Cluster Computing (2022) 25:3019–3034
https://doi.org/10.1007/s10586-021-03364-7

SDN-based bandwidth scheduling for prioritized data transfer between data centers

Aiqin Hou[1] · Chase Q. Wu[2] · Qiang Duan[3] · Dawei Quan[1] · Liudong Zuo[4] · Yangyang Li[1] · Michelle M. Zhu[5] · Dingyi Fang[1]

Received: 22 January 2021 / Revised: 12 July 2021 / Accepted: 18 July 2021 / Published online: 19 August 2021

Abstract

The widespread deployment of scientific applications and business services of various types on clouds requires the transfer of big data with different priorities between geographically distributed cloud-based data centers. As a result, Cloud Service Providers (CSP) face a significant challenge to fully utilize the expensive bandwidth resources of the links connecting data centers while guaranteeing Quality of Experience (QoE) for users. Modern data centers are increasingly adopting Software-Defined Networking (SDN) technology, which provides the capability of advance bandwidth reservation. This paper focuses on the collaborative scheduling of multiple prioritized user requests, namely, advance bandwidth reservation with a lower priority and immediate bandwidth reservation with a higher priority, to maximize the total user satisfaction. We formulate this co-scheduling problem with preemption as a generic optimization problem, which is shown to be NP-complete. We design a heuristic algorithm to maximize the number of successfully scheduled requests and minimize the number of preempted advance reservation requests, while minimizing the completion time of each request. Extensive results from simulations with randomly generated networks and emulation-based experiments on an SDN testbed show that our scheduling scheme significantly outperforms greedy approaches in terms of user satisfaction degree, a normalized quantification parameter we define to measure users' QoE.

Keywords Big data transfer · High-performance networks · Software-defined networks · Quality of Experience · Bandwidth reservation

Some preliminary results in this manuscript have been published in INDIS'19 in conjunction with SC'19 [1].

✉ Chase Q. Wu
chase.wu@njit.edu

Aiqin Hou
houaiqin@nwu.edu.cn

Qiang Duan
qxd2@psu.edu

Liudong Zuo
lzuo@csudh.edu

Michelle M. Zhu
zhumi@montclair.edu

[1] School of Information Science and Technology, Northwest University, Shaanxi International Joint Research Centre for the Battery-Free Internet of Things, Xi'an 710127, Shaanxi, China

[2] Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA

[3] Information Science and Technology Department, Penn State University, Abington, PA 19001, USA

[4] Computer Science Department, California State University, Dominguez Hills, Carson, CA 90747, USA

[5] Department of Computer Science, Montclair State University, Montclair, NJ 07043, USA

1 Introduction

Over the past decade, we have witnessed an increasing number of business services and scientific applications migrating from local computing platforms to cloud infrastructures, which often comprise geographically distributed data centers. These migrations necessitate the transfer of large amounts of data between cloud data centers for remote storage, analysis, and visualization. In fact, data transfer between cloud data centers has become a vital aspect of cloud computing and has a significant impact on cloud service performance. Consequently, networking for inter-data center data transfer with performance guarantee plays a crucial role in high-performance cloud computing. Various network control and management technologies have been employed for data transfer between cloud data centers, among which, bandwidth reservation offers a key approach to meeting the performance requirements for data transfer.

The wide variety of applications running upon the cloud infrastructure may generate diverse requests for data transfer including various types of data loads with different performance expectations, which brings challenges to bandwidth reservation. The data to be transferred between data centers and their requirements for bandwidth reservation can be generally classified into two common categories:

- Bulk data that may be accumulated over a certain time period and are typically on the order of terabytes to petabytes. Bulk data transfer in general needs to be completed by a certain deadline, and therefore network bandwidth is often reserved in advance to meet the transfer requirement, which is referred to as Advance Reservation (AR). An example of AR for bulk data transfer is to transfer the experimental data (up to 30 petabytes per year) generated by Large Hadron Collider (LHC) to multi-tier collaborating sites at different geographical locations for further processing and analysis.
- Time-critical data that typically have a much smaller size than bulk data but are more delay sensitive. Examples of this type of data include interactive communications for online collaborative analysis and urgent user requests in emergency situations. Transfer of time-critical data requires on-demand bandwidth reservation, referred to as Immediate Reservation (IR).

These two types of bandwidth reservation—AR and IR—for data transfer differ in multiple aspects. AR may achieve more efficient utilization of network resources through careful bandwidth scheduling with a global perspective but lacks prompt responsiveness to user requests. Therefore, AR is an appropriate choice for maximizing the resource utilization and revenue profit of Cloud Service Providers (CSPs). Also, AR may support the Service-Level Agreements (SLAs) between CSPs and users. On the other hand, IR offers an immediate response to on-demand requests from users; however, there is no guarantee on the availability of sufficient bandwidth to meet the stringent performance requirement. Prioritization schemes could be applied to IR for meeting SLAs, but may need to preempt some of the bandwidth previously scheduled for AR requests, thus possibly leading to service degradation or interruption of other users. Hence, an important problem arises to minimize the number of preempted AR requests or the amount of preempted bandwidth to meet diverse data transfer requirements while fully utilizing network bandwidth resources.

The emerging Software-Defined Networking (SDN) technology, which is being widely adopted in both data center networks and inter-cloud networks, offers a promising approach to addressing the challenge of bandwidth reservation for meeting diverse data transfer requirements between data centers. Essentially, SDN decouples the data plane functions for packet forwarding from the control/management plane to enable a logically centralized controller for the entire network domain in support of network programmability. Flow-based packet forwarding in SDN provides an effective mechanism to reserve bandwidth for various types of data transfer required by different users. The central SDN controller with a global view of the entire network domain greatly facilitates decision making for end-to-end path selection and bandwidth allocation to meet the performance requirements specified by users. The network programmability enabled by SDN northbound interface allows upper layer applications to define data transfer operations according to user demands without requiring knowledge about the implementation details of the underlying network infrastructure.

Although exciting progress in SDN control and management has been reported in the literature, existing work mainly focuses on data center networks and wide-area public networks. Supporting both advance and immediate reservations for different data transfer requests between data centers interconnected through SDN still remains largely unexplored.

In this paper, we investigate a collaborative bandwidth scheduling problem, referred to as BS-ARIR, which considers a combination of advance reservation (AR) with a lower priority and immediate reservation (IR) with a higher priority, to support data transfer between data centers interconnected through an SDN-based network. The goal is to maximize the total number of requests (both AR and IR) that have been satisfied and minimize the number of preempted AR requests, both of which are incorporated into a

carefully-defined performance metric, namely, overall user satisfaction (SAT). We show BS-ARIR to be NP-complete and design an efficient heuristic co-scheduling algorithm. We conduct extensive experiments based on simulations and emulations in an SDN environment to evaluate the performance of the proposed algorithm. The experimental results demonstrate that the proposed algorithm significantly outperforms existing algorithms in terms of the overall user satisfaction.

The work in this paper conducts a thorough investigation into bandwidth scheduling of high-priority IR requests and low-priority AR requests, and has potential to support and facilitate inter-cloud data transfer for high-performance cloud services.

The rest of the paper is organized as follows. We first review some representative work on bandwidth scheduling in Sect. 2. Then, we formulate the BS-ARIR problem with complexity analysis in Sect. 3. We design a heuristic co-scheduling algorithm for BS-ARIR in Sect. 4. We present performance evaluation results through simulations and emulation-based experiments in Sect. 5. We conclude our work in Sect. 6.

2 Related work

The emerging SDN paradigm offers a promising network control and management platform to support various bandwidth scheduling strategies. Recent developments in SDN technologies provide a variety of mechanisms that can be employed for realizing traffic flow scheduling or bandwidth scheduling.

There are several studies on traffic flow scheduling within data center networks. SMART [2] is an architectural enhancement aiming to realize resilient transfer for critical flows. It leverages redundancy through SDN and FlowTags middlebox architecture to ensure timely delivery of critical flows. QJump [3] applies Internet QoS-inspired techniques to data center applications and allows critical network flows to jump the queues, hence ensuring the satisfaction of SLA.

Bandwidth scheduling based on SDN for data transfer in an inter-cloud network environment has been extensively studied in the literature. Many efforts have been made to perform AR bandwidth scheduling to guarantee QoE for users. The work in this direction aims to maximize the number of accepted user requests, maximize the ratio of successfully scheduled bandwidth reservation requests (BRR), or minimize the earliest completion time (ECT). In [4], Zuo et al. investigated the problem of intelligent and flexible scheduling to achieve the optimal ratio of successfully scheduled BRRs and the average ECT of scheduled BRRs. In [5], Zuo et al. studied bandwidth scheduling problems in dedicated networks to maximize both the total amount of transferred data and the number of successfully scheduled requests. In [6], to minimize the ECT of a BRR, Lin and Wu investigated bandwidth scheduling with an exhaustive combination of different path and bandwidth constraints including (i) fixed path with fixed bandwidth (FPFB), (ii) fixed path with variable bandwidth (FPVB), (iii) variable path with fixed bandwidth (VPFB), and (iv) variable path with variable bandwidth (VPVB).

