Advance Reservation Access Control Using Software-Defined Networking and Tokens

Joaquin Chung[a,*], Eun-Sung Jung[c], Rajkumar Kettimuthu[d], Nageswara S. V. Rao[e], Ian T. Foster[d], Russ Clark[b], Henry Owen[a]

[a] *School of Electrical and Computer Engineering, Georgia Institute of Technology, USA*
[b] *College of Computing, Georgia Institute of Technology, USA*
[c] *Hongik University, South Korea*
[d] *Math and Computer Science Division, Argonne National Laboratory, USA*
[e] *Computer Science and Mathematics Division, Oak Ridge National Laboratory, USA*

Abstract

Advance reservation systems allow users to reserve dedicated bandwidth connection resources from an advanced high-speed network. A common use case for such systems is data transfers in distributed science environments in which a user wants exclusive access to the reservation. However, current advance network reservation methods cannot ensure exclusive access of a network reservation to the specific flow for which the user made the reservation. We present here a novel network architecture that addresses this limitation and ensures that a reservation is used only by the intended flow. We achieve this by leveraging software-defined networking (SDN) and token-based authorization. We use SDN to orchestrate and automate the reservation of networking resources, end-to-end and across multiple administrative domains, and tokens to create a strong binding between the user or application that requested the reservation and the flows provisioned by SDN.

Keywords: Advance reservation system, admission control, software-defined networking, tokens

1. Introduction

Research and education networks (RENs) are characterized by high-speed backbone networks that support the needs of research and education communities within a geographic region. RENs often provide dedicated connections for individual research projects, and experimenters can establish and manage these connections through advance reservation systems [1] such as the Internet2 advanced layer 2 service (AL2S) [2] and the Energy Science Network (ESNet) on-demand secure circuits and advance reservation system (OSCARS) [3]. 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 [4]. 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. Moreover, setting up these reservations is a manual process that involves many network operators and can take from five to 45 days depending on the number of domains involved, as noted by Ibarra et al. [5].

Software-defined networking (SDN) is a novel networking paradigm that enables global network programmability and rapid innovation by decoupling the control and data planes of network devices. An additional benefit of SDN is a fine-grained flow definition that allows for firewall-like services on network switches. SDN is widely used in data centers for network policy enforcement, traffic engineering, and tenant isolation. However, SDN is not enough to ensure network access control when user identification is required. Token-based authentication/authorization is widely used in web service architectures today. For instance, whenever one accesses a third-party site with a Facebook credential, the OAuth 2.0 protocol (RFC 6749), which uses tokens, is invoked to authorize the third-party site access to some resources in that individual's Facebook profile. Furthermore, OpenStack, a popular cloud orchestration open source project, uses tokens to authorize numerous application program interfaces (APIs) that require access to its services.

We present here a novel network architecture that provides advance reservation access control by leveraging SDN and token-based authorization. Our architecture is composed of three main elements: (1) an orchestrator that receives requests from users or applications and manages networking resources between sites, (2) a WAN controller that represents an advance reservation system connecting sites involved in a specific data transfer, and (3) site SDN

*Corresponding author
Email addresses: joaquin.chung@gatech.edu (Joaquin Chung), ejung@hongik.ac.kr (Eun-Sung Jung), kettimut@anl.gov (Rajkumar Kettimuthu), raons@ornl.gov (Nageswara S. V. Rao), foster@mcs.anl.gov (Ian T. Foster), russ.clark@gatech.edu (Russ Clark), henry.owen@ece.gatech.edu (Henry Owen)

controllers that manage the installation of flow rules on site switches so as to extend a reservation from the border router all the way to the data transfer node in each site. The full workflow is automated, and no involvement from a network operator is required, thus reducing the provisioning time from several days to a few minutes. Our contributions are the following:

1. A system that automates the advance reservation provisioning process using multidomain SDN orchestration.
2. A system that uses tokens to strongly bind an end-to-end flow to the user or application that requested the reservation.

The reminder of this paper is as follows. Section 2 presents background material and Section 3 presents related work. Section 4 defines our system architecture. Section 5 describes our implementation and evaluation experiments. Section 6 discusses our results, and Section 7 presents our conclusions and future work.

2. Background and Motivation

We provide background on advance reservation systems, software-defined networking, and tokens.

