On Fixed-Path Variable-Bandwidth Scheduling in High-performance Networks

Liudong Zuo, Mustafa Khaleel and Michelle M. Zhu
Department of Computer Science
Southern Illinois University
Carbondale, IL 62901, USA
Email: liudongzuo@siu.edu; mustafa@siu.edu; mzhu@cs.siu.edu

Chase Qishi Wu
Department of Computer Science
University of Memphis
Memphis, TN 38152, USA
Email: qishiwu@memphis.edu

Abstract—Many extreme-scale scientific applications are distributed in nature and oftentimes need to move vast amounts of data between multiple locations for various remote operations. Such applications require fast and reliable data transfer services with guaranteed finish time, which can be realized by making bandwidth reservation for dedicated channel provisioning in high-performance networks. From an individual perspective, a user always wishes to achieve the earliest finish time for a given data transfer request that typically specifies the maximum Local Area Network (LAN) bandwidth constraint, the data size, data available time, and the deadline. From a global perspective, the network service provider would attempt to serve as many users’ bandwidth reservation requests (BRRs) as possible to maximize the overall network resource utilization and throughput. As for the high-performance network consuming vast amounts energy and network resources, the network service provider would also want to process BRRs efficiently to save energy, network resource and the network maintenance cost. These goals are potentially conflicting and require a careful design of the bandwidth reservation and scheduling algorithm. In this paper, we focus on one particular type of scheduling problem under the constraint of fixed path and varying bandwidth (FPVB), which has been proven to be NP-complete in the literature. We develop two heuristic algorithms, namely Least Available Bandwidth of Edge (LABE) and Largest Available Bandwidth of Path (LABP), to solve this problem and conduct simulation-based performance evaluation. The extensive simulation results illustrate the superiority of these proposed algorithms in terms of execution time, success ratio, and average data transfer completion time of BRRs in comparison with two existing scheduling algorithms.

I. INTRODUCTION

Extreme-scale computations in various scientific applications are generating a colossal amount of data with the use of high-performance computing facilities and advanced scientific instruments. The scale of such data is currently on the order of terabytes and is expected to reach petabytes or exabytes in the near future. The data is usually generated at one location and must be transferred to remote sites where collaborative scientific users reside for further data analysis and knowledge discovery [1]. If not moved in a timely manner, the data may become stale and useless, causing a tremendous waste of computing, networking, and storage resources. However, the default best-effort shared IP networks such as the Internet are not well suited for such big data transfer as they generally do not provide guaranteed and reliable transport performance. High-performance networks such as ESnet [2] and Internet2 [3] that are capable of provisioning dedicated channels through advance bandwidth reservation have emerged as a promising solution to the huge data transfer needs in many mission-critical scientific applications. For example, On-Demand Secure Circuits and Advance Reservation System (OSCARS), a representative bandwidth reservation service, has been widely adopted and deployed in many network environments including ESnet, Internet2, GÉANT AutoBAHN [4], USLHCNet [5], and several other organizations for cross-domain on-demand bandwidth reservation.

In high-performance networks providing the bandwidth reservation service, the control plane is responsible for scheduling and making bandwidth reservation for users’ BRRs, if users’ BRRs finish their data transfer, the control plane would also release the reserved bandwidth for them [1]. The challenge of bandwidth reservation arises from the potentially conflicting requirements from the users, the network service provider and the whole high-performance network. From an individual perspective, a user always wishes to achieve the earliest data transfer finish time for a given BRR that typically specifies the maximum LAN bandwidth constraint, data size, data available time, and the deadline. From a global perspective, the network service provider would attempt to serve as many users’ BRRs as possible to maximize the overall network resource utilization and throughput. As for the whole high-performance network normally consuming vast amounts of energy and network resources, which would further bring more maintenance cost. So to reduce the energy and network resource needed to finish the data transfer can not only reduce the overall scheduling network cost, but also help maintain the stability of the whole scheduling network in the long run. These potentially conflicting goals require a careful design of the bandwidth reservation and scheduling algorithm. Furthermore, the rapidly increasing high-performance network size and number of users have posed an unprecedented challenge on the efficiency of the scheduling algorithm, especially under a heavy user request load. In other words, the response time of the scheduling algorithm must be fast enough to handle a large number of submitted jobs within a short period of time.

