Democratizing Network Reservations through Application-Aware Orchestration

Joaquin Chung, Rajkumar Kettimuthu, Nageswara S.V. Rao^, Ian Foster
Argonne National Laboratory. USA
^Oak Ridge National Laboratory. USA
Email: jchung@mcs.anl.gov, kettimut@mcs.anl.gov, raons@ornl.gov^, foster@anl.gov

***Abstract*—The provisioning of network connections for data transfers that provide quality of service (QoS) over research and education (R&E) networks is currently performed by network operators. For network connections that span multiple administrative domains, network operators have to reach agreements on reservation requirements. As a result, a network reservation request may take from days to weeks to be provisioned. To improve provisioning times and the success rate of multidomain network reservations, we designed and implemented an application-aware orchestration framework for multidomain R&E networks. This framework leverages latest developments in software-defined networking to automate network provisioning in order to democratize access to network reservation through novel APIs. We present the design, implementation, and evaluation of our application-aware orchestration framework. We evaluate our system using Mininet and demonstrate that it provisions 49% more reservations than current state-of-the-art systems within seconds of receiving a request.**

I. Introduction

Network quality of service (QoS) has been studied for a few decades now. Internet stream protocol (ST) defined in Internet Experiment Note 119 [1] in 1979 was developed to support (voice) applications requiring guaranteed data rates and controlled delay. Transport protocols such as Packet Video Protocol [2] and the Network Voice Protocol [3] were developed to be used over ST. In 1990 ST was revised to support a wide variety of applications and to ease implementation difficulties; the revised protocol is commonly known as ST-2 [4]. As opposed to this approach of adding a new real-time protocol in the Internet layer, an integrated services model [5], [6], [7], [8] was proposed as an extension to the Internet architecture, including protocols to support real-time service in addition to the best-effort service of IP. The proposed model, called Internet Integrated Services or IntServ, seeks to merge the advantages of two network types: (a) datagram networks maximize the network utilization by multiplexing multiple data streams but provide only a best-effort delivery service, and (b) circuit switched networks provide service guarantees but can lead to inefficient use of network resources. The goal of IntServ is to support QoS while making efficient use of network resources [5]. The IntServ model requires a signaling protocol to establish additional packet classification and forwarding state on each node along the path between sources and receivers. The resource reservation protocol (RSVP) [9] is typically used for this purpose. Scalability is a concern for IntServ because every router on that path has to maintain per flow state information. Differentiated services architecture [10] minimizes signaling and concentrates on aggregated flows. The network traffic is differentiated into a set of classes, which in turn determine the per hop behavior of network nodes. A hybrid approach has also been proposed that uses differentiated services mechanisms to aggregate integrated services state in the core of the network [11].

Earlier investigations of network QoS focused on Web- and media-streaming applications. General-purpose Architecture for Reservation and Allocation (GARA) [12] focused on QoS for high-end scientific flows. It is built using DiffServ mechanisms to support latency- and jitter-sensitive flows and high-bandwidth but latency-insensitive flows. In the early 2000s, pioneered by the DOE UltraScience net [13], [14], a significant amount of work was done on on-demand provisioning of network resources in R&E networks. These include CHEETAH [15], DRAGON [16], UCLP [17], OSCARS [18], and others [19]. While these efforts focused on dynamic provisioning of resources in the wide-area network infrastructure, projects such as TeraPaths [20], LambdaStation [21], and DYNES [22] developed methods to address the last mile issue of provisioning the resources on local-area networks to provide end-to-end QoS between scientific computing systems or instruments. These efforts have led to the production deployment of on-demand network provisioning services such as OSCARS [18] and AL2S [23]. However, despite being available for more than a decade in major R&E networks, dynamic provisioning of network resources end-to-end (for example, from a data transfer node at site A to a data transfer node at site B) remains a daunting task.

The key challenges include the following:

- Non availability of advance reservation systems at end sites, a situation that we hope will change with the advent of Software-Defined Networking (SDN)
- Lack of multidomain network orchestration tools and interfaces that capture the flexibilities needed to fulfill the application requirements
- Limited support from existing network provisioning systems in exposing the state information and/or providing alternate choices

Our work focuses on the last two challenges. To realize an end-to-end network orchestration system that provides flexible reservations, maximizes the success rate of requests, and

maintains participant's topology privacy, we have developed a reservation agent for the domain controllers and a multidomain orchestration client that interacts with the reservation agent or the legacy controller interfaces. The contributions of this paper are threefold:

1) Design of an application-aware orchestration framework to support end-to-end, flexible network reservations.
2) Two reservation schemes based on legacy reservation requests that applications can use to improve the success rate of reservations on state-of-the-art advance reservation systems.
3) Two reservation schemes based on a novel method for allocating bulk data transfers, called Bandwidth Curve, that provides 49% better success rate than the legacy reservation scheme.

The parameters usually considered for QoS are throughput, delay, jitter (maximum variance in the arrival of data at the destination), and loss rate [24]. In this work, we consider only throughput, but the tools developed can be used for other QoS parameters as well.

The remainder of this paper is as follows. Section II provides background and motivation and Sections III and IV describe the architecture and design, respectively. Section V describes the implementation, Section VI presents the evaluation results, and Section VII provides a discussion on new research directions. Section VIII discusses related work. Section IX presents conclusions and briefly outlines future work.