Some existing work on AR bandwidth scheduling considers prioritized reservation requests and allows preemption between AR. In [7], Zuo et al. studied the problem of scheduling AR requests with different priorities to minimize the number and then the total bandwidth of existing bandwidth reservations to be preempted, and minimize the total bandwidth and then the number of existing bandwidth reservations to be preempted. Xie et al. proposed a preemption scheme between high- and low-priority AR requests with a fixed transmission window using a heuristic access control algorithm based on an integer linear programming (ILP) model to determine an available path [8].

Bandwidth scheduling for a mixture of AR and IR requests with different priorities has also been studied in the literature. In [9], Dharam investigated the preemption of IR requests by AR requests using the Global Network View (GNV) in SDN and proposed several bandwidth scheduling and preemption schemes to solve the problems of IR preemption and AR blocking.

In this paper, we focus on a collaborative scheduling problem that considers low-priority AR requests reserved in advance and high-priority IR requests scheduled on-demand, and conduct emulation-based experiments on a Mininet-based SDN testbed to evaluate the effectiveness of the proposed bandwidth scheduling algorithm. This bandwidth preemption problem differs from the existing work in that the bandwidth of on-going data transfer for low-priority AR requests may be preempted by the arrival of an IR request with a higher priority. IR requests with higher priority may be made by a variety of applications that require real-time or near real-time data transfer, such as online collaborative analysis, interactive communication, urgent user request/response, and delay-sensitive data transfer. Such applications play a critical role in providing high-performance, on-demand cloud services.

3 Problem formulation

In this section, we first construct cost models for SDN-based big data transfer between data centers, and then formulate a problem of bandwidth scheduling for advance/immediate reservation with complexity analysis.

3.1 Network model

As shown in Fig. 1, an SDN-based networking system for inter-data center transfer consists of three planes. The data plane is composed of a set of OpenFlow-enabled switches and transmission links between them, which form the underlying infrastructure of the network. The control plane provides a network operating system that allows upper layer applications to define the operational behaviors of the network infrastructure. The controller communicates with the data plane via a southbound interface (typically the OpenFlow protocol) to set up and update flow tables in switches to control flow-based packet forwarding. The controller also regularly collects network states, including bandwidth availability on switches and links, and maintains a global view of the entire network domain. A variety of SDN control applications may run upon the operating system thus forming an application layer. The control applications are the actual brain of the SDN network that utilizes the global network view provided by the controller via the northbound interface to make decisions regarding various network operations such as routing and bandwidth allocation.

Considering the scenario of bandwidth management for data transfer between cloud data centers, a bandwidth scheduler may be realized as an SDN control application, which maintains a list of existing bandwidth reservations (including the transfer path, the amount of reserved bandwidth, and the flow active period on the path). Upon receiving a request for bandwidth reservation from a cloud data center, the scheduler exams the current network states and the existing reservations to determine how to make a new reservation to satisfy the request. The scheduler informs the controller once such a reservation decision has been made. The controller then updates the flow tables in all involved switches on the transfer path to set up a flow and configures these switches to allocate the required amount of bandwidth for the flow.

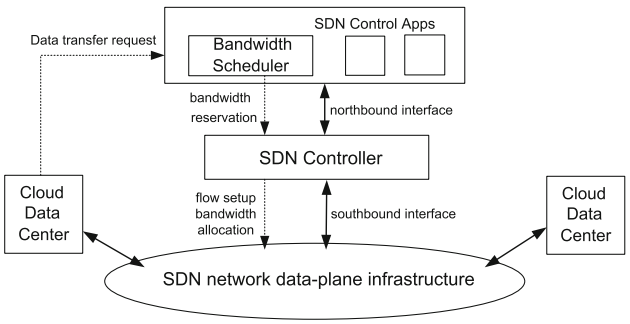


Fig. 1 SDN-based network architecture for data transfer between data centers

We denote a backbone network as a graph $G(V, E)$ with $|V|$ nodes and $|E|$ links. Figure 2 shows an example network graph, where the corresponding bandwidth of each link across different time slots from 0 to 2 is annotated on the link, respectively.

Each link $l \in E$ maintains a list of residual bandwidths specified as a segmented constant function of time [6]. A time-bandwidth (TB) list can be represented as $(t_l[i], t_l[i+1], b_l[i])$, where $b_l[i]$ denotes the residual bandwidth of link l during the ith time-slot (i.e., the time interval $[t_l[i], t_l[i+1]]$), where $i = 0, 1, 2, \ldots, T_l - 1$, and T_l is the total number of time slots in l. We combine the TB lists of all links to build an Aggregated TB (ATB) list, where we store the residual bandwidths of all links in each intersected time-slot, denoted as $(t[0], t[1], b_0[0], b_1[0], \ldots, b_{|E|-1}[0]), \ldots, (t[T-1], t[T], b_0[T-1], b_1[T-1], \ldots, b_{|E|-1}[T-1])$, where T is the total number of new time-slots after the aggregation of TB lists of all $|E|$ links. Without loss of generality, we set the smallest time-slot to be 1 time unit. Figure 3 shows the ATB table created by combining the TB tables of three links in the example network shown in Fig. 2.

The network path is an ordered set of nodes that consist of one or more links from the source node to the destination node. The bandwidth of a path is the bottleneck bandwidth of all links on it. For example, in Fig. 2, the bandwidth of path $v_s - v_1 - v_d$ in time slot 0 (i.e., time interval $(t[0], t[1])$) is the minimum bandwidth of link $v_s - v_1$ and $v_1 - v_d$ in this time slot, i.e., $B_{v_s - v_1 - v_d}[0] = \min(b_{v_s - v_1}[0], b_{v_1 - v_d}[0]) = \min(4, 5) = 4$.

3.2 BS-ARIR problem formulation

3.2.1 Model of AR (Advance Reservation) request

The bandwidth scheduler receives a batch of bandwidth reservation requests in advance and places them in a set $\mathbb{AR}$ over a period of time $[T^S, T^E]$. Each AR request is denoted as a 5-tuple $ar(v_r^s, v_r^d, D_r, [t_r^S, t_r^E], p_1)$, requesting the transfer of D_r amount of data from v_r^s to v_r^d, during a time window from the earliest transfer start time t_r^S to the latest transfer end time (deadline) t_r^E, with a specified

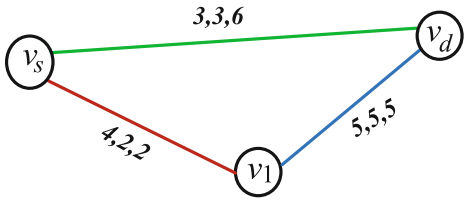


Fig. 2 A simple topology of HPN

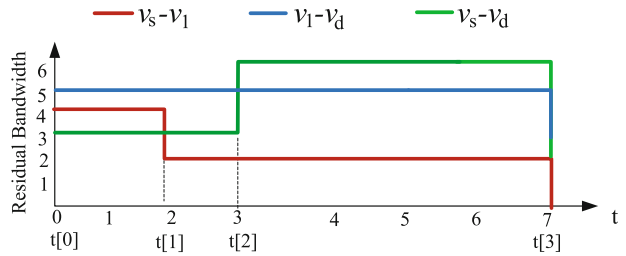


Fig. 3 An ATB with three links

priority p_1. The total transfer duration of all AR requests is $[T^S, T^E] = [\min(t_0^S, t_1^S, \ldots, t_{n-1}^S), \max(t_0^E, t_1^E, \ldots, t_{n-1}^E)]$.

Let $\mathbb{AAR}$ be a set of successfully scheduled AR requests. Each element in $\mathbb{AAR}$ is denoted as $aar(p_r, b_r, [t_r^s, t_r^e], p_1)$, meaning that aar has reserved b_r amount of bandwidth on path p_r during time interval $[t_r^s, t_r^e]$. Each aar has the same satisfaction coefficient, which is equal to the priority p_1 of the request, and satisfies the following conditions:

$$\begin{cases} b_r \cdot (t_r^e - t_r^s) = D_r, \\ t_r^S \le t_r^s < t_r^e \le t_r^E. \end{cases}$$

We define the satisfaction of each aar as

$$p_1 \cdot \frac{t_r^E - t_r^S}{[t_r^e - t_r^S] + [t_r^E - t_r^S]},$$

where t_r^E and t_r^e represent the data transfer deadline and the actual data transfer end time, respectively, t_r^S is the required earliest transfer start time, and p_1 is the satisfaction coefficient. For example, considering $p_1 = 1$, when t_r^e is equal to t_r^E, the satisfaction of the AR request is 0.5; when t_r^e is close to t_r^S, its satisfaction approaches 1.

If the reserved bandwidth of aar is preempted by an IR request with a higher priority, it is denoted as par, and the set of all preempted AR requests is denoted as $\mathbb{PAR}$. We set the satisfaction coefficient of each element in $\mathbb{PAR}$ to be a negative value of $-p_3$ (generally, $p_3 \ge p_1$). A negative satisfaction of any preempted par reflects a certain degree of punishment for preemption. The satisfaction of each preempted request par is then defined as $-p_3 \cdot \frac{t_r^E - t_r^S}{[t_r^e - t_r^S] + [t_r^E - t_r^S]}$.