Wu and Lin [1] categorized four types of advance bandwidth scheduling problems with an exhaustive combination of different path and bandwidth constraints, i.e. (i) fixed path with fixed bandwidth (FPFB), (ii) fixed path with variable bandwidth (FPVB), (iii) variable path with fixed bandwidth (VPFB), and (iv) variable path with variable bandwidth (VPVB), with the same objective to minimize the data
CARS is the most widely used bandwidth reservation service ear leading to the shortest transfer duration. Results using the polynomial-time algorithm proposed in [9]: the deadline from the user, the scheduler yields two optional restraints, the earliest data transfer start time and the data transfer via periodic reoptimization [8]. Given a local bandwidth constraint, the earliest data transfer start time and the data transfer deadline from the user, the scheduler yields two optional results using the polynomial-time algorithm proposed in [9]: the earliest completion time and the shortest transfer duration. OS-CARS is the most widely used bandwidth reservation service in high-performance networks. Guok et al. describe the path computation, circuit setup and automated device configuration, authentication and authorization and other aspects of OSCARS in [10].

More recently, Wu and Lin investigated four types of scheduling problems with the objective to minimize the data transfer end time for a given BRR [1]. Since the FPVB problem is an NP-complete problem, they proposed two greedy algorithms, GreedyFPVB and MinFPVB. The essential strategy of GreedyFPVB is to use Dijkstra’s algorithm to obtain the narrowest path from the source node to the destination node. Here, the narrowest path is defined as the path on which the maximum weight among all component edges is minimized and the weight of each edge is computed with varying bandwidth among multiple time slots. For a single link, MinFPVB randomly chooses the available bandwidth of this link among multiple time slots and sets the selected bandwidth as its available bandwidth among all time slots. The random choice for choosing the available link bandwidth leads to inaccuracy and instability of this algorithm. The performance of GreedyFPVB and MinFPVB varies greatly from actual values in terms of success ratio in the presence of multiple BRRs and completion time of each individual BRR, which are the main performance aspects we aim to improve by proposing Labe and Labp for FPVB. The design of Labe is similar to GreedyFPVB with the difference that the weight of each edge is computed with the minimum available bandwidth of that edge among all time slots. Labp focuses on paths with the largest available bandwidth in each time slot and returns the reservation option on the path with the earliest completion time.

The rest of this paper is organized as follows. Section III presents the mathematical models and problem definition. Section IV provides the details of the algorithm design and complexity analysis. Performance evaluation is conducted in Section V. Section VI concludes our work with our futher plan.

II. RELATED WORK

With the rapid development of dedicated high-performance networks, many algorithms for advance bandwidth reservation have been studied and proposed in various contexts. We provide below a brief survey of related work on advance bandwidth reservation.

Sahni et al. investigated a path computation problem from a source node to a destination node in a network with different user data transfer requirements [6]: (i) a specified bandwidth in a specified time slot, (ii) the highest available bandwidth in a specified time slot, (iii) the earliest available time with a specified bandwidth and duration, and (iv) all available time slots with a specified bandwidth and duration. Among these four scheduling problems, the classical breadth-first search algorithm comes forth as an efficient solution to tackle the first and third problems. The second problem falls in the scope of problems solvable using Dijkstra’s shortest path algorithm. The solution to the fourth one is essentially a variant of Bellman-Ford algorithm.

Sahni et al. investigated a path computation problem from a source node to a destination node in a network with different user data transfer requirements [6]: (i) a specified bandwidth in a specified time slot, (ii) the highest available bandwidth in a specified time slot, (iii) the earliest available time with a specified bandwidth and duration, and (iv) all available time slots with a specified bandwidth and duration. Among these four scheduling problems, the classical breadth-first search algorithm comes forth as an efficient solution to tackle the first and third problems. The second problem falls in the scope of problems solvable using Dijkstra’s shortest path algorithm. The solution to the fourth one is essentially a variant of Bellman-Ford algorithm.

Scheduling multiple resource reservation requests is well recognized to be an NP-hard problem. Sharma, Katramatos and Yu proposed a polynomial-time heuristic resource reservation algorithm to accommodate as many reservation requests as possible while minimizing the total time to complete the data transfers [7]. To improve the network resource utilization and reduce the job rejection ratio, Rajah and Ranka studied the optimization problems on admission control and scheduling using a network controller and a cohesive optimization-based framework comprised of the following elements: advance reservations, multipath routing, and bandwidth reassignment via periodic reoptimization [8]. Given a local bandwidth constraint, the earliest data transfer start time and the data transfer deadline from the user, the scheduler yields two optional results using the polynomial-time algorithm proposed in [9]: the earliest completion time and the shortest transfer duration. OS-CARS is the most widely used bandwidth reservation service...
varying bandwidths across transfer completion time of $E$ at timepoint $v$ easy to see path the edge on a BRR, the goal is to find a bandwidth reservation scheme allowed to change during different timesteps, the data transfer across one timestep or possible varying bandwidths should start immediately upon the reception of the BRR [1].

