Future Generation Computer Systems 95 (2019) 534–547

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Orchestrating intercontinental advance reservations with software-defined exchanges

Joaquin Chung [a,*], Rajkumar Kettimuthu [a], Nam Pho [b], Russ Clark [b], Henry Owen [c]

[a] Mathematics and Computer Science Division, Argonne National Laboratory, USA
[b] College of Computing, Georgia Institute of Technology, USA
[c] School of Electrical and Computer Engineering, Georgia Institute of Technology, USA

HIGHLIGHTS

- An architecture for multidomain, multipath advance reservations in science networks that leverages SDN and SDX.
- A negotiation protocol for multidomain, multipath 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 negotiation protocol increases the reservation success ratio from 50% on a single-path system to 99% on our multi-path system according to our simulations.
- Architectural approaches at the SDX level that enable novel science network services, with recommendations for optimal SDX-rule provisioning and bandwidth-splitting strategies that allow data transfer protocols to take advantage of our system.

ARTICLE INFO

Article history:
Received 24 January 2018
Received in revised form 20 November 2018
Accepted 21 January 2019
Available online 24 January 2019

Keywords:
Multidomain advance reservation
Orchestrator
Software-defined networking
Software-defined exchanges
Bandwidth splitting

ABSTRACT

To interconnect research facilities across wide geographic areas, network operators deploy science networks, also referred to as research and education 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 among multiple administrative domains, which need to reach an agreement on a suitable advance reservation. The success rate of finding an advance reservation decreases as the number of participant domains increases for traditional systems because the circuit is composed over a single path. To improve provisioning of multidomain advance reservations, we have designed and implemented an architecture for end-to-end service orchestration in multidomain science networks that leverages software-defined exchanges for providing multipath, multidomain advance reservations. Our orchestration architecture improves the reservation success rate from 50% in single-path systems to 99% when four paths are available.

Published by Elsevier B.V.

1. Introduction

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 [1]. 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 Advanced Layer 2 Service (AL2S) [2], Open Exchange Software Suite (OESS) [3], and the On-Demand Secure Circuits and Advance Reservation System (OSCARS) [4].

In the case of intercontinental dedicated circuits, network operators may take from days to several weeks to plan and provision a circuit over multiple science networks because these tasks are typically done manually [5]. 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

* Corresponding author.
E-mail addresses: jchung@mcs.anl.gov (J. Chung), kettimut@mcs.anl.gov (R. Kettimuthu), nampho@gatech.edu (N. Pho), russ.clark@gatech.edu (R. Clark), henry.owen@ece.gatech.edu (H. Owen).

https://doi.org/10.1016/j.future.2019.01.040
0167-739X/Published by Elsevier B.V.

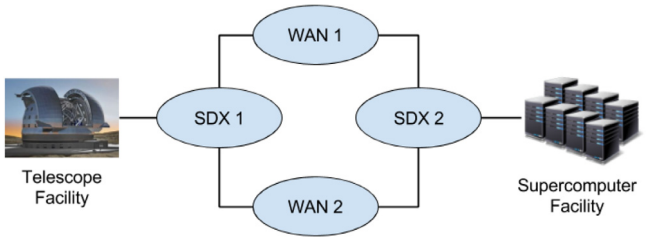


Fig. 1. SDX-enabled multidomain, multipath advance reservations scenario, with two SDXs connected through two different WANs, providing two independent paths between a telescope and a supercomputer.

times of science network services [5,6]. Because advance reservations are defined by endpoints, duration, and bandwidth, however, 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 in the specified time frame. 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, this system is not robust: a multidomain advance reservation will fail because of a single domain despite a majority of domains having available resources for the reservation. Moreover, the success rate of finding an agreement is inversely related to the number of participants. This is analogous to scheduling a multilegged flight with independent airlines from different consortia that do not share travel schedules.

Another challenge is that advance reservations terminate at the wide area network (WAN) border router of each domain and that participant domains are interconnected at single junction points [7]. As a result, multidomain advance reservations are generally provisioned over single paths, adding complexity to solving the advance reservation agreement problem. Furthermore, a data transfer has to compete with campus local area network (LAN) traffic in order to reach the advance reservation at the WAN border router of the research facility. Additionally, the interface for requesting these types of reservations is complex for domain scientists with limited networking knowledge.

Recently, software-defined exchanges (SDXs) have emerged as a new kind of cyberinfrastructure that allows independent administrative domains to share computing, storage, and networking resources by leveraging SDN [8]. 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 multidomain advance reservation. The initial benefit of adding SDXs is overcoming the limitation of single-path advance reservation. For instance, we may have two SDXs connected through two different advance reservation providers, providing two independent paths between two end sites (see Fig. 1). 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 multidomain end-to-end advance reservation, such as making reservations only at critical points or combining advance reservation and DiffServ quality of service (QoS) at the SDX. An SDX infrastructure also will enable novel science network services, for example, multipath bandwidth splitting across WANs, path migration, and multipoint-to-multipoint advance reservations.

To take advantage of an SDX-enabled advance reservation system, we require an orchestration framework. In this paper we propose a reference architecture for orchestrating end-to-end services in multidomain science networks. The contributions of this paper are threefold:

1. An architecture for multidomain, multipath advance reservations in science networks that leverages SDN and SDX. The architecture is composed of an orchestrator that requests services from participant domains and SDXs. We compare our proposed architecture using single-path vs. multipath advance reservations over multiple domains and evaluate the data transfer tools that the scientific community currently uses.
2. A negotiation protocol for multidomain, multipath 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 from 50% on a single-path system to 99% on our multipath system.
3. Architectural approaches at the SDX level that enable novel science network services. We provide recommendations for optimal SDX-rule provisioning and bandwidth-splitting strategies that allow data transfer protocols to take advantage of our system.

This paper is organized as follows. Section 2 provides background and motivation for this work. Sections 3 and 4 describe our architecture and design, respectively. Section 5 describes our implementation, Section 6 presents our evaluation results, and Section 7 discusses related work. Section 8 presents our conclusions and briefly outlines future work.

2. Motivation

Our work is motivated by the desirability of multidomain, multipath advance reservations and by recent developments in SDN and SDX technology.

2.1. Advance reservation systems

Traditionally, advance reservation 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 whether 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 advance reservations that increase the success rate of a reservation request in single-domain scenarios [9–11]. For multidomain, single-path advance reservations, however, flexible techniques lose their benefits if we cannot find an overlap among flexible reservation offered by each participant domain.

Let us consider a multidomain, single-path advance reservation with N participant domains and a success probability p for each individual domain. The success probability of the entire reservation request (i.e., all domains succeed) is p^N. For instance, according to Xiao and Hu [11], 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 multidomain advance reservation that allows path diversity and permits the required bandwidth to be distributed over multiple paths. For simplicity, let us consider two research facilities connected across two possible advance reservation systems as shown in Fig. 1. We are allowed to request advance reservations from both systems, and we obtain a successful multidomain, multipath 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

Fig. 2. Intercontinental R&E links originated from the United States.

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%. 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 3 (i.e., on average we could find three mutually exclusive paths between a source and a destination within any of these domains). Furthermore, we find this path diversity on intercontinental links originated from the United States, as shown in Fig. 2. The topology maps of ESnet [12] and Internet2 [13] report at least three links to Asia Pacific, three links to Latin America, and four links to Europe. This improvement is not possible, however, without architectural changes to the system that support end-to-end science network services. 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 multipath, multidomain advance reservations that enhance the performance of science data transfers.*

2.2. Software-defined networking

Under the SDN paradigm [14], the control and data planes of network devices are decoupled. This separation enables global, agile network programmability, rapid innovation, and independent evolution of control and data planes. The Open Networking Foundation (ONF) proposed an abstraction for SDN that divides the architecture into three layers: the infrastructure layer, representing the data plane; the control layer, representing the control plane; and the application layer, representing network applications (e.g., switching, routing, or load balancing). Additionally, the ONF proposed interfaces that enable communication between layers. The interface that allows the control layer to communicate with the infrastructure layer is commonly referred to as the southbound interface, and the interface that allows the application layer to communicate with the control layer is generally called the northbound interface.

