Scheduling End-to-End Flexible Resource Reservation Requests for Multiple End Sites

Li Shi

IEEE International Conference on Computing, Networking and Communications (ICNC 2015)
Anaheim, CA
February 16-19, 2015

Feb. 2015

Computational Science Center

Brookhaven National Laboratory

U.S. Department of Energy
Office of Advanced Scientific Computing Research

Scheduling End-to-End Flexible Resource Reservation Requests for Multiple End Sites

Li Shi
Department of Electrical and Computer Engineering,
Stony Brook University,
Stony Brook, NY 11794.
Email: li.shi@stonybrook.edu

Sushant Sharma*
Windows Azure,
Microsoft Corp.,
Redmont, WA 98052.
Email: sushant_sharma@outlook.com

Dimitrios Katramatos
Dantong Yu
Computational Science Center,
Brookhaven National Laboratory,
Upton, NY 11973.
Email: {dkat,dtyu}@bnl.gov

***Abstract*—Wide area research and education networks, such as ESnet and Internet2 in the US and GEANT in Europe, have recently deployed software that makes possible to reserve bandwidth in the form of dynamic circuits. Such circuits offer guaranteed QoS to specific data flows, significantly increasing the reliability and predictability of data transfers. In this paper, we study the problem of constructing routes and scheduling bandwidth reservations for data transfers between multiple pairs of end sites. We develop an algorithm, called RRM, to solve this problem. Our objective is to maximize the number of satisfied data transfer requests while minimizing the total data transfer times. We further prove that our problem is NP-hard and compare our algorithm with a baseline FCFS algorithm through simulations. The simulations indicate that our algorithm accommodates up to 160% more requests and achieves up to 50% shorter average data transfer times than the baseline algorithm.**

I. Introduction

In modern sciences, such as nuclear physics, astrophysics, and climate prediction, the amount of experimental data that needs to be transferred between geographically distant end sites for processing and analysis is increasing at a phenomenal rate. Researchers (or end users) regularly require data to be transferred between multiple end sites in a time-bound manner. In order to support these large-scale time-bound data transfers, the US Department of Energy (DoE) and the National Science Foundation (NSF) have invested significantly in research and education networks such as ESnet [1] and Internet2 [2]. Major efforts have also been dedicated on developing software that can be used to reserve resources on these networks (such as OSCARS [3]). There have been other efforts, such as TeraPaths [4], that extend the resource reservation capabilities from WANs all the way to the hosts sending and/or receiving data within the end-sites. These projects aim to achieve an end-to-end (host-to-host) reservation of resources by constructing virtual end-to-end paths with guaranteed Quality of Service (QoS), so as to guarantee the data transfer rates and increase the reliability of the network between pairs of end-sites for given time periods. The VNOD project [5] is a recent effort towards establishing virtual network instances comprising multiple end-to-end virtual paths to accommodate data transfers between multiple pairs of source and destination end-sites.

*The author was with the Computational Science Center at Brookhaven National Laboratory when this work was performed.

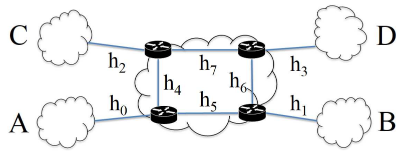


Fig. 1. Separate vs. joint optimization example.

In the context of such virtual network instances, we seek, in this paper, to develop an intelligent resource reservation algorithm to efficiently allocate available resources within wide area networks that support bandwidth reservations (BRs). We assume that users who want to transfer large amounts of data are primarily concerned with the end time by which their data transfers can finish. Therefore, the rate at which these data transfers progress is secondary, as long as deadlines can be met. We consider requests for such time-bound data transfers as flexible, because we have the flexibility to adjust when a data transfer can start and/or finish based on the transfer rates we can achieve. In order to accommodate such flexible data transfer requests, we need to construct routes that (i) interconnect the source and destination end-sites of each data transfer, and (ii) have adequate bandwidth[1] available so that the data transfer deadlines can be met. In recent work, we have developed algorithms that seek routing and resource reservation solutions between a *single* pair of end-sites [6], [7]. However, data often have to be distributed from a certain site to several recipient sites or exchanged between multiple pairs of sites.The solutions [6], [7] developed for a single pair of end-sites cannot be efficiently extended when multiple pairs of end-sites request resources at the same time. In such a multiple end-site pair scenario, resource utilization can be significantly improved if the routing and scheduling of flexible resource reservation requests is performed jointly, i.e., taking into account the whole picture instead of focusing individually on each end-site pair.

The following simple example demonstrates the concept and explains our motivation: consider four end sites connected via a WAN as shown in Fig. 1. Furthermore, assume that the bandwidth availability along all eight hops is the same and as shown in Fig. 2: bandwidth of 5 Gb/sec is available from 0 to 500 seconds and of 10 Gb/sec from 500 to 1000 seconds. Let's

[1]In this paper, we consider bandwidth as the reservable resource and use the terms “bandwidth” and “resource” interchangeably.