$\{ t^S, t^E \}$. The reserved bandwidth set $B$ has a form: $\{ b[i \cdot t^S, j \cdot t^E] : 1 \leq i \leq j \leq N \}$, where $N$ is the number of timesteps within time interval $[t^S, t^E]$, and $b[i \cdot t^S, j \cdot t^E]$ represents the reserved bandwidth on path $P$ during $t^s_k$, $i \leq k < j$. Note that the last reserved bandwidth is not necessarily reserved on an entire timestep, and $t^E$ denotes the data transfer completion time, $t^S \leq t^E$. $B(p')$ denotes the bandwidth of path $p$ in the network during timestep $t^s_i$, and is limited by the bottleneck edge of $p$, namely, the edge on $p'$ with the minimum bandwidth during $t^s_i$.

As stated in previous sections, we focus on the FPVB problem defined as follows [1]: Given a network $G(V, E)$ and a BRR, the goal is to find a bandwidth reservation scheme using a fixed path from $v_s$ to $v_d$ with a fixed bandwidth across one timestep or possible varying bandwidths across multiple consecutive timesteps such that the transfer end time is minimized. Since the reserved bandwidth along a path is allowed to change during different timesteps, the data transfer should start immediately upon the receipt of the BRR [1]. So the form of the reserved bandwidth set $B$ should be: $\{ b[i \cdot t^S, j \cdot t^E] : 1 \leq i \leq j \leq N \}$.

We use a simple example to illustrate the FPVB problem in Fig. 1. Suppose, the scheduling network receives a BRR at timepoint $t$: $(v_s, v_d, 80 MB/s, 1000 MB, 0, 20s)$. As Fig. 1 shows, network $G$ has four nodes: $V = \{ v_1, v_2, v_3, v_4 \}$ and four edges: $E = \{ v_1 - v_2, v_2 - v_3, v_3 - v_4, v_4 - v_1 \}$. At timepoint $t$, there are two timesteps, $t^s_1 = [0, 10s]$ and $t^s_2 = [10s, 20s]$, and the available bandwidths of each edge during these two timesteps are labeled on each edge. The control plane of network $G$ aims to find a QR with the earliest completion time (QRECT) for the received BRR and the returned QRECT should be made on a fixed path from $v_s$ to $v_d$ with a fixed bandwidth within $t^s_1$ or possible varying bandwidths across $t^s_1$ and $t^s_2$. As for the QRECT for above BRR, there are only two paths connecting $v_s$ an $v_d$: path $v_s - v_1 - v_d$ and path $v_s - v_2 - v_d$. It is easy to see path $v_s - v_1 - v_d$ has a relative earlier data transfer completion time of $10s + \frac{2000MB - 50MB/s \cdot 10s}{60MB/s} = 18.3s$, which is rounded to 19s. Fig. 2 shows the optimal solution for above given BRR and the QRECT is $(v_s - v_1 - v_d, \{ 50MB/s^{[0,10s]}, 60MB/s^{[10s,19s]} \}, 0, 19s)$.

Since the FPVB problem has been proven to be NP-complete [1], we focus on the design of heuristic algorithms that produce estimated QRECT with the earliest completion time as close to the optimal solution as possible.

IV. ALGORITHM DESIGN AND COMPLEXITY ANALYSIS

In this section, we present the algorithm design and complexity analysis for LABE and LABP. LABE is derived from the GreedyFPVB algorithm proposed in [1]. LABP is inspired from the fact that among those few special paths we can identify within a graph in polynomial time, the path with the highest possibility to return the QRECT is the one with the largest available bandwidth. For LABE and LABP, detailed algorithm designs and pseudocodes are shown first, following are the intuitive explanations and mechanism illustrations by using the given example in the previous section. Please note that the unit of the weight of an edge is omitted for simplicity. In this section, the source node and destination node in the QR for a given BRR will be omitted for simplicity.

For comparison purposes, we first illustrate GreedyFPVB and MinFPVB using the example shown in Fig. 1.

A. Algorithm Illustration of GreedyFPVB

The following is the algorithm illustration of GreedyFPVB by using the example given in previous section. Please refer [1] for detailed algorithm design and analysis.

**Step 1:** Compute the weight of each edge by using varying bandwidths of this edge among part of or all timesteps. Actually, the weight of each edge equals to the data transfer completion time as if we try to transfer the total data in the given BRR along that edge within the first timestep or the following multiple timesteps. For example, the total data edge $v_s - v_1$ can transfer within timestep $t^s_1 = [0, 10s]$ is $80MB/s \cdot (10s - 0) = 800MB < 1000MB$, which means the data transfer along edge $v_s - v_1$ cannot finish within timestep $t^s_1 = [0, 10s]$, then consider the second timestep $t^s_2 = [10s, 20s]$. It is easy to see the total data edge $v_s - v_1$ can transfer within timestep $t^s_2 = [10s, 20s]$ is $60MB/s \cdot (20s - 10s) = 600MB$, so the total data edge $v_s - v_1$ can transfer within timestep $t^s_1 = [0, 10s]$ and $t^s_2 = [10s, 20s]$ is $800MB + 600MB = 1400MB > 1000MB$, which means the data transfer can finish within these two timesteps. The corresponding data transfer completion time $10s + \frac{1000MB - 800MB}{60MB/s} = 13.3s$, so correspondingly, the weight of edge $v_s - v_1$ is 13.3. The weight of each edge is shown in Fig. 3.
Fig. 4. Illustration of MinFPVB.