By taking advantage of the agile programmability of SDN, Ibarra et al. [5] improved the provisioning time of international advance reservations in R&E networks from several days to a few minutes. In [6], 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 [5]. Although SDN effectively reduces provisioning times of advance reservations, however, international or intercontinental advance reservations will require WAN-optimized protocols for the coordination and composition of science network services. 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 multipath, multidomain advance reservation on the order of seconds.*

2.3. Software-defined exchange

A software-defined exchange is a meet-me point or marketplace where independent administrative domains can exchange computing, storage, and networking resources [15]. Currently, networking researchers incorporate SDN technologies into the networking infrastructure of Internet exchange points [16] and R&E exchange points [17] for agile programmability of these infrastructures. By taking into account the exchanged networking resources, we classified SDX solutions as follows [8]: (1) layer 3 SDXs, which exchange BGP updates in Internet exchange points [16, 18]; (2) layer 2 SDXs [17], which exchange multidomain Ethernet circuits in research and education networks; and (3) SDN SDXs [19], which interconnect SDN islands. SDXs will enable multipath, multidomain advance reservations. SDXs will also enable novel science network services such as multipath bandwidth-splitting across independent WAN providers, scheduled path migrations that are transparent to data transfer applications, and multipoint-to-multipoint advance reservations. Since SDX is a nascent technology, however, we need to know the advantages and disadvantages of using SDX as an interconnection point for multipath, multidomain advance reservations. For instance, 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 [20] and GridFTP [21]) that create multiple TCP streams, and we can distribute these streams over a multipath, multidomain advance reservation. Furthermore, data transfer protocols such as GridFTP support restart markers to handle failures. Hence, when networking resources are not available on a continuous time window in all domains, we propose using SDXs to transparently migrate advance reservations from one path to another in the middle of an advance reservation. We hypothesize that *by designing SDX services that take advantage of the features 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. Architecture overview

To support multipath, multidomain advance reservations, we require an architecture that takes advantage of the enriched connectivity provided by SDX. Our proposed architecture is composed of the following components (see Fig. 3):

1. Site controllers residing at 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.

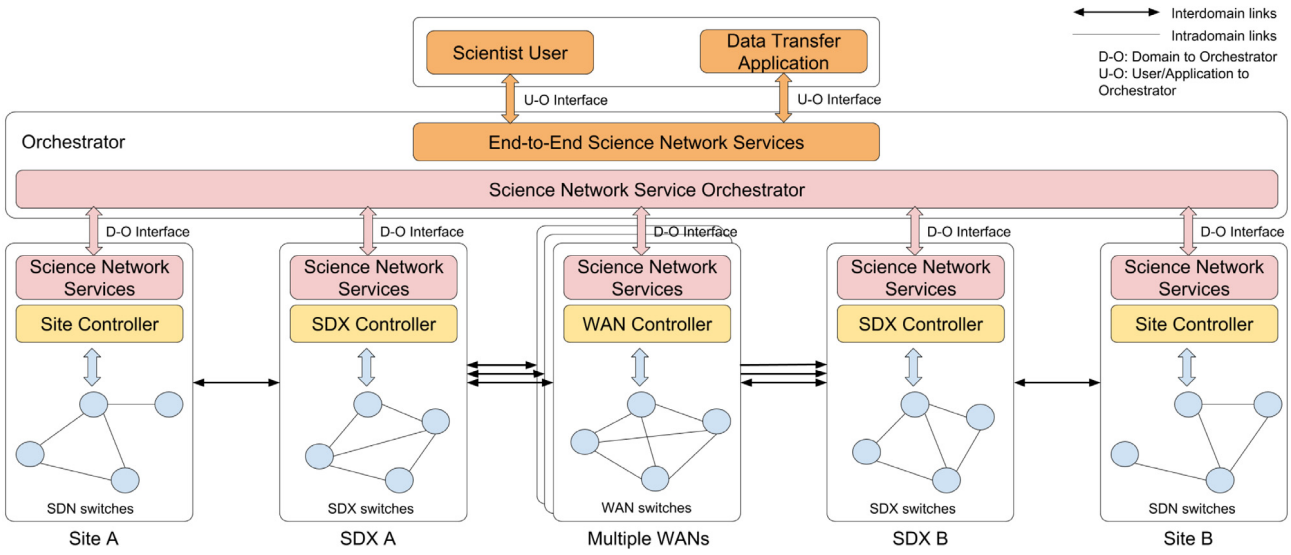


Fig. 3. Reference architecture for end-to-end service orchestration in multidomain science networks. Several independent administrative domains are connected by interdomain links and expose science network services to a centralized orchestrator through the domain-to-orchestrator (D-O) interface. The 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.

3.1. Site, WAN, and SDX controllers

The site, WAN, and SDX components of our architecture follow the same SDN abstraction proposed by the ONF, namely, infrastructure layer, control layer, and application layer. In our architecture, the application layer of SDN represents the science network services exposed by each type of controller (i.e., site, WAN, and SDX controller). In this context, the control layer of a site, WAN, or SDX controller may be any type of existing SDN controller, advance reservation system, or SDX controller. The main requirement is that the northbound interface of these controllers abstract the details of the network infrastructure and expose relevant science network services. More details about this type of interface are provided in Section 3.3.1. The infrastructure layer is composed of the data plane switches of each participant domain.

3.2. 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, to connect site A to site B in Fig. 3, the orchestrator needs to know whether 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. To be practical in multidomain environments, the orchestrator has to interact with the network resource managers at each domain in order to query status and reserve resources.

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 whether a path with the requested maximum delay is possible. Each queried domain does not have global knowledge about the end-to-end path, but each can commit to the latency of its portion of the end-to-end path. The orchestrator then has to evaluate whether a path that meets the end-to-end latency requirement can be formed. Otherwise, the orchestrator will try to find an alternative path or try to negotiate a maximum delay higher than 100 ms. With multiple paths and SDXs interconnecting them, the chances of composing a successful end-to-end service rise, as demonstrated in Section 2.

3.3. Interfaces and services

Users in our system are domain-expert scientists who in most cases do not have expertise in network operations but still need to request reservations in order to expedite their data transfers. Additionally, scientists use data transfer services (e.g., Globus [22]) 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 from the orchestrator by using interfaces that abstract network infrastructure details. 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.

3.3.1. Domain-to-orchestrator interface

The domain-to-orchestrator interface, depicted as *D-O interface* in Fig. 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 ESnet requirement review reports from 2013 to 2015 [23] 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 [21], 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 workflow such as scheduling a backup or changing the orientation of a remote telescope.

3.3.2. User-to-orchestrator interface

The user-to-orchestrator interface, depicted as *U-O interface* in Fig. 3, allows a scientist or a scientific application to request services from a science network orchestrator. The U-O interface includes flexible parameters that allow the orchestrator to negotiate a feasible 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, for this study we will focus on the D-O interface and the components pertaining the network infrastructure.

3.4. 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 satisfy a request, the orchestrator should try an alternative request that is still acceptable to the user. The orchestrator should also handle potential race conditions that arise when two or more users request the same resources at the same time. 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 multidomain, multipath advance reservations. In traditional settings, an orchestrator will be allowed to make reservations only 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; negotiate with several SDXs to interconnect these advance reservations; and compose a final end-to-end service.

4. Design

In this section we present the design challenges for a system that provides multidomain, multipath advance reservations in science networks. 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 circuit. For additional components of the orchestrator such as path computation and resource management, we take advantage of readily available implementations (e.g., Python NetworkX [24] and OpenStack Blazar [25]).

4.1. General workflow

This section describes the general workflow for requesting and composing multipath, multidomain advance reservations. We assume that multiple paths exist between two research facilities and that these paths traverse multiple administrative domains that provide connectivity and guaranteed bandwidth by using advance reservation systems. We also assume that SDXs serve as interconnection points for these administrative domains, enabling richer connectivity. An orchestrator (see Section 3.2) 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 (or bandwidth offers) the same way airlines allow us to consult flight availability. This should not be confused with open topology sharing, which is already supported by OSCARS and OESS [7].