2.1. Advance Reservation Systems

Advance reservation systems allow users to request and manage connections over a high-speed wide area network (WAN) [1]. 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 is identified by a VLAN ID. If a site does not have a high-speed Science DMZ [6], science flows have to compete with campus LAN traffic before reaching the advance reservation in the border router. Users can extend an advance reservations to the end host by manually extending VLANs on each site, but this manual provisioning of VLANs on site can take several days. For instance, the coordination of the provisioning process without automation is reported to take between five and 45 days[5]. Analysis of the durations of active OSCARS reservations on one day in 2016 shows that all are for a year or longer, and 40% (28 out of 68) are for 3.60 to 6.20 years: see Figure 1. Automating the advance reservation process reduces the provisioning time, as demonstrated in [5]. We hypothesize that automation of provisioning may provide a more flexible and dynamic system, with shorter reservation duration.

A novel internetworking paradigm, software-defined exchange (SDX), allows multiple independent administrative domains to share computing, storage, and networking resources. This effort is promoted mainly by REN users and operators. Currently, networking researchers use SDX to incorporate SDN technologies into the networking infrastructure of Internet exchange points (IXPs) [7] and academic exchange points [8]. An SDX can be regarded as a next-generation advance reservation systems, because it seeks to manage and allocate not only networking, but also computing and storage resources over multiple domains.

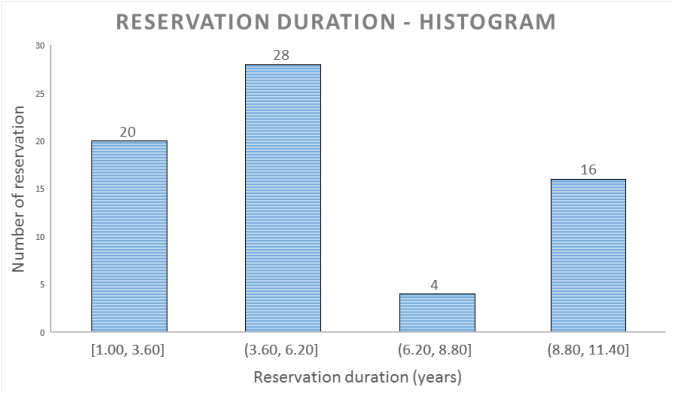

Figure 1: Histogram of OSCARS reservations duration. Data from `https://my.es.net/oscars` on August 1, 2016

2.2. Software-Defined Networking

The SDN paradigm [9] decouples the control and data planes of network devices. 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. The *data plane* comprises potentially many forwarding devices or SDN-enabled switches. The *control plane* is a logically centralized entity, generally known as an SDN controller, which can be composed of a single server or several distributed SDN controllers. The SDN controller communicates with SDN switches through the southbound interface (OpenFlow[10] being the most widely deployed) and with the network applications through the northbound interface. For distributed SDN controllers, a west-east interface can be added to enable communication between SDN controllers within the same administrative domain. If these controllers belong to independent administrative domains, a multidomain SDN [4] can be used to automate the provisioning of advance reservations.

SDN applications define the behavior of the network (e.g., switching, routing, load-balancing). The SDN controller translate this behavior into flow rules and install these rules in the data plane. An SDN flow rule typically comprises a *match* (matching field of the TCP/IP header) and an *action* (e.g., forward, drop, or send to controller). SDN is important because it allows finer and more flexible segmentation than that achievable with conventional methods (e.g., only VLANs or subnets), because a match

rule can match any permitted combination of fields from the TCP/IP header.

2.3. Tokens

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, an enforcement point has to validate the token against a centralized secure token service (STS). This approach may not need a PKI because all token information is stored in the STS.

3. Related Work

An early approach to defining a system architecture for advance reservation of bandwidth channels on research and education networks was UltraScience Net[11, 12]. More than a decade ago, this architecture defined a separate control and data plane and a bandwidth scheduler. The southbound interface of UltraScience Net was based on a transaction language communicated over a command line interface. The use of OpenFlow is a significant improvement on that work, because OpenFlow provides an open interface for data plane configuration, freeing the architecture from vendor-specific solutions.

