IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 6, 2020, accepted January 19, 2020, date of publication January 30, 2020, date of current version February 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2970542

Time-Space Decoupled SnF Scheduling of Bulk Transfers Across Inter-Datacenter Optical Networks

XIAO LIN[1], (Member, IEEE), WEIQIANG SUN[2,4], (Senior Member, IEEE), XIAOYU WANG[3], SHENGNAN YUE[2], (Student Member, IEEE), MALATHI VEERARAGHAVAN[3], (Fellow, IEEE), AND WEISHENG HU[2], (Senior Member, IEEE)

[1]College of Physics and Information Engineering, Fuzhou University, Fuzhou 350116, China
[2]State Key Laboratory of Advanced Optical Communication Systems and Networks, Shanghai Jiao Tong University, Shanghai 200240, China
[3]Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904, USA
[4]Shanghai Institute for Advanced Communication and Data Science, Shanghai 200240, China

Corresponding author: Xiao Lin (linxiaocer@fzu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61901118 and Grant 61433009, and in part by the Open Foundation of the State Key Laboratory of Advanced Optical Communication Systems and Networks under Grant 2019GZKF03003.

ABSTRACT To meet the increasing inter-datacenter traffic, datacenter storage is brought in the forwarding path. Bulk data that are delay-tolerant can be temporarily stored and forwarded (SnF) at a later time when link is less busy. However, the use of storage transforms the conventional routing problem into a scheduling problem, where both bandwidth and storage resources must be allocated and both spatial routing and temporal scheduling must be performed. Such an SnF scheduling problem is critically important for the efficiency of SnF approaches. Most prior solutions aimed to jointly solve its temporal and spatial components and formulated this problem into difficult optimization problems, which contributes to a huge expansion size of the problem and hence are too complex for large networks and dynamic traffic. In this paper, we present analytic models to quantify the performance-complexity tradeoff in the SnF scheduling problem. Our key findings reveal that desirable performance can be obtained by considering only a few pre-selected routes rather than searching in the entire network topology. Thus, we propose a time-space decoupled (TSD) SnF scheduling method. Compared to the conventional joint methods, the advantages of the TSD method are as follows: (i) by decoupling the problem and solving them separately, the TSD method reduces the quadratic complexity of the joint methods to linear complexity; (ii) by condensing the redundant states, the TSD method obtains a longer horizon of temporal scheduling, given the same computational cost; (iii) by bounding the spatial hop count of routing paths, the TSD method avoids the detour issue faced by the joint methods and hence uses bandwidth more efficiently; (iv) by formulating the problem into a routing problem, the TSD method greatly simplifies the problem for dynamic traffic. Simulations demonstrate that the TSD method can outperform the conventional joint method, especially when the traffic load is moderate-to-high.

INDEX TERMS Bulk data transfers, inter-datacenter networks, optical circuit switching, routing, store-and-forward.

I. INTRODUCTION

Emerging data-intensive applications pour massive bulk data flows into inter-datacenter networks (inter-DCNs) on a daily basis. Such bulk data flows are often long-lived and bandwidth-hungry [1]. The rapid growth of bulk data transfers not only incurs unprecedented bandwidth contention in inter-DCNs, but also requires expensive transit costs [2].

Bulk transfer demands from the data-intensive applications, such as online backups, multimedia transfers and data migration, usually do not require being delivered immediately, but need to be completed within certain time periods (e.g., hours) [3]. In other words, they are delay-tolerant.

The associate editor coordinating the review of this manuscript and approving it for publication was Rentao Gu.

This provides extra flexibility in scheduling, routing as well as resource allocation. Although many research efforts have been made to propose novel deadline-aware scheduling schemes and flexible resource allocation methods [1]–[7], they all attempted to deliver bulk data over end-to-end (E2E) connections.

A key issue in E2E-based transfers is that the residual bandwidth in inter-DCNs varies in both spatial and temporal dimensions [8]. In the temporal dimension, the user behaviour and demands in specific areas follow typical diurnal variation patterns. As a result, bandwidth contentions inevitably occur at peak hours, while massive amounts of residual bandwidth are left unused at off-peak hours. In the spatial dimension, the sender site, the intermediate sites and the receiver site may be located in various time zones. For a bulk data flow, the bandwidth bottleneck could occur on different links along its forwarding path and at different time. Thus, fully utilizing such temporal and spatial varying residual bandwidth becomes very challenging especially for the E2E-based transfers [9]. In this case, neither the time window nor the transit bandwidth of the E2E connections can satisfy the requirements of bulk data transfers.

In general, DC providers buy networking resources from ISPs or deploy dedicated inter-DCN based on peak demand. To sustain the rapid growth of bulk data flows, DC providers must keep buying more networking resources or upgrading the transmission capacity of their dedicated lines, even though the average utilization is low [3]. Moreover, due to the reliability reasons [10], the average utilization of wide area network (WAN) links usually remains 30-40%. This also suggests massive unused bandwidth.

To mitigate the peak-hour congestion and rescue the unused bandwidth in inter-DCNs, the delay tolerance of bulk data is exploited and DC storage is brought in the conventional transfer process. The intermediate DC sites can temporarily store the delay-tolerant data until the inter-DCN is less busy or there are sufficient resources for the transfer. Through the use of temporary storage, the bulk transfers can be postponed until off-peak hours. Hence, the peak-hour congestion is mitigated. In the meanwhile, since the E2E constraints on the provisioning process are relaxed, the residual bandwidth can be fully utilized. Hence, the utilization is improved. This so called Store-and-Forward (SnF) approach has been proven to be effective in inter-DCNs [11]–[16].

In today's networks, data from applications or clients are first divided into small packets before sending over the network. However, in the presence of larger data size and higher traffic volume, network operators find it difficult to deliver bulk data on a per-packet basis [17]. Many research efforts focused on offloading bulk data to optical circuit switching (OCS), which can offer great cost/power savings and performance improvement in datacenter networks [18]–[20]. While OCS is desirable to deliver bulk data efficiently, it carries data over E2E lightpaths. This suggests that OCS also faces a similar E2E dilemma. In this paper, we consider a combination of the flexibility of SnF and the efficiency of OCS for inter-DC bulk transfers [13]–[15].

However, the use of storage transforms the conventional routing problem into a scheduling problem, where both bandwidth and storage resources must be allocated and both spatial routing and temporal scheduling must be performed. Such an SnF scheduling problem is critically important for the efficiency of SnF approaches, but requires expensive computational cost to be solved [14].

Most prior studies aimed to jointly solve the spatial routing component and the temporal scheduling component of the SnF scheduling problem, and formulated this problem into difficult optimization problems [11]–[16]. Their solutions have been proven to be effective for small networks or for static traffic. However, the joint nature of their solutions contributes to a huge expansion size of the problem. Thus, they may be too complex for large networks and dynamic traffic. Instead, our proposed solution, i.e., time-space decoupled (TSD) scheduling method, aims at decoupling the spatial routing from the temporal scheduling and solving them separately, which reduces the quadratic complexity of the conventional joint solutions to linear complexity.

Portions of this work were presented at the 2019 IEEE HPSR conference [21]. For this paper, we extended our prior work as follows: (i) we conducted a comprehensive survey of existing efforts in leveraging SnF approaches to transfer bulk data as well as existing SnF scheduling methods; (ii) we extended our analytic models of the SnF scheduling problem to compute the upper bound of the probability of reservation failure; (iii) we presented a condense algorithm for the TSD method to condense the redundant network states; (iv) we ran extensive simulations to investigate the impact of the condense algorithm on the reservation window (i.e., the horizon of temporal scheduling) and the impact of the TSD method on the SnF operations.

Our key contributions are as follows:

1) Our survey on prior literature shows that most of the prior literature aims at improving network utilization or minimizing transfer cost/time. However, few studies provide practical insights on how to design efficient SnF scheduling methods. Our survey on existing SnF scheduling methods further reveals that prior methods may be either too complex or insufficient for large networks and dynamic traffic. There exists a tradeoff between performance and complexity when designing an SnF scheduling method. But little attention has been paid to this tradeoff.
2) We present analytic models to evaluate the performance-complexity tradeoff. Our key findings are as follows: (i) compared to involving more longer alternate routes in scheduling, expanding the horizon of temporal scheduling can provide more performance benefits while imposing less computational burden on scheduling; (ii) when the traffic load is light and bandwidth is readily available, the joint solution can route requests

over the entire network and hence provide performance advantages over the decoupled solution; (iii) when the traffic load is moderate-to-high and bandwidth is not readily available, the decoupled solution can obtain performance similar to the joint solution while imposing less computational burden; (iv) routing across the entire network results in detours in paths assigned to a few requests. This problem is called the detour issue. This issue escalates when the traffic load is high. Under such circumstance, a few requests occupy large amounts of bandwidth and the residual bandwidth is insufficient to accommodate new requests, which, in turn, results in more request blocking. By contrast, the decoupled solution only considers a few alternate routes, which naturally avoids the detour issue and hence provides better performance.

3) We present a time-space decoupled (TSD) SnF scheduling method. By decoupling the problem and solving them separately, the TSD method reduces the quadratic complexity of the conventional joint solutions to linear complexity. By condensing the redundant states, the TSD method obtains a longer horizon of temporal scheduling, when compared to the joint method, given the same computational cost. By considering pre-selected routes, the TSD method avoids the detour issue faced by the joint method. By formulating the problem into a routing problem, the TSD method greatly simplifies the SnF scheduling problem.
4) Our studies illustrate that the TSD method can mitigate the detour issue, and use bandwidth more effectively than the conventional joint method. Meanwhile, the TSD method can provide a wider reservation window, and reserve more resources in the future than the joint method. As a result, the TSD method has low complexity while still achieving high performance.
5) We investigate how the TSD method affects the SnF operations. Our key finding is that although more requests in the TSD method need to be stored before reaching their destinations with the traffic load increasing, majority of the stored requests only need one or two SnF operations. This suggests that transferring data through SnF will not incur high deployment cost or impose heavy management burden on the network.

The rest of the paper is organized as follows. Section II reviews the literature and argues that the design of an efficient SnF scheduling method requires a tradeoff between complexity and performance. Section III evaluates the tradeoff, and reveals that the decoupled solution has the potential to reduce complexity while maintaining high performance. Section IV presents the TSD method. Comparative evaluation of the TSD method with the conventional joint method is presented in Section V. Section VI concludes this paper.

II. RELATED WORK

In this section, we summarize the previous literature on bulk data transfer through SnF and the various goals effectively achieved by this approach. As seen in Fig. 1, SnF has been applied to bulk data transfers in WANs [9], [12], [15], [22], inter-DCNs [8], [11], [23], [24], intra-DCNs [25] and access networks [26]. Prior studies aimed to improve network performance, minimize transfer cost/time and maximize transfer utility. They have proven that SnF is an effective approach for bulk data transfers. However, solving the SnF scheduling problem is computationally expensive [14], [21]. Little work has gained practical insights into how to balance between complexity and performance when designing SnF scheduling methods. This inspires us to conduct a comprehensive survey on the SnF scheduling methods proposed in the prior studies.

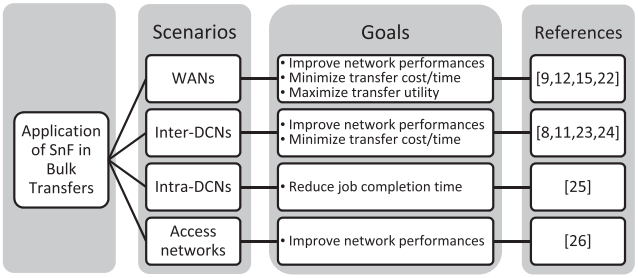


FIGURE 1. Classification of the existing studies on bulk data transfer through SnF.

A. SNF SCHEDULING METHODS

The proposed SnF scheduling methods are classified into two categories: joint solutions and decoupled solutions. These efforts are summarized in Fig. 2.

1) JOINT SOLUTIONS

The joint solutions solved the spatial and temporal components jointly. SnF scheduling problems were formulated into network flow problems (e.g., max-flow or min-cost flow problems) [13], [22]–[24], [27], [28] and linear programming (LP) problems [12]–[14]. Classical flow optimization algorithms or heuristics were used to achieve optimal solutions in both routing and scheduling for request sets. SnF scheduling problems were also formulated into shortest path problems, which were solved with classical routing algorithms [15], [29], [30].

The joint solutions require a global view of the entire network in both spatial and temporal dimensions. Intuitively, they should provide optimal results in both dimensions. However, due to the joint nature, the search space in the scheduling methods must contain multi-dimensional network states. This results in a huge expansion of the search space and hence imposes heavy computational burden on searching. To reduce the search space, constraints and assumptions were applied on their system models. For example, only an intermediate storage node was considered in each forwarding path [12]. The storage capacity constraint was relaxed by assuming unlimited storage capacity on each node [14], [23], [24], [27]–[30]. Some prior studies considered a finite time span T, which is a bounded integer indicating the number of time slots in the scheduling horizon [13], [14], [22]–[24], [27]–[30].