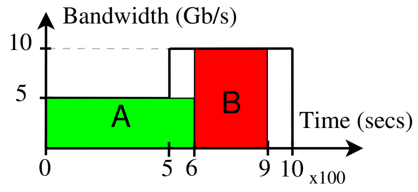


Fig. 2. Bandwidth availability for each hop shown in Fig. 1.

also assume that 3 Tb of data become available for transfer from end-site A and end-site B at time 0. The deadline to finish the data transfer from A to D is by time 1000, and from C to B by time 600. Irrespectively of the routes chosen to interconnect the end-site pairs, there will be at least one hop shared by both transfers. If scheduling is done separately (as opposed to jointly), it is possible that pair-1 may be scheduled first (e.g., according to some random selection scheme), and 5 Gb/sec bandwidth may be reserved for it for the first 600 seconds (box A). As a result, trying to schedule the second pair will fail because the transfer deadline for that pair is by time 600, but all available bandwidth for the first 600 seconds will have been already reserved for pair-1. However, if the two transfers were to be scheduled jointly and the algorithm took this potential exhaustion of bandwidth into consideration, it could eventually find a solution satisfying the deadlines of both transfers, e.g., reserve 5 Gb/sec for pair-2 for the first 600 seconds (box A) and 10 Gb/sec for pair-1 for 300 seconds starting at time 600 (box B).

The scheduling problem becomes more complex if the construction of the route interconnecting an end-site pair is also taken into account. Furthermore, consider the challenge of performing route construction and scheduling multiple data transfer requests, each with its own deadline, between multiple end-site pairs. Given these factors, our goal in this paper is to design an algorithm that performs joint optimization of route construction and bandwidth reservation for multiple pairs of end-sites in order to maximize the number of satisfied requests and minimize the total data transfer times. The rest of the paper is organized as follows: in section II, we formally define our problem and show that it is NP-hard; in section III, we develop an efficient heuristics, called RRM, to solve the problem; in section IV, we present simulation results to demonstrate the function of our heuristics; in section V, we present related work, and, finally, in section VI we summarize the presented work.

II. Problem Description

We consider multiple end sites connected to a single wide area network domain (multiple WAN domains can be combined and treated as a single domain). We model this network as a graph $G\langle\mathcal{V},\mathcal{H},\mathcal{B}\rangle$, where $\mathcal{V}$ is the set of all nodes, in which each end site and intermediate router corresponds to one node; $\mathcal{H}$ is the set of all hops; and $\mathcal{B}$ is the set of Bandwidth Availability Graphs (BAGs) for all the edges in $\mathcal{H}$, $|\mathcal{H}| = |\mathcal{B}|$. Each $BAG_{h_i} \in \mathcal{B}$ describes the time-varying bandwidth availability of hop $h_i \in \mathcal{H}$. The BAG can be represented as a step function with multiple steps $step_i$ defined as $\{start_i, end_i, bw_i\}$, where $start_i$ is the start time of $step_i$, end_i its end time, and bw_i the available bandwidth between $start_i$ and end_i. Fig. 1 shows an example of a graph G with 8 nodes and 8 hops. The BAG of all eight hops is the same and

TABLE I. Example input requests.

Sou--rce	Destin--ation	Vol. (Tb)	Start Time (sec)	Deadline (sec)	Rate Limit (Gb/s)
A	B	1	0	500	5
		1.2	0	900	2
B	C	1	500	800	5
A	D	0.8	600	1000	4

TABLE II. Example output.

Sou--rce	Destin--ation	Path	Start (sec)	End (sec)	Reserved BW (Gb/s)
A	B	$\{h_0, h_5, h_1\}$	0	200	5
			200	600	2
B	C	$\{h_1, h_6, h_7, h_2\}$	500	700	5
A	D	$\{h_0, h_4, h_7, h_3\}$	600	800	4

as shown in Fig. 2. This BAG has two steps: {0, 300, 1 Gb/s} and {300, 600, 2 Gb/s}. Note that obtaining such bandwidth availability is currently possible through recent extensions to OSCARS [3] developed by the ARCHSTONE [8] project of Internet2 [2].

Data transfers take place between certain pairs of end-sites. For each such pair, there exists a set of flexible data transfer requests. We denote the set of such pairs as $\mathcal{P}$. Each end-site pair $p_i \in \mathcal{P}$ is defined as $\{src_i, dst_i, \mathcal{R}_i\}$, where src_i is the source end site; dst_i is the destination end site; and $\mathcal{R}_i$ is the set of data transfer requests between src_i and dst_i. Each request $r_{ij} \in \mathcal{R}_i$ is defined as $\{start_{ij}, dead_{ij}, vol_{ij}, maxBW_{ij}\}$, where $start_{ij}$ is the start time when the data will become available to be transferred; $dead_{ij}$ is the deadline by which the data transfer to the destination end site should finish; vol_{ij} is the amount of data that needs to be transferred, and $maxBW_{ij}$ is the maximum achievable data transfer rate imposed by the end sites, for example, this rate could reflect the limits on the read/write speeds of the end-site storage systems. As an example, Table I shows four flexible requests that belongs to three different pairs.

