front end of the AtlanticWave/SDX controller. It is important to note that Shibboleth is not used outside of North America. As a result, other identity and access management (IAM) systems need to be tested. For example, eduGAIN [41], developed by GEANT, is the standard cross-domain IAM system used outside of North America.

4.6 Conclusions

In this chapter we presented an interface that allow domain scientists and data workflow management systems to reserve resources of a scientific network. We assessed the cyberinfrastructure of SoX by surveying nine network operators of participant institutions. From our survey, we observed that even at a regional level, big data science research is becoming more relevant, and scientist will require more science network services as they collaborate with more external partners. We mapped the components of our reference architecture to AtlanticWave/SDX, an existing SDX controller. The AtlanticWave/SDX controller exposes network services that domain scientists and data workflow management systems can easily consume through the *user/application interface*. We provided use cases that illustrate how domain-expert scientists can use the interfaces of AtlanticWave/SDX for easily requesting end-to-end network services. Furthermore, we proposed future generation science network services such as bandwidth calendaring, predictive reservation services, and green path reservation. We also described the initial prototype of the AtlanticWave/SDX controller, with special emphasis on the user/application interface, and the authentication mechanism of the AtlanticWave/SDX Web portal. A logical next step for this work would be to conduct science data network user studies to determine if the interface we implemented is in fact easier to use and more intuitive than the previous interfaces.

Chapter V

AUDITING AND ACCESS CONTROL FOR SOFTWARE-DEFINED EXCHANGES

5.1 Introduction

Recently, users and operators of Research and Education (R&E) networks have been looking for ways to allow multiple independent administrative domains to share computing, storage, and networking resources in an agile and programmatic way. By taking advantage of virtualization of computing and storage resources, software-defined networking (SDN), and software defined radio (SDR), researchers are building a new kind of cyberinfrastructure referred to as software-defined infrastructure (SDI) [88]. A central point of the SDI is a software-defined exchange (SDX), a meet-me point or marketplace where independent administrative domains can exchange computing, storage, and networking resources. The SDX also provides enhanced visibility over individual data streams along with control over those streams. Moreover, they can be customized to support specific workflows of domain science research.

Taking into account the resources exchanged (e.g., computing, storage, and networking), the spectrum of categories for an SDX ranges from networking exchanges to cloud-service exchanges [20]. A specialized instance of a network SDX is a software-defined Internet exchange point [49]. This type of SDX allows participants of an Internet exchange point (IXP) to configure networking policies in the fabric of the IXP by using SDN technologies. Similarly, R&E exchange points are introducing SDN in their infrastructure to allow network operators to provision network policies over multiple independent administrative domains [56, 21]. The goal in this case is to automate the provisioning of existing networking services in research and education

networks such as advanced reservation of Layer-2 circuits and to enable new services such as the interconnection of SDN islands.

The federated nature of R&E exchange points is based on trust between participant domains. However, an old adage says "trust, but verify", so a responsible network operator wants to verify if his or her policies have been enforced by the SDN domains participating on an SDX. Moreover, some participants of the SDX do not want to reveal internal topology information while still proving that they correctly deployed the requested policies. For these reasons, we propose FAS (Federated Auditing for SDX) [19], a federated auditing framework that allows an SDX to verify the following conditions: (1) whether configurations requested are correctly enforced by participant SDN domains, (2) whether configurations requested are correctly removed after their expiration time, and (3) whether configurations exist that are performing non requested actions. Furthermore, these verifications are conducted without revealing internal information about the participating SDN domains, and regardless of the multi-domain architecture used.

Another challenge of these cyberinfrastructures is that advance network reservation systems identify each connection by coarse-grained attributes such as endpoints (e.g., an IP address or an interface of a WAN border router), requested bandwidth, the start time, and the end time [98]. However, a major problem with using such coarse-grained attributes to identify a network reservation is that an unauthorized user or application behind the point of ingress could consume the reservation, affecting the performance of legitimate users or applications. We present here a novel network architecture that provides advance reservation access control by leveraging SDN and token-based authorization.

The contributions of this chapter are as follows:

- A federated auditing framework for configuration verification in an SDX and a communication protocol that allows an SDX to query participant domains

without exposing internal information.

- A system that uses SDN and tokens to strongly bind an end-to-end flow to the user or application that requested the reservation.

5.2 Federated Auditing for Software-Defined Exchanges (FAS)

5.2.1 Background

Federated Auditing Systems

A federation is a group of independent networked systems that share and exchange data under standardized agreements. The best example of a federated system is the Internet, and one of its most challenging problems is to diagnose routing problems. To make this task easier, Teixeira and Rexford [97] proposed Omni, a distributed system that uses passive measurements of BGP changes inside an autonomous system (AS), and allows ASs involved in a route change to collaborate in identifying problems. Similarly, Shanmugasundaram et al. proposed ForNet [92], a distributed network logging mechanism that aids digital forensics over wide area networks. Both Omni and Fornet have local servers that collect network information within the domain or AS. These servers then cooperate to identify a routing problem or a cyberattack for Omni and Fornet respectively.

Security Considerations for SDXs

SDN provides programmability and agility in network operations by centralizing the control plane. Moreover, this centralized control allows SDNs to dynamically alter network topology, routing, and even security policies on the fly. However, this centralization also introduces new attack vectors. For instance, anyone gaining access to the controller software essentially gains the keys to the kingdom, and thus, gains control over the entire network [65]. Likewise, forged or spoofed traffic flows can potentially be used to fool switches and controllers into installing erroneous or malicious flow rules [65]. Other concerns involve attacks against the control plane communication

itself and assurance of trust between controller and management applications. Of course in the context of multi-domain SDNs, the problem of finding SDN misconfigurations is similar to troubleshooting a routing problem in the Internet or conducting a forensic investigation over multiple domains.

To illustrate an attack scenario, let us consider a centralized SDX, in which an SDX controller orchestrates various SDN controllers in separate domains. The SDX controller must trust participating SDN controllers to deploy requested configurations on their respective domains. In this scenario, a compromised participant SDN controller or a malicious or uncooperative network operator might ignore the configuration request from the SDX controller or install malicious or incorrect flow rules in their corresponding data plane that puts network traffic from other participating domains at risk. For instance, a malicious participant SDN controller might duplicate and divert flows to eavesdrop on communications, alter data, or introduce man-in-the-middle attacks against unsuspecting domains. We assume that the switches in the data plane are trusted and have not been compromised by an attacker.

The NSF report from the workshop on software-defined infrastructure (SDI) and software-defined exchanges (SDX) [88] identified network slicing as a promising way of securing SDI and SDX. A network slice is a logical instantiation of a physical network that provides the isolation of user traffic. The authors emphasize that to achieve secure slicing, SDX slices should incorporate admission control, secure slice provisioning, unforgeable slice identifiers across domains, and verification and auditing mechanisms. This also assumes that SDN controllers are capable of isolating and microsegmenting network applications and possess the ability to build dynamic access control lists (ACLs) [90]. Despite the isolation provided by SDX slices, a compromised participant SDN controller could still install malicious rules in its data plane. Thus there is a need for auditing that allows the SDX controller to verify the correctness of configurations deployed by each participant domain.

5.2.2 System Architecture

A federated auditing system for an SDX should allow the verification of the following conditions: (1) whether configurations requested to the SDX are correctly enforced by participant SDN domains, (2) whether configurations requested are correctly removed after their expiration time, and (3) whether configurations exist that are performing non requested actions. Additionally, these verifications should be conducted without revealing internal information about the participant SDN domains to the SDX. Finally, the auditing system must be separated from the domain-SDN controller (equivalent of the Site controller described in Section 2.1), to avoid a compromised controller corrupting the records of the auditing system.

Considering the requirements listed in the previous paragraph, we designed the architecture of FAS (Federated Auditing for SDX) to be agnostic of the overall multi-domain architecture (see Fig. 22). FAS is composed of the following components:

1. Control plane auditor: An independent auditing system in each participant SDN domain that records the communication between the SDN controller and SDN switches.

2. Data plane auditor: An auditing system in each participant that records selected traffic in the data plane.

3. Domain FAS agent: a module in each domain that coordinates auditing elements (i.e., control plane and data plane auditors).

4. SDX FAS agent: a centralized auditing orchestrator at the SDX.

5. FAS protocol: a communication protocol that allows the SDX to query FAS agents for configuration verifications, while keeping a participant's information undisclosed.

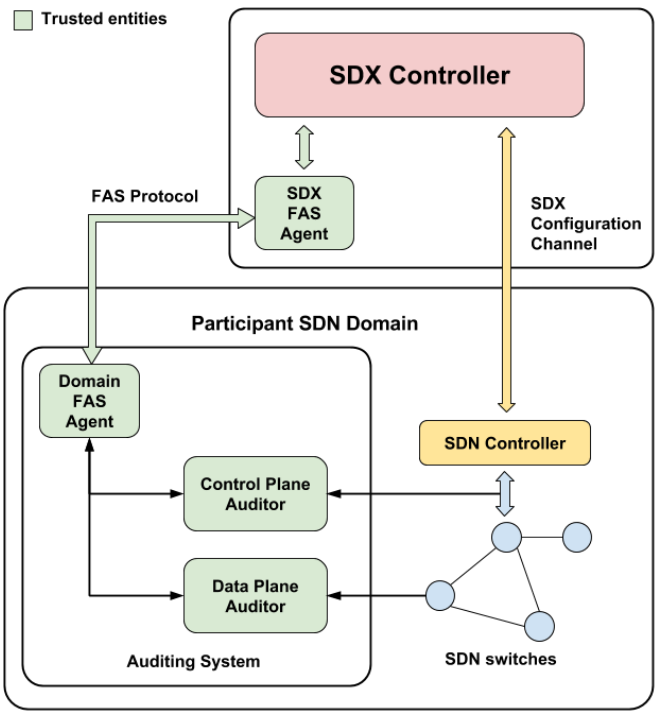


Figure 22: FAS Architecture showing a participant domain, auditing system with control plane and data plane auditors, and FAS agents communicating through the FAS protocol. Trusted entities are depicted in green

The control plane auditor module keeps records of the communication between the SDN controller and SDN switches within a domain. Similarly, the data plane auditor keeps a record of the data plane actions. Both control and data plane auditors may be implemented with current technologies such as port mirroring and a Wireshark sniffer, sFlow [91] or the Bro network security monitor [100] (implementation options will be discussed in more detail in Section 5.3.5). The SDX FAS agent allows an SDX to query participant domains and verify if configuration requests have been correctly enforced after provisioning or correctly removed after expiration. The domain FAS agent allows a participant domain to reply to these queries without revealing internal information such as topology, address space or traffic patterns. A domain FAS agent takes the information collected by control and data plane auditors and constructs a reply that provides the necessary information to the SDX while maintaining internal network

details confidentiality. Additionally, a domain FAS agent collects data plane active flow data and compares this with control plane audit data to detect non requested actions. Finally, the FAS protocol allows agents to establish auditing relationships and exchange query and response messages.

Management of Federated Auditing System

We propose that the federated auditing system be managed by an SDX operator in collaboration with participant domain network operators. The SDX operator receives notifications from the SDX FAS agent. After receiving a notification from the SDX FAS agent, the SDX operator contacts the network operator of the originating domain to start a remediation process. The SDX FAS agent will test rules right after a configuration and right after a removal for the most conservative auditing approach. A more aggressive auditing approach will require active polling with a frequency determined by the duration of the reservation and the accuracy required.

Trust within the FAS Architecture

Trust between FAS agents at the SDX and across participant domains is required for the FAS architecture to work properly. Furthermore, trust between the auditing entities and the participant domain is required. Fig. 22 depicts in green the trusted entities of the FAS architecture, which are an extension of the SDX into participant domains. The domain FAS agent subsystem should be deployed and managed by SDX operators or a designated trusted point of contact from the participant domain. Separation of roles should also be enforced. Specifically, the operator of the FAS system should not be the same person that operates the SDN controller, as we want to avoid collusion between these two roles, and mitigate the possibility that they might be used to cover the tracks of malicious activity.