Step 2: Identify the narrowest path by using modified Dijkstra’s algorithm. It is easy to see that the narrowest path in this example is \( v_s - v_2 - v_d \) with the maximum weight 17.5. According to the detailed algorithm design of GreedyFPVB in [1], 17.5 is the computed data transfer completion time, which is rounded to 18 with the unit of second to be returned as the estimated earliest completion time.

In GreedyFPVB formally stated in [1], the computed estimated earliest completion time for the data transfer is the only returned parameter, but here we slightly modify GreedyFPVB to return a QR to be a complete reservation answer for the given BRR. Hence in this example, the returned estimated QRECT is \((v_s - v_2 - v_d, 40MB/s[0,10s], 65MB/s[10s,18s]), 0, 18s)\).

B. Algorithm Illustration of MinFPVB

The following is the algorithm illustration of MinFPVB by using the example given in previous section. Please refer [1] for detailed algorithm design and analysis.

Step 1: For each edge, randomly select a timestep from timestep set and set the available bandwidth of that edge during the randomly selected timestep as its available bandwidth among all timesteps. In this example, the randomly selected available bandwidth of each edge is shown in Fig. 4.

Step 2: Compute the earliest data transfer completion time by using modified Dijkstra’s algorithm, which is essentially the data transfer completion time on the path with the largest bandwidth. It can be seen that in Fig. 4, the path with the largest bandwidth is path \( v_s - v_2 - v_d \) with bandwidth of \( 80MB/s \) because of the maximum LAN bandwidth constraint.

The data transfer completion time on path \( v_s - v_2 - v_d \) is \( 0 + \frac{1000MB}{80MB/s} = 12.5s \), so 13s will be returned as the earliest completion time for the data transfer.

Similar to GreedyFPVB, the computed estimated earliest completion time is the only returned parameter in MinFPVB. Here we also slightly modify MinFPVB to return a QR to be a complete reservation answer for the given BRR. Thus, the returned QR is \((v_s - v_2 - v_d, 80MB/s[0,13s]), 0, 13s)\) in this example.

C. LABE

1) Algorithm Design: LABE is similar to GreedyFPVB [1] with the difference that for a BRR, the weight of each edge is equal to the data size divided by the least available bandwidth of this edge among all timesteps. If the least available bandwidth of this edge is zero, we will label the weight of this edge as infinity.

The pseudocode is provided in Algorithm 1. Complexity of GreedyFPVB is \( O(|E| \cdot |TS| + |E| + |V| \cdot \log |V|) \).

Algorithm 1 LABE

**INPUT:** BRR : \((v_s, v_d, B^{max}, D, t^s, t^e)\)

**OUTPUT:** Estimated QRECT or NULL for the input BRR.

1: Get current topology of the scheduling network. Identify timestep set \( TS \) containing all timesteps within time interval \([t^s, t^e]\), initialize \( QR' \leftarrow NULL \) and \( t \leftarrow 0 \);
2: for \( e \in E \) do
3: Get the least available bandwidth of \( e \), \( MinB(e) \), among all timesteps in \( TS \);
4: if \( MinB(e) = 0 \) then
5: Set the weight of \( e \) as infinity;
6: else
7: Set the weight of \( e \) as \( \frac{D}{MinB(e)} \);
8: end if
9: end for
10: Compute the narrowest path \( p \) using modified Dijkstra’s algorithm;
11: for \( i \leftarrow 1 \) to \(|TS| \) do
12: Get \( B(p^i) \) and the total data transferred by \( p \) within \( t_s \), then further calculate the total data transferred by \( p \) from \( t_s \) to \( t_e \), represented as \( D_{pi} \);
13: if \( D_{pi} \geq D \) then
14: \( t = t_{si}^e + \frac{D_{pi}}{MinB(e)} \);
15: \( QR' = (p, \{B(p^i)[t_s,t_e]\} \ldots , B(p^i)[t_s,t_e]) ; t_{si}, t_e \);\)
16: break;
17: end if
18: end for
19: Return \( QR' \).

2) Algorithm Explanation: The following is the native step by step explanation of LABE.

Step 1: With the input BRR, get the current topology of the scheduling network. Identify timestep set \( TS \) containing all timesteps within time interval \([t^s, t^e]\).