Ibarra et al. [5] described the deployment of SDN and OpenFlow on the AmLight international research and education network, which promotes collaboration research between the US and Latin America, with the goals of improving operations efficiency and providing network programmability. Network programmability was provided by using Internet2's FlowSpace Firewall (FSF) and the Open Exchange Software Suite (OESS) SDN controller. With the new SDN AmLight, the provisioning time for a layer 2 circuit that involves up to three domains was reduced from five days and 10 emails to less than two minutes and no emails. Although SDN Amlight also automates provisioning of multidomain network reservations, its definition of a domain is a countrywide network (e.g., Internet2 and ESNet in the United Stated and RNP in Brazil). In contrast, our focus is on 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. [4] tested the use of end-to-end layer 2 paths for large dataset transfers over an existing deployment called DYNES (Dynamic Network System) [13]. The DYNES system uses OESS and OSCARS in multiple domains to establish dedicated layer 2 circuits. OESS is an intra-domain SDN controller that controls switches using OpenFlow [10], while OSCARS supports inter-domain service. 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.

Lark [14] 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 this architecture, each job is assigned to a separate network namespace [15], and each HTCondor node has a virtual switch (e.g., Open vSwitch (OVS) [16] or Linux bridge) that interconnects network namespaces to physical interfaces. This scheme allows users to change the network layer of a single Lark node when they submit batch jobs. 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 advance reservation access control system. Lark and our proposed system are thus complementary.

The Developing Applications with Networking Capabilities via End-to-end SDN (DANCES) [17] project seeks to enhance the performance of cyberinfrastructure applications (e.g., GridFTP [18] data transfers, SLASH2 [19] 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 accepted a request if: (1) resources are available on the network and (2) the user is authorized to request this amount of bandwidth.

Network access control (NAC), standardized as IEEE 802.1X [20], is a common computer security approach to authenticate endpoints and grant them access to a computer network. NAC was an early SDN application, with the main focus being policy enforcement. Casado et al. [21] 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. [22] proposed Resonance, a system for securing enterprise networks by using dynamic access control policies and network devices as enforcement points. FlowNAC [23] and FlowIdentity [24] 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.

Gommans et al. [25] proposed a token-based access control mechanism for multidomain lightpath (i.e., a fiber optics path) REN reservations. They identified and demon-

strated 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 study lies in its integration of SDN access control and token-based multidomain authorization.

4. System Architecture

We leverage SDN and token-based authorization to develop a network architecture that supports extending an advance reservation from a WAN border router to an endpoint, programmatically (i.e., without intervention of a network operator), and across multiple domains. Our architecture comprises an *orchestrator* that handles user requests and manages networking resources, a *WAN controller* representing an advance reservation system that connects 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 such as AL2S and OSCARS.

Figure 2 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.
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. Implementation and Evaluation

We conducted experiments on the ESNet infrastructure testbed. As shown in Figure 3, we used two sites,