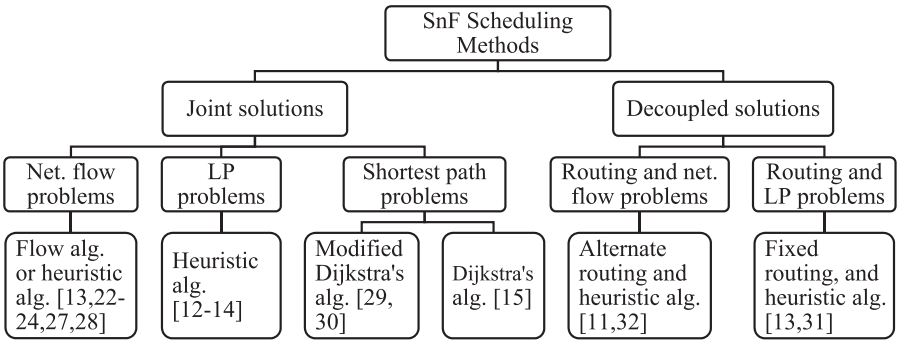


FIGURE 2. Classification of the existing SnF scheduling methods.

In [29], [30], the scheduler only maintained the state of each link in the earliest available unscheduled time slot. All subsequent time slots were assumed to be unscheduled and could be used to schedule newly arriving requests. Although the temporal scheduling problem was greatly simplified, ongoing requests cannot be scheduled within the bandwidth gaps between requests. Apparently, the reduction in the search space comes at the cost of performance degradation.

2) DECOUPLED SOLUTIONS

The decoupled solutions solved the spatial and temporal components separately. The joint solutions naturally result in a huge expansion of the search space. Alternatively, the problem can be decoupled to reduce the search space. To solve the spatial problem, fixed routing [13], [31] and alternate routing [11], [32] were used to offer a tradeoff between complexity and performance. To solve the temporal problem, prior work formulated network flow problems or LP problems, which were solved by LP solvers or proposed heuristics. To further reduce the complexity of the temporal problem, the maximum number of SnF operations used for scheduling a request was limited [32].

3) LESSONS LEARNT FROM PRIOR WORK

Most of the prior studies formulated the problem into LP problems or network flow problems, and attempted to achieve optimal solutions by using classical optimization algorithms. They are effective for small networks or for static traffic where requests arrival times are fixed and given in advance, but too complex to implement for large networks and dynamic traffic where requests arrive one after another randomly. Few studies considered dynamic traffic [15], [29], [30]. However, to reduce complexity, they had to impose constraints or assumptions on their system models for dynamic traffic at the cost of degraded performance (e.g., poor utilization). Thus, they may be insufficient for large networks and dynamic traffic.

In short, there exists a tradeoff between complexity and performance in the design of an SnF scheduling method. However, few studies have gained practical insights into how to efficiently balance this tradeoff. As a result, the existing efforts are either too complex or too insufficient for large networks and dynamic traffic. Compared to our prior joint method [15], we incorporate the decoupled solution into our prior routing framework, i.e., time-shifted multilayer graph (TS-MLG). Unlike the prior decoupled method that formulated the temporal scheduling problem into LP problems or network flow problems, the proposed method formulates the problem into a routing problem with the TS-MLG, which greatly simplifies the problem for dynamic traffic.

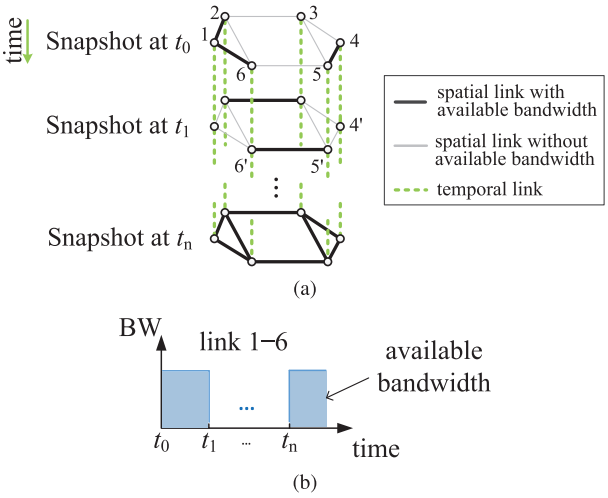


FIGURE 3. (a) Example of a TS-MLG. (b) The bandwidth usage on link 1-6.

B. A ROUTING FRAMEWORK FOR SNF: TS-MLG

1) OVERVIEW

Time-shifted multilayer graph (TS-MLG) is a routing framework proposed for bulk transfers in OCS networks with SnF [15]. Fig. 3(a) shows an example TS-MLG, which is a multilayer graph built from a set of layers. Each layer is a snapshot of the network state at a certain time instant. These layers are stacked downward in a time-increasing order to capture the dynamics of network resource usage. For example, Fig. 3(b) depicts the bandwidth usage on link 1-6, which is available at time t_0 and busy at time t_1. Correspondingly, in Fig. 3(a), link 1-6 in the layer at time t_0 is available and that at time t_1 is busy. Besides, spatial links connect different

nodes within the same layer, and temporal links connect the same nodes at different layers. When a request traverses a spatial link, it suggests data transfer from one node to another. When a request traverses a temporal link, it suggests data storage on a node for a certain period. The capacity of a spatial link or a temporal link refers to the bandwidth of that link, or the storage capacity of that node.

Consider a request r from node 1 to node 5 arrives at the network at time t_0. An E2E transmission is unavailable at t_0. However, by performing a routing algorithm, e.g., Dijkstra's algorithm, on the TS-MLG shown in Fig. 3(a), an E2E path over time and space is found, i.e., path 1-6-6'-5'. In this case, the intermediate storage node is node 6.

2) COMPLEXITY

Given a routing algorithm, its computational complexity depends on the scale of the network topology. Let V denote the number of nodes in a network topology. For Dijkstra's algorithm, its computational complexity is $O(V^2)$. However, the total number of nodes in a TS-MLG is equal to the number of layers in the TS-MLG times the number of nodes in the network topology. Thus, the number of layers dominantly determines the complexity.

A notable feature of TS-MLG is the number of its layers reflects the number of active requests in the network [15]. The active request is defined as a request which is accepted but not yet finished. The bulk data flows are generally infrequent in time [4]. However, the bursty nature of the bulk data flows may result in a temporary increase in the number of layers. This suggests high computational burden on the route search for new requests. To bound the computational burden of each request, the number of layers used for routing needs to be limited. As a result, requests are accepted or blocked depending on whether viable paths are found in the TS-MLG within a given number of layers.

III. ANALYTIC MODELS FOR SNF SCHEDULING PROBLEM

In this section, we present analytic models to evaluate the performance-complexity tradeoff in the SnF scheduling problem. We start from the case of a fixed route for a node pair (s, d), and extend to the case of multiple alternate routes for (s, d). Our studies reveal the potential of the decoupled solution to reduce the complexity while maintaining high performance.

Generally, for a request that attempts to deliver data from source s to destination d, the greater number of feasible paths, the more likely the request is to succeed. Thus, the number of feasible paths between node pairs is an important measure of network performance. Here, we investigate the number of feasible paths that can be offered to route a request. In this paper we differentiate between "path" and "route." A path traverses multiple spatial and temporal links in a TS-MLG, which differs from a conventional network graph. We use the common term "route" to refer to a path in a conventional network graph, which simply shows how to reach a destination in the spatial dimension, without considering the temporal dimension.

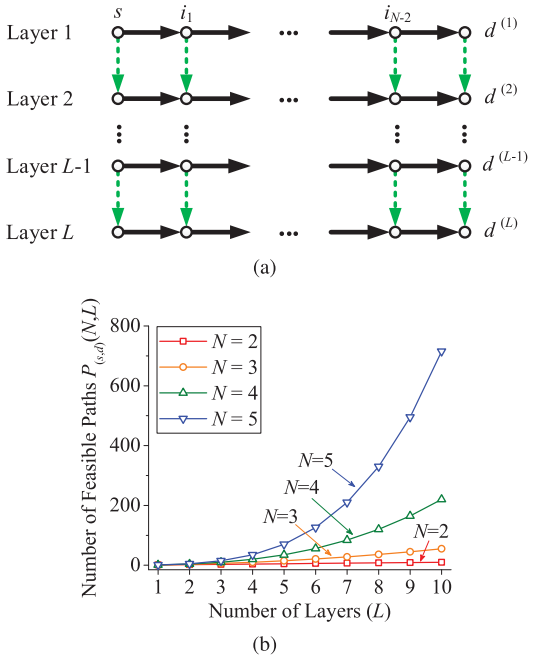


FIGURE 4. (a) SnF routing model, given a fixed route with N nodes and its TS-MLG with L layers. (b) The number of feasible paths, i.e., $P_{(s,d)}(N, L)$.

A. ANALYTIC MODELS FOR SNF SCHEDULING ON A FIXED ROUTE

Let us start by considering a simple model of a fixed route for a node pair (s, d). Consider a fixed route $\boldsymbol{R} = \{s, i_1, i_2, \ldots, i_{N-2}, d\}$, where N denotes the number of nodes on the route ($N \geq 2$). An SnF routing model for TS-MLG is depicted in Fig. 4(a). In general, spatial links are bidirectional and temporal links are unidirectional. However, in the case of Fig. 4(a), routing options for every node are confined to either delivering data to its downstream neighbor (i.e., next spatial hop) or to itself in the future (i.e., next temporal hop).

Given this model, we first investigate the number of feasible paths from s to d. Let $d^{(l)}$ denote the destination d at layer l in the model, where $l \in [1, L]$. $P_{(s,d)}(N, L)$ is defined as the total number of feasible paths from s to $d^{(\boldsymbol{L})}$, where $\boldsymbol{L} = [1, 2, \ldots, l, \ldots, L]$, i.e., the destination d at all the L layers. $P_{(s,d)}(N, L)$ can be decomposed into L different cases of $(N-1)$-node-route model from node i_1 to the destination d, i.e., $P_{(i_1,d)}(N-1, l)$. The value of l depends on the number of temporal links traversed by a feasible path before the first spatial hop. We have (see detailed proof in Appendix A)

$$P_{(s,d)}(N, L) = \begin{cases} \sum_{l=1}^{L} P_{(i_1,d)}(N-1, l) & \text{if } N \geq 3 \\ L & \text{if } N = 2 \end{cases} \quad (1)$$

In Fig. 4(b), numerical results show how $P_{(s,d)}(N, L)$ changes with L and N, respectively. $P_{(s,d)}(N, L)$ increases with L. The larger the value of N, the more significant the

increase in $P_{(s,d)}(N, L)$. For example, when $L = 10$, there are 55 feasible paths in the TS-MLG of a three-node route ($N = 3$). This number increases up to 715 when considering a five-node route ($N = 5$).

Additionally, given two non-zero integers a and b, $P_{(s,d)}(a, b) = P_{(s,d)}(b, a)$. For instance, $P_{(s,d)}(3, 10) = P_{(s,d)}(10, 3) = 55$. The difference between $P_{(s,d)}(3, 10)$ and $P_{(s,d)}(10, 3)$ lies in the former TS-MLG having more temporal links, while the latter has more spatial links. Intuitively, as the number of feasible paths offered in both cases is the same, the complexity and performance of scheduling a request in such TS-MLGs should be correspondingly similar.

The main idea of SnF is to trade storage for bandwidth. The validity of SnF lies in the fact that the cost of storage decreases faster than the cost of WAN bandwidth [9]. There is a natural fit for inter-DCNs, as sufficient storage resources are available inside DCs to enable a cost-effective SnF. In this context, requests can reserve storage on temporal links more easily than bandwidth on spatial links. This inspires us to study whether both cases can still offer similar performance.

Probability of reservation failure $F_{(s,d)}(N, L)$ is defined as the probability that a request fails to find and reserve any viable path from a set of feasible paths of size $P_{(s,d)}(N, L)$. Let p_b and p_s denote the probability that a request fails to reserve the required bandwidth on a spatial link, and the required storage on a temporal link, respectively.

However, as shown in Fig. 4(a), since each node would introduce two routing options, i.e., next spatial hop or next temporal hop, feasible paths would diverge at each intermediate nodes. Besides, various feasible paths may share common spatial links or temporal links. With the size of the routing model expanding (i.e., larger values of L or N), it would be more difficult to obtain the expression of $F_{(s,d)}(N, L)$ directly. Alternatively, we derive the upper and lower bounds of $F_{(s,d)}(N, L)$, respectively.

In essence, the complex nature of $F_{(s,d)}(N, L)$ lies in the fact that a spatial or temporal link may appear more than once in multiple feasible paths. Fortunately, the successful path method and the reliability block diagram in the reliability analysis of electrical circuits [33] have provided clues to tackle similar problems.