The FAS architecture creates a new chain of trust that follows the flow of data depicted in Fig. 23, that starts in a control or data plane auditor and ends at the SDX

FAS agent. However, a domain administrator still wants to verify that the domain FAS agent is not collecting any additional information from the domain controller and/or switches. The SDX operator can provide read-only permission to the domain administrator for the purpose of verifying what information is collected by a domain FAS agent. Moreover, both parties can coordinate the installation of rules that limit the type and amount of information that a domain FAS agent can collect. Finally, the domain FAS agent is designed to work with the minimal necessary information for responding to an SDX query. This design will be discussed more in detail in section 5.2.2.

FAS Protocol Design

The FAS architecture framework collects evidence of per-domain SDX configurations and aggregates an end-to-end configuration records for verification. Fig. 23 illustrates the flow of audit data from a single control or data plane element to the SDX FAS agent. The FAS protocol enables the collection of configuration evidence for further aggregation at the SDX.

The FAS protocol is composed of three types of messages: query, response, and notification. The query message allows an SDX FAS agent to query configurations to participant domain FAS agents; the payload contains a token representation of the rule that the SDX wants to verify. The response message allows a domain FAS agent to reply to SDX queries; the response is typically YES (flow rule exists), NO (flow rule does not exist), and NOAUTH (not authorized to query this). Finally, the notification message allows a domain FAS agent to send notification to an SDX FAS agent without a previous query from the SDX; the payload typically contains a flow rule in violation within the configured policy. This rule is abstracted to a domain level, in order to avoid disclosing internal topological information, but representative enough to allow a domain administrator to start an investigation after receiving a notification from the

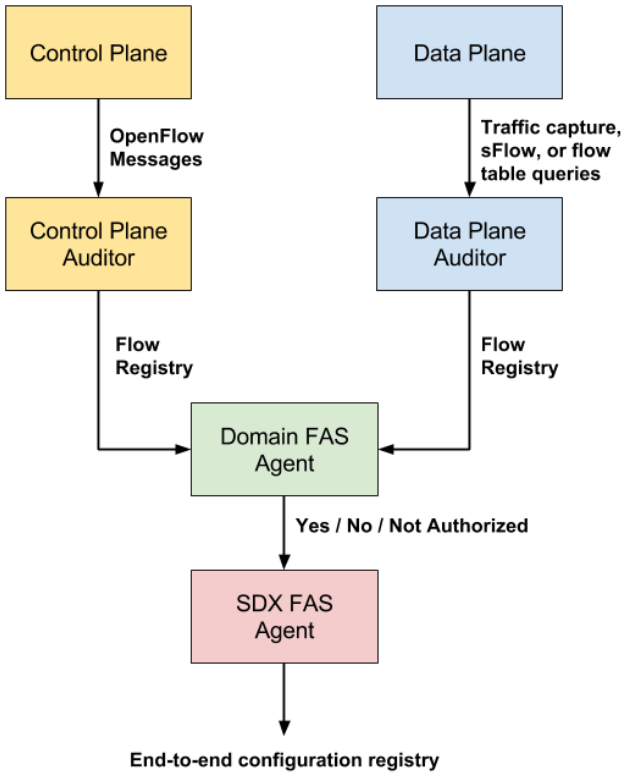


Figure 23: Flow of data in the FAS framework architecture from the control and data plane elements to the SDX. The chain of trust of the FAS architecture follows this flow of data

SDX operator. The correct balance between domain-level abstraction and explicitness of the notification is out of the scope of this work.

FAS Agent Design

FAS agents are designed to collect configuration evidence and coordinate evidence sharing over multiple domains. A domain FAS agent collects information from network elements in the control and data planes, creates flow registries from the network information collected, and replies to audit queries. Likewise, an SDX FAS agent queries domain agents for building an end-to-end flow configuration verification.

To maintain confidentiality of internal configurations, an SDX FAS agent is only allowed to ask if a configuration exists, and the response to this query is either *yes, no, or not authorized* as shown in Section 5.2.2. The answer is then compared to

the lifetime of the SDX request (i.e., start time and end time). If the SDX FAS agent gets a positive answer within the lifetime of the request, the rule was properly configured. Similarly, if the answer is negative outside the lifetime of the request, the configuration was properly eliminated (See Algorithm 4).

Algorithm 4 Verify if configuration exists

```
FR ← Set of domain flow rules
fr ← Flow rule to verify
if Authorized then
  if fr.start_time < NOW < fr.end_time then
    if fr ∈ FR then
      return RuleConfigured
    else
      return NOFOUND
    end if
  else
    if fr ∈ FR then
      return NOFOUND
    else
      return RuleEliminated
    end if
  end if
else
  return NOAUTHZ
end if
```

Once user traffic is detected following behavior that is different from the flow rules requested, the data plane auditor sends a notification to the domain FAS agent. Then, the domain FAS agent compares this information with the control plane configuration audit. If a flow detected on the data plane does not have a counterpart configuration in the control plane, the domain FAS agent has detected an unauthorized flow rule, and sends a notification to the SDX FAS agent (See Algorithm 5).

FAS Workflow

Fig. 24 shows the workflow for a user configuration request and subsequent audit verifications using the FAS framework. The detailed workflow is as follows:

Algorithm 5 Unrecognized flow notification

```
FR ← Set of domain flow rules
f ← New flow detected
if f ∈ FR then
  ActiveFlows++
else
  SendAlert()
end if
```

1. A user requests a configuration from the SDX controller.

2. The SDX controller sends requests configurations to each participant SDN controller via the SDX configuration channel.

3. Each participant SDN controller provisions its corresponding configuration in its respective domain.

4. The SDX controller requests an audit through the SDX FAS agent. The SDX FAS agent contacts each domain FAS agent through the FAS protocol.

5. Each domain FAS agent processes the request by analyzing its internal auditing system, and responds to the SDX FAS agent through the FAS protocol.

6. When the configuration request expires, steps 2-5 on the workflow will be repeated.

5.2.3 FAS Proof-of-Concept Evaluation

To evaluate the FAS architecture, we built a proof-of-concept (PoC) testbed in the GENI platform [9]. Our testbed, depicted in Fig. 25, is composed of two OpenFlow-enabled Open vSwitch (OVS) [85] virtual switches connecting two hosts, a Ryu SDN controller [84], a domain FAS agent that includes a data plane auditor, and an SDX controller.

For our implementation, we used the NSI (network service interface) [83] protocol as the model of an SDX configuration protocol. NSI allows users to request network

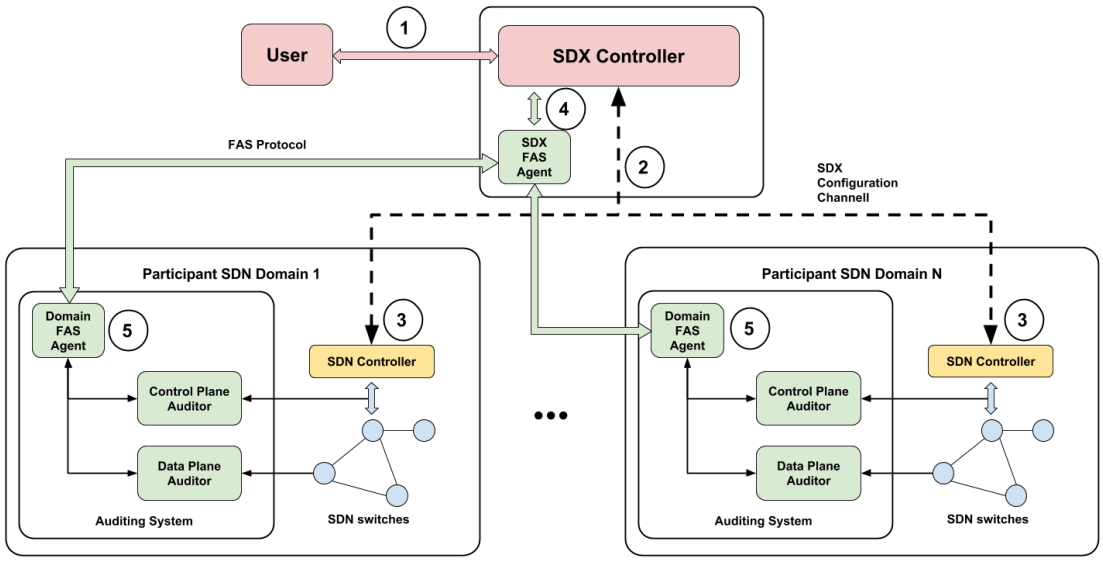


Figure 24: FAS workflow for a user configuration request and subsequent audit verifications using FAS

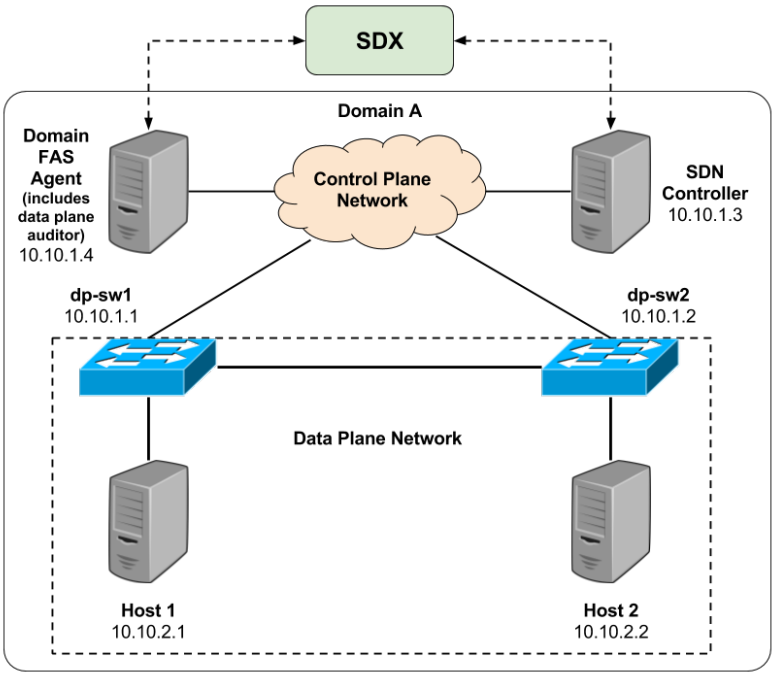


Figure 25: FAS testbed implemented on the GENI platform

services from one or more network service provider. In its most basic form, NSI allows experimenters to establish Layer-2 circuits over multiple domains. In our PoC, an SDX receives an end-to-end multi-domain configuration request from a user, breaks

down this request into several configurations for the domains involved and sends configuration requests to each domain. Each domain is in charge of implementing the requested configuration internally, by configuring each OVS in its domain.

The end-to-end configuration audit follows the reverse process of an SDX configuration. In our implementation approach, we enabled a RESTful API that allows the domain FAS agent to query the flow table of each OVS in the domain, and creates flow registries. These flow registries are then used to respond to SDX queries by composing domain level policies. Finally, the SDX FAS agent takes each domain response and composes an end-to-end response that verifies a configuration requested by a user.

5.2.4 Discussion

The implementation of our PoC sheds some light on the complexity of integrating a multi-domain auditing system as FAS. At each stage of the audit data flow, we had to make decisions that tightly coupled our implementation of the auditing system with the actual implementation of the networking infrastructure. For instance, the REST API that allows a domain FAS agent to query the flow tables of domain switches takes advantage of the *ovs-ofctl dump-flows* command of OVS. A better approach is for the control plane auditor to track flow entries querying the switch directly, out-of-band, not leveraging the same OVS commands. Similarly, the domain FAS agent has to use the same logic used by the domain SDN controller to configure the data plane switches, in order to compose the domain-level configuration from the individual flows.