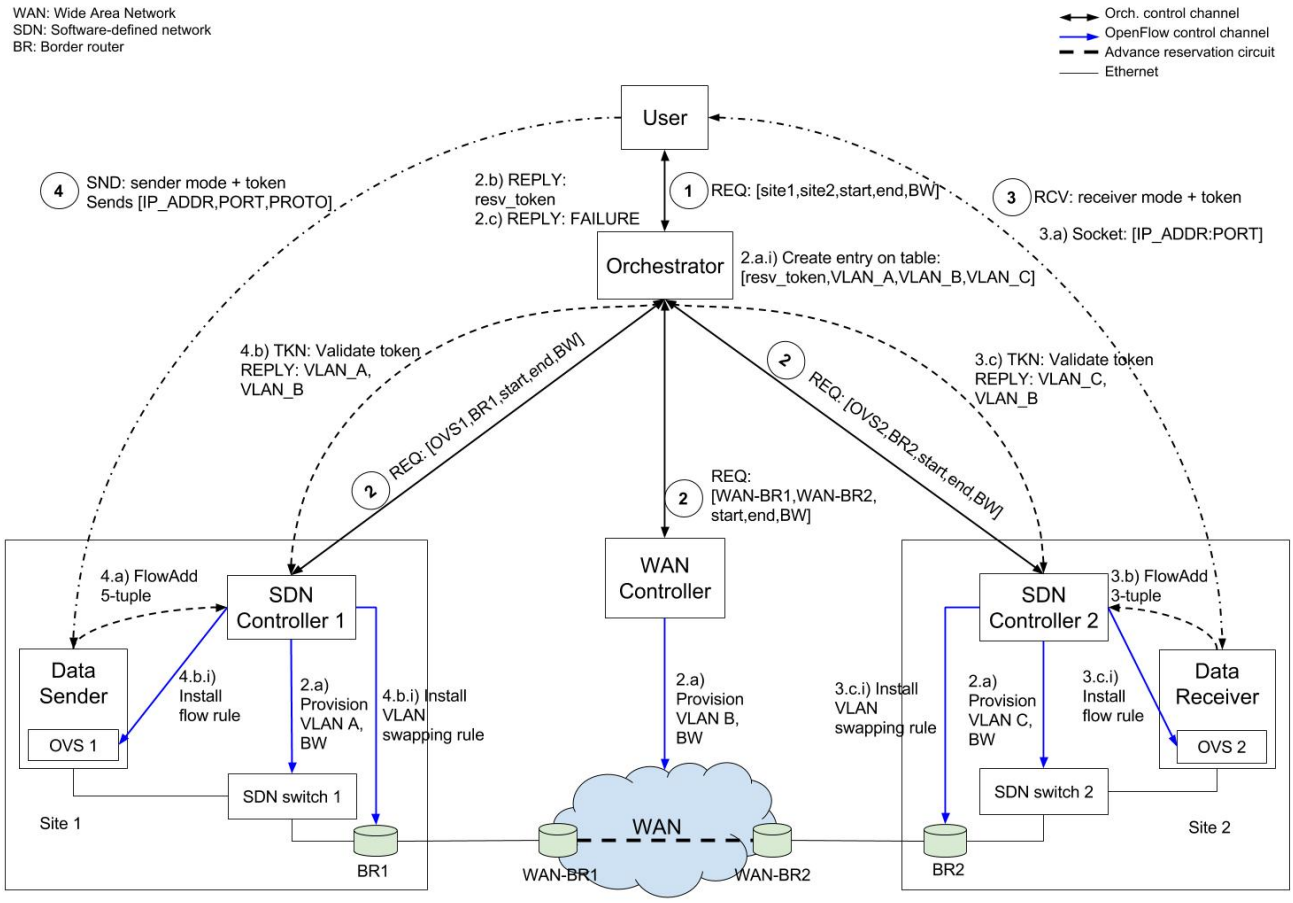

Figure 2: Block diagram of advance reservation access control using SDN and tokens. Only positive outcomes are shown

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 [16], one container endpoint, and one Ryu SDN controller [26]. The orchestrator runs on another container hosted at CERN. All containers run Ubuntu 14.04.

As shown in Figure 2, 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.

5.1. 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:

- 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.

5.2. Site Controller

A site controller manages reservation configurations at its site. The site switch in Figure 2 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

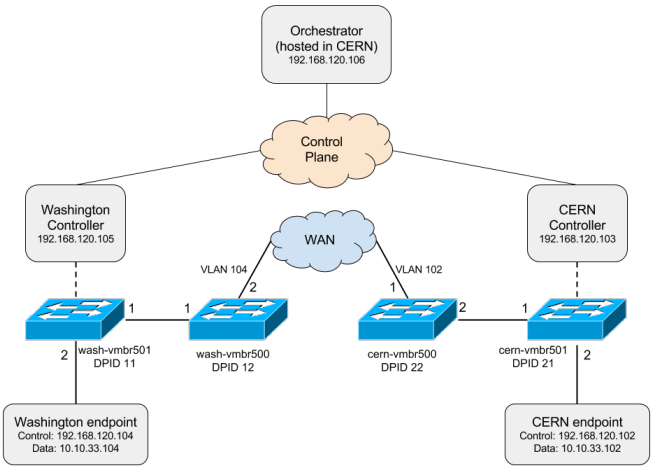

Figure 3: ESNet infrastructure testbed configuration for experiments

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.

5.3. 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 [18], 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.

5.4. 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 2). 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.

5.5. Evaluation

We evaluated the system on the ESNet testbed by measuring its latency in answering a request. We find that a circuit reservation request takes 181.98 ms on average. The time taken for sender and receiver configuration requests to data movers depends on where the orchestrator and data mover are located, around 33 ms if they are in close proximity but 1.27 seconds when they are on opposite sides of the WAN.

We find that the delay between orchestrator and participants (i.e., WAN controller and site controllers) is the main contributor to the latency of an answer. Additionally, our communication protocol implementation contacts each participant in sequence, and since we implemented an opaque token approach, token validation happens for each flow rule that has to be installed. Thus, 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 $4N$ token validation messages for a site with N OpenFlow switches. After installing all corresponding flows on each switch, we were able to verify that only the specific iperf connection was able to communicate between the two endpoints.