II. Background and motivation

Real-time data analysis is emerging as a key requirement for many science workflows. For many such workflows, the data rates from scientific instruments have been growing much faster than have computer speeds. Thus, local compute resources are often no longer sufficient to analyze data in a reasonable time. As the use of remote compute resources becomes a necessity, so too does end-to-end network QoS. However, although major R&E networks provide bandwidth reservation capabilities, such capabilities are used mostly by network operators to set up long-term circuits for big science projects and other operational needs. In the paragraphs that follow, we describe four important features that are not provided by today's state of the art systems.

Flexible reservation interfaces: The reservation interfaces provided by the major R&E networks are not flexible: a reservation request will typically fail if the exact amount of bandwidth between two endpoints is not available within the specified time frame. Balman et al. [25] proposed a flexible reservation algorithm for a single domain, where the user specifies the requirements (total volume that needs to be transferred, a maximum bandwidth that can be used and provisioned in the client sites, and a desired time window within which the transfer should be done), and the system suggests a few reservation options, including the earliest time for completion, or shortest transfer duration—leaving the choice to the user. This problem is dramatically amplified for multidomain bandwidth reservations, because participants have to reach an agreement on a suitable advance reservation that fulfills the requirements of the original request. Furthermore, despite a majority of domains having available resources for the reservation, if only one domain is not able to provide the requested resources, a multidomain advance reservation will fail. This problem is analogous to trying to reserve a multilegged flight with airlines that are not part of the same consortium and do not share flight schedules. It impacts the user's productivity. Every time a reservation fails, the systems forces the user into a cycle of trial and error until a suitable time frame is found. For legacy controllers, we address this issue by moving the trial-and-error logic into our orchestrator.

Maximization of the success rate: Even with a flexible interface, satisfying a user request that involves multiple domains may be challenging since all the domains involved must have the resources to meet the request. In earlier work [26], we incorporated a negotiation protocol and a two-phase commit protocol to deal with race conditions and splits the bandwidth reservation requests among multiple parallels paths, but we need to develop methods to maximize the success rate, while reducing the round-trip time between the orchestrator and domain controllers in the absence of path diversity.

Participant domain privacy: Currently, advance reservation systems in R&E networks support topology sharing for multidomain bandwidth reservations (e.g., DYNES, OSCARS, and OESS) [27]. However, end sites or commercial transport providers might not want to share their internal topologies [28]. The negotiation protocol that we proposed previously [26] does not require topology sharing: it is an offer-oriented protocol. Under such an approach, however, if we allow an open-ended request, we may have an infinite number of options, so evaluating all options is unproductive. We need an approach that obtains the optimal bandwidth reservation with the least number of interactions between the orchestrator and domain controllers.

Reservation at end sites: Few end sites in R&E environments provide a programmatic way to provision network resources. Although software-defined networking (SDN) [29] has gained traction in recent times, it has not been widely deployed for production use in R&E environments. Even where SDN has been deployed, it is not available for end users to provision resources. There are sociopolitical challenges in deploying policies to control access. Moreover, the lack of a standard northbound interface for applications and higher-level tools to orchestrate end-to-end reservations in multidomain environments contributes to the inertia in deploying SDN solutions and enabling users to provision network resources in a programmatic fashion.

The work that we present in this paper addresses these four requirements, with the goal of democratizing network reservations in multidomain R&E environments.

III. Architecture Overview

A centralized orchestrator is easy to reason about, but it is expensive to deploy and manage. Many questions emerge from this implementation decision: Where is the orchestrator located? Who is in charge of managing and operating the orchestrator? In this regard, we propose to deploy the orchestrator as client software that anyone can run from anywhere. Domain controllers expose science network services that these orchestration clients can consume through reservation agents. These agents enable the deployment of end-to-end bandwidth reservations across participant domains. Figure 1 shows our architecture and its reservation agents and application-aware orchestration clients. The reservation agent provides a standard northbound API for flexible reservations to domain controllers.

Orchestration clients are typically located at end sites. A client can request bandwidth reservation from any domain controller as long as the client, or the user on whose behalf the client is acting, is authorized to make the reservation. Each domain participating in end-to-end QoS negotiation deploys a reservation agent on top of its domain controller to provide flexible bandwidth reservation services.

IV. Design

We present the design considerations for our application-aware orchestration framework.

A. Network Reservation Services

The foundation for our application-aware orchestration framework is the type of services that reservation agents offer to orchestration clients. The types of network reservation services supported by each controller must be uniform so that an orchestration client can compose an end-to-end advance reservation correctly. Each network reservation service is based on an existing data transfer type. We can classify data transfers in two groups: best effort and those that require quality of service. The QoS group can be further divided into streaming data transfers and bulk data transfer. Based on these types of data transfers, we define two types of reservation services:

1) A **constant** service is used to reserve a guaranteed fixed bandwidth for a duration within a given time window. The client specifies the desired bandwidth, duration, and time window ($\geq$ duration).
2) A **moldable** service is used to reserve a variable amount of bandwidth within a given time window. The client specifies the desired data size, maximum usable bandwidth, and window ($\geq$ data size / maximum bandwidth).

In both cases, the service proceeds in two phases. In Phase I, it obtains desired parameters from the client and presents the client with zero or more choices that satisfy those parameters. In Phase II, it obtains from the client a specific choice (picked from one of the choices provided in Phase I) and commits to that choice if it is still available.

Advance reservation systems that only process reservation requests defined by start time, end time, and required bandwidth can be expressed as a special case of the constant bandwidth reservation service, in which the time window is equal to the duration. We call this special case a **rigid** reservation service.