For example, if an ar has reserved bandwidth successfully, then it is denoted as aar; when the actual transfer end time t_r^e is equal to t_r^E, its satisfaction is 0.5; if its reserved bandwidth is preempted by an IR request, it is then denoted as par. Considering $p_3 = 2$ $(p_3 \ge p_1)$, the satisfaction of this par then becomes -1. Therefore, we should design a collaborative scheduling strategy by considering not only the number of successful scheduled requests, but also the number of preempted AR requests.

3.2.2 Model of IR (Immediate Reservation) request

During time interval $[T^S, T^E]$, the number of randomly arriving IR requests follows Poisson distribution [9]. Each IR request $ir(v_r^s, v_r^d, D_r, t_r^a, d_r, p_2)$ desires to transfer D_r amount of data from v_r^s to v_r^d with the priority value of p_2 $(p_2 \ge p_1)$. The arrival time t_r^a of IR is a random number between $[T^S, T^E]$. We set the default transfer start time of IR to be the begin of the next time slot, and consider a maximum duration of d_r. The bandwidth scheduler attempts to compute an appropriate path from v_r^s to v_r^d for the IR to meet its requirement and complete the transfer as soon as possible. If the bandwidth of the computed path is insufficient to complete the desired transfer in time, then for the remaining data, the scheduler has to preempt the bandwidth of AAR with a lower priority p_1 located within its required transfer duration. The set of all successfully scheduled IR is denoted as $\mathbb{AIR}$, where each element is denoted as $air(d_r, t_r^a, t_r^e, p_2)$. We can also take the priority value p_2 as the satisfaction coefficient of each air, whose satisfaction is defined as $p_2 \cdot \frac{d_r}{[t_r^e - t_r^a] + d_r}$. Considering $p_2 = 2$, if the transfer end time t_r^e is close to its arrival time t_r^a, then its satisfaction is close to 2; if the transfer end time t_r^e is equal to its deadline $t_r^a + d_r$, then its satisfaction is 1.

The normalized overall user satisfaction for collaborative bandwidth scheduling of AR–IR is defined as follows:

$$\begin{aligned} SAT = \frac{1}{(|\mathbb{AR}| + |\mathbb{IR}|)} \cdot \Big[& \sum_{r \in \mathbb{AAR}} p_1 \cdot \frac{t_r^E - t_r^S}{[t_r^e - t_r^S] + [t_r^E - t_r^S]} \\ & + \sum_{r \in \mathbb{AIR}} p_2 \cdot \frac{d_r}{[t_r^e - t_r^a] + d_r} \\ & - \sum_{r \in \mathbb{PAR}} p_3 \cdot \frac{t_r^E - t_r^S}{[t_r^e - t_r^S] + [t_r^E - t_r^S]} \Big]. \end{aligned} \tag{1}$$

The collaborative bandwidth scheduling problem of IR and AR considers the following factors: (i) maximize the number of satisfied requests, (ii) finish the transfer of each request as early as possible, and (iii) reduce the negative effect of preempted requests on the overall satisfaction. The problem of collaborative bandwidth scheduling of AR–IR is formally defined as follows:

Definition 1 BS-ARIR (Bandwidth Scheduling for Advance Reservation and Immediate Reservation): Given a topology of HPN $G(V, E)$ with the ATB list of all links and the total transfer time interval $[T^S, T^E]$ for a batch of AR and IR requests with different priorities, our objective is to co-schedule the AR requests for bulk data transfer and time-critical IR requests to maximize the normalized overall user satisfaction as defined in Eq. 1.

3.3 BS-ARIR complexity analysis

BS-ARIR aims to maximize the overall user satisfaction *SAT* in HPNs. We consider the following objectives: (i) maximize the number of successfully scheduled AR and IR requests while minimize the transfer end time of each request; (ii) minimize the number of preempted AAR and the sum of preempted bandwidths by IR requests.

Theorem 1 *BS-ARIR is NP-complete.*

Proof The decision version of BS-ARIR is as follows: given an HPN and a set of AR and IR requests, is there a preemption strategy that returns the overall user satisfaction no less than *sat*? Given the sets of $\mathbb{AR}$, $\mathbb{IR}$, $\mathbb{AAR}$, $\mathbb{AIR}$, and $\mathbb{PAR}$, it is easy to calculate the overall user satisfaction using Eq. 1 and compare the result with a certain *SAT*. We know that BS-ARIR $\in NP$. Since the factor $\frac{1}{(|AR|+|IR|)}$ is constant, we omit this constant factor in Eq. 1 in our proof.

First, we consider a special case of BS-ARIR: suppose that the arrival time of each IR request t_r^a is a given specific value in $[T^S, T^E]$, and $ir(v_r^s, v_r^d, D_r, t_r^a, d_r, p_2)$ can be represented as $ir(v_r^s, v_r^d, D_r, [t_r^a, t_r^a + d_r], p_2)$.

In this case, the IR request actually becomes an AR request $ar(v_r^s, v_r^d, D_r, [t_r^S, t_r^E], p_2)$, and the BS-ARIR problem reduces to the problem of maximizing SAT of scheduling multiple AR $ar(v_r^s, v_r^d, D_r, [t_r^S, t_r^E], p_1/p_2)$ requests with different priority (p_1 or p_2) in HPNs.

We then consider a special case where all AR requests are of the same priority (i.e., $p_2 = p_1$), i.e.,

$ar(v_r^s, v_r^d, D_r, [t_r^S, t_r^E])$.

Obviously, no bandwidth preemption is needed. Therefore, maximizing the overall user satisfaction *SAT* reduces to maximizing SAT':

$$SAT' = \sum_{r \in \mathbb{AAR}} p_1 \cdot \frac{t_r^E - t_r^S}{[t_r^e - t_r^S] + [t_r^E - t_r^S]}. \tag{2}$$

We further suppose that the earliest transfer start time $t_r^S = 0$, and denote the AR request as $ar'(v_r^s, v_r^d, D_r, [0, t_r^E])$. We consider a particular HPN topology in our special case as shown in Fig. 4 [10] with a unique destination node v^d and the bandwidth D_r/t_r^E for each link $v_r^s - v^d$. The transfer end time of each request ar' on the unique path $v_r^s - v^d$ is t_r^E, and BS-ARIR to maximize SAT' is further transformed to maximizing the number of requests scheduled successfully, as shown in Eq. 3:

$$SAT'' = \sum_{r \in \mathbb{AAR}} p_1/2. \tag{3}$$

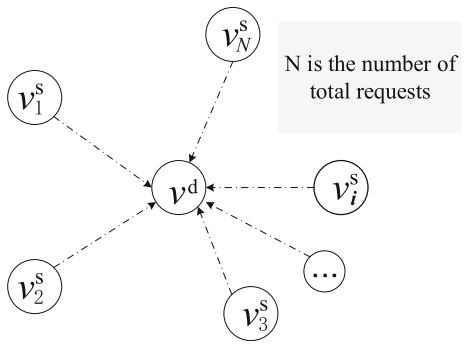


Fig. 4 An instance of a particular network structure

It is essentially equivalent to the maxR problem in [5]. That is to say, the maxR problem is a special case of our BS-ARIR problem. The maxR problem is NP-hard [5], so is our BS-ARIR problem. Along with the fact that BS-ARIR is in the class of NP, we conclude that BS-ARIR is NP-complete. □

4 Algorithm design and analysis for BS-ARIR

The NP-completeness of BS-ARIR indicates that there does not exist any polynomial-time optimal algorithm for BS-ARIR unless $P = NP$. In this section, we focus on the design of a heuristic algorithm and propose a collaborative scheduling algorithm, Max-S-ARIR. A fixed path with a fixed bandwidth is computed for each of the AR requests considering the minimum resource occupancy, and more uninterrupted bandwidth resources can be made available for future IR requests. A flexible scheduling scheme using variable paths with variable bandwidths is adopted for high-priority IR requests for the earliest completion time. If bandwidth resources become insufficient, an IR with a higher priority needs to preempt the path bandwidth of AR requests that have been previously scheduled in the overlapped time period. The overall scheduling solution is divided into the following two phases:

Phase 1: advance reservation of AR requests. Given a batch of AR requests, we design a periodic scheduling algorithm with the objective to maximize the overall satisfaction of AR requests by using a minimum resource first scheduling strategy, referred to as Min-R-AR, for multiple AR requests. We also design a comparison algorithm using the existing MBDPA algorithm in [11], referred to as Min-BHP-AR. The design details are provided in Sect. 4.1.

Phase 2: immediate reservation of IR requests. For an incoming IR request, we first compute a path with the

maximum bandwidth during the requested time periods. If this path is insufficient to complete the entire data transfer in time, we need to preempt the bandwidth of AAR reserved in the corresponding overlapped time window to transfer the remaining data. We propose a minimum preemption algorithm to identify the AARs that can be preempted in the current time window to release the bandwidths for the transfer of the IR request. A greedy algorithm for IR scheduling is also designed for comparison. The design details are provided in Sect. 4.2.