Fig. 4 depicts the general workflow for requesting multidomain, multipath advance reservations. The workflow starts with a user requesting a flexible, multidomain advance reservation from 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. The orchestrator decomposes the user's request into individual flexible requests for each domain and SDX on the path, and it requests reservation offers from each participant domain. The orchestrator then uses these offers to compose a flexible end-to-end service, commit offers, and contact SDXs to make interconnections if an end-to-end service is possible; and it aborts unused offers. Otherwise, the orchestrator aborts all offers.

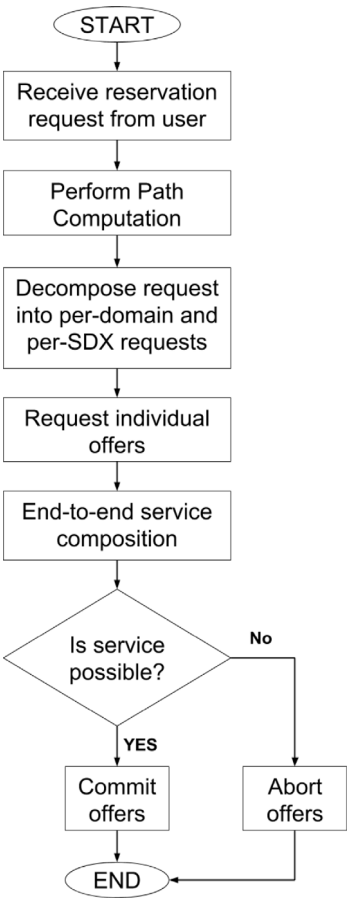


Fig. 4. General workflow for requesting multidomain, multipath advance reservations.

4.2. Negotiation protocol

In this section we take a deeper look at the negotiation protocol. The protocol is divided into two phases. Phase 1 requests offers from participant domains and composes an end-to-end service. Phase 2 commits the successful offers, aborts unused offers, and requests interconnection at SDXs. We note that not all participants are able or willing to provide reservation offers, either because they have legacy systems that do not provide reservation offers or because they have privacy concerns. We identify those domains that provide reservation offers as *flexible domains* and those that do not provide offers as *rigid domains*. Flexible domains are considered as the initial option to compose the end-to-end service. Rigid domains are considered only if flexible domains do not have enough resources. The rationale behind this strategy is that by considering rigid 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

Table 1
Negotiation protocol messages.

Message type	Description
Reservation	Message from the user to the orchestrator requesting a multidomain advance reservation
ReqOffers	Message from the orchestrator to flexible domains requesting advance reservation offers
SendOffers	Message from flexible domains to the orchestrator replying with a list of advance reservation offers
ReservationPrep	Message from the orchestrator to all participant domains and SDXs requesting the preparation of a reservation
Commit	Message from the orchestrator to all participant domains and SDXs committing a reservation already prepared
Abort	Message from the orchestrator to all participant domains and SDXs aborting a reservation already prepared
ReservationResp	Message for notifying whether a requested has succeeded or failed

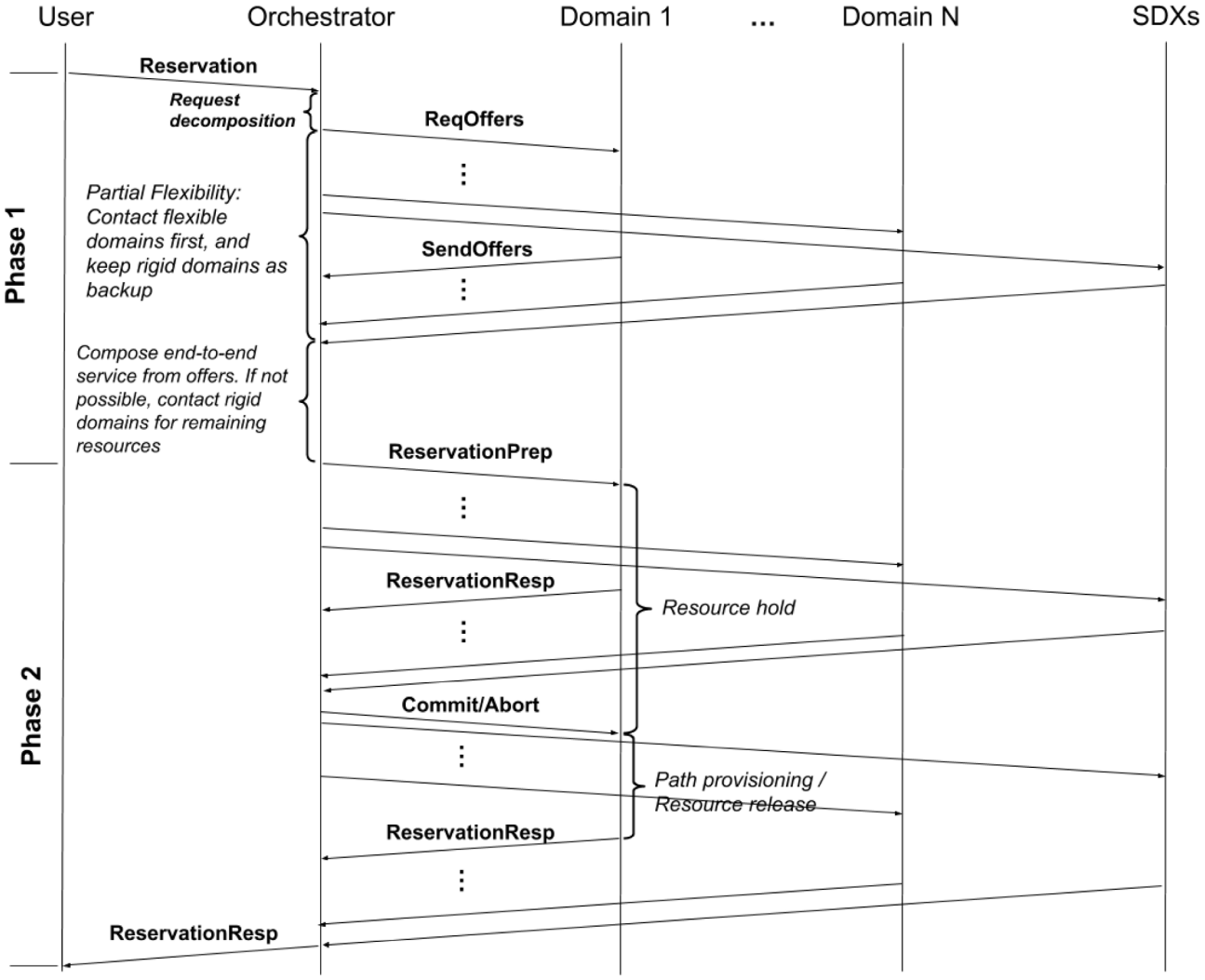


Fig. 5. Negotiation protocol for multipath, multidomain advance reservation with M flexible domains and $N - M$ rigid domains.

seven types of messages: *Reservation*, *ReqOffers*, *SendOffers*, *ReservationPrep*, *Commit*, *Abort*, and *ReservationResp*. We describe these in Table 1.

Fig. 5 shows the detailed negotiation protocol considering N participant domains, with M flexible domains and $N - M$ rigid domains. We consider three scenarios.

1. **No flexibility** ($M = 0$): All participant domains are rigid domains (i.e., only legacy advance reservation systems participate in the orchestration process).
2. **Full flexibility** ($M = N$): All participant domains are flexible domains (i.e., only systems that provide bandwidth reservation offers participate in the orchestration process).
3. **Partial flexibility** ($M \neq N$): rigid domains and flexible domains participate in the orchestration process (i.e., a mix of legacy advance reservation systems and systems that provide bandwidth offers participate in the orchestration process).

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 flexible and rigid domains participating in the process. The orchestrator sends *ReqOffers* messages to the M flexible 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 [6] 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 our 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 from the rigid 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.

4.3. SDX rules

As mentioned in Section 4.1, SDXs are considered interconnection points in our design. For simplicity, 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 virtual LANs (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.

