Orchestrating Intercontinental Advance Reservations with Software-Defined Exchanges

Joaquin Chung[a,*], Rajkumar Kettimuthu[c], Nam Pho[b], 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]*Math and Computer Science Division, Argonne National Laboratory, USA*

Abstract

To interconnect research facilities across wide geographic areas, network operators deploy science networks, also referred to as Research and Education (R&E) networks. These networks allow experimenters to establish dedicated circuits between research facilities for transferring large amounts of data, by using advanced reservation systems. Intercontinental dedicated circuits typically require coordination between multiple administrative domains, which need to reach an agreement on a suitable advance reservation. 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 multi-domain advance reservations, we propose an architecture for end-to-end service orchestration in multi-domain science networks that leverages software-defined exchanges (SDX) for providing multi-path, multi-domain advance reservations. We have implemented an orchestrator for multi-path, multi-domain advance reservations and an SDX to support these services. Our orchestration architecture enables multi-path, multi-domain advance reservations and improves the reservation success rate from 50% in single path systems to 99% when four path are available.

Keywords: Multi-domain advance reservation, Orchestrator, software-defined networking, software-defined exchanges, bandwidth splitting

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].

As advance reservations are defined by endpoints, duration, and bandwidth, the scheduling of resources is not flexible; that is, a reservation request will fail if the exact amount of bandwidth between two endpoints is not available within the specified time frame. 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 as a multi-domain 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 multi-legged flight with independent airlines from different consortiums that do not share travel schedules.

Another challenge is that advance reservations terminate at the WAN border router of each domain, and participant domains are interconnected at single junction points [5]. As a result, multi-domain 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 LAN traffic to reach the advanced 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.

*Corresponding author
Email addresses: `joaquin.chung@gatech.edu` (Joaquin Chung), `kettimut@anl.gov` (Rajkumar Kettimuthu), `nampho@gatech.edu` (Nam Pho), `russ.clark@gatech.edu` (Russ Clark), `henry.owen@ece.gatech.edu` (Henry Owen)

Recently, software-defined exchanges (SDX) have emerged as a new kind of cyberinfrastructure that allows independent administrative domains to share computing, storage, and networking resources by leveraging SDN [6]. We posit that by inserting an SDX in the junction point between participating domains in an intercontinental advance reservation, we will increase the success rate of finding a multi-domain advance reservation. The initial benefit of adding SDXs to the advance reservation process is overcoming the limitation of single-path advance reservation (i.e., SDXs enable multi-domain, multi-path advance reservations). For instance, we may have two SDXs connected through two different advance reservation providers, providing two independent paths between two end sites. 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.

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 multi-domain science networks. We demonstrate that by introducing SDXs in the provisioning process, we are able to create multi-path, multi-domain advance reservations that increases performance and efficiency over traditional methods. The contributions of this paper are:

1. An architecture for multi-domain, multi-path advance reservations in science networks that leverages SDN and SDX.
2. A negotiation protocol for multi-domain, multi-path advance reservations that increases the reservation success rate from 50% on a single-path system to 99% on our multi-path system.
3. Architectural approaches at the SDX level that enable novel science network services, while enhancing the performance of science data transfers over traditional approaches.

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

2. Motivation

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 if resources are available to fulfill a request. Current implementations try to find an exact match for constraints provided in the request, and they fail if a suitable advance reservation is not found. Researchers have proposed scheduling algorithms for flexible advance reservations that increase the success rate of a reservation request in single domain scenarios [7, 8, 9]. However, for advance reservations in which the circuit spans multiple domains and follows a single path, flexible/malleable techniques lose their benefits because participant domains have to agree on the rigid constraints of the original request to compose the end-to-end service. Fortunately, we find path diversity on intercontinental links originated from the United States, as shown in Figure 1. The topology maps of ESNet [10] and Internet2 [11] report at least three links to Asia Pacific, three links to Latin America, and four links to Europe.

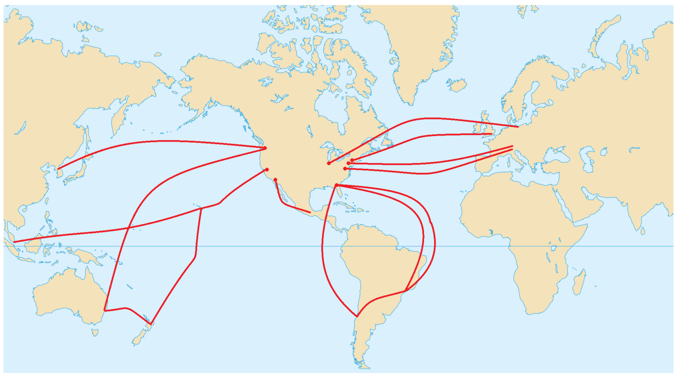


Figure 1: Intercontinental R&E links originated from the United States.