4.1 Min-R-AR algorithm design for scheduling ARs

We consider periodic scheduling of multiple AR requests r in the form of $(v_r^s, v_r^d, D_r, [t_r^S, t_r^E], p_1)$, where t_r^S and t_r^E are the earliest transfer start time and the latest transfer end time, respectively. It requests the transfer of D_r amount of data from source v_r^s to destination v_r^d within a time interval $[t_r^S, t_r^E]$. The transfer time interval, the reserved bandwidth, and the number of links on the computed path are all variables in this AR scheduling problem. A general policy would be to minimize resource occupancy and bandwidth fragmentation to support subsequent bandwidth reservations. Occupied resources include three factors: bandwidth, transmission time, and the number of links (i.e., the number of hops on a path).

Taking these three factors into consideration, we design a minimum resource first scheduling algorithm with the minimum product of data size and number of path hops, referred to as Min-R-AR, whose pseudocode is provided in Algorithm 1. This algorithm differs from Min-BHP-AR (Minimum Bandwidth Hops Product first for AR) proposed in [1], whose optimization strategy follows the Bandwidth and Distance Product Algorithm (MBDPA) proposed in [11], which considers two factors of network resources (i.e., Bandwidth and number of hops). In Min-BHP-AR, each AR request is scheduled by the minimum bandwidth-hop product, while ignoring the impact of data transfer time on the amount of resources occupied. Typically, the data size to be transferred in a given AR request is known, so the minimum bandwidth leads to the maximum transfer time and a low degree of satisfaction.

We provide an explanation of some notations used in Algorithm 1 as follows:

- t^S denotes the earliest possible transfer start time of all requests, and t^E denotes the latest possible transfer start time of all requests;
- p_r denotes the path for an AAR, b_r denotes the bandwidth of path p_r, t_r^s denotes the actual data transfer start time of ar, and t_r^e denotes the actual data transfer end time of ar;
- $\mathbb{AR}$ denotes the set of Advance Reservation requests, $\mathbb{SAR}$ denotes the set of Sorted AR by product of bandwidth and number of hops, $\mathbb{AR}'$ denotes the set of remaining unscheduled AR, $\mathbb{AAR}$ denotes the set of Accommodated AR, and $\mathbb{PAR}$ denotes the set of preempted AAR;
- $|\mathbb{AAR}|$ denotes the total number of AARs, and $|\mathbb{PAR}|$ denotes the total number of PARs;
- σ_r denotes the product of minimum bandwidth and minimum number of hops for a request r in AR;
- $\mathbb{IR}$ denotes the set of Immediate Reservation requests, and $\mathbb{AIR}$ denotes the set of Accommodated IR;
- $|\mathbb{IR}|$ denotes the total number of IRs, and $|\mathbb{AIR}|$ denotes the total number of AIRs;
- $p_r[i]$ denotes a path for an IR in time slot i, and $b_r[i]$ denotes the bandwidth of path $p_r[i]$;
- p_1 denotes the priority of AR, and p_2 denotes the priority of IR;
- $SAT1$ denotes the normalized satisfaction of all ARs, and SAT denotes the normalized satisfaction of all ARs and IRs.

The reserved bandwidth for each request is $b_r \geq B_r^{min}$. Reducing the completion time of every single request would increase the user satisfaction. The main logic flow of the algorithm is described as follows. In lines 4–13, we compute the product σ_r of data size and number of path hops for each request ar in $\mathbb{AR}'$. In lines 14–16, we select the request ar in $\mathbb{AR}'$ with the minimum σ_r, and allocate bandwidth b_r on paths p_r during a virtual transfer time interval $[t_r^s, t_r^s]$, which lies within the required transfer interval $[t_r^S, t_r^E]$. In lines 17–18, we add the successfully scheduled request to the set $\mathbb{AAR}$. In lines 19–21, we update the bandwidth of links on the corresponding paths during time interval $[t_r^s, t_r^s]$, and the remaining unscheduled AR set $\mathbb{AR}'$, respectively. We then continue to the next iteration and recompute the product σ_r of minimum bandwidth and minimum number of hops for a request $ar \in \mathbb{AR}'$, due to the bandwidth change on affected links, until we obtain the set $\mathbb{AAR}$ of successfully scheduled requests, denoted as $(p_r, B_r^{min}, [t_r^s, t_r^e])$. In line 22, we compute the normalized overall user satisfaction $SAT1$ of all ARs.

Algorithm 1 Min-R-AR(G, $\mathbb{AR}$)

Require: an HPN graph $G(V, E)$ with an ATB list of all links, and multiple ARs $(v_r^s, v_r^d, D_r, [t_r^S, t_r^E], p_1)$ in set $\mathbb{AR}$
Ensure: the successfully scheduled AR set $\mathbb{AAR}$, $|AAR|$

Initialize variable $|AAR| = 0, \mathbb{AR}' = \mathbb{AR}$;
Construct the current topology G of the HPN within time interval $[T^S, T^E]$, which contains all time dots of ARs without duplicates in the increasing order;
while $\mathbb{AR}' \neq \emptyset$ **do**
  for $ar \in \mathbb{AR}'$ **do**
    $B_r^{min} = D_r/(t_r^E - t_r^S)$;
    Prune links with available bandwidth less than B_r^{min} from G to obtain graph G';
    Employ the breadth-first search algorithm to compute a path p_r with bandwidth b_r and the minimum number h_r of hops from v_r^s to v_r^d;
    if $h_r == 0$ **then**
      no path for ar;
      $\mathbb{AR}' = \mathbb{AR}' - ar$;
      continue;
    $\sigma_r = D_r \cdot h_r$;
    $ar(D_r, p_r, b_r, \sigma_r)$;
  Select the request ar with the minimum σ_r from $\mathbb{AR}'$;
  $t_r^s = t_r^S$;
  $t_r^e = t_r^s + \frac{D_r}{b_r}$;
  $\mathbb{AAR} \leftarrow ar$;
  $|\mathbb{AAR}| = |\mathbb{AAR}| + 1$;
  Identify the time intervals overlapping with $[t_r^s, t_r^e]$;
  Update the residual bandwidths of links on path p_r within time interval $[t_r^s, t_r^e]$;
  $\mathbb{AR}' = \mathbb{AR}' - ar$;
$SAT1 = \frac{1}{|\mathbb{AR}|} \cdot \sum_{r \in \mathbb{AAR}} p_1 \cdot \frac{t_r^E - t_r^S}{[t_r^e - t_r^S] + [t_r^E - t_r^S]}$.
return $\mathbb{AAR}$.

4.2 Design of co-scheduling algorithm for BS-ARIR

During the transfer of an AAR, an IR request with a higher priority in the form of $(v_r^s, v_r^d, D_r, t_r^a, d_r, p_2)$ may arrive and request immediate bandwidth reservation. If the bandwidth of the computed path for it is insufficient to transfer D_r amount of data within time interval $[t_r^a, t_r^a + d_r]$, we may need to preempt the bandwidth of existing AARs to meet the demand of the IR request. A general guideline is to avoid preempting the bandwidth that has been reserved successfully for an AR request. This is a collaborative scheduling problem between advance reservation and immediate reservation. Our goal is to maximize the overall Quality of Experience for all users.

We first consider the heuristic algorithm, Greedy-ARIR based on a greedy strategy proposed in [1]. The main idea of this greedy algorithm is to preempt the bandwidth of the AAR with the longest overlapping time with the IR, and if there are multiple AARs with the same overlapping time period, we choose the one with the maximum bandwidth for preemption to minimize the number of preempted AARs.

To minimize the number of preempted AARs, the above greedy algorithm preempts the maximum bandwidth of the AAR that meets the criteria. However, this is not beneficial for AR transmission. Hence, we attempt to minimize the number of preempted AARs and meanwhile minimize the sum of preempted bandwidths to improve the overall user satisfaction. Furthermore, we prefer the AAR for preemption, which has transferred a less amount of data than others. We first design an algorithm with minimum preemption within time window [$ts[i]$, $ts[j]$], as shown in Algorithm 2 in Sect. 4.2.1, and then design an algorithm with maximum user satisfaction, as shown in Algorithm 3 in Sect. 4.2.2.

4.2.1 Algorithm design with minimum preemption within time window [*ts*[*i*], *ts*[*j*]]

We first design a minimum preemption algorithm Min-Preemption within time window [$ts[i]$, $ts[j]$]. If there is no sufficient network bandwidth to meet the demand of an IR request, we need to preempt the bandwidths of AARs that have been reserved successfully within the corresponding time window [$tp[i]$, $tp[j]$]. We design a scheduling strategy that minimizes the number of preempted AARs, and if there exists more than one AAR that meets the conditions, we select the one with less bandwidth. Furthermore, if there are multiple existing AARs in $\mathbb{S}_{[i,j]}$ that result in the same amount of bandwidth, we then choose the one with the least transferred data size. The pseudocode of Min-Preemption is provided in Algorithm 2, where some notations are defined as follows:

- $\mathbb{S}[i,j]$ denotes the subset of $\mathbb{AAR}$ within the time window [$ts[i]$, $ts[j]$], namely, $\mathbb{S}[i,j] \subset \mathbb{AAR}$. An IR request with priority value $p2$ needs to preempt the allocated bandwidth of $S[i, j]$ to satisfy its immediate reservation demand;
- $\mathbb{TDS}$ denotes the set of time dots, in which the available bandwidths are not sufficient to satisfy the required bandwidth b within the corresponding time window [$tp[i]$, $tp[j]$];
- b_m denotes the bandwidth of the candidate request aar_m for preemption, and $b(n)$ denotes the maximum available bandwidth from source v_r^s to destination v_r^d in the nth time slot;
- $\mathbb{S}'[i,j]$ denotes the set of preempted AARs within time window [$tp[i]$, $tp[j]$].