Step 2: For edge \( e \in E \), compute its least available bandwidth \( MinB(e) \) among all timesteps in \( TS \). If above value equals zero, then set the weight of \( e \) as infinity; otherwise, set the weight of \( e \) as \( \frac{D}{MinB(e)} \). Identify the narrowest path \( p \) using modified Dijkstra’s algorithm, then iterate through the timesteps in \( TS \). During \( t_s \in TS \), \( 1 \leq i \leq |TS| \), compute the bandwidth of \( p \) and the amount of data transferred within \( t_s \), then further calculate the total amount of data transferred along \( p \) within timesteps from \( t_s \) to \( t_e \), If the calculated amount of data is no less than \( D \), then the iteration of timesteps stops, compute the corresponding data transfer completion time of \( D \) and return the corresponding QR.

3) Algorithm Illustration: We use the example shown in Fig. 1 to illustrate the scheduling strategy of LABE.

Step 1: The topology of the scheduling network and the BRR are given, it is easy to get the timestep set \( TS = \{[0,10s], [10s,20s]\} \).

Step 2: Compute the least available bandwidth of each edge among all timesteps in \( TS \), further compute the weight of each edge, which is shown in Fig. 5. Identify the narrowest path, it is easy to see that the narrowest path in this example is \( v_s - v_2 - v_d \).
Algorithm 2 LABP

Given: \( G(V,E) \)

INPUT: \( BRR: (v_s,v_d, B_{max}, D, t^S, t^E) \)

OUTPUT: Estimated QRECT or NULL for the input BRR

1: Get current topology of the scheduling network. Identify timestep set \( TS \) containing all timestep within time interval \([t^S, t^E]\), initialize \( QR' = NULL \) and \( t \leftarrow \infty \);
2: Get the path with the largest bandwidth during each timestep in \( TS \) using modified Dijkstra’s algorithm and store them in an array \( Array \);
3: for \( i \leftarrow 1 \) to \( |TS| \) do
4: for each \( p \in Array \) do
5: Get \( B(p) \) and the total data \( p \) transfers within \( ts_i \), then further calculate the total data transferred by \( p \) from \( ts_i \) to \( ts_i \), represented as \( D_p \);
6: if \( (D_p \geq D) \) and \( (t < ts^i + \frac{D-D_p}{B(p)}) \) then
7: \( t = ts^i + \frac{D-D_p}{B(p)} \);
8: \( QR' = \{ p, \{ B(p^{ts^i, ts^i} \}, \ldots, B(p^{ts^i, t}) \}, t^i, t \} \);
9: end if
10: end for
11: end for
12: Return \( QR' \).

1) Algorithm Design: The pseudocode of the LABP algorithm is provided in Algorithm 2. The complexity of LABP is \( O(\sum_{i=1}^{\mid TS \mid} |TS| \cdot |E| + |V| \cdot \log |V|) \).

2) Algorithm Explanation: The following are the native step by step explanation of LABP.

Step 1: With the input BRR, get the current topology of the scheduling network. Identify timestep set \( TS \) containing all timesteps within time interval \([t^S, t^E]\). Use modified Dijkstra’s algorithm to get the path with the largest bandwidth within each timesteps in \( TS \) and store these paths in \( Array \).

Step 2: Iterate through the timesteps in \( TS \) and paths in \( Array \). During \( ts_i \in TS, 1 \leq i \leq |TS| \), compute the amount of data transferred within \( ts_i \), for each \( p \in Array \), further calculate the total amount of data transferred along \( p \) within timesteps from \( ts_1 \) to \( ts_i \). If the calculated amount of data is no less than \( D \), compute the corresponding data transfer completion time of \( D \) and identify the corresponding QR. Return the QR with the earliest data transfer completion time during above iteration.

3) Algorithm Illustration: We use the example shown in Fig. 1 to illustrate the scheduling strategy of LABP.

Step 1: The topology of the scheduling network and the BRR are given, it is easy to get the timestep set \( TS = \{0, 10s\}, \{10s, 20s\} \). Identify the path with the largest bandwidth in each timestep. The path with the largest bandwidth in timestep \([0, 10s]\) is \( v_s - v_1 - v_2 \) and bandwidth of \( 50MB/s \) and that within timestep \([10s, 20s]\) is \( v_s - v_2 - v_d \) with bandwidth of \( 65MB/s \), so \( Array = \{v_s - v_1 - v_2, v_s - v_2 - v_d\} \).