2.2. Software-defined Exchange (SDX)

A software-defined exchange (SDX) is a meet-me point or marketplace where independent administrative domains can exchange computing, storage, and networking resources [12]. SDXs are an architectural innovation that will enable multi-path, multi-domain advance reservations. SDXs will also enable novel science network services such as multi-path bandwidth splitting across independent WAN providers, scheduled path migrations that are transparent to data transfer applications, and multipoint-to-multipoint advance reservations. Since SDX is a nascent technology, we need to know the advantages and disadvantages of using SDX as an interconnection point for multi-path, multi-domain advance reservations. For instance, it is well known that when using hashing for load balancing, all traffic corresponding to the same hash will be sent to the same interface. Nevertheless, we can take advantage of data transfer protocols (e.g., BBCP [13] and GridFTP [14]) that create multiple TCP streams, and distribute these streams over a multi-path, multi-domain advance reservation.

3. Architecture Overview

To support multi-path, multi-domain advance reservations we require an architecture that takes advantage of the enriched connectivity provided by SDX to compose functional multi-path, multi-domain advance reservations while improving the success rate of user's requests and the performance of science data transfers. Our proposed architecture is composed of the following components (see Figure 2):

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.

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 (i.e., 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, a site, WAN, or SDX controller may be any type of existing SDN controller, advanced reservation system, or SDX controller. The main requirement is that the northbound interface of these controllers should abstract the details of the network infrastructure and expose relevant science network services. More details about this type of interface is provided in Section 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, in order to connect site A to site B in Figure 2, the orchestrator needs to know if all domains in between can provide this connectivity. To successfully compose end-to-end services, an orchestrator requires resource management, scheduling, and path computation functionalities. Our orchestrator maintains a minimal set of tables or "databases": a table of participant domains and the services they provide, and a global topology view. In order to be practical in multi-domain environments, the orchestrator has to interact with the network resource managers at each domain to query status and reserve resources.

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

3.3. Interfaces and Services

Users in our system are domain-expert scientists whom in most of the cases do not have expertise in network operations, but still need to request reservations to expedite their data transfers. Additionally, scientists use data workflow management systems (e.g., Globus [15]) to automate the process of moving and sharing data across research facilities. In our reference architecture, both scientists and applications request end-to-end science network services to the orchestrator by using interfaces that abstract network infrastructure details in our reference architecture. The following subsections provide more details about the interfaces that allow communication between site, WAN, or SDX controllers and an orchestrator, and between users or applications and orchestrators.

3.3.1. Domain to Orchestrator (D-O) Interface

The domain to orchestrator interface, depicted as *D-O interface* in Figure 2 allows a science network orchestrator to consume services from a site, WAN, or SDX controller. To understand the services that should be exposed by a site controller, we studied the Energy Science Network (ESNet) requirement review reports from 2013 to 2015 [16], and synthesized the most common scientific data transfers as follows: bulk data transfer, real-time data transfer, and management network traffic.

3.3.2. User/Application to Orchestrator (U-O) Interface

The user/application to orchestrator interface, depicted as *U-O interface* in Figure 2 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 an optimal solution to a user request, given the user constraints and the network state. Although, the *U-O interface* is an important component of the overall architecture, we will focus on the *D-O interface* and the components pertaining the network infrastructure for this study.

4. Design

In this section we present the design challenges for a system that provides multi-domain, multi-path advance