Algorithm 2 Min-Preemption: $Min - P(i, j, b)$

INPUT: (i, j, b), where i and j denote the indices of two time points in $\mathbb{TP}$, and b denotes the desired amount of available bandwidths in the HPN within $[tp[i], tp[j]]$ after preemption
OUTPUT: $NULL$ or the set of AARs to be preempted

1: Identify set $\mathbb{S}_{[i,j]}$ containing existing AARs with priority value of 1 that have time interval overlapping with $[tp[i], tp[j]]$. Initialize time dot set $\mathbb{TDS} = \emptyset$, and bandwidth preemption set $\mathbb{S}'_{[i,j]} = NULL$;
2: **for** $i \leq k < j$ **do**
3: **if** $b(k) < b$ **then**
4: Add k to $\mathbb{TDS}$;
5: **while** $|tds| > 0$ **do**
6: Identify an existing AAR aar_m that has the same source v_r^s, same destination v_r^d and the longest time interval overlapping within $[tp[i], tp[j]]$, and maximum $\sum_{n=0}^{|tds|-1} min\,(b_m, b - b(n))$. If there are multiple existing AARs in $\mathbb{S}_{[i,j]}$ that result in the same maximum value, we then choose the one with the least scheduled bandwidth. Furthermore, if there are multiple existing AARs in $\mathbb{S}_{[i,j]}$ that result in the same amount of bandwidth, we then choose the one with the least transferred data size.
7: **if** no AAR is identified **then**
8: Return $NULL$.
9: Release the bandwidths of the links on path for aar_m, and remove aar_m from $\mathbb{S}_{[i,j]}$, add it to $\mathbb{S}'_{[i,j]}$;
10: **for** $i \leq k < j$ **do**
11: **if** $b(k) \geq b$ **then**
12: Remove element k from $\mathbb{TDS}$;
13: Return $\mathbb{S}'_{[i,j]}$.

4.2.2 Algorithm design with maximum satisfaction for BS-ARIR

We design a co-scheduling algorithm in Algorithm 3, which contains two main phases:

Phase 1: we schedule AR requests in advance periodically. we call Algorithm 1 for minimum resource occupancy to reserve an FPFB path for each AR request as early as possible, since an FPFB path can maintain continuous bandwidth for future use.

Phase 2: we schedule IR requests immediately as they arrive. During the time interval $[T^S, T^E]$ that aggregates all time intervals of AR requests, we calculate a VPVB path for an arriving IR request, since an VPVB path can fully utilize the available bandwidth resources, and help maximize the number of successfully scheduled IR requests meanwhile minimizing the completion time of each request. If there is no sufficient bandwidth for the IR request, namely, the bandwidth of the calculated VPVB path for the IR request cannot complete the data transfer before the deadline, the scheduler calls Algorithm 2 to preempt some bandwidths from the AAR that have been reserved within the transfer duration of the IR request for the remaining data transfer. Preempting an AAR means releasing all bandwidths on the reserved FPFB path. We design a scheduling strategy to minimize the impact of preemption on the overall user satisfaction. To minimize the number of preempted AARs, we select the AAR with less bandwidth if there is more than one AAR. In sum, the algorithm uses the VPVB path and multiple FPFB paths preempted for the IR data transfer to maximize the overall user satisfaction of ARs and IRs collectively.

The pseudocode of Max-S-ARIR is provided in Algorithm 3, as briefly described below:

In line 1, identify time points corresponding to the reserved time interval of all AARs within time interval $[T^S, T^E]$, and sort these time points as $tp[0], tp[1], \ldots$ in the ascending order.

In line 2, call Algorithm 1 Min-R-AR to schedule the ARs within time interval $[T^S, T^E]$ in advance, and add the AR requests that are successfully reserved to set $\mathbb{ARR}$.

For an IR request with a higher priority arriving during the time interval $[T^S, T^E]$ of the scheduled ARs, the scheduler calculates a VPVB path for it immediately. If the calculated bandwidth is insufficient to transfer the data of the IR request, then call Algorithm 2 to preempt the bandwidths of the AARs overlapping with the transfer duration of the IR request. The successfully scheduled IRs are placed in set $\mathbb{AIR}$ and the AARs preempted by IRs are placed in set $\mathbb{PAR}$.

In line 4, since the beginning of an IR's transfer start time $tp[i]$ is the next time slot of its arrival time, the required transfer time is d_r, and its transfer end time is $tp[i] + d_r$. We identify the transfer time interval $tp[i]$, $tp[j]$ (i.e., time slot interval $[i, j-1]$) for the IR request.

In line 5, in each time slot k within $[i, j-1]$, calculate a path $p_r[k]$ with the maximum bandwidth $b_r[k]$ for the transfer of the IR request. We then obtain a VPVB path set during time slot $[i, j-1]$ for the IR request.

In lines 6–9, if these VPVB paths can complete the data transfer for the IR request in time, we add the successfully scheduled IR to set $\mathbb{AIR}$; otherwise, we need to identify the AARs that overlap with time slot $[i, j-1]$ as candidates to be preempted, and put them in set $\mathbb{S}_{[i,j]}$.

In line 10, within a temporary time slot window $[i, j']$, initialize the sum of bandwidth variable $B''_{sum} = +\infty$, the set of preempted AARs $\mathbb{S}''_{[i,j']} = NULL$, and the cardinality of $|\mathbb{S}''_{[i,j']}| = +\infty$.

In lines 11–19, within time window $[i, j]$, starting from time slot i, sliding time slot j', call *MinPreemption* $\left(i, j', \frac{D'_r}{tp[j'] - tp[i]}\right)$ in each time window $[i, j']$ to achieve the minimum number of preemptions. If the preempted bandwidth is sufficient to provide the bandwidth needed for the IR transfer, then the preemption process ends.

In line 20, add the successfully scheduled IR to set $\mathbb{AIR}$, and update the bandwidths on the corresponding paths.

In line 21, remove the preempted AARs from set $\mathbb{AAR}$ and add them to set $\mathbb{PAR}$.

In lines 22–23, compute the overall user satisfaction of all AR and IR requests and return the result.

Since the complexity of Dijkstra's algorithm is $O(|E| \cdot lg(|V|))$, the complexity of Algorithm 3 is $O(T \cdot (|E| \cdot lg(|V|) + |E|))$ in the worst cases.

Algorithm 3 Max-S-ARIR(G,AR,IR)

Require: an HPN graph $G(V, E)$ with an ATB list of all links within $[T^S, T^E]$, and time dot priority queue tp, a set of AR$(v_r^s, v_r^d, D_r, [t_r^S, t_r^E], p_1)$

Ensure: the normalized satisfaction degree SAT

1: Identify all time dots of ATB within time interval $[T^S, T^E]$ (including T^S and T^E), and put them in the priority queue tp in the ascending order;
2: $\mathbb{AAR}$ = Algorithm 1 within interval $[T^S, T^E]$;
3: **while** an IR$(v_r^s, v_r^d, D_r, t_r^a, d_r, p_1)$ request has arrived **do**
4: For the IR arriving at $t_r^a \in [T^S, T^E]$, identify the IR time interval $[tp[i], tp[i] + d_r]$ that overlaps with time interval $[tp[i], tp[j]]$, where $tp[i]$ is the time point next to t_r^a;
5: Compute a series of VPVB paths within time slot $[i, j-1]$ such that each path $p_r[k]$ has the maximum bandwidth $b_r[k]$ in different time slot k, for the transfer of the arriving IR within time interval $[tp[i], tp[j]]$;
6: **if** $\sum_{k=i}^{j-1} \big((tp[k+1] - tp[k]) \cdot b_r(k)\big) \geq D_r$ **then**
7: Add IR to $\mathbb{AIR}$, and update the residual bandwidths of links on path p_r within time interval $[i, j-1]$;
8: Continue;
9: Identify set $\mathbb{S}_{[i,j]}$ containing existing AARs with priority value of 1 that have time interval overlapping with $[tp[i], tp[j]]$;
10: Initialize $B''_{sum} = +\infty$, $\mathbb{S}''_{[i,j']} = NULL$ and let $|s''_{[i,j']}| = +\infty$;
11: **for** $i < j' \leq j-1$ **do**
12: $D'_r = D_r - \sum_{k=i}^{j'} \big((tp[k+1] - tp[k]) \cdot b_r(k)\big)$;
13: $\mathbb{S}'_{[i,j']} = MinPreemption(i, j', \frac{D'_r}{tp[j'] - tp[i]})$;
14: **if** $\mathbb{S}'_{[i,j']} == NULL$ **then**
15: Continue;
16: $B'_{sum} = \sum_{k=0}^{|s'_{[i,j']}|-1} b_k$;
17: **if** $\big(|\mathbb{S}'_{[i,j']}| < |\mathbb{S}''_{[i,j']}| \;||\; \big(|\mathbb{S}'_{[i,j']}| == |\mathbb{S}''_{[i,j']}| \;\&\&\; B'_{sum} < B''_{sum}\big)\big)$ **then**
18: $\mathbb{S}''_{[i,j']} = \mathbb{S}'_{[i,j']}$;
19: $B''_{sum} = B'_{sum}$;
20: Add IR to set $\mathbb{AIR}$, and update the residual bandwidths of links on path p_r within time interval $[i, j']$;
21: Add AARs in set $\mathbb{S}''_{[i,j-1]}$ to the preempted set $\mathbb{PAR}$;
22: $SAT = \frac{1}{(|\mathbb{AR}|+|\mathbb{IR}|)} \cdot \Big[\sum_{r \in \mathbb{AAR}} p_1 \cdot \frac{t_r^E - t_r^S}{[t_r^e - t_r^S] + [t_r^E - t_r^S]} + \sum_{r \in \mathbb{AIR}} p_2 \cdot \frac{d_r}{[t_r^e - t_r^a] + d_r} - \sum_{r \in \mathbb{PAR}} p_3 \cdot \frac{t_r^E - t_r^S}{[t_r^e - t_r^S] + [t_r^E - t_r^S]}\Big]$;
23: Return SAT.