On the one hand, we decompose $F_{(s,d)}(N, L)$ into L various and dependent cases. Assume that a request can start to search a viable path from s to $d^{(l)}$, i.e., the l-th case, only when it fails to find any viable path in the previous l-1 cases. This, in turn, presents an upper bound of $F_{(s,d)}(N, L)$, i.e., $F^{Upper}_{(s,d)}(N, L)$. We have (see detailed proof in Appendix B)

$$F^{Upper}_{(s,d)}(N, L) = 1 - \sum_{l=1}^{L} (1-p_s)^{l-1}(1-p_b)^{N-1} \times (p_b)^{l-1} \binom{N+l-3}{l-1} \tag{2}$$

On the other hand, we reduce $F_{(s,d)}(N, L)$ into L independent and smaller cases, where the number of nodes is reduced to $N - 1$. Assume that a request would succeed if at least a viable path is found among the L cases. This, in turn, presents

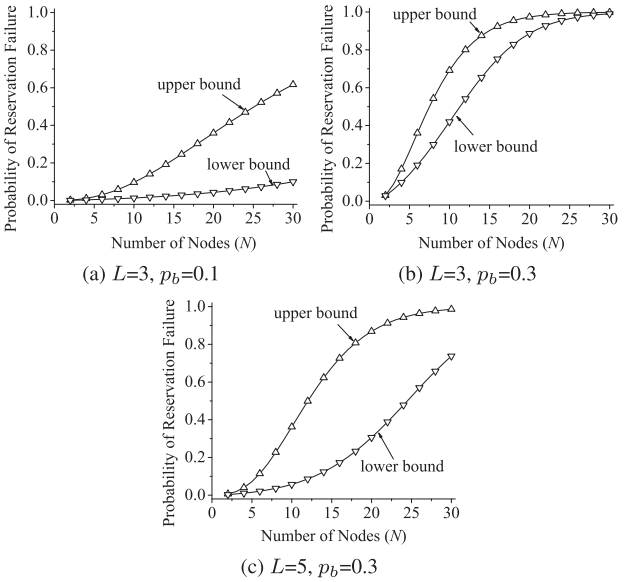


FIGURE 5. Probability of reservation failure $F^{Upper}_{(s,d)}(N, L)$ and $F^{Lower}_{(s,d)}(N, L)$ under various values of N. ($p_s = 0.01$).

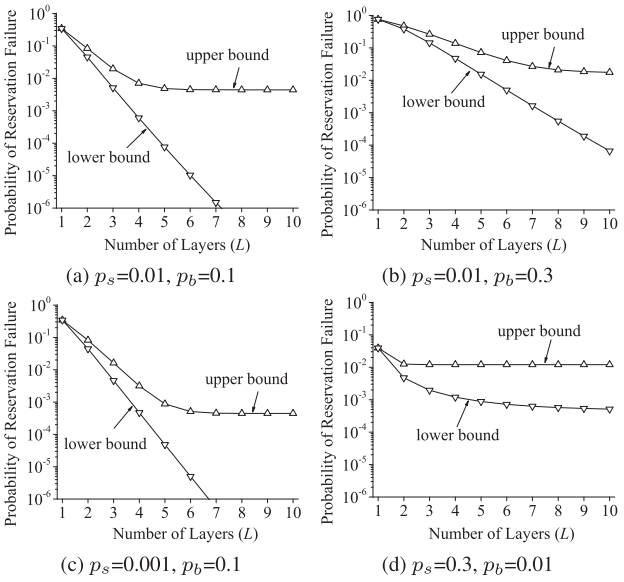


FIGURE 6. Probability of reservation failure $F^{Upper}_{(s,d)}(N, L)$ and $F^{Lower}_{(s,d)}(N, L)$ under various values of L. ($N = 5$).

a lower bound of $F_{(s,d)}(N, L)$, i.e., $F^{Lower}_{(s,d)}(N, L)$. We have (see detailed proof in Appendix C)

$$F^{Lower}_{(s,d)}(N, L) = \begin{cases} \prod_{l=1}^{L} [1 - (1-p_s)^{L-l}(1-p_b) \\ \quad \times (1 - F^{Lower}_{(i_1,d)}(N-1, l))] & \text{if } N \geq 3 \\ \prod_{l=1}^{L} [1 - (1-p_s)^{l-1}(1-p_b)] & \text{if } N = 2 \end{cases} \tag{3}$$

We investigate the properties of $F^{Upper}_{(s,d)}(N, L)$ and $F^{Lower}_{(s,d)}(N, L)$. Numerical results are shown in Figs. 5-7.

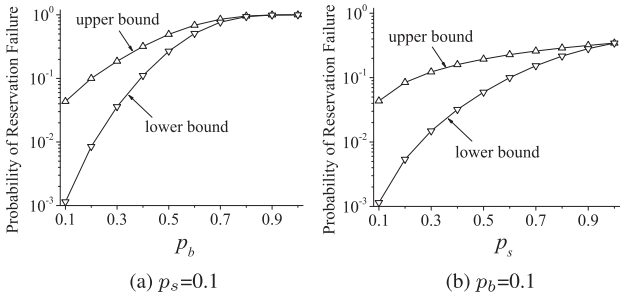


FIGURE 7. Probability of reservation failure $F^{Upper}_{(s,d)}(N, L)$ and $F^{Lower}_{(s,d)}(N, L)$ under various values of p_b and p_s. ($N = 5$ and $L = 5$).

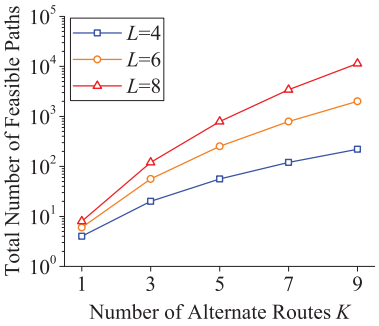


FIGURE 8. The total number of feasible paths for (s, d), i.e., $MP_{(s,d)}(K, L)$, given various values of K.

Fig. 5 shows that for a given p_b, p_s and L, $F^{Upper}_{(s,d)}(N, L)$ and $F^{Lower}_{(s,d)}(N, L)$ increase with N. Intuitively, $F^{Upper}_{(s,d)}(N, L)$ and $F^{Lower}_{(s,d)}(N, L)$ should decrease with increasing N, because $P_{(s,d)}(N, L)$ increases with N, as shown in Fig. 4. A larger N enables more feasible paths for searching, which should reduce both $F^{Upper}_{(s,d)}(N, L)$ and $F^{Lower}_{(s,d)}(N, L)$. However, when $p_s \ll p_b$, spatial hop counts are more challenging. The larger the value of N, the more the number of spatial hops on a feasible path, and the greater the difficulty in reserving. Thus, although increasing N enables more searchable feasible paths, requests are more difficult to satisfy. In other words, compared to long-spatial-hop routes, short routes provide more performance benefits while imposing less computational burden. Besides, we compare the results in Fig. 5(a) with that in Fig. 5(b). The results illustrate that the increases in both $F^{Upper}_{(s,d)}(N, L)$ and $F^{Lower}_{(s,d)}(N, L)$ become more evident when p_b increases from 0.1 up to 0.3. When the bandwidth resources become scarce, requests are more difficult to be served. Then, we compare the results in Fig. 5(b) with that in Fig. 5(c). The results illustrate that when L increases from 3 to 5, the increase becomes less evident. For example, given $N = 10$, $F^{Upper}_{(s,d)}(N, L)$ is equal to 0.69 when $L = 3$. This probability is reduced to 0.36 when $L = 5$.

Fig. 6 further reveals the impact of L on the lower and upper bounds. As seen in Fig. 6, for a given p_b, p_s and N, $F^{Upper}_{(s,d)}(N, L)$ and $F^{Lower}_{(s,d)}(N, L)$ decrease with increasing L. This is because a larger value of L enables more feasible paths for searching. On the one hand, the spatial hop count on each feasible path remains constant, as $N = 5$. On the other hand, when $p_s \ll p_b$, introducing more temporal hops in a feasible path imposes less challenge in reserving. In Fig. 6(a) and (b), when p_b increases from 0.1 to 0.3, the decreases become less evident. This suggests that increasing L becomes less effective in reducing the probability when the bandwidth resources become scarce. In addition, Fig. 6 shows that when L increases beyond a certain value, the decrease in the probability becomes slight. For example, in Fig. 6(a), given $p_s = 0.01$ and $p_b = 0.1$, the decrease in the upper bound becomes slight when L increases beyond 5. In Fig. 6(c), when p_s decreases from 0.01 to 0.001, the decrease become slight when L increases beyond 7. Therefore, the performance improvement offered by increasing L is affected by both p_b and p_s. The aforementioned studies assume $p_s \ll p_b$. Without loss of generality, we further assume $p_s \gg p_b$. In Fig. 6(d), given $p_s = 0.3$ and $p_b = 0.01$, the upper bound remains almost constant when L increases beyond 2. The decrease in the lower bound becomes slight when L increases beyond 5. This means that when the storage resources are scarce, increasing L becomes less effective in reducing the probability of reservation failure.

Fig. 7 illustrates that the impacts of p_b and p_s on the lower and upper bounds, respectively. In Fig. 7(a), given $p_s = 0.1$, when p_b increases up to 1, both the upper and the lower bounds increase to 1. In Fig. 7(b), given $p_b = 0.1$, when p_s increases up to 1, both the upper and the lower bounds increase to 0.34, which is equivalent to the case when no storage is used for routing.

B. ANALYTIC MODELS FOR MULTIPLE ROUTES

Assume there are K alternate routes for a node pair (s, d), i.e., $\boldsymbol{R}_{(s,d)} = \{R_1, \ldots, R_k, \ldots, R_K\}$. N_k denotes the number of nodes on the k-th route R_k. $MP_{(s,d)}(K, L)$ is defined as the total number of feasible paths between source s and destination d, given K alternate routes for (s, d) and L layers used for scheduling. We have

$$MP_{(s,d)}(K, L) = \sum_{k=1}^{K} P_{(s,d)}(N_k, L) \tag{4}$$

Given a certain node pair, the set $\boldsymbol{R}_{(s,d)}$ depends on specific network topologies and varies widely. It is difficult to capture how N_k changes case-by-case. Intuitively, short-hop routes should be preferred, because they are more likely to have available resources than long-hop routes. Hence, a general case is considered, where N_k increases with k. For simplicity, N_k is assumed to be $k + 1$. Exploring a more sophisticated model of N_k is an interesting, yet complicated, problem that is worthy of further study.

We investigate the properties of $MP_{(s,d)}(K, L)$. Numerical results are shown in Fig. 8. The total number of feasible paths increases with K. The larger the value of L, the more significant the increase in $MP_{(s,d)}(K, L)$.

Eqs. (2) and (3) provide the upper and the lower bounds of the probability of reservation failure in the case of a single route for a node pair (s, d). Here, we extend the upper and the

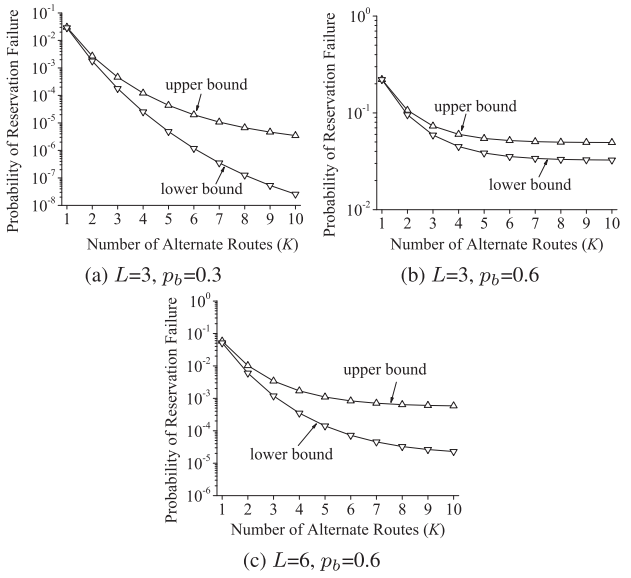


FIGURE 9. Probability of reservation failure $MF^{Upper}_{(s,d)}(K, L)$ and $MF^{Lower}_{(s,d)}(K, L)$ under various values of K. ($p_s = 0.01$).

lower bounds to the case of multiple alternate routes for (s, d). Consider K alternate routes for (s, d). Let $MF^{Upper}_{(s,d)}(K, L)$ and $MF^{Lower}_{(s,d)}(K, L)$ denote the upper and the lower bounds of the probability that a request fails to find and reserve any viable path from $\boldsymbol{R}_{(s,d)}$. Thus, we have

$$MF^{Upper}_{(s,d)}(K, L) = \prod_{k=1}^{K} F^{Upper}_{(s,d)}(N_k, L) \tag{5}$$

$$MF^{Lower}_{(s,d)}(K, L) = \prod_{k=1}^{K} F^{Lower}_{(s,d)}(N_k, L) \tag{6}$$

We investigate the properties of $MF^{Upper}_{(s,d)}(K, L)$ and $MF^{Lower}_{(s,d)}(K, L)$. Herein, $p_s \ll p_b$ and $p_s = 0.01$. The results are shown in Fig. 9. It can be seen that both $MF^{Upper}_{(s,d)}(K, L)$ and $MF^{Lower}_{(s,d)}(K, L)$ decrease with increasing K. Figs. 9(a) and (b) illustrate that the larger the p_b, the less evident the decrease in the upper and the lower bounds. As a result, given different values of p_b, the values of K needed to obtain expected $MF^{Upper}_{(s,d)}(K, L)$ and $MF^{Lower}_{(s,d)}(K, L)$ diverge. For example, when $L = 3$, to obtain a $MF^{Lower}_{(s,d)}(K, L)$ of 10^{-3}, the value of K needed in the case of $p_b = 0.3$ is 3. However, when $p_b = 0.6$, the value of K will dramatically increase, as shown in Fig. 9(b). The resulting set of feasible paths is too large and hence impractical to be searched. Similar trends are observed in $MF^{Upper}_{(s,d)}(K, L)$. Meanwhile, given $p_b = 0.6$, the decrease in $MF^{Lower}_{(s,d)}(K, L)$ is more evident in the case of $L = 6$ than in the case of $L = 3$, as shown in Fig. 9(c). To obtain $MF^{Lower}_{(s,d)}(K, L)$ of 10^{-3}, the value of K needed in the case of $L = 6$ is 4. The resulting set of feasible paths is practical to be searched. Similar trends are also observed in $MF^{Upper}_{(s,d)}(K, L)$. Increasing L not only reduces $MF^{Upper}_{(s,d)}(K, L)$ and $MF^{Lower}_{(s,d)}(K, L)$ more effectively, but also imposes less computational burden on searching, when compared to increasing K. Thus, expanding the search space in the temporal dimension (i.e., increasing L) is more effective than in the spatial dimension (i.e., increasing K) to reduce both $MF^{Upper}_{(s,d)}(K, L)$ and $MF^{Lower}_{(s,d)}(K, L)$.