Once we have defined the network reservation services supported by reservation agents, we require a reservation request format that considers current state-of-the-art fields (i.e., start_time, end_time, and bandwidth), while adding the least amount of new fields. We add two new fields (data size and transfer duration). As a result, we only require five fields to accomplish our goal:

1) Earliest start time
2) Deadline
3) Data size
4) Transfer duration
5) Bandwidth

For both types of services, the earliest start time and the deadline are required. For the constant bandwidth reservation service, the bandwidth and duration are required, but the data size is *don't care*. On the other hand, for the moldable bandwidth service, the data size and bandwidth (which represents the maximum bandwidth that can be used) are required, but the duration is *don't care*. *Don't care* fields can be represented as zeros. In the special case of a rigid request, the duration must be equal to the time elapsed from the earliest start time to the deadline.

In addition to the network reservation services, the application-aware orchestration framework must support the following underlying services:

1) Topology Service: maintains the network topology of active participating domains
2) Participant Directory: provides a directory of active participant domains

To maintain consistency of topology and participant information, we may run these services over a control network of reservation agents.

B. Reservation Agents

The reservation agent exposes network reservation services for the consumption of orchestration clients. It must track the available bandwidth within a domain across time. To this end, a network administrator may force all bandwidth reservation requests to go through the reservation agent. Another option is that the agent only serves requests from orchestration clients, but synchronizes with internal bandwidth resources managers in the domain controller.

The domain controller is in charge of handling internal bandwidth request and internal configurations. The design of the domain controller is outside the scope of this paper, since we are more interested in the end-to-end orchestration of dedicated circuits. The domain controller could easily be an advance reservation system for a REN [18], [23] or an SDN controller [30], [31] for an end site.

C. Orchestration Client

The orchestration client is a software distribution that anyone can run from anywhere to request an end-to-end

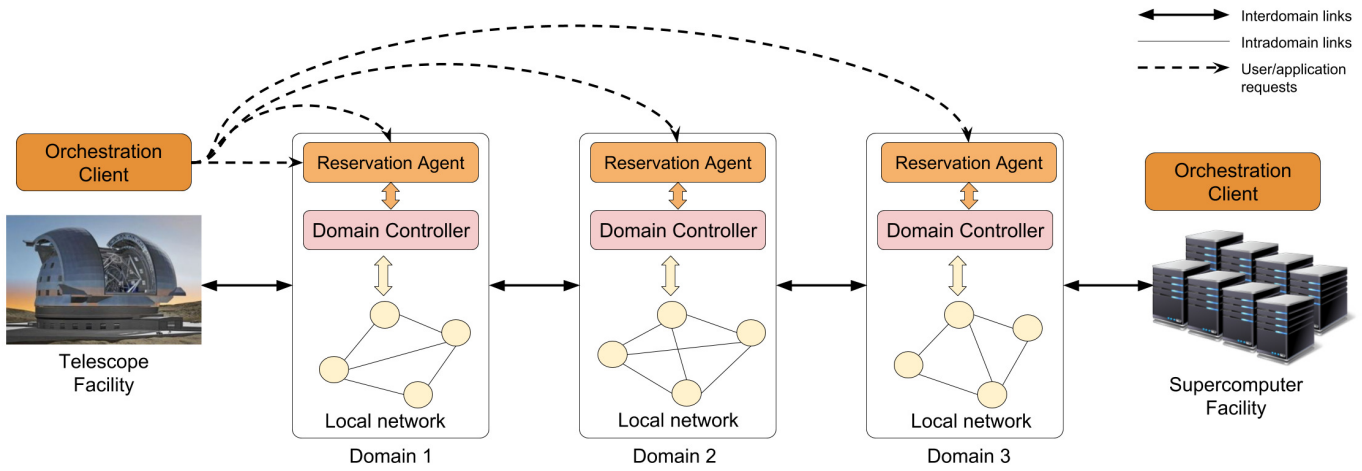

Figure 1. High-level view of our application-aware orchestration architecture for flexible bandwidth reservation, showing two scientific facilities (domains 1 and 3) connected by a backbone network (domain 2).

bandwidth reservation. The client can interact with users or can be integrated with science workflow applications to request bandwidth reservations. Before contacting participant domains for bandwidth reservation services, the orchestration client needs to know what reservation agents it should contact. First, the orchestration client runs a path-finding algorithm on the network topology. The topology may be provided by the closest agent to the client. Most likely, this agent will be located in the same site as the orchestrator clients. The path-finding algorithm may return a single path or multiple paths for redundancy, or multipath reservations. Second, the orchestration client extracts the participant domains in the path(s) and contacts them for bandwidth reservations. Third, the orchestration client computes whether the end-to-end service is possible. The next subsection discusses how the orchestration client arrives at a conclusion by using several reservation methods.

D. Reservation Methods

Our application-aware orchestration framework must be backward compatible with rigid reservation services. However, applications must not be limited by legacy advance reservation systems. In this subsection, we describe some methods that the orchestrator can use with legacy systems to improve the success rate of rigid reservation requests.