5 Performance evaluation

For performance evaluation, we implement the proposed algorithms and conduct (i) simulations in randomly generated networks in Sect. 5.1, and (ii) proof-of-concept experiments on an emulated SDN testbed based on Mininet [12] system in Sect. 5.2.

5.1 Simulation-based performance evaluation

5.1.1 Simulation setup

Within a given time interval $[T^S, T^E]$, we generated multiple random AR data transfer requests of different sizes in advance, and followed Poisson distribution to generate a series of IR requests within this period. In each network instance, we generated the ATB of all links randomly.

Our simulation experiments used random networks of different sizes, as shown in Table 1. The scheduling workloads in terms of the number of ARs/IRs are provided in Table 2.

We set the total time slot $[T^S, T^E]$ to be 20 time units, and the start time $T^S = 0$. The bandwidths of links follow the normal distribution: $b = b^{max} \cdot e^{-\frac{1}{2}(x)^2}$, where b^{max} is set to 100 Gb/s, and x is a random variable within (0, 1]. The number k of IR requests within a time unit follows Poisson distribution: $P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}$, indicating that the probability of the actual number k of request arrivals is $P(k)$ under the condition of the average arrival number λ per unit time.

Each parameter of AR $(v_r^s, v_r^d, D_r, [t_r^S, t_r^E], p_1)$ is randomly generated, where v_r^s and v_r^d are two random nodes in the network, data size D_r is a random integer, $[t_r^S, t_r^E]$ is a random interval among time interval [0, 20 s], and all ARs have the same priority $p_1 = 1$. During the time interval [0, 20 s], the ratio between the numbers of IRs and ARs is set to be 10% in our simulations, and the parameters of IR $(v_r^s, v_r^d, D_r, t_r^a, d_r, p_2)$ are set as follows: the arrival time t_r^a is a random integer between [0, 20 s], the longest transfer time d_r is less than $20 - t_r^a$, and the priority of all IRs is set to be the same value $p_2 = 2$. The way to generate all other parameters including v_r^s, v_r^d and D_r is the same as in AR. We considered $p_3 = 2p_1$ in our simulations.

5.1.2 Performance evaluation of Max-S-ARIR for co-scheduling ARs and IRs

Similarly, in each of the random networks of different sizes as shown in Table 1, using different numbers of random AR and IR requests as shown in Table 2, we run our proposed Min-R-AR algorithm to schedule a batch of ARs in advance during the time interval [0, 20 s], and then

Table 1 Network scales for testing

Index of network size	1	2	3	4	5	6
Number of nodes	10	30	60	100	150	200
Number of links	20	60	120	200	300	400

Table 2 Scheduling workloads

Index of workload	1	2	3	4	5	6	7	8
Number of ARs	100	200	400	600	800	1000	1200	1500
Number of IRs	10	20	40	60	80	100	120	150

follow Poisson distribution to generate IR requests within the time interval [0, 20 s].

According to [13], bulk transfers of big data (terabytes to petabytes) account for a large proportion of traffic, e.g., 85–95% in some inter-data center (inter-DC) WANs. However, very few literatures offer us an evidence on the ratio between the numbers of different types of transfer requests, and their actual numbers would vary widely due to the dynamics in the number of users in public clouds. Since inter-DC WANs are under increasing pressure to provide service-level agreements (SLAs), considering the requirements of SLAs and fairness, many shared production networks generally do not admit too many high-priority requests. For simplicity, we set the ratio between the numbers of IRs and ARs to be 10% in our simulations.

We run Max-S-ARIR and Greedy algorithms for 10 times, respectively. The overall user satisfaction normalized *SAT* measurements are provided in Figs. 5, 6, 7, 8, 9 and 10 for different network sizes. These simulation results show that:

(i) The overall user satisfaction normalized *SAT* of our proposed algorithm is consistently better than the greedy algorithm.
(ii) Similarly, the network size has little effect on the performance of the algorithm because the transfer performance is primarily determined by the link bandwidth.
(iii) We also observe that Min-R-AR achieves a marginal performance improvement over the comparison algorithm with the increase of data loads in smaller networks as shown in Figs. 5 and 6. This is because a small network does not have sufficient resources to support large data transfer, and neither algorithm is able to perform well in this case.

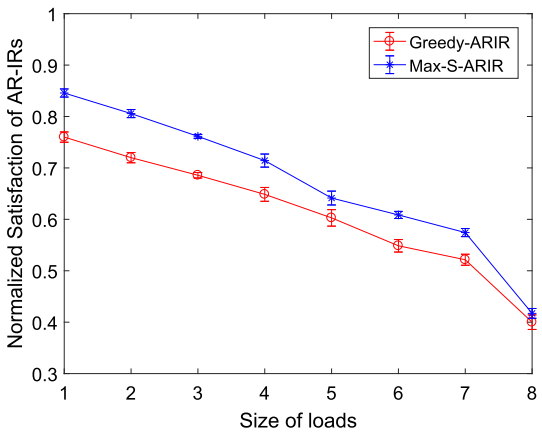


Fig. 5 In Network 1: normalized SAT evaluation with different workloads of ARs and IRs

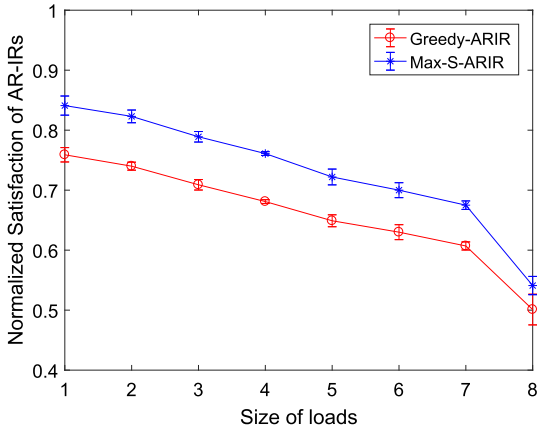


Fig. 6 In Network 2: normalized SAT evaluation with different workloads of ARs and IRs

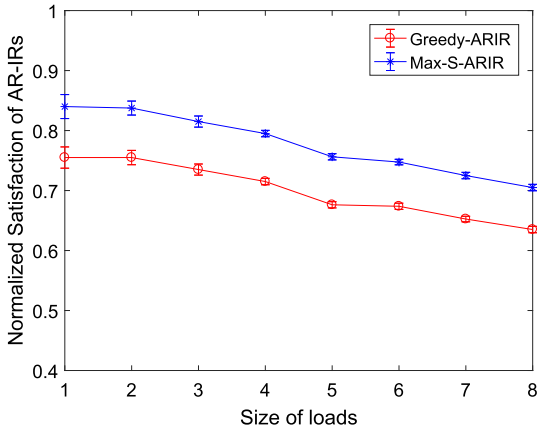


Fig. 7 In Network 3: normalized SAT evaluation with different workloads of ARs and IRs

It is worth pointing out that, although the network sizes in the simulations are limited, our approaches are scalable to larger network sizes because of their low time complexity and potential use in centralized controllers with a Global Network View (GNV) of SDN for data transfer between data centers.

