Coflow-Like Online Data Acquisition from Low-Earth-Orbit Datacenters

Huawei Huang, *Member, IEEE*, Song Guo, *Senior Member, IEEE*, Weifa Liang, *Senior Member, IEEE*, Kun Wang, *Senior Member, IEEE*, and Yasuo Okabe, *Member, IEEE*

Abstract—Satellite-based communication technology has gained much attention in the past few years, where satellites play mainly the supplementary roles as relay devices to terrestrial communication networks. Unlike previous work, we treat the low-earth-orbit (LEO) satellites as secure data storage mediums. We focus on data acquisition from a LEO satellite based data storage system (also referred to as the LEO based datacenters), which has been considered as a promising and secure paradigm on data storage. Under the LEO based datacenter architecture, one fundamental challenge is to deal with energy-efficient downloading *from space to ground* while maintaining the system stability. In this paper, we aim to maximize the amount of data admitted while minimizing the energy consumption, when downloading files from LEO based datacenters to meet user demands. To this end, we first formulate a novel optimization problem and develop an online scheduling framework. We then devise a novel coflow-like "Join the first K-shortest Queues (JKQ)" based job-dispatch strategy, which can significantly lower backlogs of queues residing in LEO satellites, thereby improving the system stability. We also analyze the optimality of the proposed approach and system stability. We finally evaluate the performance of the proposed algorithm through conducting emulator based simulations, based on real-world LEO constellation and user demand traces. The simulation results show that the proposed algorithm can dramatically lower the queue backlogs and achieve high energy efficiency.

Index Terms—LEO-based datacenter, online job-scheduling, coflow, drift-plus-penalty, energy efficiency, queue stability

1 Introduction

Although existing state-of-the-art storage systems such as NoSQL, NewSQL Databases and Big Data Querying Platforms [2], can meet the stringent requirements of users on data storage and management, it is widely admitted that those storage systems handling cloud operations and data storage are prone to cyber attacks. To mitigate the widespread global crisis of data insecurity, several well known IT organizations and companies are seeking new data storage paradigms to provide secure data storage and management. For example, a startup company named Cloud Constellation [3] intends to establish a space-based cloud storage network SpaceBelt [1], which aims to offer secure data storage for Internet service providers, large enterprises, and government organizations [1], [3].

In this paper, we concentrate on space based datacenter platforms like SpaceBelt that combine LEO satellites and well-connected secure ground networks. This system allows users to store their mission-critical data securely in space instead of ground. The advantages of such a system can be easily recognized since it can isolate data completely from the terrestrial Internet and address some jurisdictional issues [1]. Inspired by the SpaceBelt project, we believe that in the upcoming *space based cloud* era, LEO satellites will undertake more critical roles than just performing as relay devices for ground core communication networks.

In the LEO based datacenter infrastructure, the data-storage system is built upon multiple LEO satellites with each equipped with at least one data-storage server. A dataset is distributedly stored across the data-storage system with multiple duplications, according to a certain redundancy policy. Under such an infrastructure, since the contact window between a ground station and an LEO satellite is intermitted and the power budget in LEO satellites is limited, energy-efficient downloading original files from LEO based datacenters to meet dynamic demands of users poses a great challenge under time-varying channel conditions. However, existing online algorithms for job scheduling in terrestrial cloud datacenters [4], [5], [6] are not applicable to LEO based infrastructures, as they did not consider time-varying downloading opportunities in their system models. Motivated by these concerns, we here study an online data-acquisition problem in LEO based datacenters, while taking

- *H. Huang is with the School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510006, China. E-mail: huanghw28@mail.sysu.edu.cn.*
- *S. Guo is with the Department of Computing and Research Institute for Sustainable Urban Development (RISUD), The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong SAR. E-mail: song.guo@polyu.edu.hk.*
- *W. Liang is with the Research School of Computer Science, Australian National University, Canberra, ACT 0200, Australia. E-mail: wliang@cs.anu.edu.au.*
- *K. Wang is with the Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095. E-mail: wangk@ucla.edu.*
- *Y. Okabe is with the Academic Center for Computing and Media Studies, Kyoto University, Kyoto 606-8501, Japan. E-mail: okabe@media.kyoto-u.ac.jp.*

Manuscript received 22 July 2018; revised 19 May 2019; accepted 6 Aug. 2019. Date of publication 20 Aug. 2019; date of current version 3 Nov. 2020. (Corresponding author: Huawei Huang.)
Digital Object Identifier no. 10.1109/TMC.2019.2936202

both mobility of LEO satellites and time-varying downlinks into account.