Objective: Our goal is to jointly construct the paths (one path between every pair of sites) as well as the schedule of bandwidth reservations (BRs) along the hops of constructed paths. BRs along such paths should satisfy the data transfer deadlines for the submitted flexible requests. It is, however, not always possible to construct paths and reserve bandwidth so as to satisfy all the given requests. As such, our objective is *to develop an algorithm that can construct the paths and bandwidth reservation schedules in a manner that maximizes the number of satisfied requests while minimizing the total data transfer time*. We call this the PBM problem (**P**ath construction and **B**andwidth reservation for data transfer among **M**ultiple pairs of end-sites). For every pair $p_i \in \mathcal{P}$, the output can be represented as $\{path_i, \mathcal{BR}_i\}$, where $path_i$ is the route selected to transfer data between src_i and dst_i, and $\mathcal{BR}_i$ contains the BRs of all satisfied requests in $\mathcal{R}_i$. Each BR $br_{ij} \in \mathcal{BR}_i$ is defined as $\{st_{ij}, et_{ij}, bw_{ij}\}$, where st_{ij} is the beginning time; et_{ij} is the end time; bw_{ij} is the amount of bandwidth to be reserved between st_i and et_i. Table II shows an example of output for the input described in Fig. 1, Fig. 2 and Table. I.

Note that we do not consider multi-path solutions between a single pair of end sites. The reasons for this are technical rather than theoretical. In our context, bandwidth reservations are implemented in the form of virtual end-to-end paths. A virtual path comprises a guaranteed-bandwidth path segment in each end site from one or more host nodes to the site's border router and a WAN dynamic virtual circuit that interconnects these border routers. Each border router is configured to allocate a specific VLAN for the circuit and to forward selected traffic with Policy Based Routing (PBR), therefore, each circuit requires one VLAN identifier (VLAN ID) at each end site. Due to hardware and policy restrictions, the number of VLAN IDs that can be used in practice at each end site is limited to much smaller numbers than the 4096 VLAN IDs provided for by the 802.1Q standard. In our system model, given that many users may want to reserve resources within the WAN, and each reserved path requires allocating at least one VLAN tag, there is a strong incentive to minimize VLAN utilization. We, therefore limit a set of requests between two sites to use only one path (to minimize VLAN tag usage). Note, however, that different pairs of sites can still use different paths. Similarly, for technical reasons, we consider that the path and reserved bandwidth for each request remains constant throughout the reserved time period. Allowing the bandwidth or path for individual requests to change over time requires updating the configurations of network devices along that path, a time-consuming operation, During such an operation, the affected traffic may have to fall back to the default best effort service, which would violate the QoS guarantees.

A. Problem Complexity

We use an existing NP-hard hard problem, called SMR^3 [6], to prove the NP-hardness of our problem.

Theorem 1: PBM is an NP-hard problem.

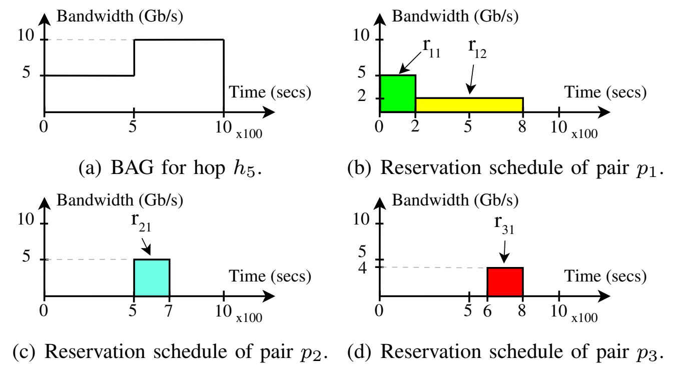
Proof: Sharma *et al.* [6] has shown that SMR^3 is an NP-hard problem. The SMR^3 problem considers a single Bandwidth Availability Graph (BAG) for a single given path and a set of requests that need to be accommodated within the BAG. An example of a BAG can is shown in Fig. 2. The objective of SMR^3 is to create a bandwidth reservation schedule for the requests in order to maximize the number of accommodated requests while minimizing the total data transfer time (same as our objective in this paper). The SMR^3 problem is a special case of PBM when there is only one pair of end sites between which multiple data transfer requests are to be accommodated, and where there is exactly one possible path between the two end sites. Since a special case of PBM is NP-hard, the PBM problem is at least NP-hard. ■

III. RRM: Resource Reservation and Route Construction for Multiple End-Site Pairs

A. Algorithm Description

We develop an algorithm, called RRM, to solve the problem described in the previous section. The pesudocode of RRM is shown in Fig. 6. The RRM algorithm runs in iterations. The goal of each iteration is to selectively obtain a solution for one pair of end sites. The number of iterations is equal to the number of pairs in the submitted requests. In each iteration, there are two phases as follows:

Phase I: During the first phase, the goal for each pair is to construct a route from source to destination and determine a bandwidth schedule on that route *independently* of other



(a) BAG for hop h_5. (b) Reservation schedule of pair p_1. (c) Reservation schedule of pair p_2. (d) Reservation schedule of pair p_3.

Fig. 3. Example of avaialble and reserved bandwidth of hop h_5.

pairs. In other words, for each pair, the RRM algorithm calculates the route and bandwidth schedule based on current bandwidth availability of each hop and without considering possible schedule of other pairs. To calculate such a route and reservation schedule,we use the RRPC heuristic [7] for each pair of end sites. This heuristic can calculate a route and a reservation schedule for a single pair of end sites. The goal is maximizing the total number of reservation requests for that pair while minimizing the total data transfer time. RRPC relies on a modified Dijkstra's algorithm for route construction and on another algorithm, called RRA [6], for calculating the resource reservation schedule along the route between source and destination. Note, that although we use RRPC as a building block for the RRM algorithm in this phase, RRPC is not a mandatory part of RRM. We could plug in any heuristic that calculates routes and reservation schedules for individual end-site pairs.