In short, the extra flexibility of SnF contributes to an exponential increase in the number of feasible paths. This increase provides significant performance benefits, since requests are more likely to find a viable path from a larger set of feasible paths. However, searching such a large set of feasible paths is computationally expensive. Therefore, there exists a performance-complexity tradeoff, which should be carefully examined when designing an SnF scheduling method.

C. POTENTIAL OF DECOUPLED SOLUTION

The joint solution attempts to jointly solve the spatial routing and the temporal scheduling problems. Thus, it requires a global view of the entire network in both spatial and temporal dimensions. In other words, to schedule a request, all alternate routes for its source-destination pair (s, d) would be considered in the joint solution. As previously discussed, the number of feasible paths for (s, d) exponentially increases with the number of alternate routes K. Such a large set of feasible paths associated with the joint solution imposes heavy computational burden on scheduling. Meanwhile, compared to long-spatial-hop routes, short routes provide more performance benefits while imposing less computational burden. Consequently, desirable performance can be obtained by considering a finite number of alternate routes rather than routing over the entire network, which suggests the potential of decoupled solution to reduce complexity and improve performance. Inspired by this, we investigate how to balance the performance-complexity tradeoff by considering a finite number of alternate routes in the decoupled solution.

We compare the joint solution with the decoupled solution in terms of the number of feasible paths and the probability of reservation failure. Due to the limited space, $MF^{Upper}_{(s,d)}(K, L)$ is used to indicate the probability of reservation failure, but $MF^{Lower}_{(s,d)}(K, L)$ would follow similar trends. Let K_j and K_d denote the number of alternate routes used for the joint solution and for the decoupled solution, respectively. Assume six layers ($L = 6$) are used for routing and the total number of alternate routes for a node pair (s, d) is 10. Thus, the 10 alternate routes would be considered in the joint solution, i.e., $K_j = 10$. By contrast, we investigate how many alternate routes (K_d) are required for the decoupled solution to obtain a $MF^{Upper}_{(s,d)}(K_d, L)$ similar to $MF^{Upper}_{(s,d)}(K_j, L)$ in the joint solution. The numerical results are shown in Fig. 10.

Fig. 10(a) shows that the number of feasible paths in the decoupled solution approaches that in the joint solution with increasing K_d. Figs. 10(b) and (c) demonstrate that the probability of reservation failure in the decoupled solution approaches that in the joint solution with K_d. In Fig. 10(b), when $p_b = 0.1$, there exists a huge performance gap between the decoupled solution and the joint solution for a small value of K_d. For instance, when $K_d = 4$, the probability in the

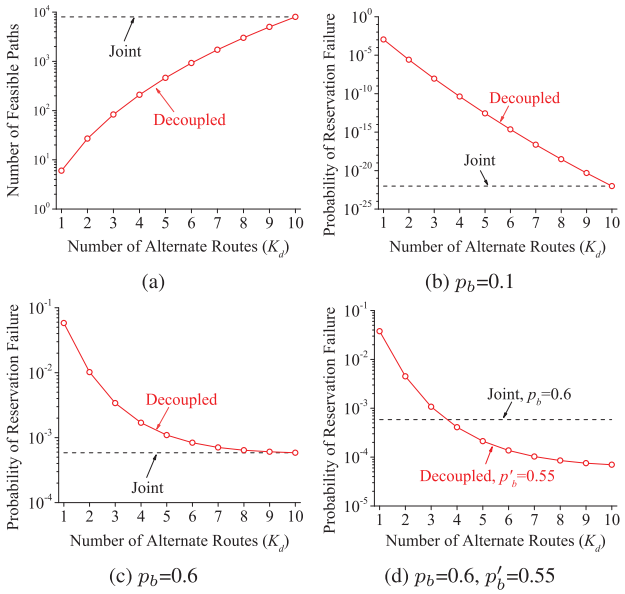


FIGURE 10. **(a) The number of feasible paths and (b)-(d) the probability of reservation failure $MF^{Upper}_{(s,d)}(K, L)$ under various values of K_d. ($p_s = 0.01$, $L = 6$ and $K_f = 10$).**

decoupled solution is 4.30×10^{-11}. Although this probability is much larger than 9.59×10^{-23} of the probability in the joint solution, it may be acceptable for some applications. On the other hand, when $K_d = 4$, the number of feasible paths in the decoupled solution is 209, which is smaller than 8007 in the joint solution. As seen in Figs. 10(a) and (b), when bandwidth is sufficient and is easy to be reserved by requests (i.e., a small value of p_b), the joint solution provides significant performance advantages over the decoupled solution. However, the decoupled solution obtains a lower computational complexity while maintaining acceptable performance for some applications. In Fig. 10(c), when p_b increases up to 0.6, the performance gap between both solutions greatly narrows. For a small value of K_d, the decoupled solution can obtain a probability similar to the joint solution. For instance, when $K_d = 4$, the probability in the decoupled solution is 1.70×10^{-3}. By contrast, the probability in the joint solution is 5.85×10^{-4}. As seen in Fig. 10(c), when bandwidth becomes scarce and difficult to be reserved (i.e., a large value of p_b), the decoupled solution can obtain performance similar to the joint solution by considering a few alternate routes rather than routing over the entire network topology.

Intuitively, during the route selection, a route with more residual bandwidth should be preferred. In a conventional network, short-hop routes are likely to have more residual bandwidth than long-hop routes. Thus, conventional dynamic routing algorithms work well by selecting the routes with more residual bandwidth while keeping the route hop short. However, in the presence of SnF, long-hop routes are possible to have sufficient residual bandwidth, because the E2E constraint is greatly relaxed. Under such circumstance, some requests could be delivered over long-hop routes if the dynamic routing algorithms do not take the hop count into account. When the traffic load is high, a few long-detoured requests may occupy large amounts of bandwidth and the residual bandwidth is insufficient to accommodate ongoing requests, which, in turn, degrades the network performance. This so called detour issue may result in long-hop routes, low bandwidth efficiency and high blocking probability [34].

Generally, the joint solution is more flexible than the decoupled solution, since it involves the entire network in scheduling. However, this also implies that the joint solution is more vulnerable to the detour issue than the decoupled solution, especially when the hop count is not carefully considered. By contrary, the decoupled solution delivers requests over a few alternate routes, which naturally avoids the detour issue and hence provides performance advantages over the joint solution.

In the above studies, both solutions are assumed to have the same value of p_b. Here, we assume the probability of reservation failure on a spatial link in the joint solution is p_b, while that in the decoupled solution is p'_b. We assume that $p_b > p'_b$ in order to consider a case when the residual bandwidth in the joint solution is scarcer than the decoupled solution due to the detour issue. Herein, $p_b = 0.6$ and $p'_b = 0.55$. We investigate the impact of the detour issue on the joint solution. The numerical results are shown in Fig. 10(d). As we can see, the decoupled solution starts to outperform the joint solution with K increasing. For instance, when $K = 4$, the decoupled solution obtains a probability of 4.08×10^{-4}, which is smaller than 5.86×10^{-4} in the joint solution. In this case, the decoupled solution has the potential to outperform the joint solution.

D. LESSONS LEARNT FROM ANALYTIC MODELS

In a typical scenario (e.g., inter-DCN) where storage is sufficient, our key findings are as follows:

1) Compared to short-spatial-hop routes, long routes provide less performance benefits while imposing more computational burden on scheduling.
2) Compared to increasing K (i.e., involving more longer alternate routes in scheduling), increasing L (i.e., expanding the horizon of temporal scheduling) can provide more performance benefits while imposing less computational burden on scheduling.
3) When the traffic load is light and bandwidth is readily available, the joint solution can provide performance advantages over the decoupled solution. This is because the joint solution can route requests over the entire network topology and hence provides more flexibility in spatial routing than the decoupled solution.
4) When the traffic load is moderate-to-high and bandwidth is not readily available, the decoupled solution can obtain performance similar to the joint solution by considering a few pre-selected routes rather than searching in the entire network topology.
5) When the traffic load is high, the detour issue may contribute to the fact that the residual bandwidth

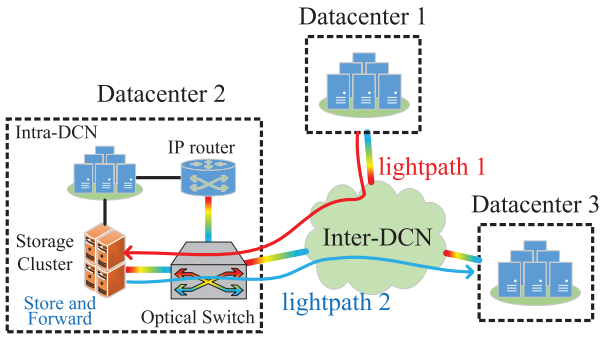


FIGURE 11. An SnF scheduling scheme on an inter-DC optical network.

is insufficient to accommodate new requests, which, in turn, results in more request blocking in the joint solution than in the decoupled solution. By contrast, the decoupled solution naturally avoids the detour issue. In this case, the decoupled solution may outperform the joint solution.

IV. TIME-SPACE DECOUPLED SNF SCHEDULING METHOD

A. NETWORK MODEL AND ASSUMPTIONS

We assume that each DC site is capable of temporarily storing bulk data at its storage server cluster, as shown in Fig. 11. The storage clusters are part of *ScienceDMZ*, which is an architecture proposed to bypass enterprise firewalls and enable high-speed network paths for bulk data transfers [35], [36]. In Fig. 11, bulk data from datacenter 1 are delivered over lightpath 1 and stored on the storage cluster of datacenter 2 for a certain period. Then, the bulk data are delivered to datacenter 3 over lightpath 2. The architecture, cost and power consumption of the storage cluster have been studied in [26]. Wavelength conversion is employed at each site. A request is blocked when, upon arrival, no path exists between the source and the destination. Each transfer request occupies a wavelength. Compared to the transmission delay, the propagation delay between two sites and the processing overhead (e.g., routing decision and lightpath establishment) is assumed to be negligible. Assume that the disk read/write speed of the storage server cluster is comparable to the transmission capacity of one wavelength. Since it is preferable to serve bulk data with dedicated resources [37], a portion of the overall network resources is assumed to be dedicated to OCS lightpaths that carry bulk data flows.

B. OVERVIEW

Our studies in Section III demonstrate that desirable performance can be obtained by considering only a few pre-selected routes. Inspired by this, a TSD SnF scheduling method is proposed.

The main idea of the TSD method is to decouple the SnF scheduling problem into a spatial routing sub-problem and a temporal scheduling sub-problem. To tackle the spatial sub-problem, K shortest routes are pre-computed for each node pair. Let $\mathbb{R}$ denote the set of pre-computed routes for all node pairs. To tackle the temporal sub-problem, the original

Algorithm 1 Time-Space Decoupled SnF Scheduling Method

1: Input: $r = \{s, d\}$, G, K, $\mathbb{R}$
2: Output: *Path*, *Find*
3: **for** $k = 1; k \leq K; k++$ **do**
4: Select a pre-computed route $\boldsymbol{R_k}$ from s to d, where $\boldsymbol{R_k} \in \mathbb{R}$
5: Use $\boldsymbol{R_k}$ to reduce G into a subgraph G'
6: Apply Algorithm 2 on G' to condense the redundant layers, and get a subgraph G'' of G'
7: Apply Algorithm 3 on G'' to find a viable path *Path*
8: **if** *Path* is valid **then**
9: **return** *Path* and *Find* = True
10: **end if**
11: **end for**
12: No viable path is found and **return** *Find* = False

Algorithm 2 Condense Algorithm

1: Input: G'
2: Output: G''
3: **for all** $Layers = [l_1, l_2, ...l_i..., l_L]$ in G' **do**
4: **if** $l_{i+1} == l_i$ **then**
5: Remove Layer l_{i+1} from G'
6: **end if**
7: **end for**
8: **return** $G'' = G'$

TS-MLG is first reduced into a smaller TS-MLG based on a pre-computed route. To further reduce the graph size, a condense algorithm is applied to condense the redundant layers in the reduced TS-MLG. Then, a search algorithm is performed on the condensed TS-MLG to find a viable path.