As energy supplies in LEO satellites are constrained by limited energy budgets that keep satellites alive and ensure communications [7], [8], we aim to (1) maximize the amount of data admitted; and (2) reduce the energy consumption of data transfer while serving admissions of user requests. To this end, we first formulate a novel optimization problem for file downloading from a system of LEO based datacenters. We then propose an online scheduling algorithm by exploiting the *drift-plus-penalty* optimization technique [9]. Particularly, we design a coflow-like parallel file-download strategy "Join the first K-shortest Queues (JKQ)" for dispatching jobs to satellite queues. To accelerate data transfer in datacenter networks, the technique of *coflow* [10], [11] is invented recently, where a coflow is a collection of parallel flows serving the same request, and the flow transfer does not stop until the completion of all its constituent flows. One advantage of utilizing the coflow-like strategy is shortening the backlogs of queues residing in LEO satellites, where the queue backlog is measured by the size (bit) of unfinished jobs in a queue. Small backlogs in queues are indicators of system stability. Inefficient control strategies will incur large queue backlogs [9]. When the system stability is not well ensured, the queues with large backlogs may lead to packet overflowing and thus packet losses.

The main contributions of this paper are summarized as follows.

- We study the data-acquisition problem under the infrastructure of mobile LEO based datacenters, to meet user requests with security concerns.
- To maximize the*amount of data admitted* while minimizing the energy consumption, we propose an online scheduling algorithm, in which a novel coflow-like job dispatch scheme is devised to lower queue backlogs. The optimality of the proposed algorithm, and its achieved queue stability are analyzed rigorously, by adopting the*drift-plus-penalty* theory framework.
- We also conduct performance evaluation through an emulator − the Satellite Tool Kit. The real-world trace driven simulations show that the proposed algorithm achieves much lower queue backlogs and higher energy efficiency than other the benchmark algorithms for traditional datacenters.

The remainder of the paper is organized as follows. Section 2 reviews the related works. Section 3 presents the system model and problem statement. The online scheduling algorithm is given in Section 4.2. Section 5 shows the performance evaluation, and Section 6 concludes the paper.

2 Related Works

2.1 Satellite based Communication Networks

Several studies on satellite based communication networks have been conducted recently. For example, Wu et al. [12] proposed a two-layer caching model for content delivery services in satellite-terrestrial networks. Jia et al. [13] studied data transmission and downloading by exploiting inter-satellite links in LEO satellite based communication networks. Cello et al. [14] proposed a selection algorithm to mitigate network congestion, using nano-satellites in LEO based networks.

In industry, several representative LEO satellite based projects have been announced recently. For example, *OneWeb satellite constellation* aims to provide global Internet broadband services to consumers as early as 2019 [15]. *SpaceX* has detailed its ambitious plan [16] to bring fast Internet access to the world by deploying a new LEO satellite system that offers greater speeds and lower latency than existing satellite networks. *Boeing* plans to launch a project GiSAT with 702 satellites [17], which can offer twice the capacity of previous digital payload designs for Cayman-Islands.

From these existing exploration efforts, we find that the LEO satellites have been only considered as supplementary extensive devices to terrestrial communication networks. The data storage mechanism under the LEO based datacenters has not been paid much attention yet. To the best of our knowledge, together with our previous study [18], this article is one of the first studies focusing on job scheduling for such LEO based datacenter infrastructures.

2.2 Green Job Scheduling on Datacenters

Several existing works in literature are leveraging stochastic optimization framework to study the energy-efficiency issues on terrestrial datacenters [19], [20], [21], [22]. For example, Deng et. al. [19], [20] studied how to minimize the operational cost of datacenters by utilizing multiple renewable energy resources. A MultiGreen online scheduling algorithm [19] was proposed by applying a two-stage Lyapunov optimization techinque. Zhou et. al. [21] proposed a fuel-cell powered cloud utility indexed maximization problem, which jointly takes the energy cost, carbon emission and geographical request routing into account. In their subsequent work [22], they designed a carbon-aware online control framework by exploiting Lyapunov optimization to find a balance on the three-way tradeoff between energy consumption, service level agreement requirement and emission reduction budget. The common feature of these mentioned stochastic optimization frameworks and together with this paper is that the job scheduling is performed under an assumption that a priori knowledge of system statistics is not given, this ensures the practicality when deploying the proposed online algorithms in the real-world environment.