As an example, given the input described in Fig. 1, Fig. 2 and Table. I, the RRM algorithm generates the following paths and bandwidth reservations: for pair p_1 (A and B), $path_1$ is $\{h_0, h_5, h_1\}$ and $\mathcal{BR}_1$ is shown in Fig. 3(b); for pair p_2 (B and C), $path_2$ is $\{h_1, h_5, h_4, h_2\}$ and $\mathcal{BR}_2$ is shown in Fig. 3(c); for pair p_3 (A and D), $path_3$ is $\{h_0, h_5, h_6, h_3\}$ and $\mathcal{BR}_3$ is shown in Fig. 3(d). Note that all paths contain the hop h_5.

Phase II. In this phase, the RRM algorithm checks the feasibility of paths and schedules constructed in the previous phase. That is, the RRM algorithm checks if there is enough bandwidth available on each hop to accommodate the reservations requested by *all* pairs of end-sites. The RRM algorithm uses the concept of *stress factor* for every hop to indicate such feasibility.

Stress Factor: Each pair of end-sites that was considered during the first phase has specific bandwidth reservation requirements. These requirements are represented by a Bandwidth Reservation Graph (BRG) as bandwidth value vs. time. For each network hop, the sum of such time-varying bandwidth reservation values can be represented as a consolidated BRG. For the previous example, the consolidated BRG for the hop h_5 whose BAG is shown in Fig. 3(a) depends on the BRGs in Fig. 3(b), Fig. 3(c), and Fig. 3(d). This consolidated BRG is shown in Fig. 4. The BRG of a hop is subtracted from the BAG of that hop to obtain a *residual* BAG. This residual BAG can be used to determine the time intervals for which the total bandwidth requested by the end site pairs on that hop exceeds the available bandwidth. The residual BAG of h_5, shown in

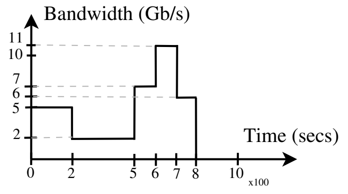


Fig. 4. Consolidated BRG for reservations in Fig.3(b), Fig.3(c), and Fig.3(d).

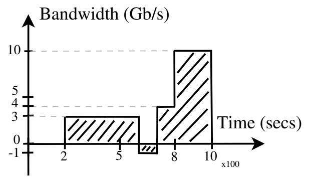


Fig. 5. Residual BAG (Shaded area) of hop h_5.

Fig. 5, is obtained by subtracting Fig. 4 from Fig. 3(a). Because for intervals $[6, 7]$ the requested bandwidth exceeds what is actually available, the residual BAG shows negative values. The RRM algorithm calculates the stress factor of the hop as equal to the area under the residual BAG for which bandwidth values are negative. If the residual BAG does not have any negative bandwidth values, the stress factor then equals to 0. It also means that the available bandwidth of this hop is large enough to accommodate all bandwidth reservations scheduled on it. In Fig. 5, the stress factor of h_5 can be calculated as equal to 1.

The RRM algorithm first calculates the stress factor for each hop, and then identifies the hop with the largest stress factor. If the largest stress factor is 0, then the RRM algorithm stops because all bandwidth requests for all pairs can be satisfied. However, if the largest stress factor is greater than 0, RRM subsequently identifies the end-site pairs which use the hop with the largest stress factor. Out of all identified pairs, RRM selects the pair that has accommodated the largest number of requests in this iteration. This pair gets the route and bandwidth schedule that was calculated for it during phase I. For the path assigned to the selected pair, RRM updates the BAG for each hop by subtracting the bandwidth assigned to that pair. The updated (reduced) bandwidth along these hops is treated as the available bandwidth for the next iteration of the RRM algorithm.

Once RRM finalizes the path and bandwidth reservation schedule for the selected pair, the algorithm does not consider the same pair again in future iterations. The next iteration executes with one less pair and reduced bandwidth availability along certain hops. Fig. 6 shows the pseudocode for RRM.

For the previous example, after calculating the stress factor for each hop, the RRM algorithm identifies hop h_5 as the hop with the largest stress factor (equals to 1). Since all the three pairs use h_5, RRM then selects the pair p_1 which has accommodated the largest number of requests. In the next, RRM finalizes the path $path_1$ and bandwidth schedules $\mathcal{BR}_1$ constructed for p_1 and starts its next iteration. In the second iteration, RRM finalizes the pair p_2 with path $\{h_1, h_6, h_7, h_2\}$ and $\mathcal{BR}_2$ shown in Fig. 3(c). In the last iteration, RRM finalizes the pair p_3 with path $\{h_0, h_4, h_7, h_3\}$ and $\mathcal{BR}_3$ shown

```
RRM Algorithm ($G\langle\mathcal{V},\mathcal{H},\mathcal{B}\rangle$, $\mathcal{P}$)
 1. while $true$ do
 2.     for each $p_i \in \mathcal{P}$ do
 3.         Calculate $path_i$ and $\mathcal{BR}_i$ by using RRPC [7].
 4.     end for
 5.     for each $h_i \in \mathcal{H}$ do
 6.         Calculate the consolidated BRG.
 7.         Calculate the residual BAG.
 8.         Calculate $StressFactor_{h_i}$.
 9.     end for
10.     if all $StressFactor_{h_i} == 0$ then break
11.     $h_{MAX} \leftarrow argmax_{h_i \in \mathcal{H}}\{StressFactorh_i\}$.
12.     $p_t \leftarrow argmax_{p_i \in \mathcal{P}}\{|\mathcal{BR}_i| \mid h_i \in path_i\}$.
13.     Finalize $path_t$ and $\mathcal{BR}_t$.
14.     for each $h_i \in path_t$ do
15.         Reduce $BAG_{h_i}$ by $BRG_{p_t}$.
16.     end for
17.     Remove $p_t$ from $\mathcal{P}$.
18.     if $\mathcal{P} == \{\}$ then break
19. end while
```