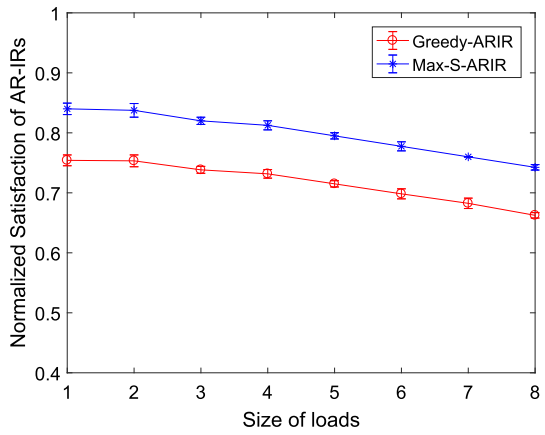


Fig. 8 In Network 4: normalized SAT evaluation with different workloads of ARs and IRs

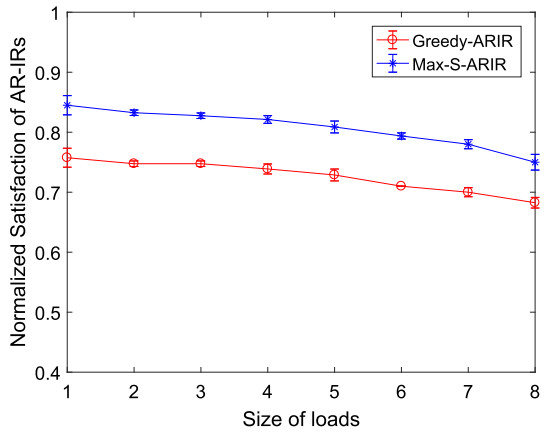


Fig. 9 In Network 5: normalized SAT evaluation with different workloads of ARs and IRs

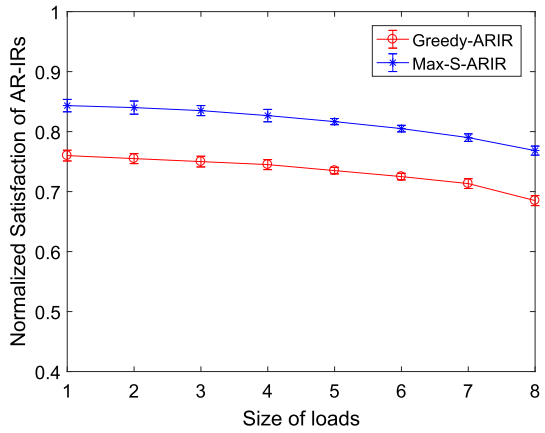


Fig. 10 In Network 6: normalized SAT evaluation with different workloads of ARs and IRs

5.2 Experiment-based performance evaluation

5.2.1 Mininet testbed setup

We construct a virtual network testbed using the Mininet emulation tool to evaluate the efficacy of the proposed bandwidth scheduling scheme in an SDN environment. Mininet is an emulation tool that has been widely used for building virtual networks [14]. The overhead introduced by the scheduling process in Mininet emulation is marginal compared to the scheduling algorithm running time and is almost negligible in large cases [15]; therefore, it offers a suitable tool for evaluating the performance of bandwidth scheduling algorithms.

In our Mininet-based testbed, we emulate an SDN environment for data transfer between three data centers. As shown in Fig. 11, the network comprises three OpenFlow switches under the control of an SDN controller and each switch is connected to a data center. Each data center is represented by a gateway server shown as a host (denoted as $h_i, i = 1, 2, 3$) in the figure. we emulate each switch as an instance of Open vSwitch [16] and choose OpenDaylight [17] as the SDN controller. We set the bandwidth capacity of the link between each pair of OpenFlow switches to be 10 Gbps. The bandwidth capacity of each link between an OpenFlow switch and a data center is limited by the gateway server capacity and is set to be 1 Gbps.

5.2.2 Performance comparison of Max-S-ARIR on the SDN testbed

We conduct a bandwidth scheduling experiment on this emulated testbed over a period of total 10 time slots denoted as $T_i\,(i = 0, \ldots, 9)$. The available amounts of bandwidth of the network links across time slots $[T_0, T_9]$ are provided in Table 3. The controller receives various data

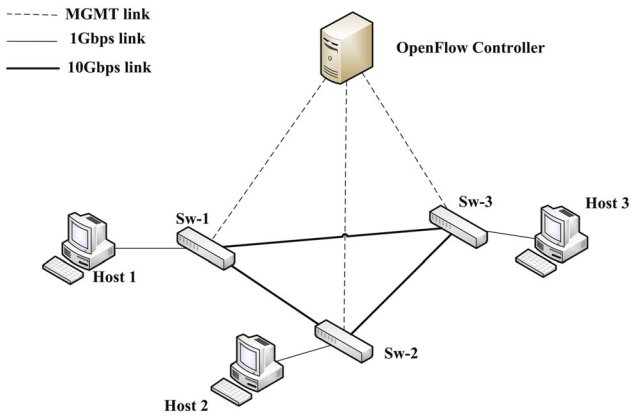


Fig. 11 A Mininet emulated testbed

Table 3 Available link bandwidths in Gb/s in Fig. 11 across [0, 9] time slots

Links	Time slots									
	0	1	2	3	4	5	6	7	8	9
$S_1 - S_2$	5.53	4.93	6.12	3.46	7.21	4.11	5.09	7.32	4.64	2.92
$S_1 - S_3$	6.34	4.47	4.19	6.28	6.72	4.53	8.05	9.22	8.61	4.19
$S_2 - S_3$	6.89	3.18	4.52	7.83	3.18	7.17	7.12	8.32	7.13	5.28

transfer requests from users running on different data centers and controls all OpenFlow switches for bandwidth allocation. In this experiment, we consider five transfer requests AR $(v_r^s, v_r^d, D_r, [t_r^S, t_r^E], p_1)$ for advance bandwidth reservation during the period $[T_0, T_9]$ and one immediate request IR $(v_r^s, v_r^d, D_r, t_r^a, d_r, p_2)$ as follows:

- $ar_1 : (h_1, h_2, 13\,\text{Gb}, [6, 10], 1)$
- $ar_2 : (h_1, h_2, 11\,\text{Gb}, [3, 8], 1)$
- $ar_3 : (h_2, h_3, 15\,\text{Gb}, [3, 9], 1)$
- $ar_4 : (h_1, h_3, 16\,\text{Gb}, [5, 9], 1)$
- $ar_5 : (h_1, h_3, 10\,\text{Gb}, [2, 7], 1)$
- $ir : (h_1, h_3, 7\,\text{Gb}, 3, 4, 2)$

In the emulated virtual SDN environment shown in Fig. 11, the controller receives AR requests from applications running on three data centers and reserves bandwidth for them using Min-R-AR according to the bandwidth availability in Table 3. During the AR transfer periods $[T_0, T_9]$, we randomly generate an immediate request IR with higher priority. We schedule them using Max-S-ARIR and the greedy heuristic algorithms for performance comparison. The scheduling outcomes and corresponding performance measurements of Max-S-ARIR and the greedy heuristic are presented in Tables 4 and 5, respectively. In both of these tables, we use $**$ to denote the AR requests that have reserved bandwidth but are preempted by the immediate request. The actual bandwidth that these AR requests obtain is less than the amount listed in the tables, and therefore their data transfer might fail due to insufficient bandwidth allocation.

From the results of our testbed emulation in Tables 4 and 5, we observe that the proposed Max-S-ARIR algorithm significantly outperforms the greedy heuristic algorithm in terms of normalized user's satisfaction (SAT). Specifically, the former achieves a normalized SAT of 0.52, while the latter achieves a normalized SAT of 0.31. This is because when an IR request arrives during the AR transfer period, these two algorithms apply different strategies in the following two aspects:

(i) The proposed Max-S-ARIR algorithm computes a series of variable paths with maximum variable bandwidth (i.e., VPVB) in different time slots, while the Greedy algorithm computes a fixed path with the maximum fixed bandwidth (i.e, FPFB) across transfer periods. Adopting a VPVB path with flexible transfer bandwidth results in less transfer time than a FPFB path.

(ii) If there is no sufficient bandwidth for the transfer of an arriving IR, Max-S-ARIR identifies those AARs that have the same pair of source and destination and overlapped transfer period with the arriving IR, and then preempts the one with the least scheduled bandwidths for the transfer of the remaining data of the IR. The greedy algorithm identifies those AARs that have the same path and overlapped transfer period with the arriving IR and then preempts the one with the largest bandwidth. Hence, these two preemption strategies result in different levels of negative satisfaction. Obviously, Max-S-ARIR has less negative satisfaction than Greedy as it preempts less bandwidth.