1) **Legacy**: A rigid reservation request is defined by start time, end time, and bandwidth, where duration is equal to end time minus start time. The legacy method treats all reservation requests as rigid requests.
2) **Legacy with Multiple Attempts** (LMA): Based on the streaming service described in subsection IV-A, an orchestration client may start by requesting a rigid reservation from the earliest start time until that time plus duration. If the reservation fails, the orchestration client may increase the start and end times by a predefined increment and retry. An orchestration client may keep doing so until it finds a reservation or the end time reaches the deadline. We propose two approaches to define the increment:
   a) Even split: Considering the latest start time to be $deadline - duration$ and a predefined number of attempts, we can define the even split increment as $(latest\_start\_time - earliest\_start\_time)/(num\_attempts - 1)$.
   b) Exponential: Inspired by the exponential backoff algorithm used to avoid congestion in Ethernet networks, we increase the size of the increment exponentially on each attempt until the end time reaches the deadline or the number of attempts reaches its quota. We set the initial increment to be a random number between 30 and 120 seconds, to ensure that in a multiple orchestrator scenario, two or more orchestrators do not choose the same reservations on their retries, effectively reducing contention and increasing success rate.

   It is important to note that for legacy advance reservation systems, Tepsuporn et al. [27] reported that the DYNES systems blocks a client for 15 minutes after a failed request. For LMA service to be possible, advance reservation systems must relax their policies, but we need to be conscious that we may enable a denial of service attack if we are careless with the maximum number of attempts. A reasonable approach is to allow the orchestration client to make a limited number of consecutive attempts.
3) **Moldable**: This method allows an orchestration client to craft rigid requests based on the earliest start time, deadline, and data size parameters. As long as the data size can be transferred before the deadline, regardless of how much bandwidth is requested, the service is possible. For instance, an orchestration client may request the minimum bandwidth from earliest start time to deadline, or it may request the maximum available bandwidth or the maximum the application can use (whichever is smaller) to finish earlier. The same limitations on how many

requests an orchestration client may craft applies to this method.

4) **Moldable with Multiple Attempts** (MMA): This method combines the multiple tries over time of the LMA method with the request crafting of the moldable method in a single method. In this approach, an orchestration client starts by crafting several rigid reservation requests. Then, it tries these reservations using the LMA method. Again, a limitation on how many consecutive requests an orchestration client may issue must be enforced.

The novelty of our application-aware orchestration framework is its ability to exploit multiple reservation options available at each domain in order to increase the success rate of user/application requests. Here, we present the *Bandwidth Curve* method, in which each reservation agent returns to an orchestration client its available bandwidth from earliest start time to deadline. Once the orchestration client collects all bandwidth curves, it produces a new bandwidth curve with bandwidth corresponding at each point in time to the minimum available bandwidth across all participant domains. If the area below this new curve, which is $\int_{time} bandwidth$, is greater than or equal to the data size, the service can fulfill the request. Then, the orchestrator picks a set of one or more *bandwidth* $\times$ *time* regions with total area equal to the data size and requests the agents to commit to these regions. Each region is represented by a start time, end time, and bandwidth. The resultant bandwidth curve may be discontinuous, but a suitable data transfer application can handle this by pausing the transfer or using a best-effort path during the discontinuity.

V. Implementation

We next present our implementation decisions for the application-aware orchestration framework.

A. Reservation Agent and Orchestration Client

We implemented both the reservation agent and orchestration client in Python using the general remote procedure call (gRPC) protocol [32] and protocol buffers [33]. The gRPC protocol is a high-performance RPC framework optimized for distributed-computing and mobile environments; it uses protocol buffers [33] for defining services and message types and for serializing data.

All reservation requests use the same structure defined in subsection IV-A: earliest start time, deadline, data size, transfer duration, and bandwidth. The orchestration client determines the type of service based on the *don't care* fields. Five types of reservation methods are supported: Legacy, LMA, Moldable, MMA, and Bandwidth Curve. The response to Legacy, LMA, Moldable, and MMA is a string "YES" if the service is possible and "NO" if it is not, as they are based on rigid reservation requests. The reply to a Bandwidth Curve request is a bandwidth curve as defined in subsection IV-D.

B. Available Bandwidth Data Structure

To be able to provide bandwidth curves while keeping a simplified implementation, we implemented the data structure for available bandwidth as an array the size of 86,400 entries. In this array, each element represents the available bandwidth for one second of the day. We can afford to have this granularity thanks to the high availability of memory in current HPC systems. For instance, considering the available bandwidth a float type, we would need only 121.5 MB to represent one year's worth of future reservations.

VI. Evaluation

We present the results of experiments that we conducted to evaluate our application-aware orchestration framework.

A. Experimental Setup

We ran our experiments on an Ubuntu 16.04 machine with 16 Intel(R) Xeon(R) CPU E5530 @ 2.40 GHz cores and 48 GB of RAM. We evaluated our application-aware orchestration framework in Mininet [34], measuring the reservation success rate and the system latency in both a single orchestrator scenario and a multiple orchestrator scenario. In the first, a logically centralized orchestrator receives all requests from users and applications. In the second, each user or application runs its own independent orchestrator. Then, we evaluated two dimensions of the requests for each scenario: flexibility and load. We define flexibility as a factor of duration that extends the end time to provide a deadline. The load refers to the volume of requests that arrive to the orchestration. We define two levels of load: original load and $1.25\times$ load.

The Mininet topology is composed of 10 participant domain controllers, one orchestrator for the single orchestrator scenario, and three orchestrators for the multiple orchestrator scenario. We used Mininet hosts to represent all 14 entities and a Mininet switch to represent the communication control channel between them. Of the 10 domain controllers, four represent scientific facilities (e.g., supercomputer center, light source facility [35], [36]), two represent backbone providers (e.g., ESnet and Internet2), two more represent regional R&E networks, and the remaining two are universities.