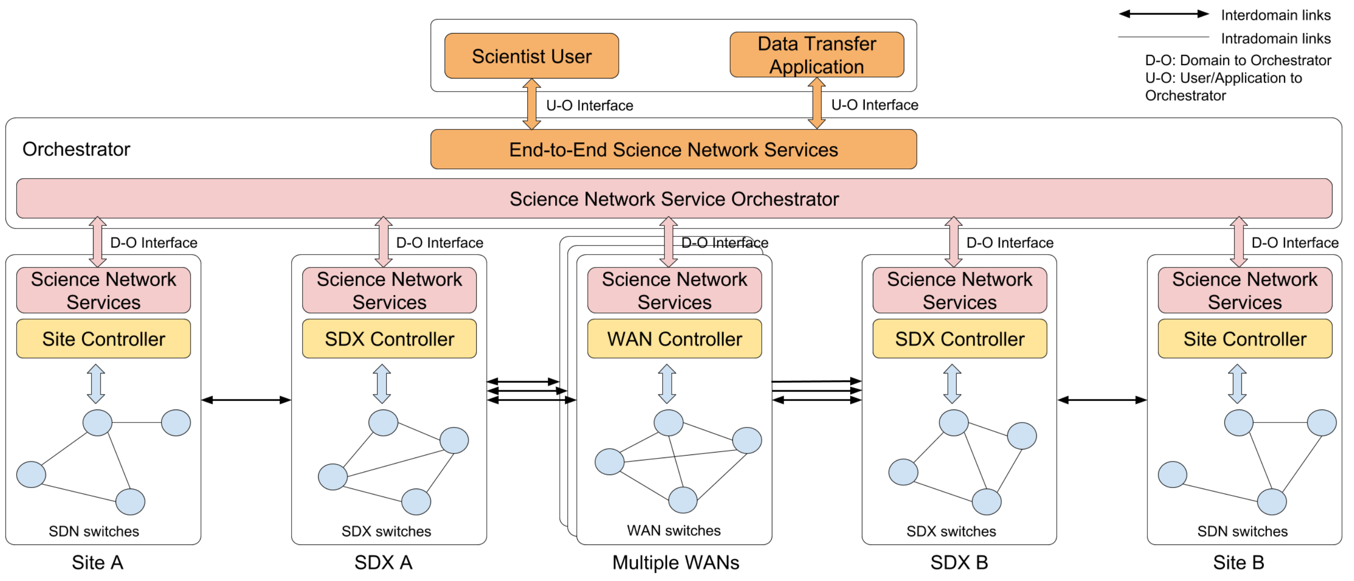


Figure 2: Reference architecture for end-to-end service orchestration in multi-domain science networks. Several independent administrative domains are connected by inter-domain links, and expose science network services to a centralized orchestrator through the domain to orchestrator (D-O) interface. The orchestrator then composes end-to-end science network services and exposes them to domain-expert scientists and data transfer applications through the user to orchestrator (U-O) interface.

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. Regarding additional components of the orchestrator such as path computation and resource management, we take advantage of readily available implementations of these standard components.

4.1. General Workflow

This section describes the general workflow for requesting and composing multi-path, multi-domain advance reservations. We assume that multiple paths exist between two research facilities, and these paths traverse multiple administrative domains that provide connectivity and guaranteed bandwidth by using advanced reservation systems. We also assume that SDXs serve as interconnection points for these administrative domains, enabling richer connectivity. An orchestrator (see subsection 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). This should not be confused with open topology sharing, which is already supported by OSCARS and OESS [5].

Figure 3 depicts the general worflow for requesting multi-domain, multipath advance reservations. The workflow starts with a user requesting a flexible, multi-domain advance reservation to the orchestrator, which performs path computation to determine the domains and SDXs on the path. Then, the orchestrator decomposes the user's

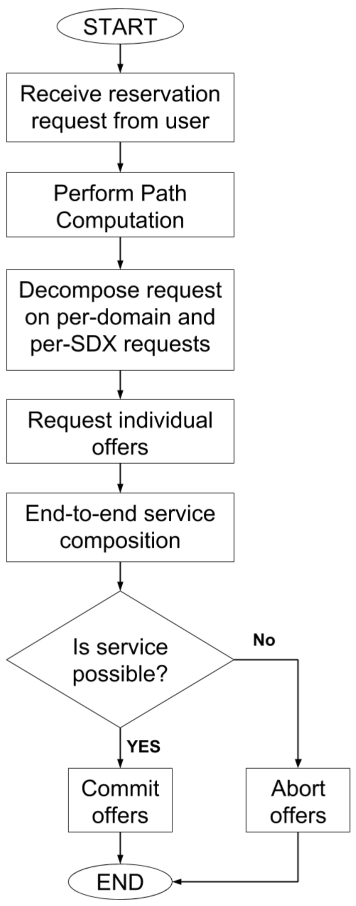

START
Receive reservation request from user
Perform Path Computation
Decompose request on per-domain and per-SDX requests
Request individual offers
End-to-end service composition
Is service possible?
No
YES
Commit offers
Abort offers
END

Figure 3: General workflow for requesting multi-domain, multipath advance reservations.

request into individual requests for each domain and SDX on the path, and it requests reservation offers from each participant domain. Finally, the orchestrator uses these offers to compose an end-to-end service, commit offers and contact SDXs to make interconnections if an end-to-end service is possible, and aborts unused offers. Otherwise, the orchestrator aborts all offers.