The FAS architecture proposes auditing of the control plane of each participant domain. Security best practices recommend using OpenFlow transport over TLS, which makes traffic capture challenging. Developers of off-the-shelf solution such as Bro have not willing to implement an OpenFlow analyzer [99]. This will force the

network operators to implement their own solution for auditing an OpenFlow-based control plane. One possible solution is to use the code base of an SDN controller and re-purpose it as an auditing tool. Also, SDN troubleshooting tools like OFRewind [104] and OpenFlow Sniffer [11] could be reengineered as control plane auditors.

Regarding data plane auditing, capturing network traffic will generate enormous amounts of data. A combination of port mirroring and network capture is not practical; it is too limited for large domains, expensive in terms of storage, while covering a few ports. A solution similar to NetAssay [32] will alleviate these problems by allowing domain FAS agents to write network filtering rules for traffic redirection. With this approach, the data plane auditor will only capture traffic of interest to the SDX FAS agent.

The control and data plane auditors can collect data following two methods: active monitoring and passive monitoring. By using active monitoring, the auditor polls networking devices for audit data, as in our PoC. On the other hand, passive monitoring collects data by passively capturing network traffic (e.g., a tap and a network analyzer in the control plane). It is important to note that the current FAS architecture is not able to detect side channels, as it relies on the trust between the SDX operator and domain administrator to set up the auditing system. Detecting side channels within a domain is outside of the scope of this work.

5.3 Advance Reservation Access Control Using SDN and Tokens

5.3.1 Tokens Background

A token authorization scheme can be implemented via either self-contained tokens or opaque tokens. In the first approach, the token contains all the information to be verified by an enforcement point. Typically, this approach requires a public key infrastructure (PKI) for signing and verifying tokens. In the second approach, a secure token service (STS) validates tokens. The token validation service may be

located either at a central point with the token issuer service, or at the enforcement point closer to the client. This approach may not need a PKI because all token information is stored in a centralized STS. However, if the meaning of the token has to be transferred to the enforcement point, a mechanism for secure token distribution requires that a PKI should be in place.

5.3.2 System Architecture

Our architecture for advance reservation access control comprises an orchestrator that handles user requests and manages networking resources, a WAN controller that represents an advance reservation system connecting sites involved in a specific data transfer, and a site SDN controller that manages the installation of flow rules on site switches. The orchestrator assigns a token to each successful reservation requested by a user, effectively creating a strong binding between the user who requested the reservation and the flows provisioned by SDN on each site and the WAN controller. Then, an authorized user can present this token to a site controller and gain access to the network reservation. After a reservation expires, all configurations are removed from the network, and the token cannot be reused. The full workflow is automated, and no involvement from a network operator is required. Our approach does not require any changes to current advance reservation systems. Figure 26 illustrates our architecture and the workflow for requesting an end-to-end circuit for a data transfer, which we describe in detail below.

The detailed workflow for requesting, provisioning, and consuming a protected advance reservation is as follows:

1. A user requests an advance reservation through an orchestrator. The user provides reservation information: identifiers for the endpoints (e.g., hostname or IP address) involved in the transfer, the start time, the end time, and bandwidth requirement.

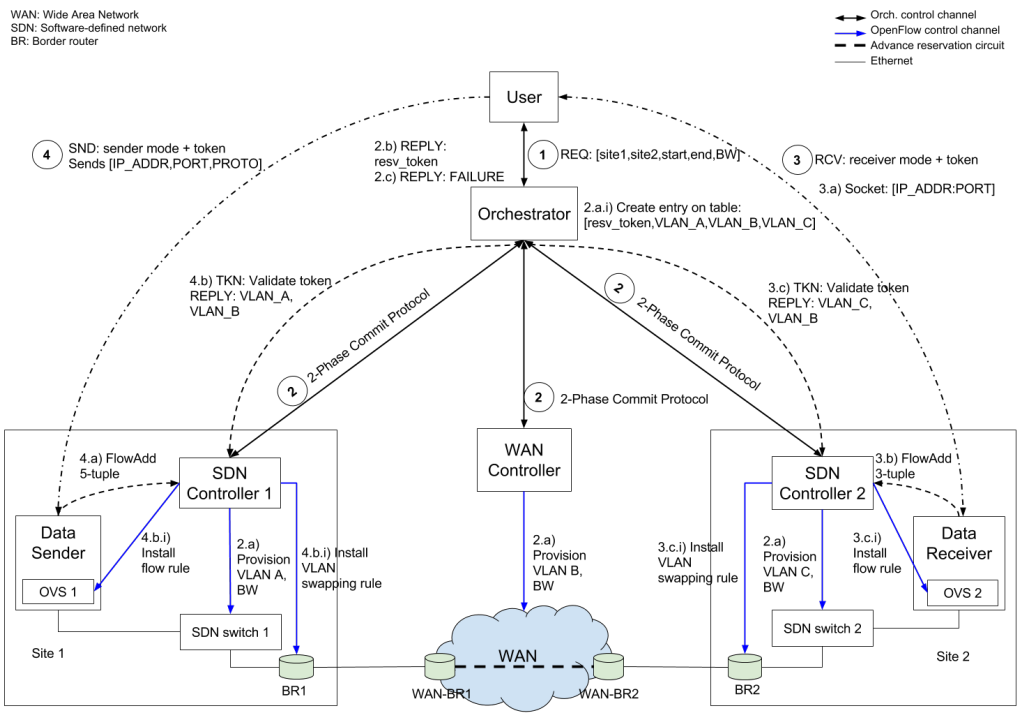


Figure 26: Block diagram of advance reservation access control using SDN and tokens. Only positive outcomes are shown

2. The orchestrator polls each site's SDN controller and a WAN controller to verify whether the bandwidth requested between the two endpoints is available during the specified time frame, i.e., from start to end.

   (a) If resources are available in every domain, each controller provisions a layer-2 circuit within its domain and reports a VLAN ID to the orchestrator. (Our approach will also work for layer 3 circuits.)

      i. The orchestrator creates a token for the reservation and associates it with the set of VLANs.

   (b) The orchestrator replies to the user with the reservation token.

   (c) If any controller does not have enough resources, the orchestrator replies to the user with a reservation failure message.

3. When it is time to start the data transfer, the user contacts the data mover on the receiver site and configures it as a receiver. The communication message includes the reservation token.

    (a) The data receiver replies with the IP address and port on which it is listening.

    (b) The data receiver request the site's SDN controller to add a flow rule matching 3-tuple [ip_addr, port, proto]. (The IP address and port of the sender site are not known at this point.)

    (c) The SDN controller validates the token against the orchestrator, which replies with reservation VLAN if valid.

        i. If valid, the controller installs the flow rule with an action send to VLAN ID of the reservation on the site's OVS and installs the flow rule on the border switch to replace the site's VLAN with the WAN's VLAN for outgoing traffic, and vice versa for incoming traffic.

        ii. Else, it rejects the request.

4. After the IP address and port are known, the user sends a request including the token, destination IP address, and destination port to the data sender.

    (a) The data sender then sends a request to the site's SDN controller to add a flow rule matching the 5-tuple [src_ip, dst_ip, src_port, dst_port, proto]. The user can then send a request to the receiver site to modify the 3-tuple to a 5-tuple flow rule.

    (b) The SDN controller validates the token against the orchestrator, which replies with a reservation VLAN if valid.

        i. If valid, the controller installs a flow rule with an action of tagging packets with a reservation VLAN on the site's OVS and installs a

flow rule on the border router for replacing the VLAN ID of the site with the VLAN ID of the WAN for outgoing traffic, and vice versa for incoming traffic.

ii. Else, it rejects the request.

5.3.3 Implementation

We conducted experiments on the ESNet infrastructure testbed. As shown in Figure 27, we used two sites, Washington DC and CERN in Geneva, Switzerland, which have an average inter-site RTT of 90 ms and up to 10 Gbps best effort for bandwidth capacity. Each site has two OVS switches [85], one container endpoint, and one Ryu SDN controller [84]. The orchestrator runs on another container hosted at CERN. All containers run Ubuntu 14.04.

As shown in Figure 26, our architecture implementation is composed of a WAN controller, one site controller per site, one data mover node per site, an orchestrator, and a user interface. Each component was coded in Python and communicates over TCP sockets sending JSON data. To communicate with the Ryu controller, we used the REST API that comes with the controller. The data transfers used iperf. The system handles three types of messages: (1) REQ for advance reservation requests, (2) RCV for data mover receiver configuration, and (3) SND for data mover sender configuration. We next provide a brief description of each component.

WAN Controller

The WAN controller emulates an advance reservation system such as OSCARS (ESNet) or AL2S (Internet2). Its northbound interface talks to the Orchestrator, while its southbound interface interacts with WAN switches. Its main functionality is to manage a pool of VLANs, assign VLANs to circuit reservation requests, and provision the circuit on the WAN infrastructure (switches). A message request from the Orchestrator has the following format:

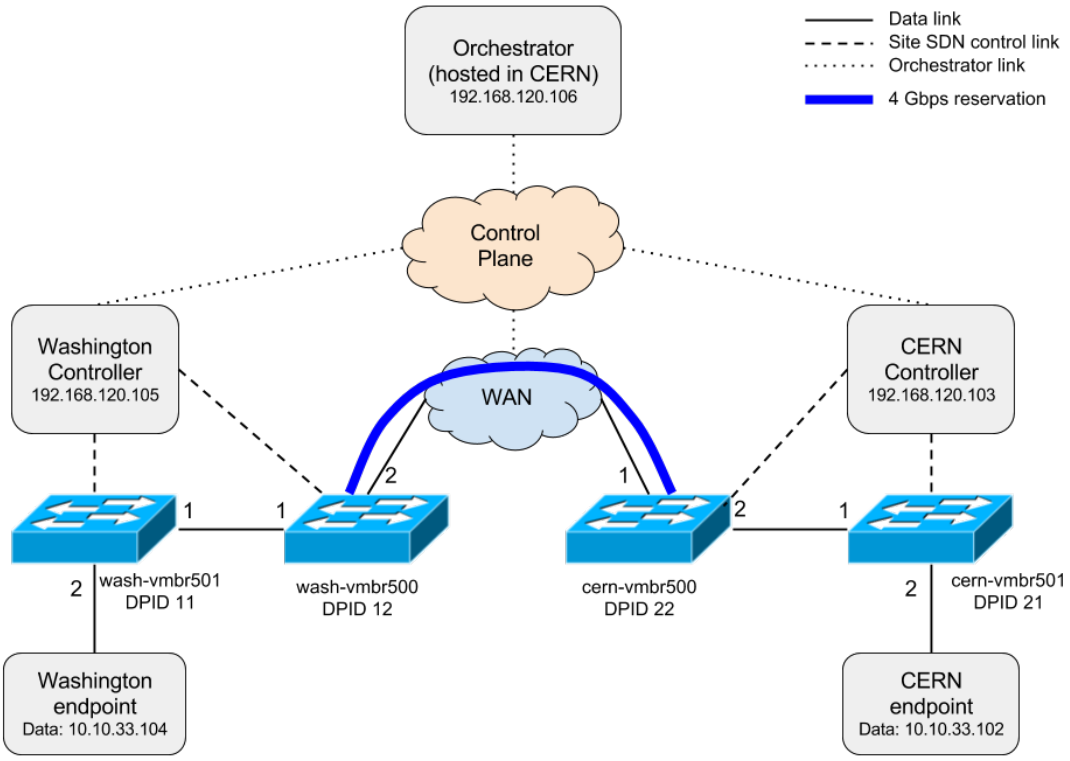


Figure 27: ESNet infrastructure testbed configuration for experiments

- Message Type: REQ
- Format: site1, site2, start time, end time, bandwidth

The actions performed by the WAN controller after receiving a request are assign VLAN to the reservation, allocate bandwidth requested, and configure switches. It may respond with the reservation VLAN ID or a failure message.