Fig. 6 presents a block diagram of the bandwidth-splitting service. Our architecture takes advantage of the multistreaming nature of data transfer protocols (e.g., GridFTP and BBCP). 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 multipath, multidomain 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 path at the SDX.

The SDN switches in Fig. 6 have two ports: a WAN port that receives all VLAN IDs representing L2 tunnels and a LAN port that connects the end site. The SDN controllers receive a pool of VLANs from the orchestrator and creates flow rules on the SDN switches that tag each new packet from a specific flow appearing 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 create flow rules that remove the VLAN tag and forward the packet to the LAN port. The SDN controller selects VLAN IDs from the pool in a round-robin fashion. In order to ensure that all the traffic belonging to a single flow traverses the same circuit, synchronization or coordination between the SDN controllers assigning the VLANs must take place. Otherwise, the forward traffic of a TCP flow might traverse one tunnel, and all the ACKs might return over another tunnel.

We consider three approaches for provisioning bandwidth-splitting rules in an SDX: synchronized, unsynchronized, and 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 of flow table entries, however, and requires prior knowledge of the TCP port ranges used in both data transfer nodes. For those reasons, we do not consider this fourth approach in this study.

4.4. 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. Alternatively, a science DMZ [26] – a dedicated network for scientific data transfers – may be used to protect the science network traffic. In the absence of either of these mechanisms, traditional QoS techniques combined with statically provisioned VLANs may be used to provide prioritized access to the network for scientific data transfers.

5. Implementation

In this section we present the implementation of an orchestrator for multipath, multidomain advance reservations and the implementation of an SDX to support these services.

5.1. Orchestrator implementation

We implemented the orchestrator in Python using an agent-based approach. Each participant domain hosts an agent that receives offer requests from an orchestrator, processes those requests internally, and sends offers back to the orchestrator. We selected an agent-based approach as opposed to simply consuming 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 by using the general remote procedure call (gRPC) protocol [27], a high-performance RPC framework optimized for distributed-computing and mobile environments. The gRPC protocol uses HTTP/2, a binary protocol that multiplexes multiple streams over a single TCP connection, for establishing communication channels between servers and stubs. In contrast, traditional REST interfaces based on HTTP/1.1 use multiple TCP connections to issue parallel requests. Another advantage of gRPC is that it uses protocol buffers [28] for defining services and message types and for serializing data.

5.2. Negotiation protocol implementation

First, we assume that the path computation component has determined the domains that provide an end-to-end path between source and destination. Second, we consider the three scenarios described in Section 4.2 (i.e., no flexibility, full flexibility, and partial flexibility). As a result, we define three variants of the negotiation protocol for bandwidth splitting:

1. **Equal Splitting:** This strategy could be applied to any scenario. It is more suitable for the *no flexibility* scenario, however, because it does not require the ability to request offers. In this approach the orchestrator divides the original bandwidth request into equal parts among the participant domains (see Algorithm 1).
2. **Partial Offers:** This approach is applicable mainly to the *partial flexibility* scenario; the orchestrator contacts the flexible 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 rigid domains (see Algorithm 2).

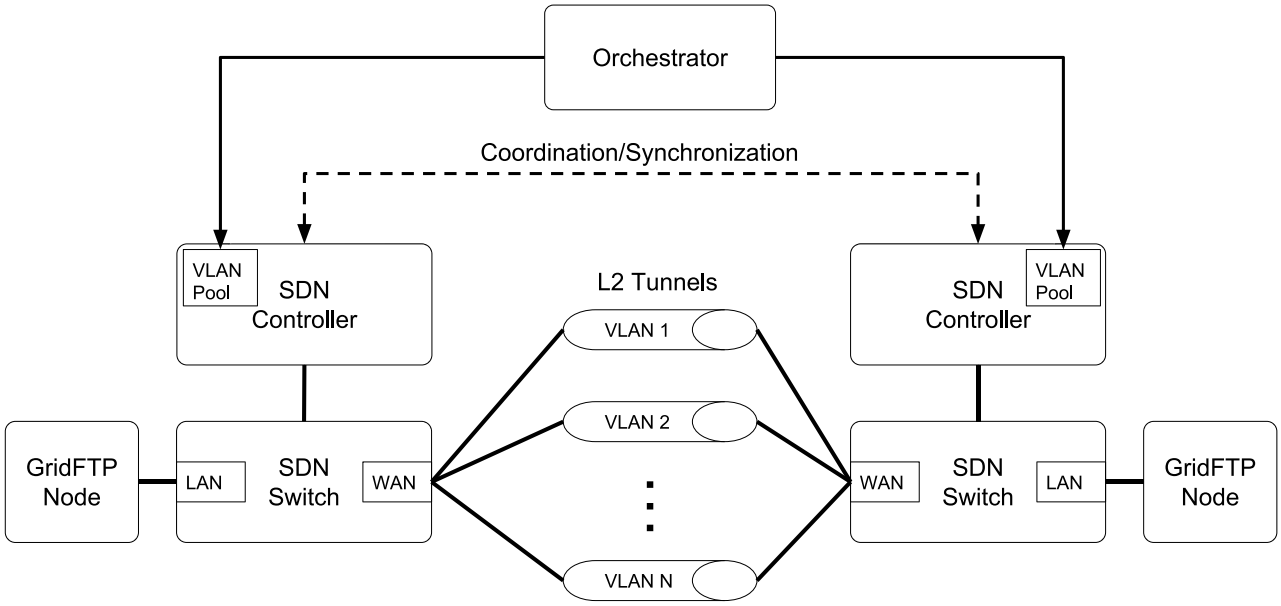


Fig. 6. Block diagram of bandwidth-splitting service components for SDX rule provisioning.

3. **Full Offers:** This approach is applicable only to the *full flexibility* 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 Equal Splitting

```
RD ← Set of rigid domain
FD ← Set of flexible domains
D ← RD ⋃ FD
N ← Total number of participant domains
M ← Number of flexible domains
EqSplitReq ← BwReq/N
for domain ∈ D do
  if EqSplitReq > AvailableBw then
    GoToPhase2()
  else
    ReservationFail()
  end if
end for
```

Algorithm 2 Partial Offers

```
RD ← Set of rigid domain
FD ← Set of flexible domains
D ← RD ⋃ FD
N ← Total number of participant domains
M ← Number of flexible domains
for domain ∈ FD do
  offers ← ReqOffers()
end for
if ∑ offers ≥ BwReq then
  GoToPhase2()
else
  EqSplitReq ← (BwReq − ∑ offers) / (N − M)
  for domain ∈ RD do
    if EqSplitReq > AvailableBw then
      GoToPhase2()
    else
      ReservationFail()
    end if
  end for
end if
```

Algorithm 3 Full Offers

```
RD ← Set of rigid domain
FD ← Set of flexible domains
D ← RD ⋃ FD
N ← Total number of participant domains
M ← Number of flexible domains
for domain ∈ D do
  offers ← ReqOffers()
end for
if ∑ offers ≥ BwReq then
  GoToPhase2()
else
  ReservationFail()
end if
```

5.3. SDX implementation

Our SDX implementation is based on AtlanticWave-SDX [29], an SDX controller written in Python that uses the Ryu SDN framework [30] as an OpenFlow [31] 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 its Web interface, but it does not 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) [32] 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 4.3.

We implemented the three rule provisioning strategies – synchronized, unsynchronized, and coordinated – as Ryu apps. All the strategies 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 control all the traffic of a single multidomain, multipath 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 controllers 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.

5.4. Practical considerations

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? It is important to note that we are still years away from full implementation of SDN in university campuses and research facilities. Moreover, network administrators are still reluctant to open production networks for third party configurations. To reduce complexity and mitigate security risks, the SDN infrastructure of end sites may run on enclaves. In the absence of SDN, pre-provisioned VLANs may be used to carry traffic of end-to-end circuits. We provide enough freedom in the orchestrator design to accommodate different levels of software-defined capabilities in the infrastructure ranging from least programmability (no state information and rigid interfaces for reservation such as OESS and OSCARS) to a full programmability. The deployment scenario can also range from a completely distributed set of orchestrators to a single centralized orchestrator. In a completely distributed scenario, each individual user runs an instance of the orchestrator. While this approach is nimble and enables faster adoption, it requires sophisticated protocols and back off mechanisms to resolve conflicts and avoid deadlocks when multiple orchestrators compete for the same resources. A single centralized orchestrator will help avoid concurrent requests (and thus conflicting demands), it becomes a single point of failure. Solutions such as replica servers [33] can be used to address this issue. A partially distributed and/or a hierarchical approach in which each science facility (or a set of major facilities) runs an orchestrator is also possible.