Since this work is more concerned with the resource orchestration than with the actual data transfer, we do not provide an actual topology of the participant domains, nor do we present a rigorous path-finding algorithm. Instead, we consider three general types of paths:

1) *Two facilities communicate with each other:* This path involves three domains: the two facilities and a backbone provider such as ESnet.
2) *Two universities communicate with each other:* This path involves five domains: the two universities, two regional networks, and a backbone provider such as Internet2.
3) *A facility and a university communicate with each other:* This path involves five domains: a facility, two backbone providers, a regional, and a university.

To make our experiments more realistic, we generated seven paths: four of type 1, one of type 2, and two of type 3.

Another important consideration of our experimental setup is the maximum available bandwidth when the experiment starts. To account for other traffic, we assigned values lower

than typical maximum capacity for each of these domains. Table I shows the values we assigned to each type of network for our experiments. We prepopulated each reservation agent with randomly generated advance reservations.

Table I
INITIAL MAXIMUM AVAILABLE BANDWIDTH FOR DIFFERENT DOMAINS

Type of Network	Max. Avail. BW
Backbone	100Gbps
Facility	80Gbps
Regional	40Gbps
University	20Gbps

B. Workload Generation

We generated our workloads (i.e., sequences of reservation requests) from Globus [37] transfer logs. We took one day's worth of data transfer logs for our experiments, considering transfers greater that 1 GB only. We assigned the transfers randomly to different paths in our experimental setup, ensuring individual requests do not exceed the maximum capacity on the paths (for example, we made sure the rates requested by transfers to a path that involve a university endpoint lower than 20Gbps).

To build each individual request, we translated the Globus transfer fields request_time, complete_time, elapsed_time, bytes, and rate into our own earliest start time, deadline, transfer duration, data size, and bandwidth. Then, we randomly assigned the request as one of the two reservation services (as defined in subsection IV-A) request, according to the following distribution: 60% streaming (constant bandwidth) requests, and 40% bulk data transfer (moldable bandwidth) requests. We also (randomly) assigned one-third of streaming requests as rigid requests (whose $deadline = earliest\_start\_time + duration$). To add flexibility for non-rigid requests, we considered two scenarios. The first one increases the complete_time (deadline) by a factor chosen at random between $1\times$ and $2\times$ the duration. The second one chooses a factor between $1\times$ and $4\times$ the duration. For the multiple-orchestrator scenario, we evenly split the reservation request among three controllers.

The result of this process is our *original* workload. We also create a *1.25*$\times$ workload in which 25% (randomly selected) of the requests in the original workload are duplicated. On each repetition of our experiments, we reordered the requests to add more stochasticity.

C. Reservation Schemes

We combined the reservation schemes defined in subsection IV-D to generate the five reservation schemes listed in Table II, Two methods for handling streaming requests – LMA with even split increment (*LMA-ES*) and LMA with exponential increment (*LMA-Exp*), and two methods for handling bulk data transfer requests – MMA and Bandwidth Curve, are combined to four reservation schemes. A fifth scheme, *Legacy*, treats all requests as rigid reservations.

Table II
RESERVATION SCHEMES

Scheme	Description
Legacy	Treats everything as rigid request
LMA-ES-MMA	Combines LMA even split increment and MMA
LMA-ES-BWC	Combines LMA even split increment and Bandwidth Curve
LMA-Exp-MMA	Combines LMA exponential increment and MMA
LMA-Exp-BWC	Combines LMA exponential increment and Bandwidth Curve

D. Measurement Results

We measured the success rate and system latency for each reservation scheme defined in Table II, for both a single orchestrator and multiple orchestrators. Table III describes the experiment configurations that we consider as we vary the two parameters under evaluation: flexibility and request load. Although the single orchestrator scenario can emulate $2\times$ load, we were forced to choose $1.25\times$ original load for our experiments, because the multi orchestrator scenario cannot support larger loads. This is due to consecutive calls to the reservation agents causing a blockage in the communication channel. We are working on optimizing this for future work and real world deployments. We show our results in Figure 2 and Figure 3.

Table III
EXPERIMENT CONFIGURATIONS

Flexibility	Load
1–2$\times$ duration of the transfer	Original load
1–2$\times$ duration of the transfer	1.25$\times$ original load
1–4$\times$ duration of the transfer	Original load
1–4$\times$ duration of the transfer	1.25$\times$ original load

Success Rate Results: Figure 2 shows that any scheme that uses Bandwidth Curve to handle bulk data transfer request allocates up to 49% more requests than Legacy for both single and multi orchestrator scenarios, while schemes that use MMA allocate up to 22% more requests in their best case scenario (1–4$\times$ flexibility with original load). In absolute numbers, the schemes that use Bandwidth Curve for bulk transfer requests can achieve up to 75% success rate, while the schemes that use MMA achieves 55%. In general, increasing flexibility from 1–2$\times$ to 1–4$\times$ improves the success rate by 14%.

It is important to note that the multiple orchestrator scenario presents better success rate at $1.25\times$ load than at original load for both 1–2$\times$ and 1–4$\times$ flexibility. These results are promising as the multiple orchestrator scenario might be able to perform better at higher loads. However, more experimentation is required to confirm this claim.