Site Controller

A site controller manages reservation configurations at its site. The site switch in Figure 26 represents the site's topology (which could involve one or more switches) and the border router represents a connection to the WAN. The controller's northbound interface talks to the orchestrator and a data mover that requests access to a reservation, while its southbound interface interacts with site switches through OpenFlow.

The controller's main functions are to manage a pool of VLANs, assign VLANs to circuit reservation requests from the orchestrator, provision the circuit on the site infrastructure (switches), validate tokens against the orchestrator, and install flow rules binding reservation VLAN to flow 5-tuple. A message request from the orchestrator has the following format:

- Message Type: REQ
- Format: site1, site2, start time, end time, bandwidth

A message request from a data mover is handled by the Ryu controller's REST API. We extended that API to accept authorization tokens when adding new flow rules.

Data Mover

The data mover's main function is to transfer data from one site to another. It can work in either sender or receiver mode. It accepts commands from a user interface and sends add flow requests to a site controller with a reservation token. In our experiments, we use iperf to perform data transfers. To emulate GridFTP behavior [2], the data mover receiver generates a random TCP port number before starting the iperf server and returns the socket on which it is listening. Likewise, the sender uses this socket to establish a connection by using iperf. A user interface can send two types of messages to a data mover:

- RCV: generates a random port number and starts an iperf server on that port; returns the socket [IP:port] to the user interface.
- SND: opens a connection to the socket provided by the client.

Every request acquires a reservation token. After every request, the data mover has to present the site controller with the request's token plus a flow to be added.

Orchestrator

The orchestrator is in charge of coordinating the reservation of an end-to-end circuit between two (or more) sites and validating the tokens presented by data movers to site controllers (refer to messages 3c and 4b in Figure 26). Its northbound interface talks to the user that requests access to a reservation; its southbound interface interacts with SDN controllers on the WAN and on each site. A user request has the following format:

- Message Type: REQ
- Format: site1, site2, start time, end time, bandwidth

If the orchestrator finds a path between the two sites, it will return a reservation token to the user; otherwise, a failure message will be sent. A token validation request from a data mover has the following format:

- Message Type: TKN
- Format: Universally Unique Identifier (UUID) v4, a 128-bit-long identifier standard defined in RFC 4122

The orchestrator will reply with a valid or invalid token message depending on the existence of a token in its token store.

5.3.4 Evaluation

We evaluated the system on the ESNet 100G SDN testbed by measuring its latency in answering a request with three token validation schemes (i.e., opaque token at orchestrator, self-contained token, and opaque token with enforcement point validation (EPV)). Figure 28 shows that advanced reservation requests (REQ messages) take around 350 ms. The main factor for this latency is the 2PC protocol that adds a second round of messages to the request. To verify this, we measured that a circuit

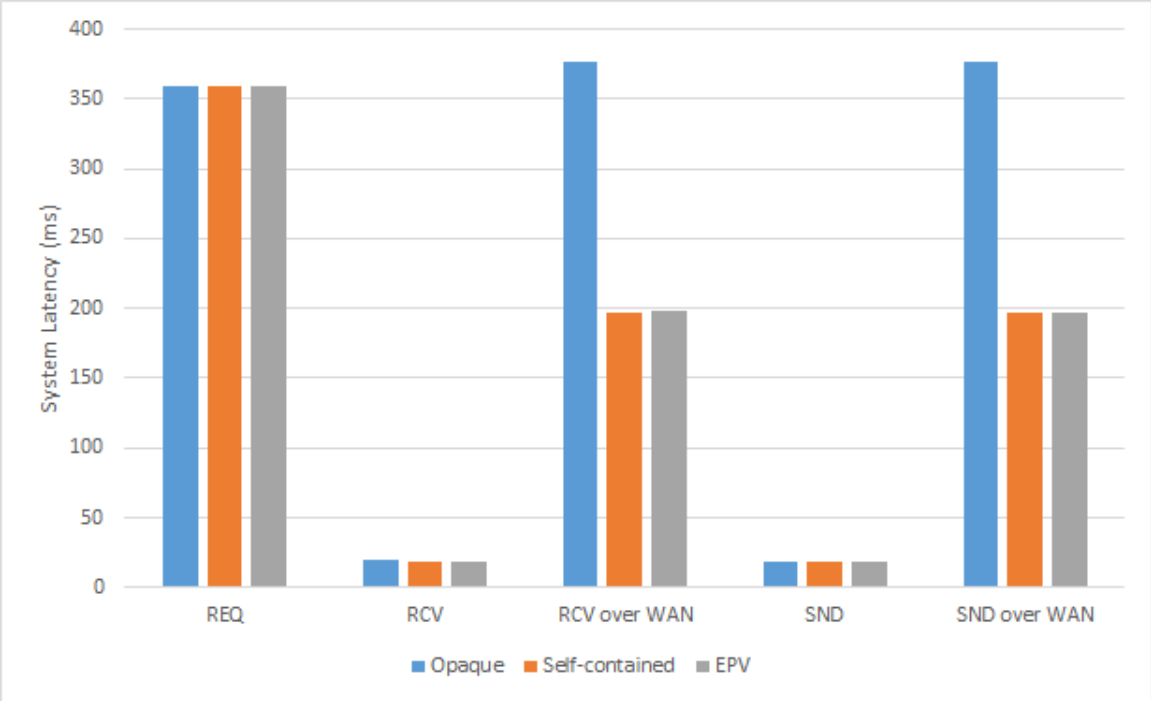


Figure 28: Latency of the system for opaque, self-contained, and enforcement point token validation (EPV).

reservation request takes around 182 ms on average for a system without 2PC, which translates into a 52% overhead. It is important to note that our proof-of-concept implementation does not consider path computation and resource scheduling functions, because these functions are outside the scope of our study.

For messages that require token validation (i.e., RCV and SND messages), the self-contained and enforcement point validation approaches have a lower latency than the opaque token with centralized validation when messages are sent over the WAN. Figure 28 shows that the opaque token approach takes around 375 ms, while the self-contained and enforcement point validation approaches take around 200 ms in this scenario. Nevertheless, for requests that stay in close proximity, all three token validation schemes have a latency of 20 ms. It is important to note that token validation happens per request (i.e., a 5-tuple connection that can be used to transfer all the files in a single data transfer), although in our proof-of-concept implementation

we need to install four flows per switch, per request—where two flows represent the incoming and outgoing traffic for the data transfer and the other two are for ARP requests—for a total of $4N$ flow rule installation request for a site with N OpenFlow switches. After installing all corresponding flows on each switch, we were able to verify that only the specific iperf3 connection was able to communicate between the two endpoints.

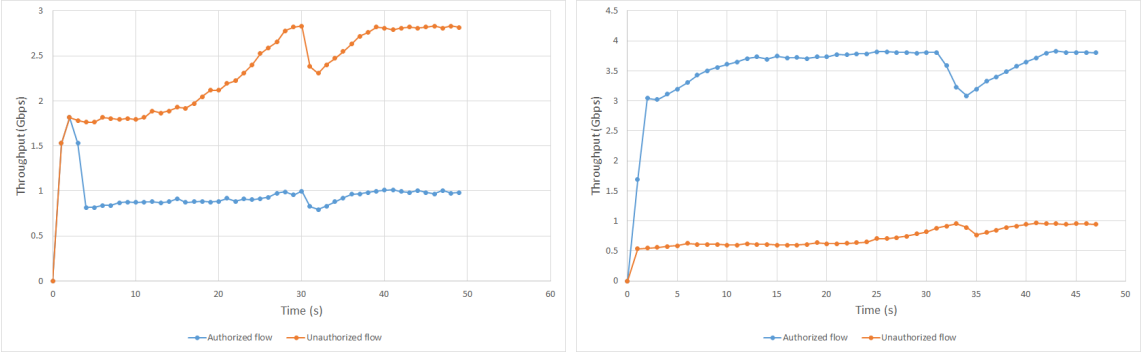
Throughput and Latency Measurements

The traffic model we used is based on the Biological and Environmental Research (BER) network requirements review of 2015 [27] that states scientists will require to transfer 20 GB of data in less than 5 minutes (i.e., 533.33 Mbps of throughput per file). To measure throughput and latency performance, we evaluated the behavior of two flows between our two sites with and without token access control. In each of the two scenarios listed below, we measured the throughput and RTT using iperf3 with the results shown in Figures 29 and 30, respectively:

Scenario 1: A reservation of 4 Gbps is manually extended to the endpoint using VLANs. This creates an interface in the endpoint that is available to all users. As a result, unauthorized users in the same endpoint still have access to the reservation.

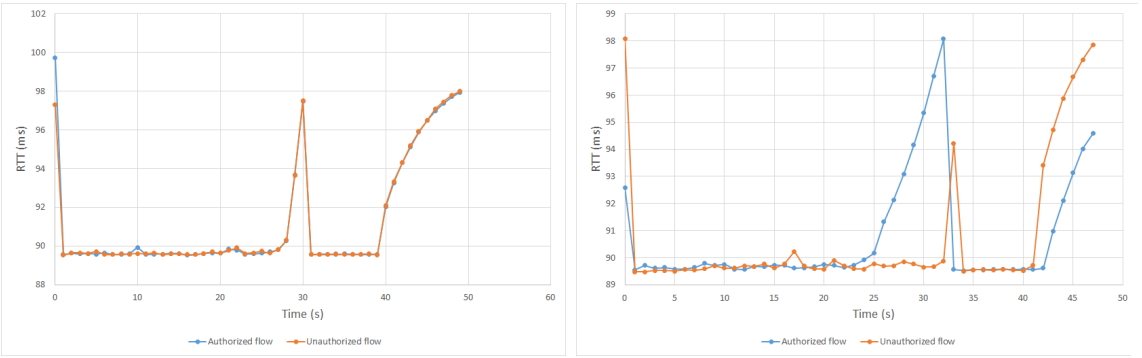
Scenario 2: The reservation is programmatically extended to the endpoint, and access is controlled by using our system. In this scenario only the authorized user (i.e., the user who made the reservation) can access the 4 Gbps reservation, while unauthorized users consume the remaining 1 Gbps available on the 5 Gbps inter-site link.

For scenario 1, we used iperf3 to transfer a 20 GB file from the endpoint at CERN to an iperf3 server running at Washington. The authorized flow, depicted in blue in Figure 29(a) consumed around 1 Gbps of the available bandwidth, while the unauthorized flow, depicted in orange in Figure 29(a) consumed around 3 Gbps.



(a) (b)

Figure 29: Throughput measurements while sending a 20 GB file using iperf3 with CUBIC TCP: (a) scenario 1 shows two flows sharing the 4 Gbps reservation, while (b) scenario 2 implements our access control solution using SDN and tokens, where the authorized flow has exclusive access to the 4 Gbps reservation and the unauthorized flow uses the remaining 1 Gbps available on the 5 Gbps link



(a) (b)

Figure 30: RTT measurements while sending a 20 GB file using iperf3 with CUBIC TCP. Both (a) scenario 1 and (b) scenario 2 present RTT measurements that range between 89 and 100 ms

For scenario 2, we generated the same type of data transfer. However, this time the authorized flow was protected by our SDN with token access control mechanism. Our system installs specific flow rules after receiving a valid token, allowing the authorized flow to use the 4 Gbps reservation. With this configuration, the authorized data transfer, depicted in blue in Figure 29(b) reported around 3.9Gbps, while the unauthorized data transfer depicted in orange in Figure 29(b) reported around 0.9 Gbps. Considering the flow installation latency of our system, which in the worst case scenario (i.e., RCV/SND message over the WAN) is 200 ms configuration, our system

only introduced a 0.40 % latency overhead to the data transfer.