In the perspective of job-dispatch, although there are a number of online control solutions in literature for job scheduling and resource allocation in cloud and datacenter networks [4], [5], [6], [23], the work in this paper is essentially different from them, due to the following reasons. These existing approaches exploit the "Join in the Shortest Queue" scheme when allocating jobs to queues under their online control frameworks. However, if the system controller sends all arrived jobs at the current time slot to a chosen queue, such a scheme will increase the queue backlog sharply, thus degrading the system stability. In contrast, we here develop a new coflow-like *JKQ* based scheme that dispatches jobs to different queues, aiming to jointly improve energy efficiency and maintain lower queue backlogs throughout the distributed satellites.

Although the proposed coflow-like JKQ based job-dispatch scheme shares similarities with existing works such as the "batch-sampling" [24] and "batch-filling" [25] load

balancing algorithms, it should be noticed that the proposed coflow-like JKQ based scheme is unique under the LEO based datacenter infrastructure. The essential differences between them [24], [25] are summarized as follows. First, the "batch-based" algorithms [24], [25] dispatch multiple redundant copies of each job to a subset of least loaded queues among the randomly *sampled* servers. However, due to the randomness of sampling over all candidate servers, the globally least loaded server may not be included in the sampling of each round. In contrast, in the proposed JKQ based scheme, the target K least loaded servers, i.e., the first K shortest queues, for each file-downloading request are selected only from those containing the associated file chunks, rather than from a randomly sampled subset. Second, under the LEO based datacenter infrastructure, the arrival file-downloading jobs at each time slot are dispatched to the first K-shortest queues associated with the currently available downlink channels. Thus, the set of K-shortest queues are varying over time due to the *mobility* of LEO satellites. This is significantly different from that of the batch-based algorithms [24], [25].

3 System Model & Problem Statement

3.1 System Model

Inspired by recent modeling studies on clouds [6], [26], we consider a discrete time-slotted system, where time slots are normalized to equal integral units $t \in \{1, 2, \dots T\}$ and the length of each time slot (denoted by δ) may range from hundreds of milliseconds to minutes [27] in reality.

We study an LEO-satellite based datacenter network $\mathcal{G} = (S \cup G, E(t))$, where S and G are a set of LEO satellites orbiting in specific planes and a set of terrestrial ground stations, respectively. In particular, $E(t)$ is a set of available time-varying downlink channels at time slot $t \in \{1, 2, \dots T\}$ between the satellites and the ground stations. Each LEO satellite is equipped with at least one data storage server, which is called a *LEO server* hereafter for brevity.

Referring to [13], [28], we elaborate the following preliminaries of the LEO system considered in our system model. Because the orbit of each LEO satellite is determined and known in priori, ground stations periodically contact with a mobile LEO satellite at predictable time slots. Multiple satellites can be in view of a ground station, and multiple ground stations could be in the footprint of a mobile satellite. Thus, a satellite can transmit its data to multiple ground stations at a time if there are sufficient downlink channels. The ground station network consists of multiple well-connected ground stations, which are located at different geographical locations, and cooperatively download different chunks of a file from different mobile satellites to meet its user demand.

Let $(i, j) \in E(t)$ denote a downlink channel between a LEO satellite $i \in S$ and a ground station $j \in G$, and let C_{ij}^t represent the channel state of (i, j) at time slot t. Note that, the state of each time-varying channel can be directly measured or predicted [23]. For example, channel states can be accurately predicted up to one second in future [29]. Thus, we consider that the channel state is known by the system controller at the beginning of each time slot and remains the same without changes at that time slot.

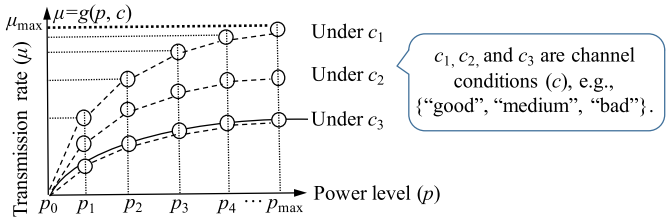


Fig. 1. The classical piecewise rate-power curve [7], [23] with parameters: power-supply level and channel condition.