Algorithm 1 presents the overall procedure of the TSD method. Assume that a request r from node s to node d, i.e., $r = \{s, d\}$, arrives at the network. First, line 4 selects a pre-computed route $\boldsymbol{R_k}$ for the node pair (s, d), where $\boldsymbol{R_k} \in \mathbb{R}$. Second, line 5 uses $\boldsymbol{R_k}$ to reduce the original TS-MLG G into a subgraph G'. Let G' denote the reduced subgraph of G. Specifically, the reduced graph G' only consists of the nodes and the links belonging to $\boldsymbol{R_k}$. Third, line 6 applies Algorithm 2 on the reduced graph G' to find and condense the redundant layers. Specifically, if two adjacent layers, say layer l_i at time t_i and layer l_{i+1} at time t_{i+1} ($t_{i+1} > t_i$), represent the same bandwidth usage, layer l_{i+1} is considered as a redundant layer and hence removed from G'. Let G'' denote the condensed subgraph of G'. Fourth, line 7 applies Algorithm 3 on G'' to find a viable path (i.e., set *Path*). If Algorithm 3 succeeds in finding one, Algorithm 1 returns set *Path*; otherwise, Algorithm 1 re-runs with the next route $\boldsymbol{R_{k+1}}$. The request r will be rejected when no path is found among $\mathbb{R}$.

Here, an example of the resulting condensed graph G'' is shown in Fig. 12. Compared to that in G shown in Fig. 3(a), routing options for every node in G'' have been confined

Algorithm 3 Forward Search Algorithm

Input: G'', L_R, N_k
Output: *Path*
Initialize $n = 1$, $Path = \{N_1^{(1)}\}$
for $l = 1; l \leq L_R; l++$ **do**
  while $n < N_k$ **do**
    if $B_{n,l}$ is valid and $S_{n,l}$ is invalid **then**
      $Path \cup N_{n+1}^{(l)}$ and $n++$
    else if $S_{n,l}$ is valid and $B_{n,l}$ is invalid **then**
      $Path \cup N_n^{(l+1)}$ and **break**
    else if both $B_{n,l}$ and $S_{n,l}$ are valid **then**
      if the link cost of $B_{n,l} \leq$ that of $S_{n,l}$ **then**
        $Path \cup N_{n+1}^{(l)}$ and $n++$
      else
        $Path \cup N_n^{(l+1)}$ and **break**
      end if
    else
      No viable path is found and **return** $Path = \varnothing$
    end if
  end while
  if $n == N_k$ **then**
    Find a viable path and **return** *Path*
  end if
end for

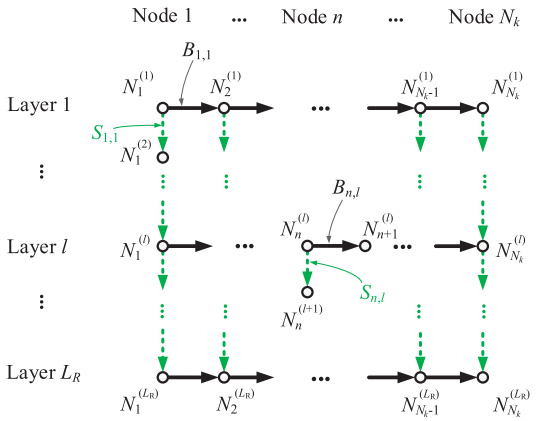


FIGURE 12. Schematic of a condensed graph G''.

to either its next spatial hop or its next temporal hop. It is unnecessary to apply a conventional routing algorithm, e.g., Dijkstra's algorithm, on G'' to find a viable path. Thus, a simple search algorithm, i.e., Forward Search algorithm, is presented, as shown in Algorithm 3.

The goal of Algorithm 3 is to address the temporal scheduling sub-problem for each request. Consider a condensed graph $G''(V'', E'')$. L_R denotes the number of layers used for routing. N_k denotes the number of nodes along $\boldsymbol{R_k}$. $N_n^{(l)}$ denotes the node n at layer l in G'', where $N_n^{(l)} \in V''$. $B_{n,l}$ denotes the spatial link between node $N_n^{(l)}$ and node $N_{n+1}^{(l)}$, where $B_{n,l} \in E''$. $S_{n,l}$ denotes the temporal link between node $N_n^{(l)}$ and node $N_n^{(l+1)}$, where $S_{n,l} \in E''$.

Algorithm 3 repeatedly determines whether $B_{n,l}$ or $S_{n,l}$ is valid for request r. The adjacent node (node $N_{n+1}^{(l)}$ or node $N_n^{(l+1)}$) on the valid link will be added to set *Path*. Algorithm 3 will finish searching when a destination d is added to set *Path*, where $d \in \{N_{N_k}^{(1)}, ..., N_{N_k}^{(L_R)}\}$. If both $B_{n,l}$ and $S_{n,l}$ are valid, Algorithm 3 will compare their link costs. If the link cost of $B_{n,l}$ is no more than that of $S_{n,l}$, node $N_{n+1}^{(l)}$ will be added to set *Path*; otherwise, node $N_n^{(l+1)}$ will be added to set *Path*. Our prior work [15] discussed how to assign spatial and temporal links with appropriate costs to reflect different routing criteria.

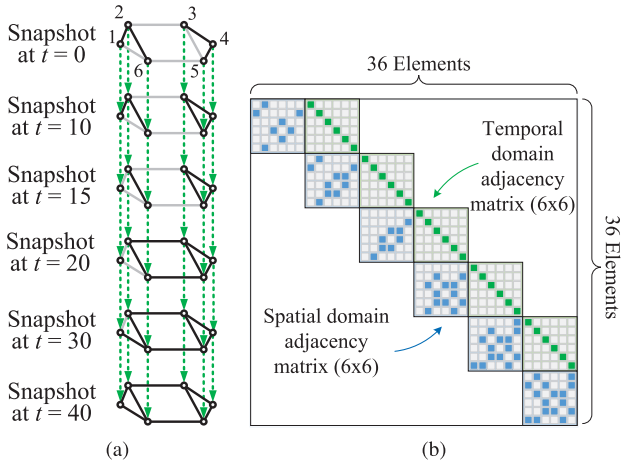


FIGURE 13. (a) The original TS-MLG G with 6 layers and 6 vertices in each layer and (b) its adjacency matrix (36×36).

C. EXAMPLE

Here, we demonstrate how the TSD method reduces the search space through an example. To illustrate this, both the TS-MLG and the corresponding adjacency matrix are presented. An adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph.

Fig. 13 shows the original TS-MLG G and its corresponding adjacency matrix. In Fig. 13(a), G consists of six layers, with each layer containing six vertices. In Fig. 13(b), as the total number of nodes in G is 36, the size of its corresponding adjacency matrix is 36×36. In this 36×36 adjacency matrix, the blue 6×6 matrices represent the spatial connections within the layers, while the green 6×6 matrices represent the temporal connections between the layers. The remaining part of this matrix is filled with zeros, which indicates no connection.

Figs. 13 to 15 depict how G is reduced by the TSD method. Assume that a request r from node 1 to node 3 arrives at the network. Based on a pre-computed shortest route, say, $\boldsymbol{R_k} = \{1, 2, 3\}$, G is reduced into G'. As shown in Figs. 14(a) and (b), the resulting graph G' consists of six layers, with each layer containing three vertices. Correspondingly, the size of its adjacency matrix is 18×18. In Fig. 14(a), the layers at $t = 0$ and $t = 10$ in G' represent the same network state. Similarly, the layers at $t = 20$ and $t = 30$ in G' also represent the same network state. These redundant layers do not provide more information, but impose extra computational burden on scheduling. Thus, Algorithm 2 is

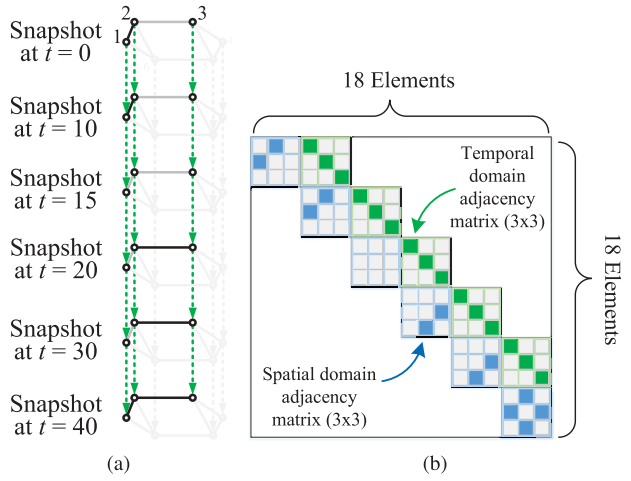


FIGURE 14. (a) The reduced TS-MLG G'' with 6 layers and 3 vertices in each layer and (b) its adjacency matrix (18×18).

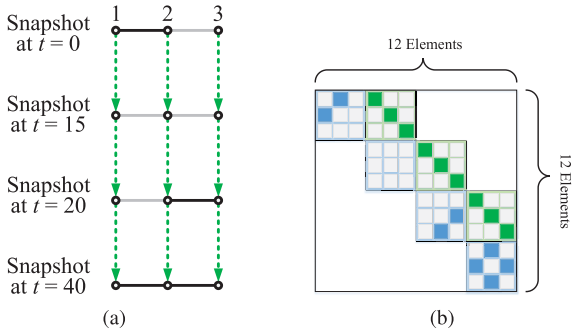


FIGURE 15. (a) The condensed TS-MLG G'' with 4 layers and 3 vertices in each layer and (b) its adjacency matrix (12×12).

performed to find and remove the redundant layers at $t = 10$ and $t = 30$. The resulting graph G'' and its adjacency matrix are shown in Figs. 15(a) and (b). G'' consists of four layers, with each layer containing three vertices. The size of its adjacency matrix is 12×12. By performing the TSD method, the size of the adjacency matrix is reduced from 36×36 to 12×12, which greatly reduces the computational complexity.

D. RESERVATION WINDOW

As mentioned in Section II-B, to bound the computational complexity, a request can only search the path within a given number of layers (i.e., L_R). The value of L_R implies how far ahead in time requests are able to reserve network resources. Thus, when redundant network states (i.e., layers) are condensed, the ability of requests to reserve the resources in the future changes correspondingly.

To study this, the reservation window is defined as the time interval between the topmost layer and the L_R-th layer (i.e., the last layer that can be used for routing). The window size (i.e., the time span of reservation window) suggests how far ahead in time the requests can reserve network resources. A larger window size increases the chances of finding viable paths for requests. The window size is related to both L_R and the time interval between the layers.

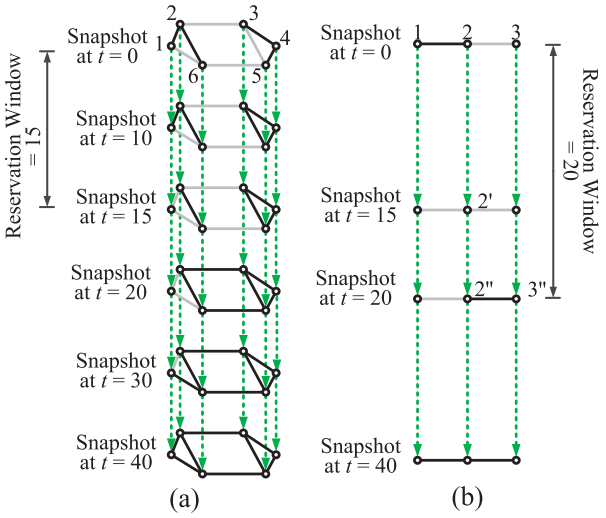


FIGURE 16. Reservation windows in (a) the original TS-MLG G and (b) the condensed TS-MLG G'' when $L_R = 3$.

We now illustrate how the reservation window size increases in the condensed TS-MLG G'' when compared to the original TS-MLG G, as depicted in Fig. 16.

Assume L_R is equal to 3. In Fig. 16(a), the reservation window offered by the joint method is the time interval between the topmost layer and the third layer, i.e., the layer at $t = 0$ and the layer at $t = 15$. Thus, the window size is 15. Similarly, in Fig. 16(b), the reservation window offered by the TSD method is also the time interval between the topmost layer and the third layer. However, because both the layers at $t = 10$ and $t = 30$ are removed, the third layer becomes the layer at $t = 20$ instead of that at $t = 15$. The time interval between the topmost layer and the third layer increases by condensing the redundant layers. Thus, although both methods have the same L_R, the TSD method can provide a wider reservation window.

Consider a request r from node 1 to node 3 arrives at the network at $t = 0$. In Fig. 16(a), as link 2-3 remains busy within the entire reservation window, r fails to find any viable path from node 1 to node 3 and hence is blocked. By contrast, in Fig. 16(b), with a larger window size, r could be routed along path 1-2-2'-2''-3''. As we can see, a wider reservation window can benefit the blocking performance, which will be further discussed in Section V-D. However, when the value of L_R is small, Algorithm 2 has a smaller chance of condensing multiple redundant layers. For example, if $L_R = 2$, the reservation windows in both methods would be 10, because no redundant layer could be condensed. Thus, the larger value of L_R, the larger the chance of condensing multiple layers, the wider the reservation window.