Figure 30 shows the RTT measurements obtained from iperf3 during the data transfers. For both scenario 1 (Figure 30(a)) and scenario 2 (Figure 30(b)), the minimum RTT is around 89 ms and the maximum RTT is 100 ms, with an average of 91 ms for all the experiments conducted. These results show that our solution has no impact on path latency, because tokens are not validated by the data plane, but by the control plane and only before the data transfer starts.

5.3.5 Discussion

Our findings demonstrated that our architecture reduces the provisioning time of an end-to-end circuit from several days (manual process) to a few minutes (automated process). Additionally, we demonstrated that, by using tokens, a specific flow can be strongly associated with the owner of the reservation. For a real deployment, however, many of our design decisions for this proof-of-concept should be optimized.

As we can observe in Figure 28, self-contained token and enforcement point validation approaches reply 150 ms faster to a client request than does an opaque token approach, because validation happens on the enforcement point. We note, however, that in this work we used a preshared password between the orchestrator and site controller, whereas a full PKI should be used in a real deployment. Moreover, in this work we assumed that a secure mechanism for token distribution was in place, and we were not concerned with token spoofing attacks.

We chose to extend the RESTful API of the Ryu controller to validate each add flow request with an authorization token. However, this approach generates too many messages between a site controller and the orchestrator, because each add flow request needs to be validated. An API that validates a single request but installs all required flows at once would be more efficient.

5.4 Conclusions

In this chapter we proposed FAS, a federated auditing framework that allows an SDX to verify whether the configurations requested by a user are correctly enforced by participating SDN domains, whether the configurations requested are correctly removed after their expiration time, and whether configurations exist that are performing non requested actions. Additionally, FAS allows end-to-end configuration verification without revealing internal information of the participant SDN domains that they are not willing to share. FAS establishes a new chain of trust by extending auditing capabilities of the SDX into participant SDN domains, and establishes trust agreements between SDX operators and domain administrators.

We presented an initial proof-of-concept architecture of FAS, implemented on the GENI platform, and discussed the administrative and technical challenges of implementing FAS. For instance, the use OpenFlow over TLS transport represents a major challenge for collecting control plane data. Similarly, bulk data plane traffic collection will require enormous amounts of storage.

We also proposed a system that provides end-to-end advance reservation access control. By using multi-domain SDN orchestration and token-based authorization, our system strongly binds an end-to-end flow to the user or application that requested the reservation. We have deployed this system in the ESNet 100G SDN testbed, and demonstrated that our solution effectively protects authorized flows from competing traffic in the network. Furthermore, the provisioning time of an end-to-end reservation can be reduced from several days (manually) to minutes (automated) on a small scale network. Moreover, the provisioning latency of our system introduces a negligible overhead (approx. 0.40%) on a large data transfer transmission time. These results opens new possibilities for future advance reservation systems in which advance reservations can be more flexible and short-lived (i.e., lasting hours instead of years), allowing finer scheduling of network resources.

Chapter VI

CONCLUSIONS

The thesis statement of this dissertation is the following: *"Given that current advance reservation systems present several challenges for deploying multi-domain intercontinental circuits, this work posits that by introducing SDXs in the reservation and provisioning process of intercontinental circuits, we are able to create multi-path, multi-domain advance reservations, which enhances the performance of science data transfers over traditional methods reported in the literature, while increasing the success rate of reservations, providing more intuitive interfaces to end users, and enabling auditing capabilities to network operators."*

Our initial investigation reveals that no existing architecture is capable of providing multi-path, multi-domain advance reservations. Hence, we propose a new architecture for multi-domain advance reservations that uses SDXs as an interconnection point for enabling multiple paths between any two locations. Our architecture is composed of site, WAN, and SDX controllers that reside in each participant domain, an orchestrator that composes multi-path, multi-domain advance reservations, and the interfaces that allow users to request science network services to the orchestrator and those that allow the orchestrator to request individual services to participant domains.

To prove the thesis statement, we evaluated our architecture by simulation and by running experiments on a physical testbed. Our simulation results show that a full deployment of our architecture with four path available achieves a reservation success rate of 99%, compared to a 50% success rate of traditional (single-domain, single-path) advance reservation systems. The conditions under which this result was

obtained were presented in section 3.4.3. In our simulations, we assume that a traditional advance reservation succeeds if the available bandwidth for a timeslot is greater than the requested bandwidth. By running experiments in GENI, we demonstrated that the system latency of our orchestrator remains below one second when we use gRPC as the communication channel between one orchestrator and eight participant domains with 300 ms RTT between them. In our physical testbed, we demonstrated that common in use scientific data transfer tools such as GridFTP can take advantage of our multi-path, multi-domain advance reservation by measuring network throughput performance under several bandwidth splitting scenarios. We also proposed an interface that allows domain expert scientists to request science network services from our orchestrator and left a user study for future work. Finally, we proposed frameworks for auditing and access control is SDXs. We implemented a proof-of-concept of our auditing framework in the GENI platform. Similarly, we evaluated our access control system in the ESNet 100G SDN testbed, and demonstrated that our solution effectively protects authorized flows from competing traffic in the network. For future work, the orchestration and auditing framework still needs to be tested on large scale real deployments.

6.1 Discussion

6.1.1 Orchestrating International Advance Reservations with Software-defined Exchanges

Advance Reservation Systems and SDN - Ibarra et al. [56] deployed SDN and OpenFlow on the AmLight international research and education network, with the goals of improving operations efficiency and providing network programmability. Although SDN Amlight also automates provisioning of multi-domain network reservations, its definition of a domain is a countrywide network. In contrast, we also consider smaller domains such as national laboratories and university campuses, as we are also concerned with automating provisioning for the last mile between the border

router and the endpoint. Furthermore, the SDN Amlight approach only provisions single-path, multi-domain reservations, while our orchestration architecture provides multi-path, multi-domain capabilities.

Tepsuporn et al. [98] tested the use of end-to-end layer 2 paths for large dataset transfers over an existing deployment called DYNES (Dynamic Network System) [109], which uses OESS and OSCARS in multiple domains to establish dedicated these layer 2 paths. The authors identified limitations with configuration overhead, scalability, path provisioning, and testing. For instance, a failed path setup attempt in OSCARS forces a user to wait 15 minutes before issuing a new request [98]. Our user interface and negotiation protocol eliminates this blocking time by allowing the orchestrator to negotiate the allocation of science network resources among participant domains.

Lark [108] enables network resource management with per-job granularity for high-throughput computing (HTC) systems such as HTCondor, using Linux containers, virtual Ethernet devices, and SDN. In their architecture, each job is assigned to a separate network namespace [69], and each HTCondor node has a virtual switch (e.g., Open vSwitch (OVS) [85] or Linux bridge) that interconnects network namespaces to physical interfaces. To demonstrate these capabilities, the developers of Lark created a bandwidth management system and a job-aware OpenFlow controller, measuring performance overhead for both implementations. The authors reported one second overhead per job, to create and configure network namespaces—a negligible delay since a typical HTC job duration is measured in hours. However, their work considers only jobs running on a single node, whereas our work focuses on the orchestration of network resources in multiple sites. Attaching a Lark node to a system running our architecture will enable job-level granularity in our orchestration framework. Lark and our proposed system are complementary.

The Developing Applications with Networking Capabilities via End-to-end SDN

(DANCES) [52] project seeks to enhance the performance of cyberinfrastructure applications (e.g., GridFTP [2] data transfers, SLASH2 [86] distributed file system data transfers, and SCP) by adding network bandwidth scheduling via SDN. The project developed a bandwidth manager, the Centralized OpenFlow and Network Governing Authority (CONGA), whose main function is to receive bandwidth requests from a resource manager or scheduling system and determine if the request can be fulfilled. CONGA accepts a request if: (1) resources are available on the network and (2) the user is authorized to request this amount of bandwidth. Although DANCES uses SDN for provisioning, their work remains as a single-path, single-domain platform.

Multi-domain SDN Architectures - Avallone et al. [4] proposed an architecture for network resource management in multi-domain scenarios using service-level specifications, while Kempf et al. [61] proposed service provider SDN (SP-SDN), an approach to rapid and flexible cross-domain service creation that complements SDN and network function virtualization (NFV). Likewise, the ONF has proposed transport SDN (T-SDN) [59] as a way to simplify transport network operations by allowing a domain to expose network services (topology, connectivity, path computation, virtual network, and notification service) that will be consumed by external domains. Similarly, the Metro Ethernet Forum (MEF) has proposed the Third Network [76] initiative to promote network as a service principles in industry, and the IETF is working on a draft document for the Abstraction and Control of Transport Networks (ACTN) [30] use cases related to Packet and Optical Integration (POI). Our architecture builds upon concepts proposed by SP-SDN, T-SDN, Third Network, and ACTN, and adapts them to the special necessities of science networks and SDXs. We demonstrated an agent-based approach in which our orchestrator communicates with agents on each participant domain. This approach allows us to control the WAN communication channel, while participant domains can independently develop their own science network services.

Flexible Reservations - Balman et al. [6] developed a flexible reservation algorithm for path-finding in the OSCARS system by taking advantage of user-provided parameters such as the total volume (in bytes) and time constraints, instead of bandwidth requirements. Similarly, Xiao et al. [105] proposed a two-dimensional relaxed reservation policy for Grid computing systems that achieves higher resource utilization and success rates (approximately 95% under low reservation rates). He et al. [53] proposed a flexible advance reservation model for cross-domain lightpath reservations in optical networks that can achieve a maximum reservation success rate of 84%. Both [6] and [105] are single domain scheduling algorithms (i.e., they complement our work when deployed inside participant domains). On the other hand, [53] is a cross-domain approach specific for optical network. Our orchestration architecture is technology agnostic, and our negotiation protocol provides higher success rates (approximately 99% when four paths are available) under the simulations assumptions made in section 3.4.3.

Network Resource Negotiation - RNAP [103] and SNAP [25] are two examples of negotiation protocols for networking and Grid computing resources, respectively. Both are based on querying resource provider for the availability of a resource before making a reservation. Venugopal et al. [101] proposed a negotiation mechanisms using an alternate offers protocol for advance reservation of compute nodes in a Grid system. We build upon the concepts of querying for resources and providing offers to create our negotiation protocol. Furthermore, we show how our negotiation protocol performs in a science network using SDXs.

TCP Striping - For more than 15 years researchers have been proposing ways of striping TCP connections across multiple diverse paths for performance enhancement, or for finding a sum of bandwidth available in a reservation system. For instance, in 2002, Hsieh et al. [54] proposed parallel TCP (pTCP), an end-to-end transport layer

protocol that allows connections to leverage the aggregate bandwidth of multiple parallel paths regardless of the individual characteristics of each path. According to the authors, pTCP can be used for bandwidth aggregation of wireless interfaces for mobile hosts, end-to-end service differentiation, and connection striping on overlay networks. Recently, Multipath TCP (MPTCP) [12] has emerged as a standard of the IETF and an implementation in the Linux kernel that allows a single transport connection to use multiple paths simultaneously. In fact, data transfer protocols (e.g. GridFTP [3]) take advantage of these ideas to implement their own TCP multistreaming capabilities. The use of multiple parallel and disjoint TCP flows is not new, however how to accomplish this in SDX-enabled science data networks is novel. Our orchestration framework uses SDXs to provision the underlying network infrastructure that allows TCP striping protocols achieve their full potential in Layer 2 advance reservation environments.