Fig. 6. Pseudocode for the RRM algorithm.

in Fig. 3(d). Note that none of the hops in the network is used by all three pairs now. As a result, although the three pairs maintain their original bandwidth reservation schedule, none of the hops has now a stress factor larger than 0.

B. Runtime Complexity

In this section, we analyze the worst case runtime complexity of the RRM algorithm. Let the number of hops in the network be $|\mathcal{H}|$ and the number of end-site pairs be $|\mathcal{P}|$. The RRM algorithm finalizes the route and bandwidth schedule of one pair during each iteration. As a result, in the worst case, the number of iterations in the RRM algorithm will be equal to the number of end-site pairs (i.e., $|\mathcal{P}|$).

During every iteration, the RRM algorithm performs phase I and phase II. In phase I, RRM executes the RRPC algorithm for all pairs of end sites. Runtime complexity of the RRPC algorithm (for a single pair j) is given in [7], and can be written as $O\left(|\mathcal{H}| \cdot \left[N_j^3 + N_j^2 \cdot \sum_{b_i \in \mathcal{B}} M_{b_i}\right]\right)$, where N_j is the number of requests between the j^{th} pair of end sites, $\mathcal{B}$ is the set of BAGs along all hops in the network, and M_{b_i} is the number of steps in BAG $b_i \in \mathcal{B}$. As a result, the total complexity of phase I is $O\left(|\mathcal{H}| \cdot \sum_{j=1}^{|\mathcal{P}|} \left[N_j^3 + N_j^2 \cdot \sum_{b_i \in \mathcal{B}} M_{b_i}\right]\right)$.

In phase II, the first step is to calculate the stress factor for every hop in the network. For each hop, the cost of calculating the BRG in the worst case is $O(\sum_{j=1}^{|\mathcal{S}|} N_j)$. The cost of comparing a BRG with a BAG b_i is equal to the sum of the number of steps in the BAG and the BRG, which is $O\left(\sum_{b_i \in \mathcal{B}} [M_{b_i} + \sum_{j=1}^{|\mathcal{P}|} N_j]\right)$. The RRM algorithm then calculates the stress factor of each hop by using the consolidated BRG and the BAG for that hop. In the worst case, the cost can be expressed as $O\left(\sum_{b_i \in \mathcal{B}} \left[M_{b_i} + \sum_{j=1}^{|\mathcal{P}|} N_j\right]\right)$. In the next, the RRM algorithm identifies the most stressed hop and selects the end-site pair hat is using that hop and has the largest number of accommodated requests. The cost of this operation is $O(|\mathcal{H}| + |\mathcal{P}|)$. After assigning a route and a reservation schedule to the selected pair, the algorithm updates the BAGs for all the hops along that particular route. The cost of this step is $O\left(|\mathcal{H}| \cdot (M_{b_i} + N_j)\right)$.

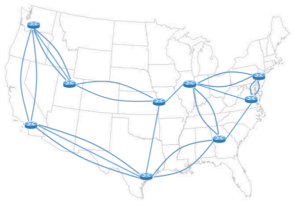


Fig. 7. Network topology of Internet2.

After integrating the complexity of each step together, the total complexity of the RRM algorithm can be written as:

$$O\left(|\mathcal{P}|\cdot\left\{|\mathcal{H}|\cdot\sum_{j=1}^{|\mathcal{P}|}\left[N_j^3+N_j^2\cdot\sum_{b_i\in\mathcal{B}}M_{b_i}\right]+\sum_{b_i\in\mathcal{B}}\left[M_{b_i}+\sum_{j=1}^{|\mathcal{P}|}N_j\right]\right\}+\sum_{j=1}^{|\mathcal{P}|}\left[|\mathcal{P}|+|\mathcal{H}|\cdot(M_{b_i}+N_j)\right]\right) \quad (1)$$

The above expression can be simplified into the following:

$$O\left(|\mathcal{H}|\cdot|\mathcal{B}|\cdot|\mathcal{P}|^2\cdot N_{\max}^3\cdot M_{\max}\right) \quad (2)$$

where

$$N_{\max}=\max(N_j), j\in[1,|\mathcal{S}|] \quad (3)$$

and

$$M_{\max}=\max(M_{b_i}), b_i\in\mathcal{B} \quad (4)$$

Note that even in large WANs (e.g., Internet2 shown in Fig. 7), the value of $|\mathcal{H}|$ (number of hops), $|\mathcal{B}|$ (number of BAGs), and $|\mathcal{S}|$ (end-site pairs) is usually much smaller than N (number of input pairs) or M (number of steps in a BAG). As such, $N_{\max}^3$ usually dominates the complexity of the RRM algorithm.