E. COMPUTATIONAL COMPLEXITY

The complexity of our prior method is $O((V \cdot L_R)^2)$, where V denotes the number of nodes in a network topology [15]. In Algorithm 1, Algorithm 3 is performed to find a viable path within G'' (line 7). The complexity of the TSD method depends on the complexity of Algorithm 3, which is

determined by the hop count of a viable path. Furthermore, the hop count is determined by both N_k and L_R. In the worst case, a request will be stored at the source s until layer L_R and then routed to reach the destination $N_{N_k}^{(L_R)}$. In other words, one of the longest paths in G'' could be path $N_1^{(1)}$-$N_1^{(2)}$-...-$N_1^{(L_R)}$-$N_2^{(L_R)}$-...-$N_{N_k}^{(L_R)}$. In the worst case, line 7 in FS will be run $L_R - 1 + N_k - 1$ times and line 9 will be run $L_R - 1$ times. Thus, the complexity of the TSD method is $O(K \cdot (L_R + N_k))$. In Section V, simulations show that the TSD method can obtain high performance, given a small value of K, e.g., $K = 3$. In this case, the complexity can be approximately $O(L_R + N_k)$. Compared to the quadratic complexity in the joint methods, e.g., $O((V \cdot L_R)^2)$, the TSD method reduces its complexity to linear.

V. EVALUATIONS AND DISCUSSIONS

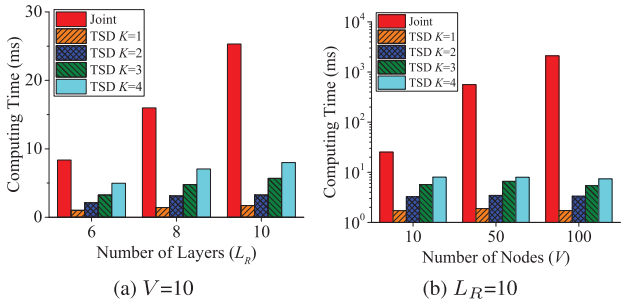
In this section, we simulate dynamic network environments in Matlab to compare the TSD method with the conventional joint method.

The assumptions in Section IV-A are adopted. The NSFNET topology is used. Each request randomly selects a source-destination pair from the network topology. Request arrivals follow a Poisson process with an arrival rate of λ requests per unit time. Request durations follow the negative exponential distribution with an average of $1/\mu$ time units. Thus, the traffic load ρ is equal to λ/μ. L_R denotes the number of layers used for routing. Let w denote the number of wavelengths per link and K denote the number of shortest alternate routes for each node pair. For simplicity, the storage capacity constraint on each node is assumed to be unlimited. Spatial and temporal links are assigned with the same cost, i.e., 1 unit. The routing problem herein is therefore formulated into a shortest-hop-count problem. Each simulation generates 50,000 requests and 20 independent simulations are performed to obtain the average results.

A. COMPUTING TIME

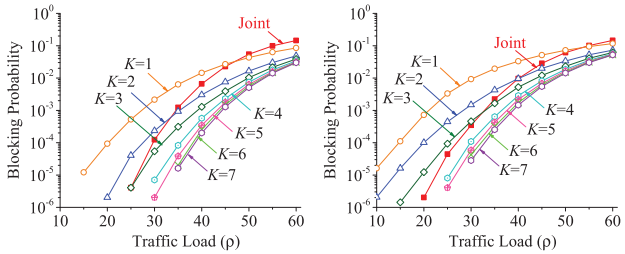
Section IV-E shows that the TSD method reduces the quadratic complexity of the joint method to linear complexity. To demonstrate this, we investigate the computing time of both methods to perform the path selection on a prespecified TS-MLG. Network topologies are randomly generated with density 0.6. Here, we define the density as the probability of an edge between any two nodes. Let V denote the number of nodes in a random topology and L_R denote the number of layers in the prespecified TS-MLG.

Fig. 17 shows the impact of V and L_R on the computing time of both methods. Fig. 17(a) shows that when V remains 10, the computing time of both methods increases with L_R. The increase in the joint method is more evident than that in the TSD method. Fig. 17(b) shows that when L_R remains 10, the computing time of the joint method increases with V, while that of the TSD method remains almost constant. This occurs because by decoupling the problem, the complexity of the TSD method can be reduced to $O(L_R + N_k)$, which is independent of V. Therefore, compared to the joint method, the TSD method obtains lower complexity, especially in a large-scale network.



(a) V=10 (b) L_R=10

FIGURE 17. Computing time under various values of (a) L_R and (b) V.



(a) L_R=8 (b) L_R=5

FIGURE 18. Blocking probability under various traffic load ρ. (w = 5).

B. TRAFFIC LOAD

Then, we investigate how the blocking probability varies with ρ. Generally, ρ can be increased by either increasing λ or by decreasing μ. Our prior work [21] demonstrated that the results obtained in both cases were similar. Thus, in the following simulations, $\lambda = 1$, and we increase ρ by varying μ.

Fig. 18 shows the simulation results given $w = 5$. In Fig. 18(a), given $L_R = 8$, both methods yield a blocking probability of 0, when $\rho = 10$. With ρ increasing, request blocking begins to occur in the cases of the TSD method with $K = 1$ and 2. When ρ increases up to 25, request blocking occurs in the TSD method with $K = 3$ and in the joint method. However, the increase in blocking probability is more significant in the joint method than in the TSD method. When ρ increases up to 25, the TSD method with $K = 3$ outperforms the joint method. Similarly, when ρ increases beyond 35 and 50, the TSD method with $K = 1$ and 2, respectively, outperforms the joint method. The result in Fig. 18(b) follows similar trends, but the TSD method outperforms the joint method at larger values of ρ. This result occurs because L_R is smaller in Fig. 18(b) than in Fig. 18(a). Algorithm 2 has a smaller chance of condensing multiple redundant layers in the case of $L_R = 5$ than in the case of $L_R = 8$, as discussed in Section IV-D. As a result, requests have narrower reservation windows in the case of $L_R = 5$ than in the case of $L_R = 8$.

Besides, the larger value of K, the lower the blocking probability, as seen in Fig. 18. However, since K shortest routes are pre-computed, the value larger value of K, the more the long alternate routes introduced. These longer alternate routes provide less performance benefits while imposing more

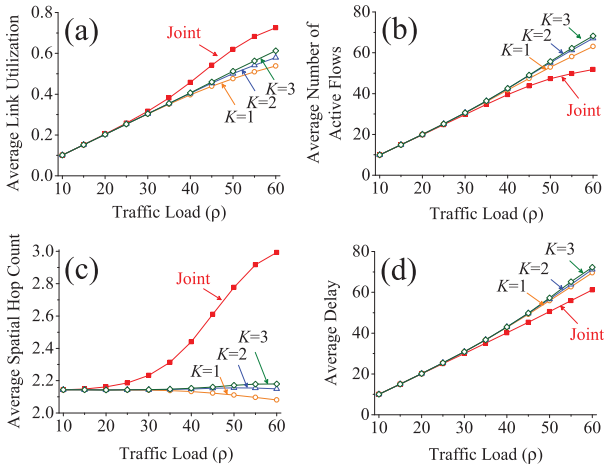


FIGURE 19. Network performance: (a) average link utilization, (b) average number of active flows, (c) average spatial hop count, and (d) average delay.

computational burden, as discussed in Section III-D. Therefore, we focus on small values of K in this paper, say, $K \leq 3$.

C. NETWORK PERFORMANCE

The joint method performs quite well when the traffic load is low. However, when the traffic load is moderate-to-high, the TSD method can provide performance advantages over the joint method. To understand this finding, we focus on the results in Fig. 18(a), and investigate how various performance metrics vary with ρ. The results are shown in Fig. 19.

In Fig. 19(a), the joint method outperforms the TSD method in the average link utilization metric when ρ increases beyond 30. However, by contrast, the increase in the average number of active flows is greater in the TSD method when compared to the joint method for ρ larger than 30, as seen in Fig. 19(b). When the traffic load is moderate-to-high, the network can accommodate more requests while maintaining lower utilization in the TSD method, when compared to the joint method. In other words, the TSD method yields higher bandwidth efficiency than the joint method. This result is caused by detours. In the joint method, since the entire network topology can be used for routing, a request may choose a long-spatial-hop path as long as the path reaches the destination. As large amounts of bandwidth are occupied by a few requests, the scheduler finds it difficult to find available bandwidth for ongoing requests.

On the other hand, in the TSD method, requests can only be routed on K shortest routes, which naturally bounds the spatial hop count of paths, thereby mitigating the detour issue. Fig. 19(c) illustrates that the average spatial hop count greatly increases in the joint method than in the TSD method. With requests that need a long-spatial-hop detour being blocked, more bandwidth remains available for accommodating other requests. The TSD method can use bandwidth more effectively than the joint method, but at the cost of longer delay, as shown in Fig. 19(d). Herein, delay is defined as the time between the request arrival instant and the end of data transfer. Average delay is averaged over all successful requests.

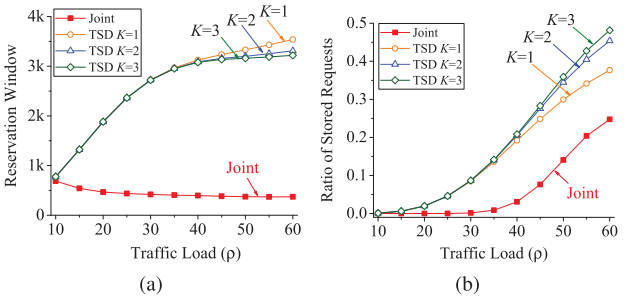


FIGURE 20. (a) Reservation window and (b) ratio of stored requests under different ρ.

D. RESERVATION WINDOW

Next, we investigate how the reservation window varies with ρ. In Fig. 20(a), significant increases in the window sizes are observed in the TSD method, while a slight decrease in that is observed in the joint method. This happens because the network reconfigurations become more frequent when ρ increases. In other words, the time intervals between the network states as well as the time intervals between the layers in the TS-MLG decrease with ρ. Consequently, for a given L_R, the reservation window slightly narrows with ρ in the joint method. By contrast, benefiting from condensing the redundant layers, the TSD method can search the layers further ahead in time and can provide a wider window to reserve more resources in the future, when compared to the joint method, as discussed in Section IV-D. Thus, more requests can be served through SnF. Correspondingly, the ratio of stored requests increases with ρ, as shown in Fig. 20(b). The ratio of stored requests is defined as the ratio between the number of the stored requests and the total number of requests.

E. SNF OPERATIONS

Generally, the more the SnF operations involved in scheduling, the heavier the control and management burden on the network. Besides, each SnF operation would introduce an expensive OEO conversion as well as extra power consumption and storage usage [11]. Thus, we investigate the number of SnF operations. The results are depicted in Fig. 21. With ρ increasing, more requests need SnF operations to reach their destinations. However, majority of the stored requests only need one or two SnF operations. For example, when $\rho = 20$, over 99% of requests in both methods are transferred without SnF. This percentage decreases to 80% when $\rho = 60$. The percentages of requests that need more than three SnF operations are less than 0.03% in both the methods when $\rho = 60$. This suggests that the extra complexity and cost introduced by SnF operations are slight.

F. LINK CAPACITY

Here, we investigate the impact of the number of wavelengths per link (w), i.e., link capacity. Herein, $\rho = 40$ and $L_R = 5$. As seen in Fig. 22(a), the blocking probability decreases with w increasing. When the value of w is small, the TSD method

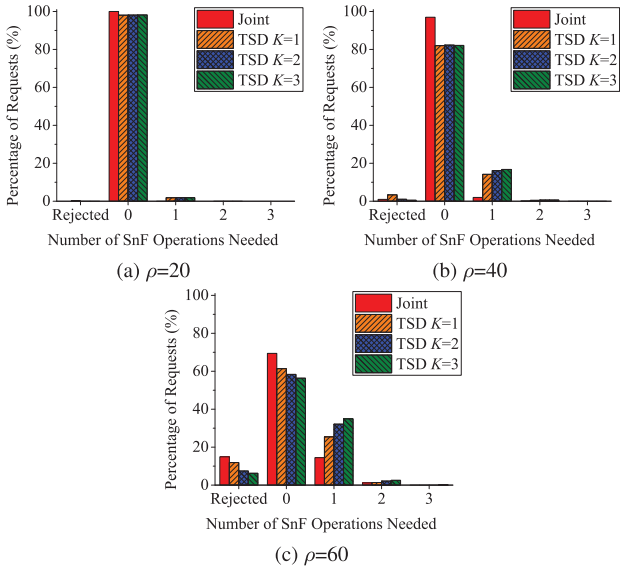


FIGURE 21. Percentage of requests that need SnF operations under various ρ, given $w = 5$ and $L_R = 8$.

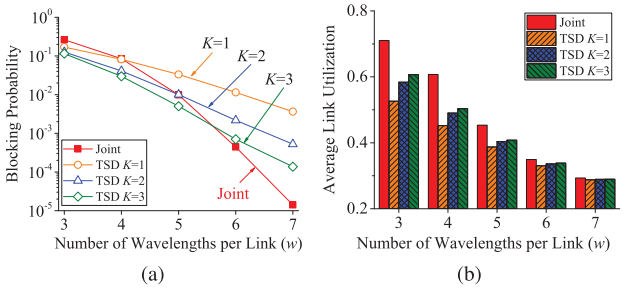


FIGURE 22. (a) Blocking probability and (b) link utilization under various w.

outperforms the joint method. When $w = 6$, the joint method obtains better blocking performance than the TSD method.