Another important observation is that schemes that use Bandwidth Curve for handling bulk data transfers always provides the best success rate for each scenario and testbed configuration. However, it is not clear whether using LMA with even split or LMA with exponential increment is better

for handling streaming requests. More experimentation is required to confirm which one is the best.

System Latency Results: Figure 3 shows that any scheme that uses Bandwidth Curve suffers a penalty of approximately three orders of magnitude larger system latency compared to other schemes. We note that the streaming requests do not incur high latency as the Bandwidth Curve method is applied only for bulk transfer requests. Although we expect that this latency will decrease in the multiple orchestrator scenario (as the requests that arrive around the same time can be processed in parallel), we observe an increase in latency. We suspect that Mininet might be causing serialization of the requests in the experiment. However, this requires more investigation.

In terms of the absolute values, while the schemes that rely on MMA have latency on the order of tens of milliseconds, the system latency for those that rely on Bandwidth Curve is on the order of few seconds. In general we do not observe significant changes in system latency as the flexibility factor increases, because the tasks that the orchestrator and reservation agents have to complete are the same regardless of the flexibility.

These results mean that a user or an application with reservation requests for bulk data transfer can trade off success rate and system latency. If an application can afford to wait a couple of seconds for a response, it could get higher success rates by using schemes that rely on Bandwidth curve. On the other hand, if it requires fast response times we recommend schemes that use MMA for handling bulk data transfers.

VII. Discussion

We discuss a variation of the Bandwidth Curve method and how the malleable nature of the QoS needs of bulk data transfers can be exploited to accommodate more QoS flows.

A. Top K Offers – An Alternate Reservation Method

We can imagine what we call a *Top K Offers* reservation method, in which each reservation agent provides its top K offers for start time, bandwidth, and duration to an orchestration client. If the orchestration client finds an overlap among the offers from different agents, it composes a reservation request and instructs the agents to commit that reservation. As each agent picks its top K offers without communicating with (or knowing the state of) its peers, finding an overlap among the offers from different domains may be difficult. An orchestration client may need to provide a high flexibility factor (i.e., a deadline far into the future), or request many offers to increase the success rate of requests. This does not happen in the Bandwidth Curve scheme because agents return the available bandwidth over a time period and it is the orchestration client that composes the final reservation. To improve the success rate and reduce the latency of a *Top K Offers* method, an orchestration client might request the top K offers from the reservation agent that is most likely to be the bottleneck. Then, it could use the offers to request reservations to the other agents on the path. Other optimizations can be imagined. We leave the implementation and evaluation of *Top K Offers* methods for future work.

B. Malleability

As mentioned in subsection IV-A, the bandwidth provided to bulk transfers (with QoS requirements) can vary so long as the aggregate bandwidth provided is sufficient to complete the transfers before the deadline. Depending on the environment, these transfers can be either moldable (able to adjust to different bandwidth choices before the transfer starts) or moldable as well as malleable (able to adjust to bandwidth changes during the transfer), within their QoS constraints (being able to complete the transfers by the deadline). In this work, we considered these transfers to be moldable but not malleable. Exploiting the malleability of these transfers can help accommodate more QoS transfers and is an interesting research topic. A significant amount of work has been done on exploiting moldability [38], [39], [40] and malleability [41], [42], [43] in the context of parallel job scheduling. Although significant differences exist between parallel job scheduling and end-to-end bandwidth scheduling in multidomain environments, inspirations can be drawn from the work in the parallel job scheduling context.

VIII. Related Work

Balman et al. [25] 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 [44] proposed a two-dimensional relaxed reservation policy for Grid computing systems that achieves higher resource utilization and success rates (approximately 95% under low reservation rates). Both approaches are single-domain scheduling algorithms and thus complement our work when deployed inside participant domains.

He et al. [45] proposed a flexible advance reservation model for cross-domain lightpath reservations in optical networks that can achieve a maximum reservation success rate of 84%. This model is a cross-domain approach specific for optical networks, whereas our application-aware orchestration framework is technology agnostic.

We previously [26] developed an architecture for multidomain, multipath advance reservations in science networks and software-defined exchanges that increases the reservation success rate from 50% with a single path to 99% with four paths available. On that work, we treat all request as rigid reservations, and we split the bandwidth across multiple parallel paths.

IX. Conclusions

We have proposed an application-aware orchestration framework for end-to-end network reservations in R&E networks. Our framework is composed of reservation agents, which reside on each participant domain, and application-aware orchestration clients, which can be run by anyone anywhere. We presented the design, implementation, and evaluation of our application-aware orchestration. Furthermore, we defined two