Table 4 Performance measurements of Max-S-ARIR on the emulated testbed

Requests	Time slots	Paths	Bandwidths (Gb/s)	ECT (s)	Normalized SAT
ar_1	6–9	$S_1 - S_3 - S_2$	3.52	9.69	0.52
ar_2	3–6	$S_1 - S_2$	3.46	6.18	
ar_3	3–7	$S_2 - S_3$	3.18	7.72	
ar_4	5–8	$S_1 - S_3$	4.53	8.53	
$ar_5(**)$	2–4	$S_1 - S_3$	4.19	4.39	
ir	3–4	$S_1 - S_3$	3.50	5.00	

Table 5 Performance measurements of Greedy-ARIR on the emulated testbed

Requests	Time slots	Paths	Bandwidths (Gb/s)	ECT (s)	Normalized SAT
ar_1	6–9	$S_1 - S_3 - S_2$	3.52	9.69	0.31
ar_2	3–6	$S_1 - S_2$	3.46	6.18	
ar_3	3–7	$S_2 - S_3$	3.18	7.72	
$ar_4(**)$	5–8	$S_1 - S_3$	4.53	8.53	
ar_5	2–4	$S_1 - S_3$	4.19	4.39	
ir	3–6	$S_1 - S_3$	1.75	7.00	

6 Conclusion

In this paper, we investigated a collaborative scheduling problem for a batch of AR and IR requests, referred to as BS-IRAR. This problem includes two subproblems: periodic scheduling of AR requests and immediate scheduling of IR requests arriving on the fly. If the bandwidth is insufficient to accommodate an IR request, the IR request with a higher priority may preempt the bandwidths of lower-priority AR requests that have been previously reserved. We proved both of these problems to be NP-complete. We proposed a heuristic algorithm Min-R-AR for periodic scheduling of AR requests, which outperforms Min-BHP-AR based on an existing algorithm for periodic scheduling of AR requests. We further proposed Max-S-ARIR, a collaborative scheduling algorithm for BS-ARIR. For comparison, we also designed a heuristic algorithm Greedy-ARIR for collaborative scheduling of BS-ARIR. The performance superiority of Max-S-ARIR over Greedy-ARIR was verified by extensive simulations in randomly generated networks in terms of overall user satisfaction

We plan to implement and test the proposed bandwidth scheduling solution in real-life high-performance networks such as OSCARS of ESnet [18]. It is also of our future interest to combine the proposed scheduling solution with existing high-performance transport methods to support big data transfer between data centers over wide-area networks.

Acknowledgements This research is sponsored by the Key Research and Development Plan of Shaanxi Province, China under Grant No. 2018GY-011, and Xi'an Science and Technology Plan Project under Grant No. 2019218214GXRC018CG019-GXYD18.2 with Northwest University, China.

Author contributions The authors of this paper made the following contributions to the work in our submission. AH: Conceptualization, Methodology, Writing-Original Draft, Formal analysis, Investigation, Funding acquisition. CW: Conceptualization, Formal analysis, Methodology, Supervision, Investigation, Writing-Review and Editing. QD: Resources, Investigation, Writing-Review and Editing. DQ and YL: Visualization, Validation, Investigation, Data curation. LZ and MZ: Methodology, Data curation, Writing-Review and Editing. DF: Data curation, Writing-Review and Editing

Data availability The simulation and experimental data will be made available upon request.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Ethical approval This research does not involve any human participants and/or animals.

References

1. Hou, A., Wu, C., Zuo, L., Quan, D., Li, Y., Zhu, M., Duan, Q., Fang, D.: Co-scheduling of advance and immediate bandwidth reservations for inter-data center transfer. In: 2019 IEEE/ACM Workshop on Innovating the Network for Data-Intensive Science (INDIS), in Conjunction with the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'19), Denver, Co, USA, 2019
2. Kathiravelu, P., Veiga, L.: SDN middlebox architecture for resilient transfers. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 560–563. https://doi.org/10.23919/INM.2017.7987329
3. Grosvenor, M.P., Schwarzkopf, M., Gog, I., Watson, R.N.M., Moore, A.W., Hand, S., Crowcroft, J.: Queues don't matter when you can jump them! In: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI '15), Oakland, CA, USA, 4–6, 2015
4. Zuo, L.: Intelligent and flexible bandwidth scheduling for data transfers in dedicated high-performance networks. IEEE Trans. Netw. Serv. Manag. **17**(4), 2364–2379 (2020). https://doi.org/10.1109/TNSM.2020.3012888
5. Zuo, L., Zhu, M.M., Wu, C.Q.: Bandwidth reservation strategies for scheduling maximization in dedicated networks. IEEE Trans. Netw. Serv. Manag. (2018). https://doi.org/10.1109/TNSM.2018.2794300
6. Lin, Y., Wu, Q.: Complexity analysis and algorithm design for advance bandwidth scheduling in dedicated networks. IEEE/ACM Trans. Netw. **21**(1), 14–27 (2013)
7. Zuo, L., Wu, C., Rao, N., Hou, A., Chang, C.: Bandwidth preemption for high-priority data transfer on dedicated channels. In: IEEE International Conference ICCCN, Hangzhou, China, 2018
8. Xie, C., Ghani, N.: Admission control in networks with prioritized advance reservation. In: Proceedings of International Conference on Computer Communications and Networks, 2009. ICCCN 2009, pp. 1–6 (2009)

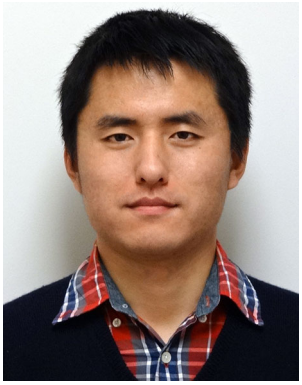
9. Dharam, P.: QoS routing for big data transfer in software-defined networks. PhD Thesis, The University of Memphis, Memphis (2014)
10. Hou, A., Wu, C., Fang, D., Zuo, L., Zhu, M., Zhang, X., Qiao, R., Yin, X.: Bandwidth scheduling for big data transfer with deadline constraint between data centers. In: Workshop on Innovating the Network for Data-Intensive Science (INDIS), in Conjunction with the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'18), Dallas, Texas, USA, 2018
11. Lin, Y., Wu, Q.: On design of bandwidth scheduling algorithms for multiple data transfers in dedicated networks. In: ACM/IEEE Symposium on Architecture for NETWORKING and Communications Systems, ANCS 2008, San Jose, California, USA, 2008, pp. 151–160
12. Mininet: an instant virtual network on your laptop (or other PC). http://mininet.org/. Accessed 1 Jan 2019
13. Jin, X., Li, Y., Wei, D., Li, S., Gao, J., Xu, L., Li, W.X. G., Rexford, J.: Optimizing bulk transfers with software-defined optical WAN. In: Proceedings of SIGCOMM, 2016, pp. 87–100. https://doi.org/10.1145/2934872.2934904
14. Paasch, C., Ferlin, S., Bonaventure, O.: Experimental evaluation of multipath TCP schedulers. In: ACM SIGCOMM Workshop on Capacity Sharing Workshop, 2014. https://doi.org/10.1145/2630088.2631977
15. Aktas, M.F., Haldeman, G., Parashar, M.: Scheduling and flexible control of bandwidth and in-transit services for end-to-end application workflows. Future Gener. Comput. Syst. **56**, 284–294 (2015). https://doi.org/10.1016/j.future.2015.09.011
16. Open vSwitch: an open virtual switch. http://openvswitch.org/. Accessed 1 Jan 2019
17. SDN controller. http://www.opendaylight.org/. Accessed 1 Jan 2019
18. OSCARS. http://www.es.net/oscars. Accessed 1 May 2018

Aiqin Hou is an Associate Professor with the School of Information Science and Technology, Northwest University of China. Her research interests include big data, high performance network, and bandwidth scheduling. She received the Ph.D. Degree in School of Information Science and Technology from Northwest University in 2018. She received the B.S. Degree in Information Theory from Xidian University of China in 1989.

Chase Q. Wu completed his Ph.D. Dissertation at Oak Ridge National Laboratory and received his Ph.D. Degree in Computer Science from Louisiana State University in 2003. He was a Research Fellow at Oak Ridge National Laboratory during 2003–2006 and an Associate Professor at University of Memphis during 2006–2015. He is currently a Professor at New Jersey Institute of Technology. His research interests include big data, parallel and distributed computing, machine learning, high-performance networking, sensor networks, and cyber security.

Qiang Duan is a Professor of Information Sciences and Technology at the Pennsylvania State University Abington College. His general research interest is about data communications and computer networking. Currently, his active research areas include the next generation Internet architecture, network virtualization, software-defined networking, cloud-native network design, and convergence of networking and cloud/edge computing. He has published 100+ journal articles and conference papers and (co)authored 4 books and 6 book chapters. He is serving on the editorial boards of multiple international research journals and has served on the technical program committees for numerous international research conference. He is a Senior Member of IEEE.

Dawei Quan is currently a Master Student in Software Engineering at Northwest University, China. He received the B.S. Degree in Software Engineering from Northwest University, China in 2018. His research interests include bandwidth scheduling and high-performance networks.



Liudong Zuo is currently an Assistant Professor in Computer Science Department at California State University, Dominguez Hills. He received the B.E. Degree in Computer Science from University of Electronic Science and Technology of China in 2009, and the Ph.D. Degree in Computer Science from Southern Illinois University Carbondale in 2015. His research interests include computer networks, high-performance networks, algorithm design, big data, and machine learning.

Yangyang Li is currently a Master Student in Electronic and Communication Engineering at Northwest University, China. He received the B.S. Degree in Automation from Hunan University in 2016. His research interests include bandwidth scheduling and high-performance networks.

Michelle M. Zhu is a Professor and Associate Chair of the Computer Science Department at Montclair State University. Her research areas focus on parallel and distributed computing and big data system. She has published about 120 peer-reviewed articles in various journals and conferences. She received a total of over two million research and education grants from various agencies such as NSF, DOE, ORNL, Illinois State Board of Education and NVidia.

Dingyi Fang is a Professor with the School of Information Science and Technology, Northwest University of China. His current research interests include mobile computing and distributed computing systems, network and information security, localization, social networks, and wireless sensor networks.