6. Evaluation

In this section we evaluate the success rate of the three variations of our negotiation protocol. We also evaluate the performance of several provisioning strategies for a multipath, multidomain advance reservation service.

6.1. Orchestrator microbenchmark

We evaluated the system latency of our orchestrator for requesting resources from eight participant domains, while varying the round-trip time (RTT) between the orchestrator and participants. We compared REST and gRPC for the communication channel between the orchestrator and participant domains. We used the GENI (Global Environment for Network Innovations) platform [34] to conduct our experiment. The results, plotted in Fig. 7, show 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 since the system latency remains under one second for all the values of RTT tested.

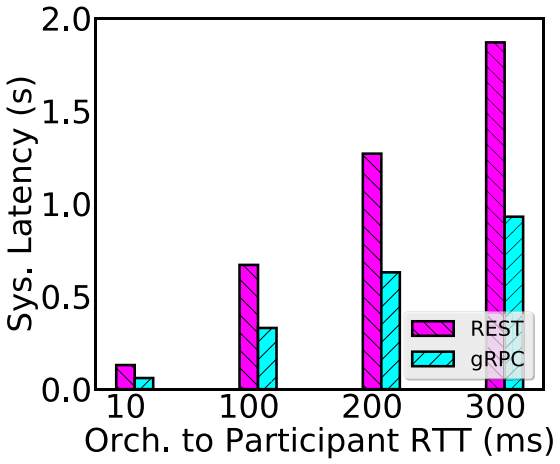


Fig. 7. System latency microbenchmark for an orchestrator requesting resources from eight participant domains using REST and gRPC and varying the RTT between participants and the orchestrator.

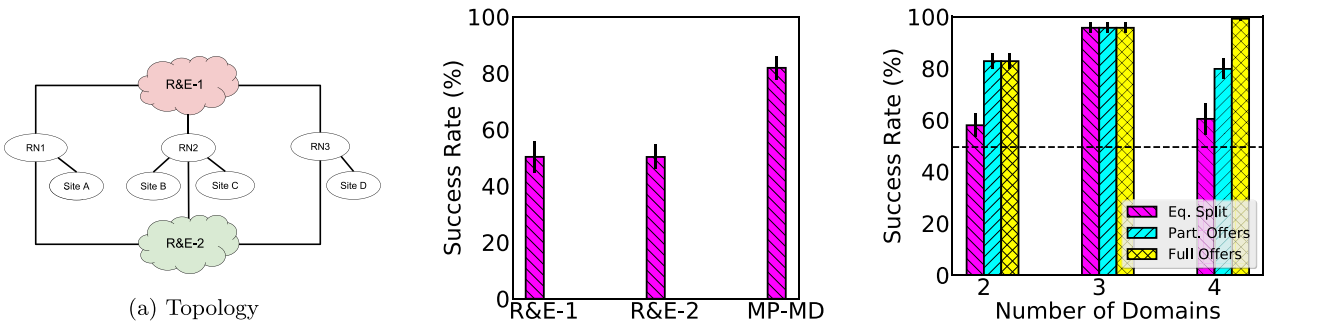
6.2. Multipath, multidomain advance reservations

To evaluate our multipath, multidomain advance reservation, we considered the topology depicted in Fig. 8(a). This topology is composed of four end sites (sites A, B, C, and D), connected to three regional networks (RN1, RN2, and RN3) where an SDX might reside. These three regional networks are further connected to two R&Es (R&E-1 and R&E-2). For our simulation, we created a registry of advance reservations for both R&Es. 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 generated a random request composed of a time window, a required bandwidth, a source, and a destination. We sent this request to both domains individually and evaluated whether the domains have enough available resources. For our multipath, multidomain advance reservation service we evaluated whether the sum of the available bandwidth in both domains satisfies the request. Fig. 8(b) shows that our multipath, multidomain approach has an 85% success rate when two independent paths are available, compared with an approximately 50% success rate for the state-of-the-art (single-path) approach.

6.3. Negotiation protocol success rate

To evaluate the success rate of the three variants of our negotiation protocol – equal splitting, partial offers, and full offers – we simulated a scenario in which an orchestrator can request advance reservations from up to four participant domains in order to compose a multipath, multidomain advance reservation. We chose four domains because this is a reasonable number of multiple intercontinental paths between two sites, as mentioned in Section 2.1. For each participant domain, we randomly 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 1000 entries. We ran the simulation 32 times and took the averages for each scenario.

Fig. 8(c) shows the results of our simulations. The horizontal line represents the success rate for a single domain, which is



(a) Topology (b) State-of-the-art (c) Negotiation protocol

Fig. 8. Simulation topology and results: (a) topology for multipath, multidomain advance reservation evaluation simulation; (b) success rate for multipath, multidomain advance reservation evaluation compared with the state-of-the-art methods; and (c) negotiation protocol success rate for three bandwidth-splitting strategies and up to four participant domains.

49.56% under our assumptions. All of our strategies outperform the baseline. In the worst-case scenario (i.e., equally splitting under no flexibility), the success rate of our orchestrator is approximately 58%. Under the best conditions (four flexible domains), our orchestrator (with full offers) achieves an approximately 99% success rate, or 2*X* improvement. We note that for this simulation the commit phase of the negotiation protocol is not executed; that is, the system is in its initial state for each simulation run. Intuition tells us that as more reservations are committed, the success rate of future requests will decrease. We expect to see lower success rate results in real-world scenarios, and we leave the modeling of these conditions and its evaluation for future work.

6.4. SDX experimental setup

Fig. 9 shows the topology of our experimental setup, and Table 2 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 [35] (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 is paired with an SDX instance. The orchestrator runs on another Docker container and communicates with the agents by using gRPC. We used two customized Supermicro servers as GridFTP endpoints. Each server runs a Docker container with either a GridFTP server or a GridFTP client (globus-url-copy), an Open vSwitch (OVS) [32] virtual switch, and a Ryu SDN controller [30]. We used the OVS switches and Ryu controllers at the endpoints because of limited available ports on the Corsa switch to create more virtual switch instances. 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].

6.5. Data transfer methods

We measured the throughput baseline of a data transfer over a single-path, multidomain advance reservation versus a data transfer over a multipath, multidomain 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, a well-known bandwidth-measuring tool, as a

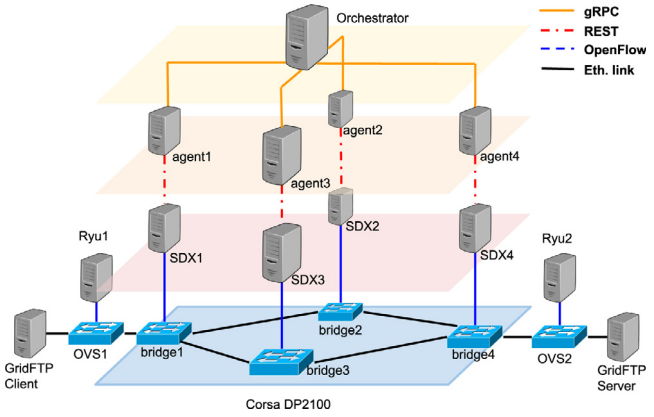


Fig. 9. Experimental setup topology.

Table 2
Experimental setup and equipment specifications.