Multi-path Advance Reservations - OLiMPS (OpenFlow Linklayer Multi-Path Switching) [79] is an OpenFlow application that allows load balancing over multiple switched paths. The authors integrated their OpenFlow application with the OSCARS system, and tested several load-balancing algorithms on a dedicated testbed. Likewise, Plante et al. [87] proposed a multipath extension to the OSCARS client that enables end users to reserve multiple paths, providing session survivability and increasing parallelism. Although similar to our work, both of these solutions are for single-domain reservations, each one focuses on a single piece of the overall problem. While OLiMPS cares about provisioning OpenFlow rules, Plante's work is more concerned with the scheduling aspect of the problem. Furthermore, Plante's work assumes identical bandwidth demands for every parallel virtual circuit (i.e., all advance reservation requests are equal to the original user request). We provide a bandwidth splitting service that makes more efficient use of network resources, instead of requesting the same bandwidth in all available paths. Furthermore, our multi-domain

architecture is easily adaptable to single domain scenarios as those proposed by [79] and [87].

6.1.2 Novel Network Services for Supporting Big Data Science Research

Researchers have already proposed the use of SDN for enhancing scientific application resource management and performance over a WAN connection. For instance, the Lark project [108] proposed a flexible and fine-grained mechanism to manage network resources in high-throughput computing (HTC) systems [7]. Similarly, the DANCES project uses SDN to enhance scientific application resource management and performance of a WAN connection by developing applications with networking capabilities via end-to-end SDN [52]. However, these solutions do not provide interfaces that allow domain-expert scientists to request scientific data transfer services abstracting network details. While the Lark project built an SDN controller that allows scheduling of high-throughput computing (HTC) jobs in a HTCondor system [39], the DANCES project uses the same abstraction of endpoints, start time, end time, and requested bandwidth that current science network reservation systems use. Our architecture proposes to provide abstractions that enable domain-expert scientists to request end-to-end services on scientific networks while hiding the details of the network. Furthermore, Lark and DANCES focus on single-domain SDN, while our solution focuses on multi-domain science networks.

Inside the SDN community, members are aligned with the development of intent-based networking interfaces [14, 66] that use a prescriptive rather than descriptive approach to network configuration; that is, network operators and applications describe a goal, and the SDN controller decides how to implement it. Within the research community, Kiran et al. [62] proposed the iNDIRA (Intelligent Network Deployment Intent Renderer Application) tool [62], which uses natural language processing to capture the network service requirements of the user. iNDIRA has been

deployed on the Energy Science Network (ESNet), where it interacts with Globus data transfer tools. Similarly, the *SDX user interface* of AtlanticWave/SDX seeks to implement intents by describing the network services in a high level language, and enabling negotiation between the SDX and local controllers.

6.1.3 Auditing and Access Control for Software-Defined Exchanges

Multi-domain SDN Auditing - The closest work to our federated auditing for software-defined exchanges is AudIt, a multi-domain SDN auditing system proposed by Maldonado-Lopez et al. [71] which identifies whether an origin policy has been enforced by foreign domains. The AudIt architecture comprises four main elements: (1) a model of the network topology, policies, and flows; (2) the AudIt protocol for gathering information about the configuration deployed on an external domain; (3) a validation engine for detecting policy violations; and (4) an extension to the OpenFlow protocol to enable external auditing. The AudIt protocol works by sending related flow tables to an origin domain every time an external switch receives an AudIt packet. Although this architecture can verify if an external policy has been enforced in less than one second, and it can prove if an external policy has been violated in 1.5 seconds, the auditing system is tightly integrated to the SDN controller. This means that an attacker that compromised the SDN controller would also compromise the auditing system. Moreover, this architecture requires the modification of SDN switches and the OpenFlow protocol. FAS on the other hand works as a third party auditor of the SDN infrastructure, and requires no changes to OpenFlow.

Single-domain SDN Auditing - In single-domain SDN, many researchers have proposed the use of packet histories or network traffic statistics for network diagnosis. NetSight [51] is an extensible platform that captures packet histories and enables applications to retrieve histories of interest by using a regex-like language called packet history filter (PHF). To assemble packet histories, NetSight uses postcards, which are

event records created whenever a packet traverses a switch and contains the packet header, switch ID, output ports and version number of the switch forwarding state. Similarly, FlowMon [60] detects compromised switches through real-time analysis of network traffic statistics collected by OpenFlow in an SDN controller. To achieve its goal, FlowMon extends an SDN controller with two extra functional blocks: a malicious switch detection and prevention (MSDP) block and a policy block. While MSDP continually and transparently analyzes communication between the controller and switches, the policy block contains a set of rules enforced whenever a malicious switch is detected. Contrary to NetSight and Flowmon, packet trackeback [107] takes advantage of the global view of SDN controller to analyze the policy at the controller, rather than monitoring the data plane. Our federated auditing for software-defined exchanges is based upon ideas from the single-domain SDN monitoring and extends them to federated auditing for SDX.

Federated Auditing - Federated auditing has also been proposed in areas outside SDN and SDX. ForNet [92], a federated forensics system, is composed of two functional components: a synopsis appliance called SynApp, and a forensic server that manages a set of SynApps within an administrative domain. The SynApp appliance is designed to summarize and remember network events in its vicinity. The architecture of ForNet is hierarchical, with all SynApps within a domain associated with a centralized forensic server of that domain. Forensic servers can be networked for inter-domain collaboration that forms the second level of the hierarchy. Likewise, Omni [97] is a passive measurement tool for diagnosing routing problems in the Internet. The Omni architecture requires an Omni server per AS that constructs a comprehensive view of its routing system. ASes involved in a routing change cooperate for pinpointing the problem. If an AS needs to contact an Omni server in other ASes, the Omni server sends a query that follows the forwarding path. Although, Fornet and Omni identify the necessity of cooperation between domains, both architectures

overlook the desire of exchanging audit data without revealing internal information of the participant domains, which most network operators are not willing to share.

SDN-based Access Control - Network access control (NAC), standardized as IEEE 802.1X [57], is a common computer security approach to authenticate endpoints and grant access to a computer network. NAC was an early SDN application, with the main focus being policy enforcement. Casado et al. [16] proposed a Secure Architecture for the Networked Enterprise (SANE), which defines a single protection layer that governs all routing and access control decisions in the network. Similarly, Nayak et al. [78] proposed Resonance, a system for securing enterprise networks by using dynamic access control policies and network devices as enforcement points. FlowNAC [74] and FlowIdentity [106] adapt IEEE 802.1X by using SDN principles. FlowNAC performs authorization by a set of predefined flow rules per network service, whereas FlowIdentity enforces a policy through a stateful role-based firewall that is updated dynamically at the SDN controller. These studies were all conducted in a single domain, such as a campus or enterprise network. Our system builds upon these ideas and extend them to multi-domain science networks.

Token-based Access Control - Gommans et al. [46] proposed a token-based access control mechanism for multidomain lightpath (i.e., a fiber optics path) REN reservations. They identified and demonstrated three ways to enforce access control policies by using tokens: at the IP packet layer, by using a token-base switch; at the control plane, by including a token in a specific field of the resource reservation protocol - traffic engineering (RSVP-TE) signaling protocol for networks based on generalized multiprotocol label switching (GMPLS); and at the service layer signaling, by implementing an authentication, authorization, and accounting server, a token enforcement point, and a lightpath resource allocation system. However, while this work extended to multiple domains, it did not consider SDN. The originality of our work lies in its integration of SDN access control and token-based multi-domain

authorization.

6.2 Contributions

The contributions in each chapter are summarized as follows:

1. Chapter 3 presented the design, implementation, and evaluation of our architecture for intercontinental multi-domain, multi-path advance reservations in science networks and SDXs. The architecture is composed of an orchestrator that request services from participant domains and SDXs, and a negotiation protocol that allows the orchestrator to compose end-to-end services taking advantage of alternative paths provided by the enriched connectivity of SDXs. We evaluated our proposed architecture in a dedicated testbed using single-path vs. multi-path advance reservations over multiple domains and the data transfer tools that the scientific community currently uses. We demonstrated that our orchestration framework and negotiation protocols increases the reservation success rate from approximately 50% using single path to approximately 99% with four paths available under the constraints specified in section 3.4.3. Chapter 3 also presents architectural approaches at the SDX level that enable novel science network services, while enhancing the performance of science data transfers over traditional approaches. We evaluated our solution using GridFTP, one of the most popular tools for data transfers in the scientific community. In our experiments, we tested our system under several conditions of bandwidth splitting ratios, SDN rule provisioning strategies at the edge, and number of GridFTP streams, and generated recommendations for the optimal performance of our system.

2. Chapter 4 proposed an intuitive interface that users and other systems can use to request science network services from our orchestration framework. Contrary

to current interfaces that were designed *by* network operators *for* network operators, our interface allows a domain-expert scientist to specify the size of their datasets and a deadline for the data transfer. Then our orchestrator negotiates a suitable advance reservation across all participant domains. We also proposed novel science network services enabled by our proposed architecture. These proposals need further work to fully evaluate their impact on existing science data networks.

3. Chapter 5 presented a federated auditing framework (FAS) that allows an SDX to verify whether the configurations requested by a user are correctly enforced by participating SDN domains, whether the configurations requested are correctly removed after their expiration time, and whether configurations exist that are performing non requested actions. Additionally, FAS allows end-to-end configuration verification without revealing internal information of the participant SDN domains that they are not willing to share. FAS establishes a new chain of trust by extending auditing capabilities of the SDX into participant SDN domains, and establishes trust agreements between SDX operators and domain administrators. We presented an initial proof-of-concept architecture of FAS, implemented on the GENI platform, and discussed the administrative and technical challenges of implementing FAS. For instance, the use OpenFlow over TLS transport represents a major challenge for collecting control plane data. Similarly, bulk data plane traffic collection will require enormous amounts of storage.

6.3 Future Research

This thesis has demonstrated that by introducing SDXs in the reservation and provisioning process of multi-domain intercontinental advance reservations, we can improve

the success rate of reservations and define novel science network services such as multi-path bandwidth splitting across independent WAN providers, scheduled path migrations that are transparent to data transfer applications, and multipoint-to-multipoint advance reservations. However, many questions remain open regarding real deployment of orchestration systems similar to the one proposed in this thesis, and the future implications of SDX in next generation network infrastructures. In the following subsections we describe some of the research opportunities that science networks SDXs and SDXs in general will enable in the future.

6.3.1 Large Scale Deployment of Science Network SDXs

In this dissertation we evaluated our orchestration system on a dedicated testbed, and emulated the parameter of real-world science network. However, there are conditions that are better tested on real large scale testbeds. For instance, what is the effect of paths with different delays on the composition of multi-path, multi-domain advance reservations? Where is the optimal placement of the orchestrator? In the case of a distributed orchestrator, what is the best approach to maintain consistency and consensus among replicas? How does the negotiation protocol behaves under a high load of requests, or when multiple orchestrators compete for the same resources, and how do site controllers will behave under these conditions?

Some of the results of this dissertation (bandwidth offers and U-O interface) have influenced the development of the AtlanticWave/SDX controller. However, a detailed user study is required to determine how effective and intuitive the interface really is. We are currently involved in the definition of the roadmap of future SDX services for the AtlanticWave/SDX project. We plan to continue to contribute to the design and evaluation of these services as they are deployed at large scale.

Similarly, both our federated auditing for SDX (FAS) and our advance reservation access control frameworks need to be tested in large scale deployments. One should

evaluate the robustness of privacy mechanism in the FAS protocol, as well as new data structures to represent network configurations from the data plane. How the auditing system could detect side channels in the science data network infrastructure is another area of investigation. Regarding access control, one could explore the robustness of token approaches against spoofing.

6.3.2 Network Function Virtualization (NFV) and Programmable Dataplanes

OpenFlow has historically been considered as the main component in the implementation of SDN, and by default it was the only enabler for SDXs. However, with the advent network function virtualization (NFV) [38], and programmable dataplanes such as P4 [13] and the Berkeley extensible software switch (BESS) [50] we envision new types of services. For instance, we may be able to deploy security network functions on demand at the SDX, or perform bandwidth splitting at packet level for applications that do not use multiple TCP streams. A question that remains is how can we realize these services at line-rate speeds in the order of tens or hundreds of gigabits per second?