4.2. Negotiation Protocol

In this section we take a deeper look on the negotiation protocol that allows the orchestrator to compose multi-path, multi-domain advance reservations. The negotiation protocol is divided in two phases: phase 1 requests offers from participant domains and composes an end-to-end service, and phase 2 commits the successful offers, aborts unused offers, and request interconnection at SDXs. It is important to note that not all participants are willing to provide reservation offers, either because they have legacy systems, or because they have privacy concerns. We identify those domains that provide reservation offers as visible domains, and those that do not provide offers as blind domains. Visible domains are considered as the initial option to compose the end-to-end service. Blind domains are only considered if visible domains do not have enough resources. The rationale behind this strategy is that by considering blind domains for remaining resources, we increase the chances of success because it is easier to allocate smaller amounts of bandwidth. Our negotiation protocol is composed of seven types of messages: *Reservation*, *ReqOffers*, *SendOffers*, *ReservationPrep*, *Commit*, *Abort*, and *ReservationResp*, that we describe in Table 1.

Figure 4 shows the detailed negotiation protocol considering N participant domains, with M visible domains and $N - M$ blind domains. We consider three scenarios:

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

2. **Full visibility ($M = N$):** All participant domains are visible domains (i.e., only systems that provide bandwidth offers participate in the orchestration process).

3. **Partial visibility ($M \neq N$):** blind domains and visible domains participate in the orchestration process (i.e., a mix of traditional 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 visible and blind domains participating in the process. The orchestrator sends *ReqOffers* messages to the M visible domains. These domains respond with *SendOffers* messages to the orchestrator, which uses these offers to compose an end-to-end service. Each *SendOffers* message contains a token ID [17] 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 to the blind domains and tries to compose a new end-to-end service. If the service composition succeeds, the orchestrator transitions to phase 2, otherwise the reservation request fails.

4.3. SDX Rules

As mentioned in subsection 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 VLANs at each interconnection point. As a result, an SDX allows rules that bridge a VLAN in an inbound port to another VLAN in an outbound port, split traffic among several outbound ports, and create the corresponding mirror policies for bidirectional traffic.

Figure 5 illustrate the bandwidth splitting service block diagram. Our architecture takes advantage of the multi-streaming nature of data transfer protocols (e.g., BBCP and GridFTP). We propose that an SDN switch and an SDN controller create flow rules that assign a new VLAN ID to every new TCP flow. Ideally, these switches and SDN controllers will be provisioned on demand for each new multi-path, multi-domain 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 Figure 5 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. To ensure that all the traffic belonging to a single flow traverses the same circuit, a synchronization or coordination between the SDN controllers assigning the VLANs might exist. Otherwise, we might have the forward traffic of a TCP flow travers-

Table 1: Negotiation Protocol Messages

Message Type	**Description**
Reservation	Message from the user to the orchestrator requesting a multi-domain advance reservation
ReqOffers	Message from the orchestrator to visible domains requesting advance reservation offers
SendOffers	Message from visible 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

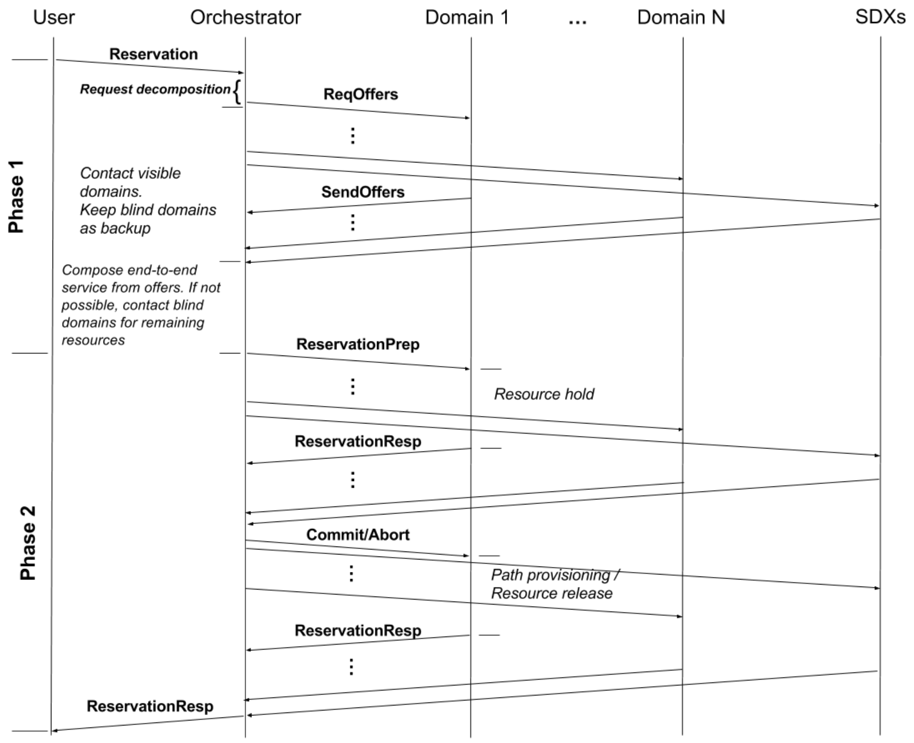

User
Orchestrator
Domain 1
...
Domain N
SDXs
Reservation
Request decomposition
ReqOffers
Phase 1
Contact visible domains. Keep blind domains as backup
SendOffers
Compose end-to-end service from offers. If not possible, contact blind domains for remaining resources
ReservationPrep
Resource hold
ReservationResp
Phase 2
Commit/Abort
Path provisioning / Resource release
ReservationResp
ReservationResp