To understand this, we further investigate the link utilization under various w. The results are shown in Fig. 22(b). When the link capacity is small, e.g., $w = 3$, the network is difficult to accommodate new requests since the link capacity is insufficient. However, the link utilization is higher in the joint method than in the TSD method. Thus, compared to the TSD method, the impact of the detour issue on the joint method is more evident, since the joint method searches in the entire network. When the link capacity increases, e.g., $w = 6$, the network still has sufficient resources to accommodate new requests, even when some requests detour via long-spatial-hop paths. In this case, the impact of the detour issue on both methods is slight, since the link utilization in both methods is low. However, in the TSD method, requests are routed via a few pre-computed routes. This results in high utilization in some bottleneck links, but low average utilization of the network. A transient burst of request arrivals could more easily cause request blocking in the TSD method than in the joint method. Thus, when the link capacity is sufficient and the detour issue is not a concern, the joint method provides more flexibility in scheduling and hence outperforms the TSD method.

VI. CONCLUSION

In this paper, we developed analytic models to study the complexity-performance tradeoff in the SnF scheduling problem. Our studies showed that the joint solution could provide performance advantages over the decoupled solution, but at the cost of higher complexity. The studies further revealed that short routes could provide more performance benefits while imposing less computational burden, when compared to long-spatial-hop routes. This implied that a decoupled solution had the potential to strike this tradeoff by considering a few pre-selected routes. Thus, we proposed a TSD scheduling method. By decoupling the scheduling problem, the TSD method obtains the linear complexity, when compared to the quadratic complexity in the joint method. By condensing the redundant network states, the TSD method can provide wider reservation windows to schedule more resources in the future. When the traffic load is moderate-to-high, the joint method suffers from the detour issue, since a few long-detoured requests occupy a large amount of network resources. In contrast, the TSD method delivers requests over K pre-selected routes. This naturally bounds the spatial hop count of routing paths and hence uses bandwidth more efficiently. Simulation results show that the TSD method outperforms the joint method, especially when the traffic load is moderate-to-high. Overall, the joint solutions are effective for small networks or for static optimization. For larger networks and online scheduling, decoupling the problem is more practical to reduce complexity.

APPENDIXES

APPENDIX A PROOF OF EQUATION 1

To obtain the expression of $P_{(s,d)}(N, L)$, we first illustrate its decomposition feature through an example. Let us start by considering a simple model of a two-node route with L layers, i.e., $P_{(s,d)}(2, L)$, as shown in Fig. 23(a). Because the route has only a spatial hop, the number of feasible paths from the source s to the destination d is L. Thus, we get $P_{(s,d)}(2, L) = L$. Without loss of generality, the destinations in different layers, i.e., $d^{(1)}$, ..., $d^{(L)}$, are connected via temporal links. However, a feasible path cannot traverse multiple destinations. For instance, in Fig. 23(a), a path s-i_1-$d^{(1)}$-$d^{(2)}$ does not exist.

Now, let us consider a model of a three-node route with L layers, i.e., $P_{(s,d)}(3, L)$. As shown in Fig. 23(b), $P_{(s,d)}(3, L)$ is more complex than $P_{(s,d)}(2, L)$, and its expression seems to be difficult to obtain directly. However, $P_{(s,d)}(3, L)$ can be decomposed into L various cases, with source i_1 and destination d, as shown in Fig. 23(c). In each case, the two-node-route model has various l, where $1 \leq l \leq L$. The value of l depends on how many temporal links have been traversed by a feasible path before the first spatial hop. By decomposing the three-node model into L two-node models, we get $P_{(s,d)}(3, L) = \sum_{l=1}^{l=L} P_{(i_1,d)}(2, l)$. Similarly, $P_{(s,d)}(N, L)$ can be decomposed into L various cases of a (N-1)-node route. Thus, we have $P_{(s,d)}(N, L) = \sum_{l=1}^{l=L} P_{(i_1,d)}(N-1, l)$, if $N \geq 3$.

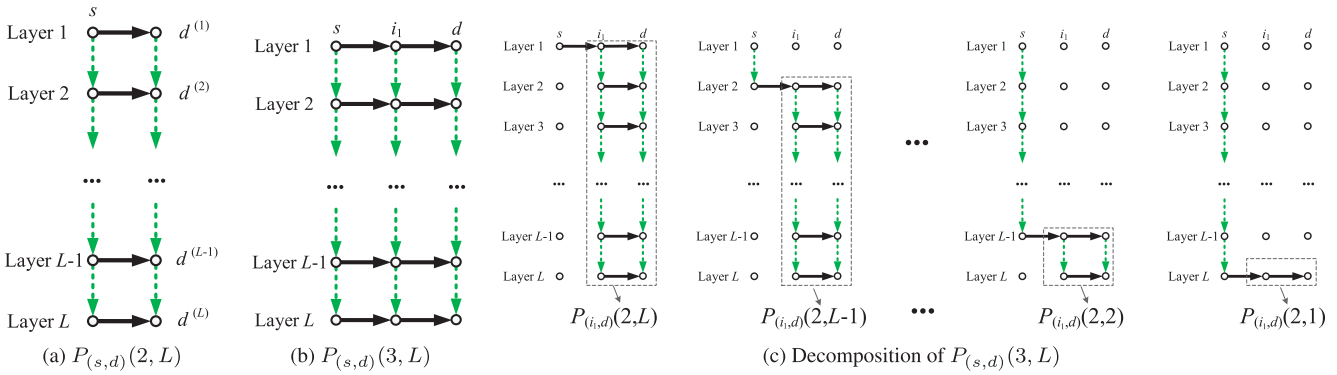


FIGURE 23. TS-MLGs of $P_{(s,d)}(2, L)$ and $P_{(s,d)}(3, L)$. Decompose $P_{(s,d)}(3, L)$ into L different cases of $P_{(i_1,d)}(2, l)$.

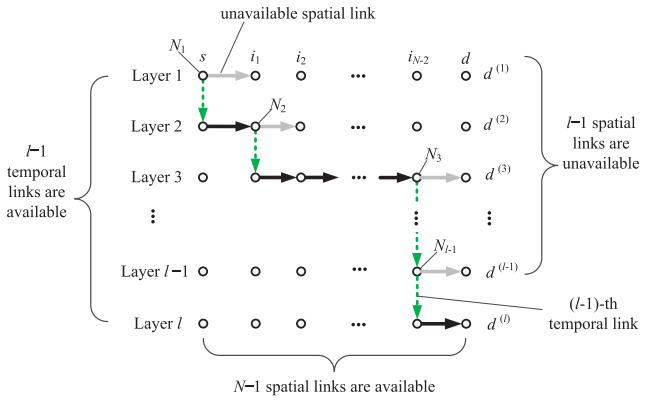


FIGURE 24. TS-MLG of the l-case in $F^{Upper}_{(s,d)}(N, L)$.

APPENDIX B PROOF OF EQUATION 2

The key idea of the successful path method [33] is based on the following concept: The system fulfills its required function if there is at least one path between the input and the output upon which all elements perform their required function. Inspired by this, we derive $F^{Upper}_{(s,d)}(N, L)$ by reducing the fixed-route model shown in Fig. 4(a) into L cases. In the l-th case, a request attempts to search a viable path from s to $d^{(l)}$, where $l \in [1, L]$. In addition, we assume that a request would start to search in the l-th case only when it fails in all the previous l-1 cases, i.e., it cannot find any viable path for $(s, d^{(\boldsymbol{l})})$, where $\boldsymbol{l} = [1, \ldots, l-1]$.

Fig. 24 shows an example where a request succeeds in the l-th case. While there exist various feasible paths that can reach $d^{(l)}$, the common features shared by them are as follows: (i) each feasible path consists of $N-1$ spatial links and $l-1$ temporal links; (ii) there exists at least an unavailable spatial link on each previous layer (i.e., layer $1 \sim$ layer $l-1$), which leads to the reservation failure in the previous l cases. For simplicity, we assume that the unavailable spatial link at each previous layer is the next spatial hop of the last available spatial link, as shown in Fig. 24. Therefore, given a feasible path from s to $d^{(l)}$, the probability that a request succeeds on this path is equal to $(1-p_s)^{l-1}(1-p_b)^{N-1}(p_b)^{l-1}$.

Then, we investigate how many feasible paths exist between s and $d^{(l)}$. As previously mentioned, routing options for every node in Fig. 24 are confined to either next spatial hop or to next temporal hop. Thus, the feasible paths would diverge at each node before reaching $d^{(l)}$. Since there are l-1 temporal links along each feasible path, the total number of feasible path for $(s, d^{(l)})$ is determined by various locations of the l-1 temporal links.

Let N_t denote the node connected by the t-th temporal link, where $t \in [1, l-1]$ and $N_t \in [s, i_1, \ldots, i_{N-2}]$. More specifically, the t-th temporal link connects node N_t at layer t to node N_t at layer $t+1$. Due to the limited routing options, we have $N_1 \leq \ldots \leq N_t \leq \ldots \leq N_{l-1}$. Thus, the problem of the total number of feasible path for $(s, d^{(l)})$ is formulated into a combinations problem, i.e., selecting l-1 nodes out of N-1 nodes, which should satisfy $N_1 \leq \ldots \leq N_t \leq \ldots \leq N_{l-1}$. This is a classical combination of multisets problem [38]. Therefore, the total number of feasible path for $(s, d^{(l)})$ is equal to $\binom{N+l-3}{l-1}$. We further obtain the probability that a request succeeds in the l-th case shown in Fig. 24, which is equal to $(1-p_s)^{l-1}(1-p_b)^{N-1}(p_b)^{l-1}\binom{N+l-3}{l-1}$. Finally, We get the probability that a request fails to find any viable path from the L cases, i.e., $F^{Upper}_{(s,d)}(N, L) = 1 - \sum_{l=1}^{L}(1-p_s)^{l-1}(1-p_b)^{N-1}(p_b)^{l-1}\binom{N+l-3}{l-1}$.

APPENDIX C PROOF OF EQUATION 3

Let us start by considering simple models of $P_{(s,d)}(N, 1)$ and $P_{(s,d)}(2, L)$, as shown in Fig. 25. In Fig. 25(a), there exists only a feasible path for the node pair (s, d), where $N-1$ spatial links are connected in series. We get $F^{Lower}_{(s,d)}(N, 1) = 1-(1-p_b)^{N-1}$. The more the spatial hops, the higher the value of $F^{Lower}_{(s,d)}(N, 1)$. In other words, requests are more difficult to find a viable path on a long-spatial-hop route. In Fig. 25(b), there exists L feasible paths for (s, d). For the l-th feasible path, it consists of $l-1$ temporal links and a spatial link, which are connected in series. We get the reservation failure probability of the l-th feasible path, which equals to $1-(1-p_s)^{l-1}(1-p_b)$. For all the L feasible paths in $P_{(s,d)}(2, L)$, they share some common temporal links. Thus, their reservation failure probability should be dependent. For simplicity, we assume that they are independent and are connected in parallel, which, in turn, presents a lower bound of $F^{Lower}_{(s,d)}(2, L)$. We obtain $F^{Lower}_{(s,d)}(2, L) = \prod_{l=1}^{l=L}[1-(1-p_s)^{l-1}(1-p_b)]$. The larger the value of L, the more the layers

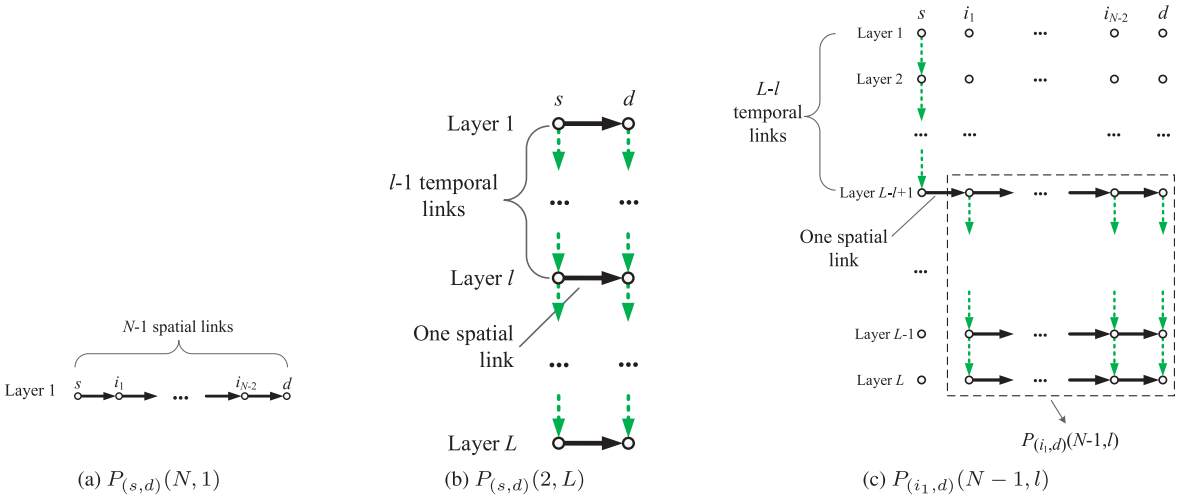


FIGURE 25. TS-MLGs of $P_{(s,d)}(N, 1)$, $P_{(s,d)}(2, L)$ and the i-case when decomposing $P_{(s,d)}(N, L)$.

in $P_{(s,d)}(2, L)$, the lower value of $F^{Lower}_{(s,d)}(2, L)$. Increasing the number of layers used for routing can make requests easier to find viable paths.