Step 2: Iterate through timestep set \( TS \) and path array \( Array \). During \([0, 10s]\), the total data path \( v_s - v_1 - v_2 \) can transfer is \( 50MB/s \cdot (10s - 0) \approx 500MB \) while the total data path \( v_s - v_2 - v_d \) can transfer is \( 40MB/s \cdot (10s - 0) = 400MB \). Both path \( v_s - v_1 - v_2 \) and path \( v_s - v_2 - v_d \) cannot finish the data transfer of the required amount of data, so the iteration of timesteps continues. During \([10s, 20s]\), the total data path \( v_s - v_2 - v_d \) can transfer is \( 65MB/s \cdot (20s - 10s) = 600MB \) while the total data path \( v_s - v_2 - v_d \) can transfer is \( 65MB/s \cdot (20s - 10s) = 650MB \). The total amount of data path \( v_s - v_1 - v_d \) can transfer within \([0, 10s]\) and \([10s, 20s]\) is \( 500MB \) while path \( v_s - v_2 - v_d \) can transfer within these two timesteps is \( 400MB \) and \( 650MB \), which means both path \( v_s - v_1 - v_d \) and path \( v_s - v_2 - v_d \) cannot finish the data transfer before the given deadline \( 20s \). The data transfer completion time path on \( v_s - v_1 - v_d \) is \( 10s + \frac{500MB}{50MB/s} = 18.3s \approx 19s \) while that on path \( v_s - v_2 - v_d \) is \( 10s + \frac{650MB}{65MB/s} = 19.2s \approx 20s \). Obviously, the data transfer on path \( v_s - v_1 - v_d \) has a relative earlier completion time, so the return QR is \( v_s - v_1 - v_d, \{50MB/s, 60MB/s\} \), which is the estimated QRECT by using LABP.

From the above illustration of LABE and LABP, we observe that both algorithms return the optimal solution for the given example. GreedyFPVB and MinFPVB also return a bandwidth reservation result and seem to successfully schedule the input BRR: GreedyFPVB returns \( v_s - v_2 - v_d, \{40MB/s, 65MB/s\} \) and MinFPVB returns \( v_s - v_2 - v_d, \{80MB/s\} \). The question is if the returned path actually has the capability to finish the data transfer at the returned data transfer completion time point. In this example, path \( v_s - v_2 - v_d \) cannot finish the data transfer at timepoint 18s or 13s. The total amount of data path \( v_s - v_2 - v_d \) can transfer at timepoint 18s is \( 40MB/s \cdot 18s = 720MB \) while that path \( v_s - v_2 - v_d \) can transfer at timepoint 13s is \( 65MB/s \cdot 13s = 845MB \).
is \(40MB/s \cdot 10s + 65MB/s \cdot 3s = 595MB < 1000MB\). We will further discuss the weakness of GreedyFPVB and MinFPVB in the following section.

V. PERFORMANCE EVALUATION

Currently, OSCARS is the most widely used bandwidth reservation service developed by the ESnet group to support high-bandwidth virtual circuits setup and teardown across multi-domains. OSCARS provides interoperable, effective, reliable, and high performance network communications and collaboration services [11], [12], [13], [14], [15] for more than 40 DOE research sites, including the entire National Laboratory system, together with its supercomputing facilities, the major scientific instruments and other 140 research and commercial networks around the world [12], [13]. To simulate the real ESnet scenario, we perform scheduling experiments on the topology data gathered from ESnet as shown in Fig. 6 [16]. The edges with bandwidth capacity less than 1,000Mbps are deleted for a cleaner view.

In our simulation, a certain number of BRRs are randomly generated with the same earliest start time \(t^E = 0\) and latest finish time \(t^F = 20s\). For each BRR represented as \((v_s, v_d, B^{max}, D, t^S, t^F)\), \(v_s\) and \(v_d\) are randomly selected from the collected nodes, \(B^{max}\) is a random integer between 1 and 1,000 multiplied by 10, and \(D \leq B^{max} \cdot (t^F - t^S)\). We conduct 10 sets of simulation experiments, each of which contains five different groups of 1,000 to 10,000 randomly generated BRRs. We implement GreedyFPVB, MinFPVB, LABLE and LABP to process the same group of BRRs and measure three performance metrics for each algorithm, namely algorithm running time, job success ratio, which is defined as the percentage of BRRs that have been successfully scheduled within the defined time interval, and the average BRR completion time of all BRRs in this group.