Figure 4: Negotiation protocol for multi-path, multi-domain advance reservation with M visible domains and $N - M$ blind domains.

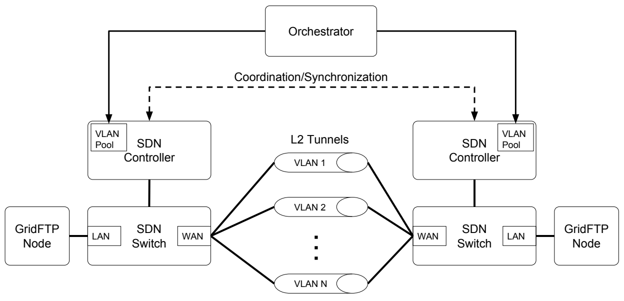


Figure 5: Block diagram of bandwidth splitting service components for SDX rule provisioning.

ing one tunnel, and all the ACKs returning over another tunnel.

5. Implementation

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

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, process those requests internally, and send 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 using the general remote procedure call (gRPC) protocol [18], a high-performance RPC framework optimized for distributed computing and mobile environments.

5.2. Negotiation Protocol Implementation

First, we assume 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 subsection 4.2 (i.e., no visibility, full visibility, and partial visibility). 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. However, it is more suitable for the *no visibility* scenario, because it does not require the ability to request offers. In this approach the orchestrator divides the original bandwidth request in equal parts among the participant domains.

2. **Partial Offers:** This approach is mainly applicable to the *partial visibility* scenario; the orchestrator contacts the visible domains for bandwidth offers. If the orchestrator is able to compose an end-to-end service with these offers only, the orchestrator proceeds with Phase 2 of our negotiation protocol (i.e., provisioning). Otherwise, the orchestrator tries to request the remaining bandwidth from blind domains.

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

5.3. SDX Implementation

Our SDX implementation is based on AtlanticWave/SDX [19], an SDX controller written in Python that uses the Ryu SDN Framework [20] as an OpenFlow [21] speaker, and has a REST API and Web application for management. Currently, AtlanticWave/SDX supports advance reservation of L2 tunnels using the Web interface or the REST API. We added the bandwidth query functionality through a REST API in AtlanticWave/SDX to support our negotiation protocol. We verified that OSCARS supports a similar functionality through their Web interface, but it does not 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.

6. Evaluation

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

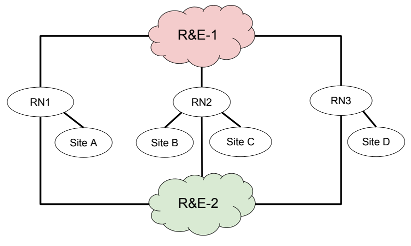
6.1. Multi-path, Multi-domain Advance Reservations

To evaluate our multi-path, multi-domain advance reservation we consider the topology depicted in Figure 6a. 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&E (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 generate a random request composed of a time window, a required bandwidth, a source, and a destination. We send this request to both domains individually, and evaluate whether the domains have enough available resources. For our multi-path, multi-domain advance reservation service we evaluate whether the sum of the available bandwidth in both domains satisfies the request. Figure 6b shows that our multi-path, multi-domain approach has an 85% success rate when two independent paths are available, compared to approximately 50% success rate for the state-of-the-art (single path) approach.

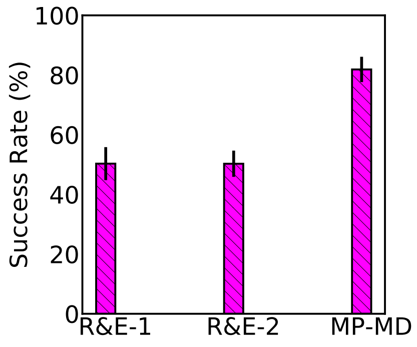
6.2. Negotiation Protocol Success Rate

To evaluate the success rate of the three variants of our negotiation protocol (i.e., 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 to compose a multi-path, multi-domain advance reservation. We chose four domains, because this is a reasonable number of multiple intercontinental paths between two sites as mentioned in section 2. For each participant domain, we generated a bandwidth schedule of 1000 entries that provide the available bandwidth at a given time point. A user generates 100 random bandwidth requests within the time window defined by the aforementioned 1000 entries. We ran the simulation 32 times and took the averages for each scenario.