The total frequency bandwidth of each satellite is divided into multiple beams, and each beam then is broken into narrow channels via the Frequency Division Multiple Access (FDMA) technology. The overall bandwidth within a frequency band can increase, by using the frequency reuse technique such as the orthogonal polarization. The total number of downlink channels in LEO satellite $i \in S$ is represented by σ_i. Furthermore, the Time Division Multiple Access (TDMA) [30] has become a mature satellite communication technology, where a transmission channel is reused by different transmitters at different time slots.

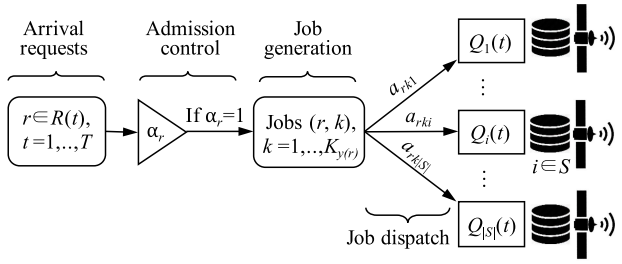
By defining $\overrightarrow{C}$ as a vector of observed channel conditions of downlinks, the relationship between the power allocation and the transmission rate can be described by referring a classical concave rate-power curve $g(p, c)$ [7], [23] as shown in Fig. 1, where p is the allocated power, and $c \in \overrightarrow{C}$ denotes the current channel condition. In reality, linear piece-wise power function with a finite set $\overrightarrow{P} = [p_1, p_2, \dots, p_{\max}]$ of discrete operating power-allocation levels rather than a continuous concave function, is adopted [7], [23]. Thus, the transmission rate of an LEO-satellite downlink is determined by the power level allocated and the current channel condition. Furthermore, as shown in Fig. 1, the maximum transmission rate of each downlink is $\mu_{\max}$, i.e.,

$$g(p, c) \leq \mu_{\max}, \forall p \in \overrightarrow{P}, c \in \overrightarrow{C}.$$

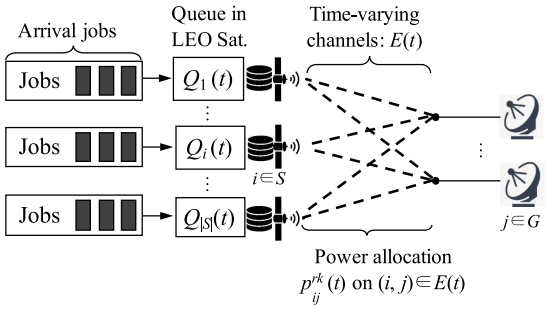
Many types of files $\mathbf{Y} = \{1, 2, \dots, Y\}$ are assumed to have been stored at LEO based datacenters in advance. Each file is with a unique size and stored in multiple copies throughout all LEO servers, following a certain redundancy policy, such as repetition based or network coding mechanisms. If the former is adopted, the duplications need to be randomly distributed over LEO servers for parallel downloading; otherwise, the maximum-distance separable (MDS) code [31] is an option. Under either of the mechanisms, we define a pair of general storage parameters (N_y, K_y) for any file $y \in \mathbf{Y}$: each file is duplicated to N_y chunks, any K_y ($\leq N_y$) chunks among the N_y chunks can reconstruct the original file. Each of the K_y chunks of the file is of size Z_y.

At the beginning of each time slot t, all arrival data-acquisition requests (denoted by $R(t)$) are sent to the system controller. For each request $r \in R(t)$, we define a mapping function $y(r)$ that returns the file type indicated by r. The corresponding original file $y(r)$ is with storage parameters $(N_{y(r)}, K_{y(r)})$. Then, all the file chunks with size $Z_{y(r)}$ are randomly distributed to the $|S|$ LEO servers. Let $f(r)$ represent the original size of a file $y(r)$, we have $f(r) = K_{y(r)} \cdot Z_{y(r)}$.