6.3.3 Other Applications for SDXs

SDX has already been proposed as a solution to mitigate distributed denial of service (DDoS) [34]. However, how can we make several SDXs to cooperate in the task of mitigating a global scale DDoS attack? SDXs may also be applied in new areas. For instance, in the context of smart cities and smart communities, we have proposed MetroSDX [31], a neutral network design that increases the resiliency of edge networks and global and local services, improves isolation of network functions, and preserves data from edge devices when they are disconnected. One might study deployment strategies for MetroSDX, as every city and every community has their own peculiarities.

6.4 Conclusions

In this thesis, we studied architectural approaches to a science network SDX, and proposed an orchestration system for advance reservation of multi-domain, multi-path intercontinental links. Our orchestration system not only increases the success rate of multi-domain intercontinental advance reservation, but also enhances the performance of science data transfers, provides interfaces for domain scientist users, and enables auditing and access control capabilities for network operators. First, we provided a literature survey of advance reservation systems and SDXs. Second, we introduced a reference architecture of our orchestration system. Third, we designed, implemented, and evaluated our orchestration system for reservation and provisioning of multi-path, multi-domain intercontinental advance reservations. Fourth, we proposed an interface for domain-expert scientists to request multi-domain intercontinental advance reservations on our system, adapting our reference architecture to the AtlanticWave/SDX project and leave as future work a user study. Fifth, we proposed a federated auditing system that allows network operators to verify that their network policies have been correctly installed in an SDX, regardless of what architecture are they using, and an access control system that uses SDN and tokens to strongly bind network flows to the user who requested an advance reservation.

REFERENCES

[1] Al-Shabibi, A., De Leenheer, M., Gerola, M., Koshibe, A., Parulkar, G., Salvadori, E., and Snow, B., "Openvirtex: Make your virtual sdns programmable," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, (New York, NY, USA), pp. 25–30, ACM, 2014.

[2] Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Liming, L., and Tuecke, S., "GridFTP: Protocol extensions to FTP for the grid," *Global Grid ForumGFD-RP*, vol. 20, pp. 1–21, 2003.

[3] Allcock, W., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I., and Foster, I., "The globus striped gridftp framework and server," in *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, SC '05, (Washington, DC, USA), pp. 54–, IEEE Computer Society, 2005.

[4] Avallone, S., D'Antonio, S., Esposito, M., Romano, S. P., and Ventre, G., "Resource allocation in multi-domain networks based on service level specifications," *Journal of Communications and Networks*, vol. 8, pp. 106–115, March 2006.

[5] Bailey, J., Pemberton, D., Linton, A., Pelsser, C., and Bush, R., "Enforcing rpki-based routing policy on the data plane at an internet exchange," in *Proceedings of the third workshop on Hot topics in software defined networking*, pp. 211–212, ACM, 2014.

[6] Balman, M., Chaniotakisy, E., Shoshani, A., and Sim, A., "A flexible reservation algorithm for advance network provisioning," in *2010 ACM/IEEE*

International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–11, Nov 2010.

[7] Basney, J., Livny, M., and Tannenbaum, T., "High throughput computing with condor," *HPCU news*, vol. 1, no. 2, p. 1, 1997.

[8] Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., and Parulkar, G., "Onos: Towards an open, distributed sdn os," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, (New York, NY, USA), pp. 1–6, ACM, 2014.

[9] Berman, M., Chase, J. S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R., and Seskar, I., "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, no. 0, pp. 5 – 23, 2014. Special issue on Future Internet Testbeds Part I.

[10] Betge-Brezetz, S., Kamga, G. B., and Tazi, M., "Trust support for sdn controllers and virtualized network applications," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pp. 1–5, April 2015.

[11] Bezerra, J., Ibarra, J., Galiza, H., and Schwarz, M., "AmLights OpenFlow Sniffer dissected: Troubleshooting production networks." http://amlight.net/wp-content/uploads/2015/03/WPEIF-2016-OpenFlow-Sniffer.pdf, 2016.

[12] Bonaventure, O., Handley, M., and Raiciu, C., "An overview of multipath TCP," *; login:*, vol. 37, no. 5, p. 17, 2012.

[13] Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G.,

and Walker, D., "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–95, July 2014.

[14] Cao, W. W., Zhou, T., Jiang, S., Xiao, Y., and Hares, S., "Nemo an application's interface to intent based networks." http://www.nemo-project.net/.

[15] Carrozzo, G., Monno, R., Belter, B., Krzywania, R., Pentikousis, K., Broadbent, M., Kudoh, T., Takefusa, A., Vieo-Oton, A., Fernandez, C., and others, "Large-scale sdn experiments in federated environments," in *Smart Communications in Network Technologies (SaCoNeT), 2014 International Conference on*, pp. 1–6, IEEE, 2014.

[16] Casado, M., Garfinkel, T., Akella, A., Freedman, M. J., Boneh, D., McKeown, N., and Shenker, S., "Sane: A protection architecture for enterprise networks.," in *Usenix Security*, 2006.

[17] Casado, M., Koponen, T., Shenker, S., and Tootoonchian, A., "Fabric: A retrospective on evolving sdn," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, (New York, NY, USA), pp. 85–90, ACM, 2012.

[18] Charbonneau, N., Vokkarane, V. M., Guok, C., and Monga, I., "Advance reservation frameworks in hybrid ip-wdm networks," *Communications Magazine, IEEE*, vol. 49, no. 5, pp. 132–139, 2011.

[19] Chung, J., Cox, J., Clark, R., and Owen, H., "FAS: Federated auditing for software-defined exchanges," in *SoutheastCon 2017*, pp. 1–8, March 2017.

[20] Chung, J., Clark, R., and Owen, H., "SDX architectures: A qualitative analysis," in *IEEE SoutheastCon 2016*, pp. 1–8, IEEE, 2016.

[21] Chung, J., Cox, J., Ibarra, J., Bezerra, J., Morgan, H., Clark, R., and Owen, H., "Atlanticwave-sdx: An international sdx to support science data applications," *Software Defined Networking (SDN) for Scientific Networking Workshop, SC'15*, pp. 1–7, Nov 2015.

[22] Chung, J., Donovan, S., Bezerra, J., Morgan, H., Ibarra, J., Clark, R., and Owen, H., "Novel network services for supporting big data science research," in *Gateways 2017*, October 2017.

[23] Chung, J., Jung, E.-S., Kettimuthu, R., Rao, N. S., Foster, I. T., Clark, R., and Owen, H., "Advance reservation access control using software-defined networking and tokens," *Future Generation Computer Systems*, 2017.

[24] Chung, J., Kettimuthu, R., Pho, N., Clark, R., and Owen, H., "Orchestrating intercontinental advance reservations with software-defined exchanges," *4th International Workshop on Innovating the Network for Data Intensive Science (INDIS) 2017*, pp. 1–11, Nov 2017.

[25] Czajkowski, K., Foster, I., Kesselman, C., Sander, V., and Tuecke, S., *SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems*, pp. 153–183. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002.

[26] Dart, E. and others, "ESNet requirement review reports." https://www.es.net/science-engagement/science-requirements-reviews/requirements-review-reports/. Accessed: 2017-01-14.

[27] Dart, E. and others, "Biological and environmental research network requirements review."

https://www.es.net/assets/Hester/RequirementsReviews/BER-Net-Req-Review-2015-Final-Report.pdf, 2015.

[28] Dart, E., Rotman, L., Tierney, B., Hester, M., and Zurawski, J., "The Science DMZ: A network design pattern for data-intensive science," *Scientific Programming*, vol. 22, no. 2, pp. 173–185, 2014.

[29] de Laat, C., Gross, G., Gommans, L., Vollbrecht, J., and Spence, D., "Rfc 2903 - generic aaa architecture." https://tools.ietf.org/html/rfc2903, 2000. Accessed: 2016-05-04.

[30] Dhody, D., Zhang, X., de Dios, O. G., Ceccarelli, D., and Yoon, B., "Packet optical integration (POI) use cases for abstraction and control of te networks (ACTN)." https://datatracker.ietf.org/doc/draft-dhody-actn-poi-use-case/. Accessed: 2017-01-15.

[31] Donovan, S., Chung, J., Sanders, M., and Clark, R., "MetroSDX: A resilient edge network for the smart community," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 575–580, March 2017.

[32] Donovan, S. and Feamster, N., "Intentional network monitoring: Finding the needle without capturing the haystack," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, p. 5, ACM, 2014.

[33] Donovan, S., Skandalakis, J., and Lamba, A., "Atlanticwave/sdx controller prototype." https://github.com/atlanticwave-sdx/atlanticwave-proto. Accessed: 2017-08-11.

[34] Eddy, W., Clark, G., and Dailey, J., "Customer-controlled filtering using sdn." https://datatracker.ietf.org/doc/draft-eddy-sdnrg-customer-filters/.

[35] ESNet, "Esnet - network maps." https://www.es.net/engineering-services/the-network/network-maps/. Accessed: 2017-08-28.

[36] ESNet, "Linux tuning." https://fasterdata.es.net/host-tuning/linux/. Accessed: 2017-09-07.

[37] ESNet, "oscars-newtech." https://github.com/esnet/oscars-newtech. Accessed: 2017-07-25.

[38] ETSI, "Network functions virtualisation architectural framework," *ETSI GS NFV*, vol. 2, p. v1, 2013.

[39] for-High-Throughput-Computing-at UW-Madison, C., "What is htcondor?." https://research.cs.wisc.edu/htcondor/description.html. Accessed: 2017-01-19.

[40] Fukushima, M., Kuroki, K., and Hayashi, M., "Unified metamodel for orchestrating different domains in sdi," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pp. 1–6, April 2015.

[41] GEANT, "eduGAIN." https://www.geant.org/Services/Trust_identity_and_security/eduGAIN.

[42] Global-NOC, "Internet2 international networks." https://noc.net.internet2.edu/i2network/live-network-status/maps-graphs/internet2-international-network.html. Accessed: 2017-08-28.

[43] GlobalNOC, "Flowspace firewall." http://globalnoc.iu.edu/sdn/fsfw.html.

[44] GlobalNOC, "OESS: Open exchange software suite." https://globalnoc.iu.edu/sdn/oess.html.

[45] Globus, "Globus - research data management system." https://www.globus.org/. Accessed: 2017-01-20.

[46] Gommans, L., Xu, L., Demchenko, Y., Wan, A., Cristea, M., Meijer, R., and de Laat, C., "Multi-domain lightpath authorization, using tokens," *Future Generation Computer Systems*, vol. 25, no. 2, pp. 153 – 160, 2009.

[47] Google-Developers, "Protocol buffers." https://developers.google.com/protocol-buffers/. Accessed: 2017-09-06.

[48] gRPC, "gRPC." https://grpc.io/. Accessed: 2017-09-06.

[49] Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S. P., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., and Katz-Bassett, E., "SDX: A software defined internet exchange," in *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 551–562, ACM, 2014.

[50] Han, S., Jang, K., Han, D., Panda, A., Edupuganti, S., Reich, J., and Ratnasamy, S., "BESS berkeley extensible software switch," tech. rep., University of California at Berkeley, 2015.

[51] Handigol, N., Heller, B., Jeyakumar, V., Mazières, D., and McKeown, N., "I know what your packet did last hop: Using packet histories to troubleshoot networks," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, (Seattle, WA), pp. 71–85, USENIX Association, Apr. 2014.

[52] Hazlewood, V., Benninger, K., Peterson, G., Charcalla, J., Sparks, B., Hanley, J., Adams, A., Learn, B., Budden, R., Simmel, D., Lappa, J., and Yanovich, J., "Developing applications with networking capabilities via end-to-end sdn (dances)," *XSEDE16*, pp. 1–7, July 2016.