We then reimplemented our system using self-contained tokens and measured the latency of the system to answer a request. We selected JSON web tokens (JWT) on Python for our implementation. We used a preshared password for testing purposes instead of a full PKI. Table 1 shows the results compared with the opaque token approach.

We next evaluated the behavior of advance reservations under three scenarios:

Table 1: Latency of the system for opaque and self-contained tokens. All measurements are in msec.

	Token	
Request	**Opaque**	**Self-contained**
REQ	182.0	180.4
RCV	32.0	17.7
RCV over WAN	1,270.0	196.4
SND	34.7	17.7
SND over WAN	1,270.0	198.3

1. The reservation ends at the border router, and the LAN is congested. Traffic in the LAN is not necessarily going to the WAN, but it is affecting the performance of a user who wants to reach the border router. This is how most systems work currently.
2. The reservation is manually extended to the endpoint using VLANs. However, unauthorized users in the same endpoint still have access the reservation.
3. The reservation is programmatically extended to the endpoint, and access is controlled by using our system. In this scenario only authorized users can access the reservation.

We did not present bandwidth graphs because the main concern of our solution is access control. A bandwidth graph for scenario 2 will show two flows sharing the circuit, while in scenario 3 the unauthorized flow will not exist. Scenario 1 is more concerned with a quality of service (QoS) solution that combines advance reservation and application adaption [27]. Note that our solution protects the performance of only those flows that made an advance reservation. Hence, this solution should be paired with a best-effort channel in order to avoid an unintentional denial of service of flows between the same endpoints that did not make a reservation.

6. Discussion

We have presented a first attempt to develop a system for orchestrating the end-to-end provisioning of an advance reservation using SDN and token-based authorization. We demonstrated that our solution can reduce 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 design decisions in this proof-of-concept should be optimized. In the following subsections we provide more details about possible improvements to our work.

6.1. Token Scheme

As we can observe in Table 1, a self-contained token approach replies from 15 ms to 1 second 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.

6.2. SDN Site Controller

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. For instance, we could use an intent-based networking [28] API extended with token-based authorization.

7. Conclusion and Future Work

We have described a system that provides end-to-end advance reservation access control. By using multidomain SDN orchestration, our system automates the advance reservation provisioning process. Furthermore, by using 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 testbed and demonstrated that the provisioning time of an end-to-end reservation can be reduced from several days (manually) to a few minutes (automated). This result 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. In future work, we want to explore how the addition of QoS in an end-to-end advance reservation can improve utilization of network resources.

Acknowledgments

This work was supported in part by the U.S. Department of Energy under contract number DEAC02-06CH11357 and SDN-SF project, and the National Science Foundation, under grant ACI-1440761. We thank Eric Pouyoul from ESnet for his help in setting up the testbed. We also thank Sean Donovan for his feedback.

References

[1] N. Charbonneau, V. M. Vokkarane, C. Guok, I. Monga, Advance reservation frameworks in hybrid IP-WDM networks, IEEE Communications Magazine 49 (5) (2011) 132–139.

[2] Internet2, Layer 2 services, `http://www.internet2.edu/products-services/advanced-networking/layer-2-services/`, accessed: 2016-05-04.

[3] I. Monga, C. Guok, W. E. Johnston, B. Tierney, Hybrid networks: lessons learned and future challenges based on esnet4 experience, IEEE Communications Magazine 49 (5) (2011) 114–121. doi:10.1109/MCOM.2011.5762807.

[4] S. Tepsuporn, F. Al-Ali, M. Veeraraghavan, X. Ji, B. Cashman, A. J. Ragusa, L. Fowler, C. Guok, T. Lehman, X. Yang, A multi-domain SDN for dynamic layer-2 path service, in: 5th International Workshop on Network-Aware Data Management, NDM '15, ACM, New York, NY, USA, 2015, pp. 2:1–2:8, http://doi.acm.org/10.1145/2832099.2832101.

[5] J. Ibarra, J. Bezerra, H. Morgan, L. Fernandez Lopez, M. Stanton, I. Machado, E. Grizendi, D. Cox, Benefits brought by the use of OpenFlow/SDN on the AmLight intercontinental research and education network, in: International Symposium on Integrated Network Management, 2015, pp. 942–947. doi:10.1109/INM.2015.7140415.