Equipment	Specifications
Corsa DP2100	OpenFlow 1.5, multiple flow tables, multicontext virtualization, 48 Gb packet buffer, 100 Gbps line rate
Dell PowerEdge R220	Ubuntu Server 16.04, 16 GB RAM, four Intel(R) Xeon(R) CPU E3-1220 v3 @ 3.10 GHz processors, four-port Gigabit Ethernet card
Customized Supermicro	Ubuntu Server 16.04, 8 GB RAM, four Intel(R) Xeon(R) CPU X3430 @ 2.40 GHz, two Gigabit Ethernet interfaces

reference. Fig. 10 shows the results of performing the 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 37.5 GB of data at line rate over a 1 Gbps link. For GridFTP disk-to-disk, we transferred a 20 GB file, which is a reasonable size for a scientific dataset [23]. As the maximum disk throughput of our GridFTP endpoints we obtained 92.34 MB/s or 738.72 Mbps on average.

Fig. 10(a) shows that iperf3 reaches only 514 Mbps of throughput for a single L2 tunnel of 1 Gbps of bandwidth, while GridFTP reaches only 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 able

Table 3
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

to use only approximately 67% (670.36 Mbps on average) of the link with parallel streams.

Fig. 10(b) shows the throughput baseline after splitting the bandwidth reservation between two 500 Mbps L2 tunnels. For one and two TCP streams per tunnel, iperf3 achieves 936 Mbps of throughput. However, it is able to achieve only 883 Mbps with four parallel TCP streams per tunnel. GridFTP memory-to-memory shows more consistent results: with 889.12 Mbps using one and two TCP streams and 873.04 Mbps using four streams. In contrast, GridFTP disk-to-disk obtains a slight improvement after using four TCP streams, achieving 733.44 Mbps of throughput compared with 632 Mbps and 660.8 Mbps obtained with one and two parallel streams, respectively.

6.6. 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 3. The main goal of the orchestrator in this scenario is to split a bandwidth reservation into 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. Fig. 10(c) shows that for two and four parallel TCP streams, the throughput of a data transfer stays 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 Fig. 10(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 poor overall performance. For eight streams per tunnel, the throughput results are not optimal since many TCP streams are competing for the same resources. These results are important because the orchestrator has to return meaningful recommendations to the end users in order for their data transfers to run optimally. For instance, given that two streams per tunnel provide optimal performance, our orchestrator should recommend using four parallel TCP streams, 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.

6.7. 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 5.3): synchronized VLANs, unsynchronized VLANs, and coordinated VLAN. We also consider the same bandwidth-splitting strategies described in Table 3. Fig. 11 shows the throughput measurement results for this experiment.

Figs. 11(a)–11(c) show the results for 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 is 3.47%. In the best-case 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 that 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, the *eight streams per tunnel* strategy provides nonoptimal results, for the reasons already explained in Section 6.6 and should not be considered for production environments.

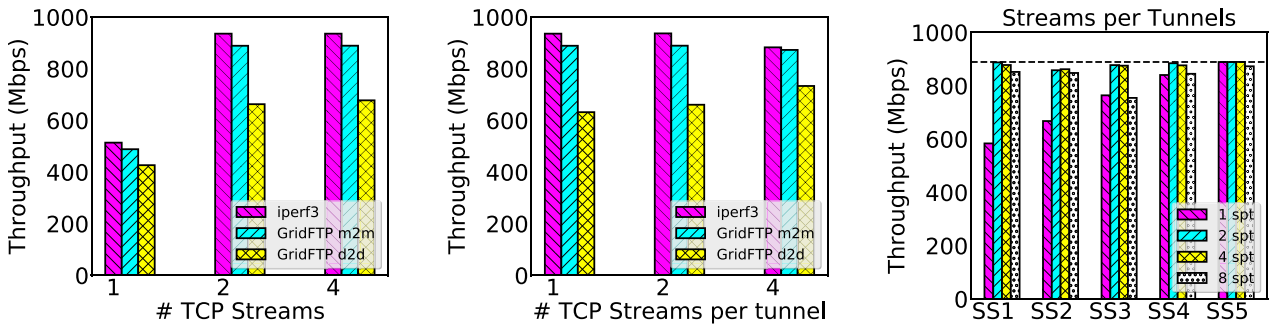
OpenFlow rule provisioning is known to add an extra delay to the transmission of the first packet of a flow. Nevertheless, we do not observe significant overhead on throughput because OpenFlow's delay is on the order of milliseconds and the total transmission time for this experiment is five minutes. 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 using the coordinated VLANs approach. As mentioned in Section 5.3, this approach 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 to the packets arriving at its interface. In the unsynchronized approach, however, or if 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 multipath, multidomain advance reservations.

6.8. Oversubscription

We also 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. We therefore 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 600 Mbps for an aggregate of 1.2 Gbps. Fig. 12 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.

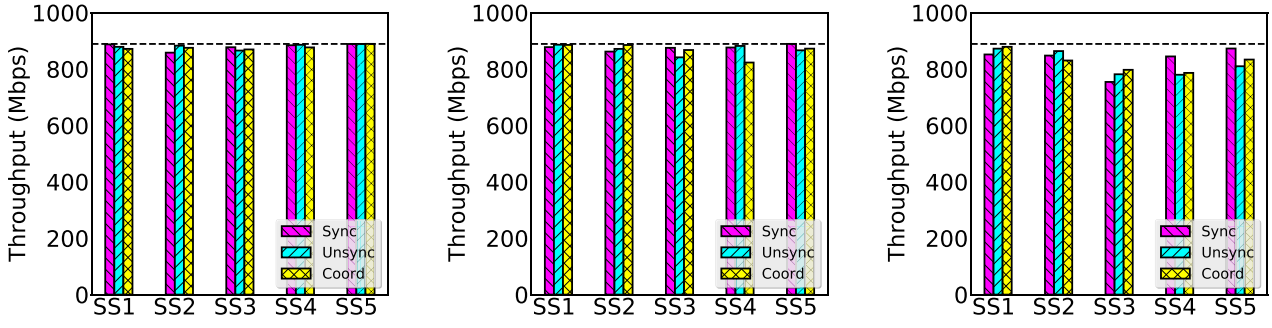
7. Related work

Multidomain SDN architectures. Avallone et al. [37] proposed an architecture for network resource management in multidomain scenarios using service-level specifications. Kempf et al. [38] proposed service provider SDN (SP-SDN), an approach to rapid and flexible cross-domain service creation that complements SDN and network function virtualization. The ONF, the Metro Ethernet Forum, and the IEFT have proposed similar architectures for service providers [39–41]. In the context of science networks, Zurawski



(a) Baseline 1 tunnel @ 1 Gbps (b) Baseline 2 tunnels @ 500 Mbps (c) Num. streams

Fig. 10. 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) baseline for a single L2 tunnel of 1 Gbps, (b) baseline for two L2 tunnels of 500 Mbps each, and (c) 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.



(a) Two TCP streams per tunnel (b) Four TCP streams per tunnel (c) Eight TCP streams per tunnel

Fig. 11. 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), and (c) show the results for two, four, and eight TCP streams per tunnel, respectively. We observe that two streams per tunnel is the recommended setting in order to achieve optimal performance. The baseline for each scenario is represented as a horizontal dashed line.

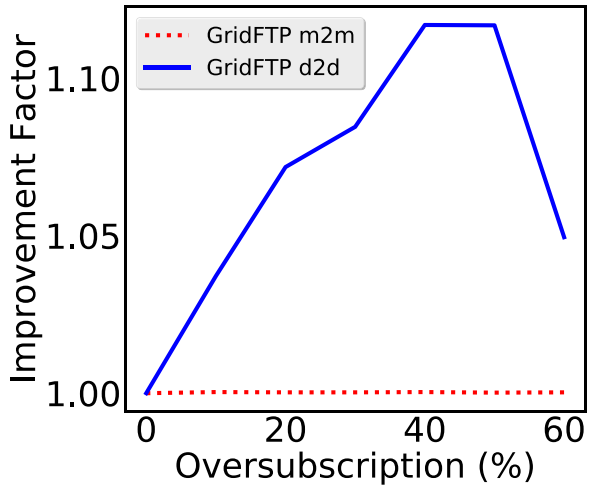


Fig. 12. Improvement factor in GridFTP's average throughput for oversubscribing the physical links, while maintaining multipath, multidomain reservations within limits. For instance, requesting two 600 Mbps L2 tunnels for an aggregate of 1.2 Gbps gives us 20% oversubscription on a 1 Gbps link.

et al. [42] proposed DYNES (Dynamic Network System), a system that uses OESS and OSCARS in multiple domains to establish dedicated layer 2 circuits. Similarly, Monga et al. [43] proposed SENSE (SDN for End-to-end Networked Science at the Exascale), a new paradigm for application to network interactions. Our architecture builds on concepts proposed by [37] and [38] and draw parallels with the architectures proposed in [42] and [43]. We adapt these concepts to the special needs of science networks and SDXs for multipath, multidomain advance reservations.