$P_{(s,d)}(N, L)$ could be decomposed into L cases of $P_{(i_1,d)}(N-1, l)$, where $1 \le l \le L$. In the l-th case, all the feasible paths traverse $L-l$ temporal links and a spatial link before reaching node i_1. Then, they diverge in $P_{(i_1,d)}(N-1, l)$, as shown in Fig. 25(c). $L-l$ temporal links, a spatial link and $P_{(i_1,d)}(N-1, l)$ are connected in series. Thus, we have the reservation failure probability of the l-th case, i.e., $1-(1-p_s)^{L-l}(1-p_b)(1-F_{(i_1,d)}(N-1, l))$. Assuming these L cases are independent and are connected in parallel, we present a lower bound of $F^{Lower}_{(s,d)}(N, L)$. Thus, we get $F^{Lower}_{(s,d)}(N, L) = \prod_{l=1}^{l=L}[1-(1-p_s)^{L-l}(1-p_b)(1-F_{(i_1,d)}(N-1, l))]$, if $N \ge 3$.

REFERENCES

[1] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang, "Guaranteeing deadlines for inter-data center transfers," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 579–595, Feb. 2017.

[2] W. Li, X. Zhou, K. Li, H. Qi, and D. Guo, "TrafficShaper: Shaping inter-datacenter traffic to reduce the transmission cost," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1193–1206, Jun. 2018.

[3] W. Hu, J. Liu, T. Huang, and Y. Liu, "A completion time-based flow scheduling for inter-data center traffic optimization," *IEEE Access*, vol. 6, pp. 26181–26193, 2018.

[4] M. Noormohammadpour and C. S. Raghavendra, "Datacenter traffic control: Understanding techniques and tradeoffs," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1492–1525, 2018.

[5] S. M. Srinivasan, T. Truong-Huu, and M. Gurusamy, "Deadline-aware scheduling and flexible bandwidth allocation for big-data transfers," *IEEE Access*, vol. 6, pp. 74400–74415, 2018.

[6] M. Noormohammadpour, C. S. Raghavendra, S. Kandula, and S. Rao, "QuickCast: Fast and efficient inter-datacenter transfers using forwarding tree cohorts," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 225–233.

[7] S. Ji, S. Liu, and B. Li, "Deadline-aware scheduling and routing for inter-datacenter multicast transfers," in *Proc. IEEE Int. Conf. Cloud Eng. (ICE)*, Apr. 2018, pp. 124–133.

[8] L. Tong, X. Zheng, Y. Xia, and M. Li, "Delay tolerant bulk transfers on inter-datacenter networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2016, pp. 1–6.

[9] N. Laoutaris, G. Smaragdakis, R. Stanojevic, P. Rodriguez, and R. Sundaram, "Delay-tolerant bulk data transfers on the Internet," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1852–1865, Dec. 2013.

[10] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Aug. 2013. [Online]. Available: http://doi.acm.org/10.1145/2534169.2486019

[11] P. Lu and Z. Zhu, "Data-oriented task scheduling in fixed- and flexible-grid multilayer inter-DC optical networks: A comparison study," *J. Lightw. Technol.*, vol. 35, no. 24, pp. 5335–5346, Dec. 15, 2017.

[12] C. Lee and J.-K.-K. Rhee, "Efficient design and scalable control for store-and-forward capable optical transport networks," *J. Opt. Commun. Netw.*, vol. 9, no. 8, p. 699, Aug. 2017.

[13] C. Sun, W. Guo, Z. Liu, M. Xia, and W. Hu, "Performance analysis of storage-based routing for circuit-switched networks," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 8, no. 5, p. 282, May 2016.

[14] Y. Wang, S. Su, A. X. Liu, and Z. Zhang, "Multiple bulk data transfers scheduling among datacenters," *Comput. Netw.*, vol. 68, pp. 123–137, Aug. 2014.

[15] X. Lin, W. Sun, M. Veeraraghavan, and W. Hu, "Time-shifted multilayer graph: A routing framework for bulk data transfer in optical circuit-switched networks with assistive storage," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 8, no. 3, pp. 162–174, Mar. 2016. [Online]. Available: http://jocn.osa.org/abstract.cfm?URI=jocn-8-3-162

[16] X. Lin, W. Sun, M. Veeraraghavan, and W. Hu, "Slotted store-and-forward optical circuit-switched networks: A performance study," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 9, no. 7, pp. 563–576, Jul. 2017. [Online]. Available: http://jocn.osa.org/abstract.cfm?URI=jocn-9-7-563

[17] W. Sun, S. Yue, Q. Liu, X. Lin, and W. Hu, "Bulk data transfer with store-and-forward in wide area networks," in *Proc. 21st Int. Conf. Transp. Opt. Netw. (ICTON)*, Jul. 2019, pp. 1–4.

[18] Y. Mori, M.-E. Ganbold, and K.-I. Sato, "Design and evaluation of optical circuit switches for intra-datacenter networking," *J. Lightw. Technol.*, vol. 37, no. 2, pp. 330–337, Jan. 15, 2019.

[19] Z. Li and H. Shen, "Co-scheduler: Accelerating data-parallel jobs in datacenter networks with optical circuit switching," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 186–195.

[20] P. J. Argibay-Losada, G. Sahin, K.-I. Kitayama, and C. Qiao, "On whether OCS maximizes application throughput in all-optical datacenter networks," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 7, no. 12, pp. 1135–1147, Dec. 2015.

[21] X. Lin, X. Wang, S. Yue, W. Sun, M. Veeraraghavan, and W. Hu, "Design of an SnF scheduling method for bulk data transfers over inter-datacenter WANs," in *Proc. IEEE Int. Conf. High Perform. Switching Routing (HPSR)*, May 2019, pp. 1–8.

[22] G. Iosifidis, I. Koutsopoulos, and G. Smaragdakis, "Distributed storage control algorithms for dynamic networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1359–1372, Jun. 2017.

[23] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, p. 74, Oct. 2011.

[24] Y. Wu, Z. Zhang, C. Wu, C. Guo, Z. Li, and F. C. M. Lau, "Orchestrating bulk data transfers across geo-distributed datacenters," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 112–125, Jan. 2017.

[25] X. Wang, M. Veeraraghavan, Z. Lin, and E. Oki, "Optical switch in the middle (OSM) architecture for DCNs with Hadoop adaptations," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–7.
[26] X. Wang, X. Lin, W. Sun, and M. Veeraraghavan, "Comparison of two sharing modes for a proposed optical enterprise-access SDN architecture," in *Proc. Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Nov. 2018, pp. 1–8.
[27] Y. Feng, B. Li, and B. Li, "Postcard: Minimizing costs on inter-datacenter traffic with store-and-forward," in *Proc. 32nd Int. Conf. Distrib. Comput. Syst. Workshops (ICDCS Wkshps)*, 2012, pp. 43–50.
[28] Y. Li, H. Wang, P. Zhang, J. Dong, and S. Cheng, "D4D: Inter-datacenter bulk transfers with ISP friendliness," in *Proc. IEEE Int. Conf. Cluster Comput.*, 2012, pp. 597–600.
[29] A. N. Patel, Y. Zhu, and J. P. Jue, "Routing and horizon scheduling for time-shift advance reservation," in *Proc. Opt. Fiber Commun. Conf. Nat. Fiber Opt. Eng. Conf. (OFC/NFOEC)*, 2009, pp. 1–3, Paper OThO4. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2009-OThO4
[30] A. Patel, Z. Yi, S. Qingya, and J. Jue, "Routing and scheduling for time-shift advance reservation," in *Proc. 18th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2009, pp. 1–6.
[31] A. Patel, M. Tacca, and J. P. Jue, "Time-shift circuit switching," in *Proc. Opt. Fiber Commun. Conf. Nat. Fiber Opt. Eng. Conf. (OFC/NFOEC)*, 2008, pp. 1–3, Paper OThI6. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2008-OThI6
[32] D. Feng, W. Sun, and W. Hu, "Joint provisioning of lightpaths and storage in store-and-transfer wavelength-division multiplexing networks," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 9, no. 3, pp. 218–233, Mar. 2017.
[33] A. Birolini, *Reliability Engineering: Theory and Practice*. New York, NY, USA: Springer-Verlag, 1999.
[34] X. Chu and B. Li, "Dynamic routing and wavelength assignment in the presence of wavelength conversion for all-optical networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, pp. 704–715, Jun. 2005.
[35] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The science DMZ: A network design pattern for data-intensive science," in *Proc. IEEE/ACM Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC)*, Nov. 2013, pp. 1–10.
[36] X. Wang, M. Veeraraghavan, M. Brandt-Pearce, T. Miyazaki, N. Yamanaka, S. Okamoto, and I. Popescu, "A dynamic network design for high-speed enterprise access links," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–7.
[37] C. Castillo, G. N. Rouskas, and K. Harfoush, "On the design of online scheduling algorithms for advance reservations and QoS in grids," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, Mar. 2007, pp. 1–10.
[38] O. Sokratova. (2008). *Combinations of Multisets*. [Online]. Available: http://homepage.divms.uiowa.edu/ sokratov/2008m150/multicomb.pdf/

XIAO LIN (Member, IEEE) received the Ph.D. degree in information and communication engineering from Shanghai Jiao Tong University. He was a Visiting Scholar of the Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia. He is currently an Assistant Professor with the College of Physics and Information Engineering, Fuzhou University (FZU). His research interests include intelligent edge computing, optical switches, reliability, and bulk data transfer over optical networks.

WEIQIANG SUN (Senior Member, IEEE) is currently a Full Professor with the Department of Electronic Engineering, Shanghai Jiao Tong University (SJTU). He also works with the Shanghai Institute for Advanced Communication and Data Science, Shanghai, China. He is actively involved in the research of high speed networks, network control and management, and network applications. He has about 60 publications in peer-reviewed journals and conferences. He was actively involved in the standardization of Generalized MultiProtocol Label Switching (GMPLS) performance measurement in IETF. He is the Co-Editor of RFC 6777 and RFC 5814. He has served as an Invited Speaker on many international conferences and is member of Technical Program Committee of COIN, GreenTouch, PGC and organizer of the Sino-Korea Workshop on IPTV and NGN, from 2007 to 2010. He is one of the TPC co-chairs of IEEE HPSR 2017.

XIAOYU WANG is currently pursuing the Ph.D. degree with the Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia (UVA). Her research interests include dynamic optical networking, data-center networks, bulk data transfers, distributed systems, and network optimization.

SHENGNAN YUE (Student Member, IEEE) is currently pursuing the Ph.D. degree in electronic information and electrical engineering with Shanghai Jiao Tong University (SJTU). Her research interests include bulk data transfer and delay-tolerant networks.

MALATHI VEERARAGHAVAN (Fellow, IEEE) received the B.Tech. degree from the Indian Institute of Technology Madras, and the M.S. and Ph.D. degrees from Duke University. After receiving the Distinguished Member of Technical Staff award and a ten-year career at Bell Laboratories, she joined the Faculty at Polytechnic University, Brooklyn, New York, where she was an Associate Professor of Electrical Engineering, from 1999 to 2002. She joined the University of Virginia, in 2003, and is currently a Professor of electrical and computer engineering. She is currently a Professor with the Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia (UVA). Her research work has been primarily in high-speed networking, wireless networking, and network security. Her research funding has been mainly from the National Science Foundation, the U.S. Department of Energy, and DARPA. She holds 30 patents, has over 138 publications, and has received six Best paper awards. She served as the Technical Program Committee Co-Chair for the High-Speed Networking Symposium in IEEE ICC 2013, as a Technical Program Committee Chair for IEEE ICC 2002, and an Associate Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING. She was an Associate Editor of the IEEE TRANSACTIONS ON RELIABILITY, from 1992 to 1994.

WEISHENG HU (Senior Member, IEEE) received the B.Sc. degree from Tsinghua University, in 1986, the M.Eng. degree from the University of Science and Technology Beijing, in 1989, and the Ph.D. degree from Nanjing University, in 1997. He has been a Postdoctoral Fellow with Shanghai Jiao Tong University, since 1997, and as a Professor, since 1999, where he was promoted to Distinguished Professor, in 2009. He has served as the Deputy Director and the Director of the State Key Laboratory of Advanced Optical Communication Systems and Networks, from 2002 to 2012. He has published about 400 peer-reviewed journal/conference papers. He serves on several editorial boards, including *Optics Express*, the *Journal of Lightwave Technology*, *Chinese Optics Letters*, and *China Communications*. He has served on the program committees of a number of international conferences, including OFC, ICC, and INFOCOM.