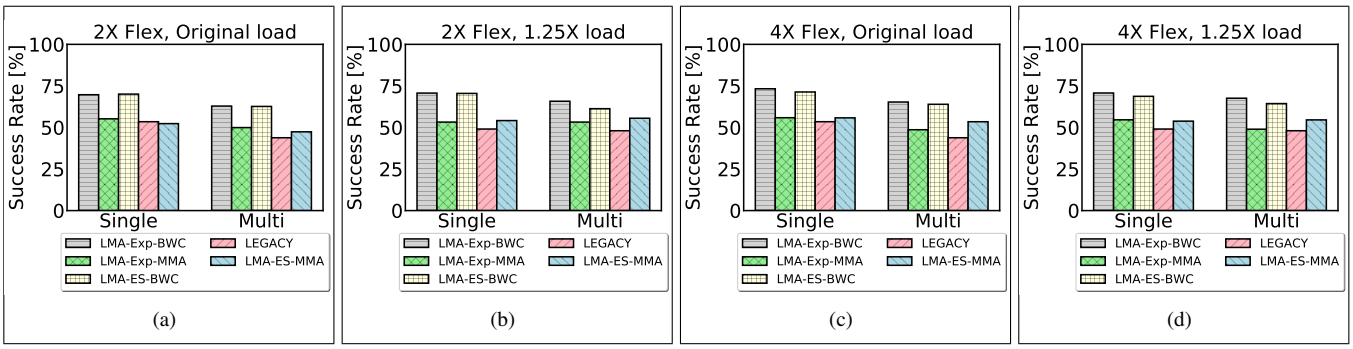

Figure 2. Success rate results for flexibility: (a) 1–2× with original load, (b) 1–2× with 1.25× load, (c) 1–4× with original load, (d) 1–4× with 1.25× load. In each case, we show results for both a single controller and multiple controllers, and for the five reservation schemes described in Table II.

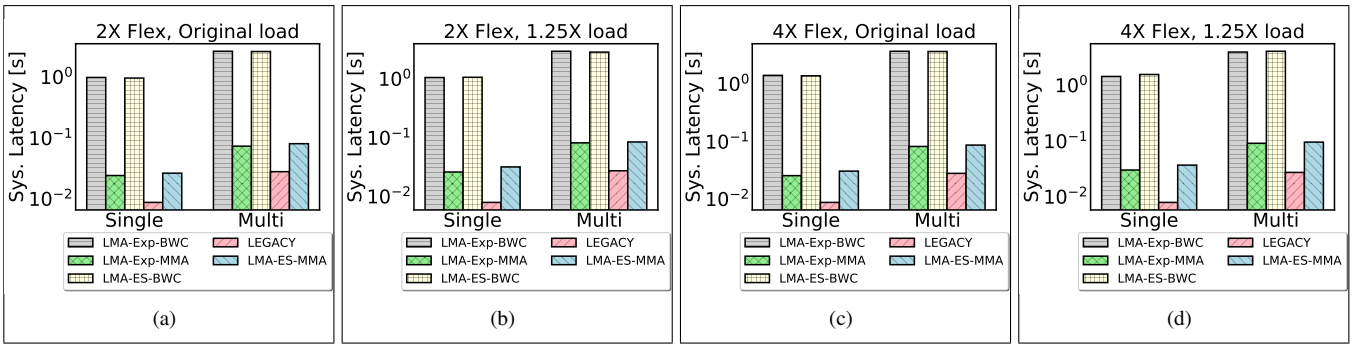

Figure 3. Success rate results for flexibility: (a) 1–2× with original load, (b) 1–2× with 1.25× load, (c) 1–4× with original load, (d) 1–4× with 1.25× load. In each case, we show results for both a single controller and multiple controllers, and for the five reservation schemes described in Table II.

reservation schemes based on legacy reservation requests that applications can use to improve the success rate of reservations on state-of-the-art advance reservation systems. We also presented two reservation schemes based on a novel method for allocating bulk data transfer reservations, Bandwidth Curve, that provide 49% better success rate than the legacy reservation scheme. We conducted our experiments using Mininet, with a topology that models current data transfers across multiple scientific facilities.

Acknowledgments

This material was based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

References

[1] J. W. Forgie, "ST - A Proposed Internet Stream Protocol," IEN 119, Sep. 1979. [Online]. Available: https://www.rfc-editor.org/ien/ien119.txt

[2] E. Cole, "PVP–A packet video protocol," 1981.

[3] D. Cohen, "Specifications for the network voice protocol (nvp)," United States, 1977.

[4] C. Topolcic, "Experimental internet stream protocol: Version 2 (ST-II)," United States, 1990.

[5] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: An overview," IETF, United States, 1994.

[6] D. D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism," *SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 4, pp. 14–26, Oct. 1992. [Online]. Available: http://doi.acm.org/10.1145/144191.144199

[7] J. S. Turner, "Design of an integrated services packet network," *SIGCOMM Comput. Commun. Rev.*, vol. 15, no. 4, pp. 124–133, Sep. 1985. [Online]. Available: http://doi.acm.org/10.1145/318951.319028

[8] D. Ferrari, "The Tenet experience and the design of protocols for integrated-services internetworks," *Multimedia Syst.*, vol. 6, no. 3, pp. 179–185, May 1998. [Online]. Available: http://dx.doi.org/10.1007/s005300050086

[9] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource ReSerVation protocol," *IEEE Network*, vol. 7, no. 5, pp. 8–18, Sept 1993.

[10] D. Black and P. Jones, "Differentiated services (DiffServ) and real-time communication," Tech. Rep., 2015.

[11] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine, "A framework for integrated services operation over DiffServ networks," IETF, United States, 2000.

[12] I. Foster, A. Roy, and V. Sander, "A quality of service architecture that combines resource reservation and application adaptation," in *IWQOS 2000. Eighth International Workshop on Quality of Service*. IEEE, 2000, pp. 181–188.

[13] N. S. V. Rao, W. R. Wing, S. M. Carter, and Q. Wu, "UltraScience Net: Network testbed for large-scale science applications," *IEEE Communications Magazine*, vol. 43, no. 11, pp. S12–S17, Nov 2005.