[6] E. Dart, L. Rotman, B. Tierney, M. Hester, J. Zurawski, The Science DMZ: A network design pattern for data-intensive science, Scientific Programming 22 (2) (2014) 173–185.

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

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

[9] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proceedings of the IEEE 103 (1) (2015) 14–76. doi:10.1109/JPROC.2014.2371999.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review 38 (2) (2008) 69–74.

[11] N. S. V. Rao, W. R. Wing, S. M. Carter, Q. Wu, Ultrascience net: network testbed for large-scale science applications, IEEE Communications Magazine 43 (11) (2005) S12–S17. doi:10.1109/MCOM.2005.1541694.

[12] N. S. V. Rao, Q. Wu, S. Ding, S. M. Carter, W. R. Wing, A. Banerjee, D. Ghosal, B. Mukherjee, Control plane for advance bandwidth scheduling in ultra high-speed networks, in: 25TH IEEE International Conference on Computer Communications, 2006, pp. 1–5. doi:10.1109/INFOCOM.2006.35.

[13] J. Zurawski, R. Ball, A. Barczyk, M. Binkley, J. Boote, E. Boyd, A. Brown, R. Brown, T. Lehman, S. McKee, B. Meekhof, A. Mughal, H. Newman, S. Rozsa, P. Sheldon, A. Tackett, R. Voicu, S. Wolff, X. Yang, The DYNES instrument: A description and overview, Journal of Physics: Conference Series 396 (4) (2012) 042065, http://stacks.iop.org/1742-6596/396/i=4/a=042065.

[14] Z. Zhang, B. Bockelman, D. W. Carder, T. Tannenbaum, Lark: Bringing network awareness to high throughput computing, in: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2015, pp. 382–391. doi:10.1109/CCGrid.2015.116.

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

[16] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, M. Casado, The design and implementation of Open vSwitch, in: 12th USENIX Symposium on Networked Systems Design and Implementation, USENIX Association, Oakland, CA, 2015, pp. 117–130, https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/pfaff.

[17] V. Hazlewood, K. Benninger, G. Peterson, J. Charcalla, B. Sparks, J. Hanley, A. Adams, B. Learn, R. Budden, D. Simmel, J. Lappa, J. Yanovich, Developing Applications with Networking Capabilities via End-to-End SDN (DANCES), XSEDE16 (2016) 1–7.

[18] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Tuecke, GridFTP: Protocol extensions to FTP for the grid, Global Grid Forum, GFD-RP 20 (2003) 1–21.

[19] P. S. Center, SLASH2 file system, https://github.com/pscedu/slash2.

[20] 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) (2010) 1–205doi:10.1109/IEEESTD.2010.5409813.

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

[22] A. K. Nayak, A. Reimers, N. Feamster, R. Clark, Resonance: Dynamic access control for enterprise networks, in: 1st ACM Workshop on Research on Enterprise Networking, WREN '09, ACM, New York, NY, USA, 2009, pp. 11–18, http://doi.acm.org/10.1145/1592681.1592684. doi:10.1145/1592681.1592684.

[23] J. Matias, J. Garay, A. Mendiola, N. Toledo, E. Jacob, FlowNAC: Flow-based network access control, in: 2014 Third European Workshop on Software Defined Networks, 2014, pp. 79–84. doi:10.1109/EWSDN.2014.39.

[24] S. T. Yakasai, C. G. Guy, FlowIdentity: Software-defined network access control, in: IEEE Conference on Network Function Virtualization and Software Defined Network, 2015, pp. 115–120. doi:10.1109/NFV-SDN.2015.7387415.

[25] L. Gommans, L. Xu, Y. Demchenko, A. Wan, M. Cristea, R. Meijer, C. de Laat, Multi-domain lightpath authorization, using tokens, Future Generation Computer Systems 25 (2) (2009) 153 – 160, http://www.sciencedirect.com/science/article/pii/S0167739X08001179. doi:http://dx.doi.org/10.1016/j.future.2008.07.013.

[26] Ryu SDN Framework, http://osrg.github.io/ryu. Visited September 10, 2016.

[27] I. Foster, A. Roy, V. Sander, A quality of service architecture that combines resource reservation and application adaptation, in: 8th International Workshop on Quality of Service, IEEE, 2000, pp. 181–188.

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

Disclaimer