[53] He, E., Wang, X., and Leigh, J., "A flexible advance reservation model for multi-domain wdm optical networks," in *2006 3rd International Conference on Broadband Communications, Networks and Systems*, pp. 1–10, Oct 2006.

[54] Hsieh, H.-Y. and Sivakumar, R., "ptcp: an end-to-end transport layer protocol for striped connections," in *10th IEEE International Conference on Network Protocols, 2002. Proceedings.*, pp. 24–33, Nov 2002.

[55] Hu, F., Hao, Q., and Bao, K., "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 2181–2206, Fourthquarter 2014.

[56] Ibarra, J., Bezerra, J., Morgan, H., Fernandez Lopez, L., Stanton, M., Machado, I., Grizendi, E., and Cox, D., "Benefits brought by the use of openflow/sdn on the amlight intercontinental research and education network," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pp. 942–947, May 2015.

[57] IEEE, "IEEE standard for local and metropolitan area networks–port-based network access control," *IEEE Std 802.1X-2010 (Revision of IEEE Std 802.1X-2004)*, pp. 1–205, Feb 2010.

[58] Internet2, "Layer 2 services." http://www.internet2.edu/products-services/advanced-networking/layer-2-services/. Accessed: 2017-07-25.

[59] Janz, C., Ong, L., Sethuraman, K., and Shukla, V., "Emerging transport sdn architecture and use cases," *IEEE Communications Magazine*, vol. 54, pp. 116–121, October 2016.

[60] Kamisiński, A. and Fung, C., "Flowmon: Detecting malicious switches in software-defined networks," in *Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense*, SafeConfig '15, (New York, NY, USA), pp. 39–45, ACM, 2015.

[61] Kempf, J., Korling, M., Baucke, S., Touati, S., McClelland, V., Mas, I., and Backman, O., "Fostering rapid, cross-domain service innovation

in operator networks through service provider sdn," in *Communications (ICC), 2014 IEEE International Conference on*, pp. 3064–3069, IEEE, 2014.

[62] Kiran, M., Pouyoul, E., Mercian, A., Tierney, B., Guok, C., and Monga, I., "Enabling intent to configure scientific networks for high performance demands," *Future Generation Computer Systems*, pp. –, 2017.

[63] Koponen, T., Amidon, K., Balland, P., Casado, M., Chanda, A., Fulton, B., Ganichev, I., Gross, J., Ingram, P., Jackson, E., Lambeth, A., Lenglet, R., Li, S.-H., Padmanabhan, A., Pettit, J., Pfaff, B., Ramanathan, R., Shenker, S., Shieh, A., Stribling, J., Thakkar, P., Wendlandt, D., Yip, A., and Zhang, R., "Network virtualization in multi-tenant datacenters," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, (Seattle, WA), pp. 203–216, USENIX Association, 2014.

[64] Kreutz, D., Ramos, F. M. V., Verssimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S., "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, pp. 14–76, Jan 2015.

[65] Kreutz, D., Ramos, F. M., and Verissimo, P., "Towards secure and dependable software-defined networks," *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, p. 55, 2013.

[66] Lenrow, D., "Intent-based networking seeks network effect." https://www.sdxcentral.com/articles/contributed/intent-based-networking-seeks-network-effect-david-lenrow/2015/09/.

[67] Lin, P., Bi, J., Wolff, S., Wang, Y., Xu, A., Chen, Z., Hu, H., and Lin, Y., "A west-east bridge based sdn inter-domain testbed," *Communications Magazine, IEEE*, vol. 53, no. 2, pp. 190–197, 2015.

[68] Lin, P., Hart, J., Krishnaswamy, U., Murakami, T., Kobayashi, M., Al-Shabibi, A., Wang, K.-C., and Bi, J., "Seamless interworking of sdn and ip," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, (New York, NY, USA), pp. 475–476, ACM, 2013.

[69] Linux, "ip-netns - process network namespace management." http://man7.org/linux/man-pages/man8/ip-netns.8.html. Accessed: 2016-05-04.

[70] linuxcontainers.org, "Linux containers." https://linuxcontainers.org/.

[71] Maldonado-Lopez, F. A., Calle, E., and Donoso, Y., "Checking multi-domain policies in SDN," *International Journal of Computers Communications & Control*, vol. 11, no. 3, pp. 428–440, 2016.

[72] Mambretti, J., Chen, J.-J., and Yeh, F., "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*, pp. 1–6, IEEE, 2014.

[73] Mambretti, J., Chen, J., and Yeh, F., "Software-defined network exchanges (sdxs): Architecture, services, capabilities, and foundation technologies," in *Teletraffic Congress (ITC), 2014 26th International*, pp. 1–6, IEEE, 2014.

[74] Matias, J., Garay, J., Mendiola, A., Toledo, N., and Jacob, E., "Flownac: Flow-based network access control," in *2014 Third European Workshop on Software Defined Networks*, pp. 79–84, Sept 2014.

[75] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J., "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008.

[76] MEF and others, "An industry initiative for third generation network and services." https://www.mef.net/third-network/third-network-white-paper. Accessed: 2017-01-15.

[77] Monga, I., Guok, C., Johnston, W. E., and Tierney, B., "Hybrid networks: lessons learned and future challenges based on esnet4 experience," *IEEE Communications Magazine*, vol. 49, pp. 114–121, May 2011.

[78] Nayak, A. K., Reimers, A., Feamster, N., and Clark, R., "Resonance: Dynamic access control for enterprise networks," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, WREN '09, (New York, NY, USA), pp. 11–18, ACM, 2009.

[79] Newman, H. B., Barczyk, A., and Bredel, M., "OLiMPS. openflow link-layer multipath switching," tech. rep., California Institute of Technology, Pasadena, CA (United States), 2014.

[80] Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., and Turletti, T., "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 1617–1634, Third 2014.

[81] ONF, "Openflow switch specification version 1.5.1 (protocol version 0x06)." https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf, Mar 2015. Accessed: 2016-05-04.

[82] ONRL, "Transferring data with bbcp." https://www.olcf.ornl.gov/kb_articles/transferring-data-with-bbcp/. Accessed: 2017-09-04.

[83] Open-Grid-Forum, "Network service interface." https://redmine.ogf.org/projects/nsi-wg. Accessed: 2017-01-14.

[84] OSRG, "Ryu SDN framework." https://osrg.github.io/ryu/, 2017.

[85] Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., Amidon, K., and Casado, M., "The design and implementation of open vswitch," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, (Oakland, CA), pp. 117–130, USENIX Association, 2015.

[86] Pittsburgh-Supercomputing-Center, "SLASH2 file system." https://github.com/pscedu/slash2.

[87] Plante, J. M., Davis, D. A. P., and Vokkarane, V. M., "Parallel and survivable multipath circuit provisioning in esnet's oscars," *Photonic Network Communications*, vol. 30, pp. 363–375, Dec 2015.

[88] Ricci, R. and Feamster, N., eds., *Report of the NSF Workshop on Software Defined Infrastructures and Software Defined Exchanges*, (Washington, DC), Feb. 2016.

[89] Schiff, L. and Schmid, S., "Study the past if you would define the future: Implementing secure multi-party sdn updates," in *2016 IEEE International Conference on Software Science, Technology and Engineering (SWSTE)*, pp. 111–116, June 2016.

[90] SDxCentral, "[Market Report] The Future of Network Virtualization and SDN Controllers." https://www.sdxcentral.com/wp-content/uploads/2016/09/SDxCentral-Future-of-Network-Virtualization-and-SDN-Controllers-Reoprt-2016-D.pdf, 2016.

[91] sflow.org, "sFlow - making the network visible." http://www.sflow.org/, 2017.

[92] Shanmugasundaram, K., Memon, N., Savant, A., and Bronnimann, H., *ForNet: A Distributed Forensics Network*, pp. 1–16. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

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

[94] Shibboleth-Consortium, "Shibboleth." https://shibboleth.net/. Accessed: 2017-01-22.

[95] SoX, "Southern crossroads." http://www.sox.net/about-us/.

[96] Stringer, J., Pemberton, D., Fu, Q., Lorier, C., Nelson, R., Bailey, J., Correa, C. N., Esteve Rothenberg, C., and others, "Cardigan: SDN distributed routing fabric going live at an internet exchange," in *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pp. 1–7, IEEE, 2014.

[97] Teixeira, R. and Rexford, J., "A measurement framework for pin-pointing routing changes," in *Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting: Research, Theory and Operations Practice Meet Malfunctioning Reality*, NetT '04, (New York, NY, USA), pp. 313–318, ACM, 2004.

[98] Tepsuporn, S., Al-Ali, F., Veeraraghavan, M., Ji, X., Cashman, B., Ragusa, A. J., Fowler, L., Guok, C., Lehman, T., and Yang, X., "A multi-domain sdn for dynamic layer-2 path service," in *Proceedings of the Fifth International Workshop on Network-Aware Data Management*, NDM '15, (New York, NY, USA), pp. 2:1–2:8, ACM, 2015.

[99] The-Bro-Project, "[Bro-Dev] OpenFlow Analyzer." http://mailman.icsi.berkeley.edu/pipermail/bro-dev/2016-October/012224.html, 2017.

[100] The-Bro-Project, "The bro network security monitor." https://www.bro.org/, 2017.

[101] Venugopal, S., Chu, X., and Buyya, R., "A negotiation mechanism for advance resource reservations using the alternate offers protocol," in *2008 16th Interntional Workshop on Quality of Service*, pp. 40–49, June 2008.

[102] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and Spence, D., "Rfc 2904 - aaa authorization framework." https://tools.ietf.org/html/rfc2904, 2000. Accessed: 2016-05-04.

[103] Wang, X. and Schulzrinne, H., "Rnap: A resource negotiation and pricing protocol," in *in International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV99), Basking*, Citeseer, 1999.

[104] Wundsam, A., Levin, D., Seetharaman, S., and Feldmann, A., "Ofrewind: Enabling record and replay troubleshooting for networks," in *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'11, (Berkeley, CA, USA), pp. 29–29, USENIX Association, 2011.

[105] Xiao, P. and Hu, Z., "Two-dimension relaxed reservation policy for independent tasks in grid computing," *Journal of Software*, vol. 6, no. 8, pp. 1395–1402, 2011.

[106] Yakasai, S. T. and Guy, C. G., "Flowidentity: Software-defined network access control," in *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*, pp. 115–120, Nov 2015.

[107] Zhang, H., Reich, J., and Rexford, J., "Packet traceback for software-defined networks," *Princeton University*, 2015.

[108] Zhang, Z., Bockelman, B., Carder, D. W., and Tannenbaum, T., "Lark: Bringing network awareness to high throughput computing," in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pp. 382–391, May 2015.

[109] Zurawski, J., Ball, R., Barczyk, A., Binkley, M., Boote, J., Boyd, E., Brown, A., Brown, R., Lehman, T., McKee, S., Meekhof, B., Mughal, A., Newman, H., Rozsa, S., Sheldon, P., Tackett, A., Voicu, R., Wolff, S., and Yang, X., "The dynes instrument: A description and overview," *Journal of Physics: Conference Series*, vol. 396, no. 4, p. 042065, 2012.

VITA

Joaquin Chung received both his B.S. in Electrics and Communications Engineering (2007) and his M.Sc. in Communication Systems Engineering with Emphasis in Data Networks (2013) from University of Panama, Panama. He received his Ph.D in Electrical and Computer Engineering in December 2017, under the supervision of Dr. Henry Owen and Dr. Russell Clark at Georgia Institute of Technology, Atlanta, USA. He is a Fulbright alumni and an IEEE member. His research interests include software-defined networking, software-defined exchanges, network function virtualization, network security, and the Internet of Things.