System controller knows of the storage parameters of each original file and the locations of its chunks. Specifically,



(a) Admission control and job dispatch



(b) Power allocation on time-varying downlinks

Fig. 2. Decisions for arrived file-acquisition requests: admission control, job dispatch, and power allocation on time-varying downlink channels.

we use $I_{y(r)}(\subset S)$ to denote the set of LEO servers that contain the chunks of file $y(r)$. Thus, we have $|I_{y(r)}| = N_{y(r)}$, and any $K_{y(r)}$ out of the $|I_{y(r)}|$ chunks can be used to reconstruct the original file $y(r)$. In addition, to enforce the Service Level Agreement (SLA) [32] between LEO based datacenter providers and users, each data-acquisition request has a maximum tolerable downloading delay, which is measured by time slots and denoted by $\overline{T}_r$, $\forall r \in R(t)$. Even if a chunk cannot be completely downloaded within a contact window between an LEO satellite and a ground station, the downloading job will continue in the next contact opportunity through the seamless service handover technology [33].

During each time slot, if a chunk is under downloading, we consider the tight Service Level Agreement [32] between the LEO based datacenter provider and its users. Here SLA refers to as a contract between the service provider and its users, serving as the foundation for expected QoS attributes such as the data amount downloaded and traffic rate. In our system model, we consider that the SLA as the desired downloading rate for each chunk should not be violated while a downlink channel is being exploited for a chunk-downloading. The major notations and variables are summarized in Table 1.

3.2 Problem Statement

3.2.1 Three-Step Planning for Each Request

As shown in Fig. 2a, an arrived file-acquisition request $r \in R(t)$ is first filtered by an *admission control* strategy. Only if request r is admitted, the system controller will generate $K_{y(r)}$ equal-sized chunk-downloading jobs (r,k), $\forall k = 1,2,\ldots,K_{y(r)}, \forall r \in R(t)$ for it. For brevity, we denote by $\mathbb{K}_r = \{1,2,\ldots,K_{y(r)}\}$ the job indices for request r. Next, jobs need to be dispatched to queues, each of which resides in an LEO satellite. Particularly, to download file-chunks for each request in parallel, we perform a *coflow-like* policy on dispatching jobs to queues. That is, jobs (r,k), $\forall k \in \mathbb{K}_r$, generated for a same request $r \in R(t)$ must be dispatched to $K_{y(r)}$ different queues for parallel chunk-downloading. Notice that, we here consider the*preemptive* scheduling at each queue, which implies that a chunk-downloading job can be interrupted by another job at two consecutive time slots. The*preemptive* model also implies that every downloading job allocated in each queue will be rescheduled at the beginning of each time slot until its downloading is finished. As shown in Fig. 2b, the third step is to allocate power to time-varying downlink channels to decide their transmission rates.

3.2.2 Problem Formulation

Now under the aforementioned system model, we need to make crucial control decisions: (1) admission control to all file-acquisition requests; (2) workload scheduling for chunk-downloading jobs; and (3) power allocations for downlink channels.

Variables. To determine whether a request $r \in R(t)$ is admitted, we first define a binary variable α_r, which is 1 if the request is admitted; 0 otherwise. Further, let $\widehat{Q}_i(t)$ $(i \in S)$ denote the set of jobs dispatched to the queue that resides in satellite $i \in S$ at time slot t, we define another binary variable a_{rki} $(\forall r \in R(t), \forall k \in \mathbb{K}_r)$ to represent the event that chunk-downloading job (r,k) is allocated to $\widehat{Q}_i(t)$. If $a_{rki} = 1$, we have $\widehat{Q}_i(t) \leftarrow \widehat{Q}_i(t) \cup \{(r,k)\}$. Each assigned job (r,k) will be removed from $\widehat{Q}_i(t)$ once its maximum tolerant delay $\overline{T}_r$ is violated. That is, the file acquisition of request r is unsuccessful. In the control of power allocation, for a job $(r,k) \in \widehat{Q}_i(t)$, we define a real-valued variable $p_{ij}^{rk}(t)$, which is chosen from a vector $\overrightarrow{P}$, to indicate the power allocation level on the downlink channel $(i,j) \in E(t)$ at time slot t.