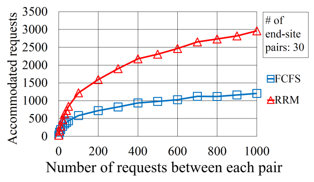
IV. Simulation Results

In this section, we demonstrate the performance of the RRM algorithm using simulations.

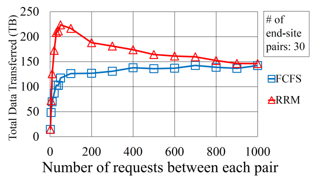
A. *Simulation Setup*

Instead of generating a random network, we consider the actual topology of Internet2 as the topology of the network on which we are going to test our algorithms. The topology of Internet2 is available from [10] in xml format. Figure 7 shows the topology of Internet2. It consists of 9 routers. Routers are connected to other routers through one or more direct links. Typically, end-sites (e.g., educational institutions) are connected to the Internet2 routers through regional provider networks utilizing links of dedicated capacity. For simplicity, however, in our simulations we consider that these routers represent individual end sites.

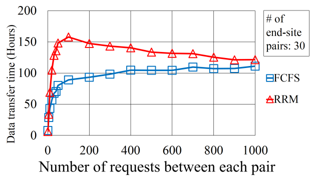
To the best of our knowledge, no algorithm exists currently in the literature that can be used to solve the same or similar problems. Therefore, we compare RRM to a simple heuristic algorithm, called First Come First Serve (FCFS). FCFS represents the typical method that resource reservations take place in a network, which does not have a system that can schedule reservations with optimization strategies. The FCFS algorithm processes data transfer requests in the order of their submission



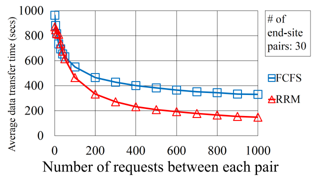
(a) Total accommodated requests.



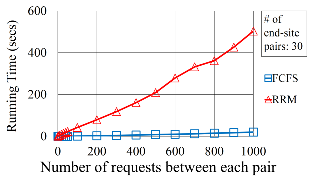
(b) Total data transferred.



(c) Total transfer time.



(d) Average transfer time.



(e) Algorithm runtime comparison.

Fig. 8. Performance of RRM with increasing number of request per pair.

time. Once a data transfer request arrives, the FCFS algorithm calculates the BAG of the default routing path between the two end-sites specified in the request, and then tries to reserve bandwidth for the request, in a manner that minimizes the transfer duration. In our simulation, we set the default routing path between two end sites to be the shortest path. If a suitable reservation slot with sufficient bandwidth cannot be found, FCFS rejects the request. In our simulations, each data transfer request is generated in random. The file size of each request is between 0 and 35 TB. The rate limit is between 0 and 10 Gb/s. Furthermore, we generate a random submission time for each request with the restriction that it be earlier than the earliest start time for that request. This submission time is used by the FCFS algorithm.

B. *Performance with Increasing Number of Requests per End-Site Pair*

In this simulation, we study the performance of the RRM algorithm as the number of requests per pair increases from 1 to 1000. We present the results for 30 random pairs of end-sites. All results in this section were obtained with the exact same set of end-site pairs. For every hop in the network, we generate a multi-step BAG in which the available bandwidth of each step is randomly selected between 0 and 10 Gb/s. Every data point in our simulation is an average of 50 simulation runs.

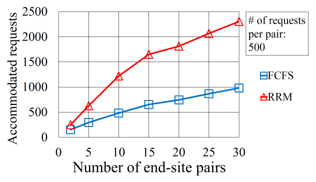
Figure 8(a) shows the number of accommodated requests for the selected range of requests. The performance of the RRM is nearly 150% superior to that of the FCFS algorithm. We attribute this to our algorithms having the flexibility select the hops of a path for each end-site pair. With this strategy, RRM may route requests of different end-site pairs to different paths, and thereby alleviate the occurrence of congestion. Figure 8(b) shows the total amount of data for the accommodated requests. The RRM algorithm transfers 20%-80% more data than the FCFS algorithm. There is an interesting observation one can make based on Fig. 8(b): even though the number of accommodated requests increases (Fig. 8(a)), the total data transferred actually decreases slightly for RRM algorithm. This happens for the following reasons: first, the maximum amount of data that can be transferred given a certain bandwidth availability is constant; second, RRM tries to accommodate as many requests as possible while fully utilizing the available bandwidth; finally, given a certain bandwidth availability, as the number of accommodated requests increases, the fragmentation of available bandwidth also increases. This results in much more wasted bandwidth and lower amounts of total data transferred. Also, note that maximization of total data transferred is not one of our objectives.

Due to these reasons, Fig. 8(c) shows that the total sum of times required to transfer the data for accommodated requests also decreases for RRM algorithm. As we can see, the total data transfer time of RRM is longer than that of FCFS. The reason is that the RRM algorithm accommodates many more requests. Fig. 8(d) shows the average data transfer time of requests. We define as average transfer time the total data transfer time divided by the number of accommodated requests. Although the RRM algorithm has longer total data transfer time, Fig. 8(d) shows that the average data transfer time of RRM is nearly 50% shorter than that of FCFS. Finally, Fig. 8(e) shows the actual running time of the three algorithms on a 2.7 GHz processor. The running time required by FCFS minimally increases with increasing numbers of requests and/or end-sites due to the simplicity of the heuristic. However, the runtimes of RRM are in the order of several seconds to a few minutes and are negligible compared to the actual data transfer times shown in Fig. 8(c). For example, the longest running time shown this figure (for 30 pairs and 1000 requests) is approximately 12 minutes corresponding to a solution for a set of data transfers that takes approximately 130 hours (0.15% of the total duration). For 100 requests, the algorithm takes just 50 seconds for a solution corresponding to 160 hour-long data transfers. This indicates the efficiency of our algorithms.