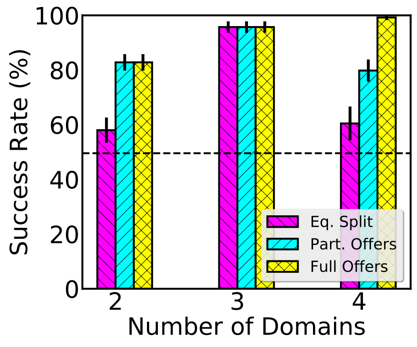
Figure 6c shows the results of our simulations. The horizontal line represents the success rate for a single domain, which is 49.56% under our assumptions. Any of our strategies outperform the baseline. In the worst case scenario (i.e., equal splitting under no visibility), the success rate of our orchestrator is approximately 58%. Under the best conditions (four visible domains), our orchestrator achieves approximately 99% success rate, which translate to a $2X$ improvement. The optimal solution is found when three multiple paths are available, because all three negotiation strategies achieve greater than 95% success rate.



(a) Topology



(b) State-of-the-art



(c) Negotiation protocol

Figure 6: Simulation topology and results: (a) topology for multi-path, multi-domain advance reservation evaluation simulation; (b) success rate for multi-path, multi-domain advance reservation evaluation compared to the state-of-the-art methods; and (c) Negotiation protocol success rate for three bandwidth splitting strategies and up to four participant domains.

6.3. SDX Experimental Setup

Figure 7 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 [19] (SDX1, SDX2, SDX3, and SDX4) running on a Docker container inside a Dell PowerEdge R220 server. This server also hosts our orchestration system: four instances of our orchestration agents (agent1, agent2, agent3, and agent4), and one orchestrator. Each orchestrator agent runs on a Docker container, and each one is paired with an SDX instance, while the orchestrator runs on another Docker container and communicates with the agents using gRPC. We used two customized Supermicro 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) [22] virtual switch, and a Ryu SDN controller [20]. 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 [23].

6.4. Data Transfer Methods

We measured the throughput baseline of a data transfer over a single-path, multi-domain advance reservation versus a data transfer over a multi-path, multi-domain advance reservation on our testbed using two data transfer methods: GridFTP memory-to-memory (m2m), and GridFTP disk-to-disk (d2d) data transfers. We used iperf3, a well-known bandwidth measuring tool as a reference. Figure 8 shows the results of performing the aforementioned data transfer over a 1 Gbps link with 90 ms RTT.

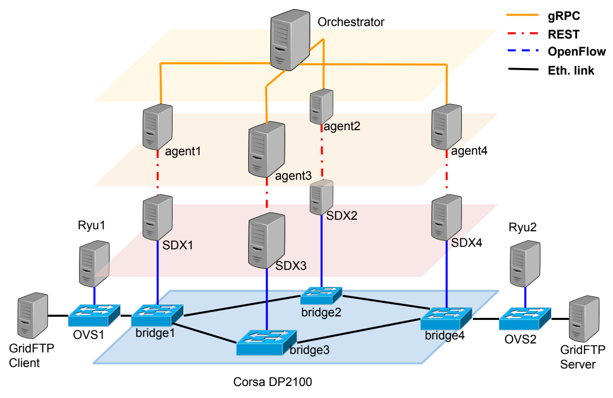


Figure 7: Experimental setup topology.

For iperf3 and GridFTP memory-to-memory, we sustained the data transfer for five minutes, or the equivalent of transferring a 37.5 GB of data at line-rate over a 1 Gbps link, while for GridFTP disk-to-disk we actually transferred a 20 GB file, which is a reasonable size for a scientific dataset [16]. We measured the maximum disk throughput of our GridFTP endpoints, and we obtained 92.34 MB/s or 738.72 Mbps on average.

Figure 8a shows that iperf3 only reaches 514 Mbps of throughput for a single L2 tunnel of 1 Gbps of bandwidth, while GridFTP only reaches 488.56 Mbps and 426.72 Mbps of throughput for memory-to-memory and disk-to-disk, respectively. The reason for this low performance is that our endpoints are optimized for parallel TCP streams. As we see for two and four parallel TCP streams, iperf3 utilized 93.6% of the link (936 Mbps of throughput on average), and GridFTP memory-to-memory used 88.92% (or 889.24 Mbps on average). However, GridFTP disk-to-disk is only able to use approximately 67% (670.36 Mbps on average) of the link with parallel streams.