Flexible reservations. Balman et al. [9] 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 and Hu [11] proposed a two-dimensional relaxed reservation policy for Grid computing systems that achieves higher resource utilization and lower rejection rates. He et al. [44] proposed a flexible advance reservation model for cross-domain lightpath reservations in optical networks. These flexible advance reservation frameworks work on a single administrative domain, while our architecture focuses on multiple administrative domains.

Network resource negotiation. RNAP [45] and SNAP [46] are two examples of negotiation protocols for networking and Grid computing resources, respectively. Both protocols are based on querying the resource provider about the availability of a resource before making a reservation. Venugopal et al. [47] proposed a negotiation mechanisms using an alternative offers protocol for advance reservation of compute nodes in a Grid system. To create our negotiation protocol, we build on the concepts of querying for resources and providing offers.

TCP stripping. 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. [48] 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 striping on overlay networks. Recently, Multipath TCP (MPTCP) [49] 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 [50]) take advantage of these ideas to implement their own TCP multistreaming. Our orchestration framework uses SDXs to provision the underlying network infrastructure that allows TCP striping protocols in order to achieve their full potential.

Multipath advance reservations. OLiMPS (OpenFlow Link-layer MultiPath Switching) [51] 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. Plante et al. [52] 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 focuses on a single piece of the overall problem. While OLiMPS cares about provisioning OpenFlow rules, the work of Plante et al. is more concerned with the scheduling aspect of the problem. Furthermore, their work assumes identical bandwidth demands for every parallel virtual computer. We provide bandwidth splitting, which makes more efficient use of network resources; and our multidomain architecture is easily adaptable to single-domain scenarios.

8. Conclusions

In this paper we described the design and implementation of an architecture for end-to-end service orchestration in multidomain science networks that leverages SDXs for providing multipath, multidomain advance reservations. 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 multidomain advance reservations from approximately 50% when using single-path circuits to almost 99% when four paths are available. 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 and number of GridFTP streams, and we generated recommendations for the optimal performance of our system. Although all our tests were conducted at 1 Gbps, we expect our results to hold at higher speeds (10, 40, 100 Gbps) because the abstractions provided are not hardware dependent.

In future work, we will deploy our orchestrator in a large-scale testbed (e.g., AtlanticWave-SDX, ESnet 100GB SDN testbed, or GENI) and evaluate larger bandwidths, as well as the effects of path latency on our bandwidth-splitting service. We will also implement and evaluate novel science network services such as scheduled path migrations that are transparent to the data transfer applications and multipoint-to-multipoint advance reservations.

Acknowledgments

We thank Sean Donovan for his help in setting up the AtlanticWave-SDX testbed. This material was based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357, and National Science Foundation awards 1451024 and 1440761.

References

[1] N. Charbonneau, V.M. Vokkarane, C. Guok, I. Monga, Advance reservation frameworks in hybrid IP-WDM networks, IEEE Commun. Mag. 49 (5) (2011) 132–139.
[2] Internet2, Layer 2 services, http://www.internet2.edu/products-services/advanced-networking/layer-2-services/. (Accessed: 2017-07-25).
[3] GlobalNOC, OESS: Open exchange software suite, https://docs.globalnoc.iu.edu/sdn/oess.html. (Accessed: 2017-10-18).
[4] I. Monga, C. Guok, W.E. Johnston, B. Tierney, Hybrid networks: Lessons learned and future challenges based on ESnet4 experience, IEEE Commun. Mag. 49 (5) (2011) 114–121, http://dx.doi.org/10.1109/MCOM.2011.5762807.
[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: 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM, 2015, pp. 942–947, http://dx.doi.org/10.1109/INM.2015.7140415.
[6] J. Chung, E.-S. Jung, R. Kettimuthu, N.S. Rao, I.T. Foster, R. Clark, H. Owen, Advance reservation access control using software-defined networking and tokens, Future Gener. Comput. Syst. 79 (Part 1) (2018) 225–234.
[7] 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: Proceedings of the Fifth International Workshop on Network-Aware Data Management, NDM '15, ACM, New York, NY, 2015, pp. 2:1–2:8, http://dx.doi.org/10.1145/2832099.2832101, URL http://doi.acm.org/10.1145/2832099.2832101.
[8] J. Chung, R. Clark, H. Owen, SDX architectures: A qualitative analysis, in: IEEE SoutheastCon 2016, IEEE, 2016, pp. 1–8.
[9] M. Balman, E. Chaniotakisy, A. Shoshani, A. Sim, A flexible reservation algorithm for advance network provisioning, in: 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, 2010, pp. 1–11, http://dx.doi.org/10.1109/SC.2010.4.
[10] S. Venugopal, X. Chu, R. Buyya, A negotiation mechanism for advance resource reservations using the alternate offers protocol, in: 2008 16th International Workshop on Quality of Service, 2008, pp. 40–49, http://dx.doi.org/10.1109/IWQOS.2008.10.
[11] P. Xiao, Z. Hu, Two-dimension relaxed reservation policy for independent tasks in grid computing, J. Softw. 6 (8) (2011) 1395–1402.
[12] ESnet - network maps, https://www.es.net/engineering-services/the-network/network-maps/. (Accessed: 2017-08-28).
[13] Internet2 international networks, https://noc.net.internet2.edu/i2network/live-network-status/maps-graphs/internet2-international-network.html. (Accessed: 2017-08-28).
[14] D. Kreutz, F.M.V. Ramos, P.E. Verssimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103 (1) (2015) 14–76, http://dx.doi.org/10.1109/JPROC.2014.2371999.
[15] Report of the NSF workshop on software defined infrastructures and software defined exchanges, Washington, DC, 2016.
[16] 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: Proceedings of the 2014 ACM Conference on SIGCOMM, ACM, 2014, pp. 551–562.
[17] J. Chung, J. Cox, J. Ibarra, J. Bezerra, H. Morgan, R. Clark, H. Owen, AtlanticWave-SDX: An international SDX to support science data applications, in: Software Defined Networking (SDN) for Scientific Networking Workshop, SC'15, 2015, pp. 1–7.
[18] J. Stringer, D. Pemberton, Q. Fu, C. Lorier, R. Nelson, J. Bailey, C.N. Correa, C. Esteve Rothenberg, et al., Cardigan: SDN distributed routing fabric going live at an internet exchange, in: 2014 IEEE Symposium on Computers and Communication, ISCC, IEEE, 2014, pp. 1–7.
[19] P. Lin, J. Bi, S. Wolff, Y. Wang, A. Xu, Z. Chen, H. Hu, Y. Lin, A west-east bridge based SDN inter-domain testbed, IEEE Commun. Mag. 53 (2) (2015) 190–197.
[20] Transferring data with BBCP, https://www.olcf.ornl.gov/kb_articles/transferring-data-with-bbcp/. (Accessed: 2017-09-04).
[21] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Tuecke, GridFTP: Protocol extensions to FTP for the grid, in: Global Grid Forum, GFD-RP, vol. 20, 2003, pp. 1–21.
[22] Globus - research data management system, https://www.globus.org/. (Accessed: 2017-01-20).
[23] E. Dart, et al., ESnet requirement review reports, https://www.es.net/science-engagement/science-requirements-reviews/requirements-review-reports/. (Accessed: 2017-01-14).
[24] Python networkX, https://networkx.github.io/. (Accessed: 2018-06-02).
[25] OpenStack blazar, https://wiki.openstack.org/wiki/Blazar. (Accessed: 2018-06-02).
[26] E. Dart, L. Rotman, B. Tierney, M. Hester, J. Zurawski, The Science DMZ: A network design pattern for data-intensive science, Sci. Program. 22 (2) (2014) 173–185.
[27] GRPC, https://grpc.io/. (Accessed: 2017-09-06).
[28] Protocol buffers, https://developers.google.com/protocol-buffers/. (Accessed: 2017-09-06).