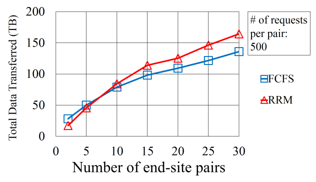
Summarizing, in this simulation, the RRM algorithm exhibits a significant advantage over the baseline FCFS algorithm by accommodating nearly 160% more requests and achieving approximately 50% shorter average data transfer times. Additionally, RRM manages to transfer up to 80% more data than the baseline, although this is not explicitly pursued by its objectives.

C. Performance with Increasing Number of End-Site Pairs

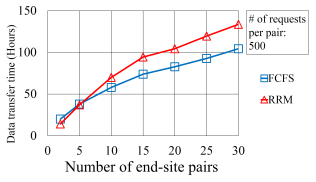
We now study the performance of RRM when the number of end-site pairs increases from 2 to 30. We present the results obtained for a number of 500 requests for each pair. These results are in agreement with those obtained for increasing number of requests per end-site pair, but from a different perspective. Fig. 9(a) shows the number of accommodated requests as the number of pairs increases. RRM accommodates about 150% more requests than FCFS. In Fig. 9(a), the difference between RRM and FCFS is increasing as the number of pairs increases. This is because the flexibility of selecting paths is increasing along with the increasing of number of pairs. This higher flexibility can better alleviate competition for bandwidth, and thereby accommodate more requests. Fig. 9(b) shows the amount of data transferred. We



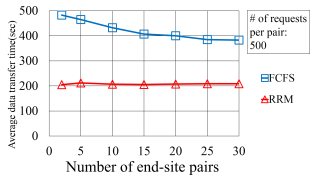
(a) Total accommodated requests.



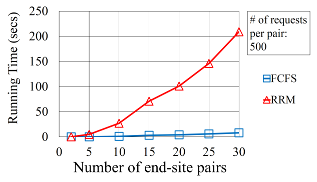
(b) Total data transferred.



(c) Total transfer time.



(d) Average transfer time.



(e) Algorithm runtime comparison.

Fig. 9. Performance of RRM with increasing number of end-site pairs.

can see that the trend in Fig. 9(b) (for constant number of requests and increasing number of pairs) is different than the trend shown in Fig. 8(b) for constant number of pairs and increasing number of requests. In Fig. 9(b), when the number of pairs is small, the amount of data transferred is also small compared to Fig. 8(b) where the amount of data peaks. The reason for this is our problem setting. Due to the use of circuits, we are restricted to selecting a single path between a single pair of end sites (see Section II). As a result, the requests between 30 different end-site pairs corresponding to Fig. 8(a), may follow up to 30 different paths. However, for smaller numbers of end-site pairs, the same number of accommodated requests has to follow a correspondingly smaller number of paths, thus the overall available bandwidth is also smaller. Fig. 9(c) shows the sum of data transfer times for individual requests and Fig. 9(d) shows that the average data transfer time of RRM is about 50% shorter than that of FCFS. We can see that the average data transfer time of RRM is about 50% shorter than that of FCFS. Fig. 9(e) shows the running time of the algorithms on a 2.7 GHz processor. Again, the running time of RRM is shown to be negligible compared to the duration of the data transfers.

V. Related Work

A significant amount of research on network QoS exists in the literature. A large portion of the existing research, however, is focused on providing QoS guarantees when users submit requests without any flexibility in terms of reservation times and the amount of resources they would like to reserve. Such QoS guarantees are usually achieved by developing QoS or constraint-based routing mechanisms [11]. In this section, we will focus on network QoS for "flexible" rather than "inflexible" resource reservation requests.

Sharma *et al.* [6] presents an algorithm, called RRA, to accommodate multiple flexible requests between a single pair of end sites along a pre-determined path. Sharma *et al.* [7]

continues the work of [6] with developing an algorithm, called RRPC, which can calculate a routing solution as well as accommodate flexible requests between a single pair of end sites. In contrast to the work presented in this paper, the RRA and RRPC algorithms cannot be used to perform *joint* bandwidth reservation and route construction between multiple pairs of end sites.

Balman *et al.* [12] solves the problem of scheduling a single flexible reservation with an objective of optimizing the completion time or transfer duration. In contrast to this work, the RRM algorithm schedule reservations for multiple requests between multiple sources and destinations.

Gu *et al.* [13] considers accommodating requests,in the order that they arrive, between a single pair of end sites. Such a first-come-first-serve mechanism of accommodating requests has been shown to perform worse than the RRA algorithm [6]. Furthermore, in contrast to the RRM algorithm, the work in [13] cannot handle resource reservations when the route construction is part of the problem.