As stated in previous section, both GreedyFPVB and MinFPVB share the same weakness: they do not check if the path in the returned estimated QRECT actually has the capability to finish the data transfer at the returned completion time point. Consequently, the outcome returned by these two algorithms might differ greatly from the actual performance along the identified path. In our experiments, we double check if the computed paths in the estimated QRECTs returned by GreedyFPVB, MinFPVB and LABLE have the ability to finish the data transfer at the returned data transfer completion time points as above returned estimated QRECTs show. Please note that at any rate, we can identify a path by using any of above three algorithms. For example, we can get the narrowest path by using GreedyFPVB and LABLE although these two algorithms use different strategies to calculate the weight of each edge. We can also compute the path with the largest available bandwidth using MinFPVB. A path with the largest available bandwidth during a timestep can be identified in any case using LABP. We define the actual data transfer completion time for each BRR scheduled by GreedyFPVB, MinFPVB and LABLE within the following different situations. For a given BRR, if the computed path in the estimated QRECT returned by one of above four algorithms can finish the data transfer at the returned data transfer completion time point, then its actual data transfer completion time equals the returned data transfer completion time; otherwise, we check if the returned path has the ability to finish the data transfer before the data transfer deadline 20s. If the returned path has the ability to finish the data transfer before 20s, the actual data transfer completion time of this BRR equals to the computed data transfer completion time on the returned path; otherwise, the total amount of data transferred during \([0, 20s]\) along the identified path is computed, represented as \(d\). If \(d > 0\), then the completion time of this BRR is defined as \((20 \frac{d}{s})\). On the other hand, if \(d = 0\), which indicates that the corresponding BRR cannot be scheduled, we set the actual data transfer completion time of this BRR as the largest actual completion time for all BRRs in the same group. As for the returned estimated QRECT for a BRR by using LABP, we do not check the path in its estimated QRECT like we do for returned estimated QRECTs by using GreedyFPVB, MinFPVB and LABLE, because the path in the estimated QRECT returned by LABP is a certain unambiguous path. We only compute the actual data transfer time for BRRs by using LABP for comparison purpose.

The detailed experiment data of GreedyFPVB, MinFPVB, LABLE and LABP is shown in Table I. Please note that in Table I, the detailed experiment data of GreedyFPVB, MinFPVB and LABLE have a form \((a, b, c, d, e)\), they represent the average returned data transfer completion time, the average returned success ratio, the average actual data transfer completion time, the average actual success ratio and the average processing time when processing those five groups of randomly generated BRRs in corresponding set by corresponding algorithm, respectively. As for LABP, its experiment data has a form \((a, b, d, c)\), they represent the average returned data transfer completion time, the average returned success ratio, the average actual data transfer completion time and the average returned processing time when processing those five groups of randomly generated BRRs in corresponding set, respectively. The returned success ratio of LABP is also its actual success ratio. The returned data transfer completion time for a BRR is defined by using the following scheme: if the data transfer completion time in the returned estimated QRECT is no larger than the data transfer deadline 20s, then the returned data transfer completion time equals to that in the returned estimated QRECT; otherwise, it equals to the largest one among all data transfer completion times of BRRs in the corresponding group. The returned success ratio is defined as the division between the BRRs whose data transfer completion times in their estimated QRECTs are no larger than 20s and the number of all BRRs in that group. The actual success ratio is defined as the division between the BRRs whose actual data transfer completion times are no larger than 20s and the number of all BRRs in that group. We plot the mean value and the corresponding variance with 95% confidence level of these experimental results in Fig. 8, Fig. 7 and Fig. 9. In Fig. 8, GreedyFPVB represents the returned completion time while GreedyFPVB-actual represents the actual completion time of data transfer on the same path. MinFPVB and MinFPVB-actual, and LABE and LABLE-actual have the similar meanings. In Fig. 7, above notations have the same meanings.

These performance figures clearly show that the completion time (around 6s) and actual time (around 132s) of GreedyFPVB vary significantly, so do the success ratio (around 88.4%) and actual success ratio (around 1.9%) of GreedyFPVB. The same phenomenon is observed on
Fig. 6. A simplified topology of ESnet network [16].

MinFPVB: the completion time returned is around 25s while the actual completion time is around 81s, and the success ratio is around 82.8% while the actual success ratio is around 22.5%. MinFPVB is the least stable in terms of completion time since MinFPVB randomly selects a timestep for an edge, and sets the available bandwidth for that edge during that selected timestep as the available bandwidth among all timesteps. This random choice causes the instability in the transfer completion time. On the contrary with GreedyFPVB and MinFPVB, the actual success ratio of LABE (around 59%) is larger than its returned success ratio (around 53%) while its actual data transfer completion time (around 19s) is less than its returned data transfer completion time (around 30s). LABP has the highest actual success ratio (around 69.0%), the least returned completion time (around 13s) and the least actual completion time (around 16s), however, its running time is larger than the other three algorithms when processing the same group of BRRs.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed two improved bandwidth scheduling algorithms, namely LABE and LABP, for FPVB, a known NP-complete problem. For either of these two proposed algorithms, detailed algorithm design, intuitive explanations and mechanism illustrations by using the given example are given. To simulate the real network topology of DOE’s ESnet, we construct our high-performance network using the node and edge information from ESnet. Extensive scheduling experiments using multiple sets of randomly generated BRRs illustrated the performance superiority of our proposed algorithms over two existing algorithms. Both GreedyFPVB and MinFPVB have the same weakness: they do not check if the computed path in the corresponding estimated QRECT has the ability to finish the data transfer at the computed data transfer completion time point. The inaccuracy and instability of MinFPVB is caused by that for each edge, MinFPVB randomly select a timestep out of the timestep set and set the bandwidth of that edge among all timesteps as its bandwidth during the selected timestep. LABE and LABP significantly improve the success ratio for multiple BRRs and reduce the average data transfer completion time for each individual BRR. Furthermore, the low algorithm running time allows the scheduling component of dedicated networks to make a real-time scheduling decision under a heavy scheduling demand in a large-scale network and even cross-domain networks. In LABE, we choose the least bandwidth of an edge to compute the transfer completion time instead of the varying bandwidth, thus yielding a higher probability of successful schedule on the narrowest path. In LABP, we identify paths with the largest available bandwidth during each timestep and use greedy strategy to compute the amount of data each path transferred from the first timestep to the current timestep. The QR with the earliest completion time during above iteration will be returned as the estimated QRECT. The significant improvement of the success ratio of LABE and LABP compared with GreedyFPVB and MinFPVB will not only improve the efficiency of the scheduling high-performance network, but also save energy, network resource and the network maintenance cost. It is our future interest to study the reliability issue in the FPFB problem category since each edge in the high-performance network has a certain failure probability. We wish to explore effective scheduling method to move data across the network with a transfer reliability guarantee.