Figure 8b shows the throughput baseline after splitting the bandwidth reservation among two 500 Mbps L2 tunnels. For one and two TCP streams per tunnel, iperf3 achieves 936 Mbps of throughput. However, it is only able

Table 2: Experimental setup, equipment specifications

Equipment	Specifications
Corsa DP2100	OpenFlow 1.5, multiple flow tables, multi-context 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 @ 3.10GHz processors, four port Gigabit Ethernet card
Customized Supermicro	Ubuntu Server 16.04, 8 GB RAM, four Intel(R) Xeon(R) CPU X3430 @ 2.40GHz, two Gigabit Ethernet interfaces

to achieve 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. To the contrary, GridFTP disk-to-disk obtains a slight improvement after using four TCP streams, achieving 733.44 Mbps of throughput compared to 632 Mbps and 660.8 Mbps obtained with one and two parallel streams, respectively.

6.5. 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 among two L2 tunnels, obtaining an aggregate bandwidth of 1 Gbps. For instance, one strategy is to split the bandwidth into two 500 Mbps tunnels. Another strategy is to split the request into one tunnel of 100 Mbps and another tunnel of 900 Mbps.

Table 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

Figure 8c shows that for two and four parallel TCP streams, the throughput of a data transfer stays very close to the no-splitting baseline of 889.24 Mbps. For one stream per tunnel, the throughput increases as the bandwidth splitting strategy is more balanced. This behavior can be explained from our observation in Figure 8a. The TCP stream using a tunnel with a larger bandwidth reservation cannot fill the pipe, because the endpoints are optimized for parallel streams. Meanwhile, the stream using the smaller reservation is limited, resulting in a poor overall performance. In the case of eight streams per tunnel, the throughput results are not optimal as many TCP streams are competing for the same resources. These results are important because the orchestrator has to return meaningful recommendations to the end user for their data transfers to run optimally. For instance, given that two streams per tunnel provides optimal performance, our orchestrator should recommend the end user to four parallel TCP on her application, because the reservation was split among two tunnels. In the case of splitting the bandwidth among three tunnels, the orchestrator's recommendation should be six parallel streams.

7. Related Work

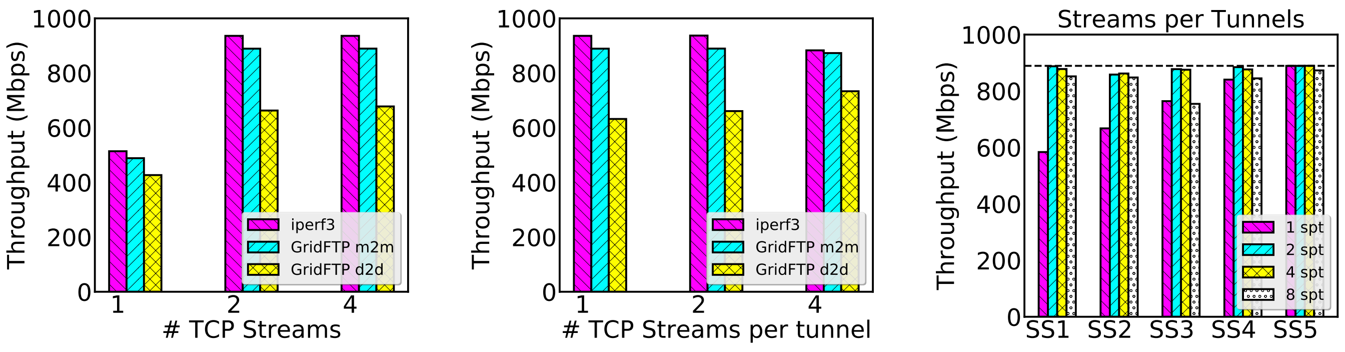
Multi-domain SDN Architectures - Avallone et al. [24] proposed an architecture for network resource management in multi-domain scenarios using service-level specifications, while Kempf et al. [25] proposed service provider SDN (SP-SDN), an approach to rapid and flexible cross-domain service creation that complements SDN and network function virtualization (NFV). Our architecture builds upon concepts proposed by [24] and [25], and adapts them to the special necessities of science networks and SDXs.

Network Resource Negotiation - RNAP [26] and SNAP [27] are two examples of negotiation protocols for networking and Grid computing resources, respectively. Both protocols are based on querying resource provider for the availability of a resources before making a reservation. Venugopal et al. [28] proposed a negotiation mechanisms using an alternate offers protocol for advance reservation of compute nodes in a Grid system. We build upon the concepts of querying for resources and providing offers to create our negotiation protocol.