In [14], multiple requests between a single pair of end sites are again processed sequentially in the order of their arrival. In [15], the scheduling of bandwidth reservation is considered as an optimization problem that can answer the question if all requests can be satisfied or not. For a given set of requests, the solution procedure in [15] rejects all requests following a particular request that cannot be satisfied. This procedure may lead to lower performance and rejection of subsequent requests that could otherwise be accommodated. The RRPC algorithm, which is used in phase I of the RRM algorithm in the paper, does not suffer from such drawback as it selectively rejects the requests that cannot be accommodated and accepts those that can be.

Ghosh *et al.* [16] presents a distributed algorithm that performs QoS routing in IP networks. Norder *et al.* [17] considers the QoS routing problem at the inter-domain level, and reviews the critical issues that need to be considered when designing new QoS routing protocols. The QoS routing aims at selecting a specific path that satisfies a set of QoS requirements for a data flow. It cannot, however, reserve (guarantee) resources for data flows. Our approach in this paper is quite different because we assume the utilization of recently developed mechanisms to schedule and enforce end-to-end reservations of network resources through the establishment of virtual end-to-end paths with guaranteed QoS.

VI. Conclusion

In this paper, we studied the problem of scheduling bandwidth reservations on various hops within a wide area network for efficient data transfers between multiple pairs of end-sites. We proved that this problem is NP-hard and developed a joint routing and bandwidth scheduling algorithm with the objective to maximize the number of satisfied data transfer requests while minimizing the total data transfer times. Furthermore, we calculated the time complexity of this algorithm and compared its performance, through simulations, with a baseline FCFS algorithm. Our simulations indicate that our algorithm accommodates more requests than the baseline algorithm and achieves shorter average data transfer times. We have observed scenarios where our algorithm accommodates up to 160% more requests and achieves up to 50% shorter average data transfer times than the baseline FCFS algorithm, while transferring up to 80% more data. Furthermore, the running time of our algorithm is low – practically negligible – when compared with the overall duration of the data transfer requests submitted for scheduling.

References

[1] “Energy sciences network,” http://www.es.net/

[2] “Internet 2,” http://www.internet2.edu/

[3] “OSCARS: On-Demand Secure Circuits and Advance Reservation System,” http://code.google.co-m/p/oscars-idc/

[4] D. Katramatos, D. Yu, K. Shroff, S. McKee, and T. Robertazzi, “TeraPaths: End-to-end network resource scheduling in high-impact network domains,” *International Journal On Advances in Internet Technology,* vol 3, no. 1–2, pp. 104–117, 2010.

[5] “VNOD: Virtual Network On Demand,” https://wiki.bnl.gov/CSC/index.php/VNOD

[6] S. Sharma, D. Katramatos, and D. Yu, “End-to-end network QoS via scheduling of flexible resource reservation requests,” In *Proc. ACM/IEEE Supercomputing Conference (SC),* Seattle, WA, November 12–18, 2011.

[7] S. Sharma, D. Katramatos, D. Yu, and L. Shi “Design and implementation of an intelligent end-to-end network QoS system,” In *Proc. ACM/IEEE Supercomputing Conference (SC),* Salt Lake City, UT, November 10–16, 2012.

[8] “ARCHSTONE (Advanced Resource Computation for Hybrid Service and TOpology NEtworks) Project,” http://archstone.east.isi.edu/

[9] C. Guok, D. Robertson, E. Chaniotakis, M. Thompson, W. Johnston, and B. Tierney “A User Driven Dynamic Circuit Network Implementation,” In *Proc. IEEE Distributed Autonomous Network Management Systems (DANMS) Workshop,* New Orleans, LA, November 30, 2008.

[10] “Internet 2–Topology,” http://patdev1.internet2.edu/TopologyViewer/

[11] O. Younis and S. Fahmy, “Constraint-based routing in the internet: Basic principles and recent research,” *IEEE Communications Surveys & Tutorials,* vol. 5, no. 1, pp. 2–13, 2003.

[12] M. Balman, E. Chaniotakis, A. Shoshani, and A. Sim, “A flexible reservation algorithm for advance network provisioning,” In *Proc. ACM/IEEE Supercomputing Conference (SC),* Seattle, WA, November 12–18, 2011.

[13] J. Gu, D. Katramatos, X. Liu, V. Natarajan, A. Shoshani, A. Sim, D. Yu, S. Bradley, and S. McKee, “StorNet: Co-scheduling of end-to-end bandwidth reservation on storage and network systems for high-performance data transfers,” in *Proc. IEEE INFOCOM, Workshop on High-Speed Networks,* Shanghai, China, April 10–15, 2011.

[14] S. Naiksatam and S. Figueira, “Elastic reservations for efficient bandwidth utilization in LambdaGrids,” *The International Journal of Grid Computing*, vol. 23, no. 1, pp. 1–22, January 2007.

[15] K. Rajah, S. Ranka, and Y. Xia, “Advance reservation and scheduling for bulk transfers in research networks,” *IEEE Transactions on Parallel and Distributed Systems.* vol. 20, no. 11, pp. 1682–1697, 2009.

[16] D. Ghosh, V. Sarangan, and R. Acharya, “Quality-of-service routing in IP networks,” In *IEEE Transactions on Multimedia,* vol. 3, no. 2, pp. 200-208, 2001.

[17] S. Norden, and J. Turner, “Inter-domain QoS routing algorithms,” Washington University, Department of Computer Science Technical Report, WUCS-02-03 (2002): 199-215.