REFERENCES

<table>
<thead>
<tr>
<th></th>
<th>GreedyFPVB</th>
<th>MinFPVB</th>
<th>LABE</th>
<th>LABP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.8762</td>
<td>6.12</td>
<td>0.0140</td>
<td>133.47</td>
</tr>
<tr>
<td></td>
<td>(0.8280, 23.17)</td>
<td>0.2198</td>
<td>85.78</td>
<td>209.2</td>
</tr>
<tr>
<td>2000</td>
<td>0.8844</td>
<td>6.02</td>
<td>0.0192</td>
<td>139.2</td>
</tr>
<tr>
<td></td>
<td>(0.8252, 28.22)</td>
<td>0.2686</td>
<td>2848.4</td>
<td>1516.0</td>
</tr>
<tr>
<td>3000</td>
<td>0.8832</td>
<td>6.04</td>
<td>0.0181</td>
<td>131.3</td>
</tr>
<tr>
<td></td>
<td>(0.8284, 24.32)</td>
<td>0.2491</td>
<td>79.82</td>
<td>269.8</td>
</tr>
<tr>
<td>4000</td>
<td>0.8836</td>
<td>6.04</td>
<td>0.0193</td>
<td>138.2</td>
</tr>
<tr>
<td></td>
<td>(0.8284, 24.32)</td>
<td>0.2491</td>
<td>79.82</td>
<td>269.8</td>
</tr>
<tr>
<td>5000</td>
<td>0.8836</td>
<td>6.04</td>
<td>0.0193</td>
<td>138.2</td>
</tr>
<tr>
<td></td>
<td>(0.8284, 24.32)</td>
<td>0.2491</td>
<td>79.82</td>
<td>269.8</td>
</tr>
<tr>
<td>6000</td>
<td>0.8830</td>
<td>6.05</td>
<td>0.0198</td>
<td>134.1</td>
</tr>
<tr>
<td></td>
<td>(0.8286, 25.54)</td>
<td>0.2246</td>
<td>72.79</td>
<td>247.2</td>
</tr>
<tr>
<td>7000</td>
<td>0.8830</td>
<td>6.05</td>
<td>0.0198</td>
<td>134.1</td>
</tr>
<tr>
<td></td>
<td>(0.8286, 25.54)</td>
<td>0.2246</td>
<td>72.79</td>
<td>247.2</td>
</tr>
<tr>
<td>8000</td>
<td>0.8836</td>
<td>6.02</td>
<td>0.0192</td>
<td>131.3</td>
</tr>
<tr>
<td></td>
<td>(0.8284, 24.32)</td>
<td>0.2491</td>
<td>79.82</td>
<td>269.8</td>
</tr>
<tr>
<td>9000</td>
<td>0.8830</td>
<td>6.05</td>
<td>0.0198</td>
<td>134.1</td>
</tr>
<tr>
<td></td>
<td>(0.8286, 25.54)</td>
<td>0.2246</td>
<td>72.79</td>
<td>247.2</td>
</tr>
<tr>
<td>10000</td>
<td>0.8838</td>
<td>6.07</td>
<td>0.0192</td>
<td>131.3</td>
</tr>
<tr>
<td></td>
<td>(0.8292, 24.92)</td>
<td>0.2278</td>
<td>97.37</td>
<td>212.2</td>
</tr>
</tbody>
</table>

Fig. 7. Comparison of success ratio between GreedyFPVB, MinFPVB, LABE and LABP.

Fig. 8. Comparison of data transfer completion time between GreedyFPVB, MinFPVB, LABE and LABP.

Fig. 9. Comparison of execution time between GreedyFPVB, MinFPVB, LABE and LABP.