Multi-path Advance Reservations - OLiMPS (OpenFlow Link-layer MultiPath Switching) [29] is an OpenFlow application that allows load balancing over multiple switched paths. Likewise, Plante et al. [30] proposed a multi-path extension to the OSCARS client that enables end users to reserve multiple paths, providing session survivability and increasing parallelism. Although similar to our work, both of these solutions are for single-domain reservations, each one focuses on a single piece of the overall problem. We provide bandwidth splitting, which makes more efficient use of network resources, and our multi-domain architecture is easily adaptable to single domain scenarios.

8. Conclusions

In this paper we presented an architecture for end-to-end service orchestration in multi-domain science networks that leverages SDXs for providing multi-path, multi-domain advance reservations. We implemented an orchestrator for multi-path, multi-domain advance reservations and an SDX to support these services. Our implementation uses an agent-based approach in which site agents



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

Figure 8: Throughput measurements while performing data transfers using iperf3, GridFTP memory-to-memory (m2m) and GridFTP disk-to-disk (d2d) over a 1 Gbps link with 90 ms RTT: (a) shows the baseline for a single L2 tunnel of 1 Gbps, (b) shows the baseline for two L2 tunnels of 500 Mbps each, and (c) represents the 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.

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 multi-domain 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 generated recommendations for the optimal performance of our system. In future work, we will deploy our orchestrator in a large scale testbed (e.g., ESNet 100GB SDN testbed, AtlanticWave/SDX or GENI) and evaluate 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.

9. Acknowledgments

The authors would like to thank Sean Donovan for his help in setting up the AtlanticWave/SDX testbed.

References

[1] N. Charbonneau, V. M. Vokkarane, C. Guok, I. Monga, Advance reservation frameworks in hybrid IP-WDM networks, Communications Magazine, IEEE 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 suit, `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 Communications Magazine 49 (5) (2011) 114–121. `doi:10.1109/MCOM.2011.5762807`.

[5] 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, USA, 2015, pp. 2:1–2:8. `doi:10.1145/2832099.2832101`.
URL `http://doi.acm.org/10.1145/2832099.2832101`

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

[7] 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. `doi:10.1109/SC.2010.4`.

[8] 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. `doi:10.1109/IWQOS.2008.10`.

[9] P. Xiao, Z. Hu, Two-dimension relaxed reservation policy for independent tasks in grid computing, Journal of Software 6 (8) (2011) 1395–1402.

[10] Esnet - network maps, `https://www.es.net/engineering-services/the-network/network-maps/`, accessed: 2017-08-28.

[11] Internet2 international networks, `https://noc.net.internet2.edu/i2network/live-network-status/maps-graphs/internet2-international-network.html`, accessed: 2017-08-28.

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

[13] Transferring data with bbcp, `https://www.olcf.ornl.gov/kb_articles/transferring-data-with-bbcp/`, accessed: 2017-09-04.

[14] 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.

[15] Globus - research data management system, `https://www.globus.org/`, accessed: 2017-01-20.

[16] E. Dart, et al., Esnet requirement review reports, `https://www.es.net/science-engagement/science-requirements-reviews/requirements-review-reports/`, accessed: 2017-01-14.

[17] 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 Generation Computer Systems.

[18] gRPC, `https://grpc.io/`, accessed: 2017-09-06.

[19] Atlanticwave/sdx controller prototype, `https://github.com/atlanticwave-sdx/atlanticwave-proto`, accessed: 2017-09-07.

[20] Framework Community, Ryu SDN Framework,, `http://osrg.github.io/ryu`, accessed: 2017-09-07.

[21] 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. doi:10.1145/1355734.1355746.
URL http://doi.acm.org/10.1145/1355734.1355746

[22] 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

[23] Linux tuning, https://fasterdata.es.net/host-tuning/linux/, accessed: 2017-09-07.

[24] S. Avallone, S. D'Antonio, M. Esposito, S. P. Romano, G. Ventre, Resource allocation in multi-domain networks based on service level specifications, Journal of Communications and Networks 8 (1) (2006) 106–115. doi:10.1109/JCN.2006.6182910.

[25] 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.

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

[27] 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. doi:10.1007/3-540-36180-4_9.
URL https://doi.org/10.1007/3-540-36180-4_9

[28] 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. doi:10.1109/IWQOS.2008.10.

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

[30] J. M. Plante, D. A. P. Davis, V. M. Vokkarane, Parallel and survivable multipath circuit provisioning in esnet's oscars, Photonic Network Communications 30 (3) (2015) 363–375. doi:10.1007/s11107-015-0535-x.
URL https://doi.org/10.1007/s11107-015-0535-x