[29] S. Donovan, J. Skandalakis, A. Lamba, AtlanticWave-SDX controller prototype, https://github.com/atlanticwave-sdx/atlanticwave-proto. (Accessed: 2017-08-11).

[30] Framework Community, Ryu SDN framework,, http://osrg.github.io/ryu. (Accessed: 2017-09-07).

[31] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: Enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74, http://dx.doi.org/10.1145/1355734.1355746, URL http://doi.acm.org/10.1145/1355734.1355746.

[32] 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, NSDI 15, USENIX Association, Oakland, CA, 2015, pp. 117–130, URL https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/pfaff.

[33] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, G. Parulkar, ONOS: Towards an open, distributed SDN oS, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN '14, ACM, New York, NY, USA, 2014, pp. 1–6, http://dx.doi.org/10.1145/2620728.2620744, URL http://doi.acm.org/10.1145/2620728.2620744.

[34] M. Berman, J.S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, I. Seskar, GENI: A federated testbed for innovative network experiments, Comput. Netw. 61 (2014) 5–23, http://dx.doi.org/10.1016/j.bjp.2013.12.037, URL http://www.sciencedirect.com/science/article/pii/S1389128613004507, (special issue on Future Internet Testbeds Part I).

[35] AtlanticWave-SDX controller prototype, https://github.com/atlanticwave-sdx/atlanticwave-proto. (Accessed: 2017-09-07).

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

[37] S. Avallone, S. D'Antonio, M. Esposito, S.P. Romano, G. Ventre, Resource allocation in multi-domain networks based on service level specifications, J. Commun. Netw. 8 (1) (2006) 106–115, http://dx.doi.org/10.1109/JCN.2006.6182910.

[38] J. Kempf, M. Korling, S. Baucke, S. Touati, V. McClelland, I. Mas, O. Backman, Fostering rapid, cross-domain service innovation in operator networks through service provider SDN, in: Communications, ICC, 2014 IEEE International Conference on, IEEE, 2014, pp. 3064–3069.

[39] C. Janz, L. Ong, K. Sethuraman, V. Shukla, Emerging transport SDN architecture and use cases, IEEE Commun. Mag. 54 (10) (2016) 116–121, http://dx.doi.org/10.1109/MCOM.2016.7588279.

[40] MEF, et al., An industry initiative for third generation network and services, https://www.mef.net/third-network/third-network-white-paper. (Accessed: 2017-01-15).

[41] D. Dhody, X. Zhang, O.G. de Dios, D. Ceccarelli, B. Yoon, 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).

[42] 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, J. Phys. Conf. Ser. 396 (4) (2012) 042065, URL http://stacks.iop.org/1742-6596/396/i=4/a=042065.

[43] I. Monga, C. Guok, J. MacAuley, A. Sim, H. Newman, J. Balcas, P. Demar, L. Winkler, T. Lehman, X. Yang, SDN for end-to-end networked science at the exascale (SENSE), in: The 5th Innovating the Network for Data-Intensive Science, INDIS Workshop, Dallas, TX, USA, 2018.

[44] E. He, X. Wang, J. Leigh, A flexible advance reservation model for multi-domain WDM optical networks, in: 2006 3rd International Conference on Broadband Communications, Networks and Systems, 2006, pp. 1–10, http://dx.doi.org/10.1109/BROADNETS.2006.4374310.

[45] X. Wang, H. Schulzrinne, RNAP: A resource negotiation and pricing protocol, in: International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV99, Basking, Citeseer, 1999.

[46] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, S. Tuecke, SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 153–183, http://dx.doi.org/10.1007/3-540-36180-4_9.

[47] S. Venugopal, X. Chu, R. Buyya, A negotiation mechanism for advance resource reservations using the alternate offers protocol, in: 2008 16th Interntional Workshop on Quality of Service, 2008, pp. 40–49, http://dx.doi.org/10.1109/IWQOS.2008.10.

[48] H.Y. Hsieh, R. Sivakumar, PTCP: An end-to-end transport layer protocol for striped connections, in: 10th IEEE International Conference on Network Protocols, 2002. Proceedings, 2002, pp. 24–33, http:dx.doi.org/10.1109/ICNP.2002.1181383.

[49] O. Bonaventure, M. Handley, C. Raiciu, An overview of multipath TCP, ; login: 37 (5) (2012) 17.

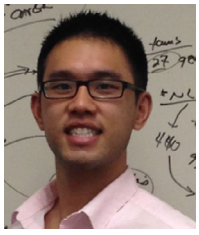
[50] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster, The globus striped gridFTP framework and server, in: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, SC '05, IEEE Computer Society, Washington, DC, USA, 2005, p. 54, http://dx.doi.org/10.1109/SC.2005.72, URL https://doi.org/10.1109/SC.2005.72.

[51] H.B. Newman, A. Barczyk, M. Bredel, OLiMPS. Openflow Link-Layer Multipath Switching, Tech. rep., California Institute of Technology, Pasadena, CA (United States), 2014.

[52] J.M. Plante, D.A.P. Davis, V.M. Vokkarane, Parallel and survivable multipath circuit provisioning in ESnet's OSCARS, Photonic Netw. Commun. 30 (3) (2015) 363–375, http://dx.doi.org/10.1007/s11107-015-0535-x.

Joaquin Chung is a postdoctoral appointee at the Mathematics and Computer Science Division at Argonne National Laboratory. He 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 under the supervision of Dr. Henry Owen and Dr. Russ Clark at Georgia Institute of Technology, Atlanta, USA in December 2017. He is a Fulbright alumnus and an IEEE member. His research interests include software-defined networking, software-defined exchanges, network function virtualization, edge computing, network security, and the Internet of Things.

Raj Kettimuthu is a Computer Scientist in the Mathematics and Computer Science Division at Argonne National Laboratory, and a Senior Fellow in the Computation Institute at The University of Chicago and Argonne National Laboratory. He received the B.E. degree from Anna University, Chennai, India, and an M.S. and Ph.D. from the Ohio State University, all in Computer Science and Engineering. His research is focussed on high-speed transfer of large-scale data, software defined networking, rapid execution of data-intensive science workflows, parallel job scheduling and large-scale data analysis. He has co-authored more than 80 articles in the above-mentioned areas. He is a recipient of R&D 100 award. He is a senior member of both IEEE and ACM.



Nam Pho is the Director for Research Computing at the University of Washington. He has a research interest in biomedical informatics, high-performance computing, machine learning, and computer networks. He is a graduate of Johns Hopkins University and the University of Maryland.

Russ Clark is a senior research scientist in Georgia Tech's School of Computer Science, and the co-director of the Georgia Tech Research Network Operations Center (GTRNOC), which supports research efforts across campus. Dr. Clark received the B.S. in Mathematics and Computer Science from Vanderbilt University in 1987. He received the M.S. and Ph.D. degrees in Information and Computer Science from Georgia Institute of Technology in 1992 and 1995. For the years 1997–2000 he was a Senior Scientist with Empire Technologies, a network management software company. His research interests include Internet infrastructure and operating systems, mobile and wireless communications, network security, software-defined networking, and software-defined exchanges.

Henry L. Owen received the B.E.E., M.S.E.E., and Ph.D. degrees from the Georgia Institute of Technology in 1980, 1983, and 1989 respectively. During 1991 he was on a leave of absence and he worked for the telecommunications firm ALCATEL-SEL in Stuttgart, Germany. Between 1992 and 1995, he spent summers performing research for ALCATEL-SEL on site in Stuttgart. At Georgia Tech Dr. Owen has implemented the Internet Programming and Internet Design laboratories and classes that provide networking equipment as well as an environment for hands on laboratory experimentation and project implementation. His research interests are internetworking, computer networks, quality of service in the Internet, network protocol implementations in operating systems, and software-defined networking.