[14] N. S. V. Rao, Q. Wu, S. Ding, S. M. Carter, W. R. Wing, A. Banerjee, D. Ghosal, and B. Mukherjee, "Control plane for advance bandwidth scheduling in ultra high-speed networks," in *25TH IEEE International Conference on Computer Communications*, April 2006, pp. 1–5.

[15] X. Zheng, M. Veeraraghavan, N. S. V. Rao, Q. Wu, and M. Zhu, "CHEETAH: Circuit-switched high-speed end-to-end transport architecture testbed," *IEEE Communications Magazine*, vol. 43, no. 8, pp. s11–s17, Aug 2005.

[16] T. Lehman, J. Sobieski, and B. Jabbari, "DRAGON: A framework for service provisioning in heterogeneous grid networks," *IEEE Communications Magazine*, vol. 44, no. 3, pp. 84–90, March 2006.

[17] J. Wu, M. Savoie, S. Campbell, H. Zhang, G. v. Bochmann, and B. S. Arnaud, "Customer-managed end-to-end lightpath provisioning,"

International Journal of Network Management, vol. 15, no. 5, pp. 349–362. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.581

[18] I. Monga, C. Guok, W. E. Johnston, and B. Tierney, "Hybrid networks: Lessons learned and future challenges based on ESnet4 experience," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 114–121, May 2011.

[19] M. Campanella, R. Krzywania, V. Reijs, and A. Sevasti, "The bandwidth on demand service for the European research and education networks," in *International Conference on Photonics in Switching*, Oct. 2006, pp. 1–4.

[20] D. Katramatos, D. Yu, B. Gibbard, and S. McKee, "The TeraPaths testbed: Exploring end-to-end network QoS," in *2007 3rd International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities*, May 2007, pp. 1–7.

[21] A. Bobyshev, M. Crawford, P. DeMar, V. Grigaliunas, M. Grigoriev, A. Moibenko, D. Petravick, R. Rechenmacher, H. Newman, J. Bunn, F. V. Lingen, D. Nae, S. Ravot, C. Steenberg, X. Su, M. Thomas, and Y. Xia, "Lambda Station: On-demand flow based routing for data intensive grid applications over multitopology networks," in *3rd International Conference on Broadband Communications, Networks and Systems*, Oct 2006, pp. 1–9.

[22] 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, and X. Yang, "The DYNES instrument: A description and overview," *Journal of Physics: Conference Series*, vol. 396, no. 4, p. 042065, 2012. [Online]. Available: http://stacks.iop.org/1742-6596/396/i=4/a=042065

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

[24] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[25] M. Balman, E. Chaniotakisy, A. Shoshani, and A. Sim, "A flexible reservation algorithm for advance network provisioning," in *2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2010, pp. 1–11.

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

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

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

[29] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

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

[31] GlobalNOC, "OESS: Open Exchange Software Suite," https://docs.globalnoc.iu.edu/sdn/oess.html, accessed: 2017-10-18.

[32] "gRPC," https://grpc.io/, accessed: 2017-09-06.

[33] "Protocol buffers," https://developers.google.com/protocol-buffers/, accessed: 2017-09-06.

[34] "Introduction to Mininet," https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

[35] ANL, "Advanced photon source - overview," https://www.aps.anl.gov/About/Overview, accessed: 2018-05-10.

[36] LBL, "Advanced light source," https://als.lbl.gov/, accessed: 2018-05-10.

[37] B. Allen, J. Bresnahan, L. Childers, I. Foster, G. Kandaswamy, R. Kettimuthu, J. Kordas, M. Link, S. Martin, K. Pickett, and S. J. Tuecke, "Software as a service for data scientists," *Communications of the ACM*, vol. 55, no. 2, pp. 81–88, 2012.

[38] G. Sabin, M. Lang, and P. Sadayappan, "Moldable parallel job scheduling using job efficiency: An iterative approach," in *12th International Conference on Job Scheduling Strategies for Parallel Processing*, ser. JSSPP'06. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 94–114. [Online]. Available: http://dl.acm.org/citation.cfm?id=1757044.1757049

[39] W. Cirne and F. Berman, "Adaptive selection of partition size for supercomputer requests," in *Workshop on Job Scheduling Strategies for Parallel Processing*, ser. IPDPS '00/JSSPP '00. London, UK: Springer-Verlag, 2000, pp. 187–208. [Online]. Available: http://dl.acm.org/citation.cfm?id=646381.689677

[40] S. Srinivasan, V. Subramani, R. Kettimuthu, P. Holenarsipur, and P. Sadayappan, "Effective selection of partition sizes for moldable scheduling of parallel jobs," in *9th International Conference on High Performance Computing*, ser. HiPC '02. London, UK, UK: Springer-Verlag, 2002, pp. 174–183. [Online]. Available: http://dl.acm.org/citation.cfm?id=645448.652950

[41] G. Allen, D. Angulo, I. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel, and J. Shalf, "The Cactus Worm: Experiments with dynamic resource selection and allocation in a grid environment," *International Journal of High Performance Computing Applications*, vol. 15, no. 4, 2001.

[42] L. V. Kale, S. Kumar, and J. DeSouza, "A malleable-job system for time-shared parallel machines," in *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002.*, May 2002, pp. 230–230.

[43] Jansen and Porkolab, "Linear-time approximation schemes for scheduling malleable parallel tasks," *Algorithmica*, vol. 32, no. 3, pp. 507–520, Mar 2002. [Online]. Available: https://doi.org/10.1007/s00453-001-0085-8

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

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