Performance Metrics. For cloud and datacenter networks, system throughput is an important performance metric [5], [6], [8]. Particularly, under the LEO based datacenter platform, we consider a metric that is equivalent to throughput, i.e., the*amount of data admitted* in each time slot t. We denote this meteric by $\phi(t)$, which is calculated as follows.

$$\phi(t) = \sum_{r \in R(t)} \alpha_r f(r) = \sum_{r \in R(t)} \alpha_r K_{y(r)} Z_{y(r)}. \tag{1}$$

As mentioned, the data-transmission in satellites is constrained by a pre-defined power-budget [7], [8]. If the power allocation on transmission channels cannot be carefully scheduled, e.g., allocating too large power level to a downlink with a bad channel condition, much energy will be wasted, thus degrading the amount of data admitted. Therefore, the *energy consumption* should be minimized when satellites are transmitting their data to ground stations. The total energy consumption $\zeta(t)$ on data transmission of all satellites at time slot t is then defined as follows.

$$\zeta(t) = \sum_{i \in S} \sum_{(r,k) \in \widehat{Q}_i(t)} \sum_{(i,j) \in E(t)} \delta \cdot p_{ij}^{rk}(t). \tag{2}$$

Objectives. In order to maximize overall the amount of data admitted while minimizing the energy consumption simultaneously, we define a penalty function. Because we aim to describe an online system, the objective is to minimize a time-average penalty that is denoted by $\overline{Pen}$.

TABLE 1
Notations and Variables

$S; G$	the set of LEO data storage servers; ground stations
$E(t)$	the set of time-varying downlink channels available in time slot t between LEO satellites and ground stations
$\delta; T$	the length of a time slot; the # of all time slots
$\mathbf{Y}$	= {1,2, . . . ,Y}, all file types in storage system
$R(t)$	the set of all arrival requests at the beginning of time slot t, each wants to access its associated original file
(N_y, K_y)	storage parameters of an original file $y \in \mathbf{Y}$, which is duplicated into N_y chunks, only K_y out of N_y chunks can recover the original file
Z_y	size of each chunk for file $y \in \mathbf{Y}$
$\overline{T}_r$	data acquisition deadline required by $r \in R(t)$, measured by time slots
$y(r)$	function that returns the file type demanded by $r \in R(t)$
$\mathbb{K}_r$	= $\{1, 2, \ldots, K_{y(r)}\}$, the set of job indices for request r
$I_{y(r)}$	$\subset S$, the set of LEO servers containing the file chunks of original file type $y(r)$, $\lvert I_{y(r)}\rvert = N_{y(r)}$
(r, k)	the kth ($k = 1, \ldots, K_{y(r)}$) job generated for request r
(i, j)	$\in E(t)$, downlink channel between $i \in S$ and $j \in G$
σ_i	total number of channels in LEO satellite $i \in S$
$\overrightarrow{P}; \overrightarrow{C}$	vector of power levels; vector of channel conditions
α_r	binary variable denoting admission control of $r \in R(t)$
$f(r)$	function returning the size of the original file $y(r)$, $f(r) = K_{y(r)} Z_{y(r)}$
a_{rki}	binary variable indicating the event that dispatches job (r, k) to the queue Q_i ($i \in S$) at the beginning of t
$f(a_{rki})$	function returning the size of the job (r, k) if $a_{rki} = 1$; 0 otherwise. $f(a_{rki}) = Z_{y(r)} \cdot a_{rki}$.
$p_{ij}^{rk}(t)$	$\in \overrightarrow{P}$, the variable indicating the power allocated on channel $(i, j) \in E(t)$, for job (r, k)
$g(p, c)$	transmission-rate function of satellite channels, with parameters of power level $p \in \overrightarrow{P}$ and channel state $c \in \overrightarrow{C}$
$\mathbf{1}_{\{\Pi\}}$	a binary indicator that returns 1 if condition Π is met; 0 otherwise

To make the formulation concise, let Π represent the condition $g(p_{ij}^{rk}(t), c(t)) > 0$, which indicates that the downloading rate of downlink $(i, j) \in E(t)$ is larger than 0 with the allocated power $p_{ij}^{rk}(t)$ and channel condition $c(t)$ at time slot t. Furthermore, since the concept of *mean-rate stability* of a queue is used in problem formulation, we give its definition as follows.

Definition 1. *A queue $\widehat{Q}(t)$ is mean-rate stable [9] if it satisfies*

$$\limsup_{t\to\infty} \frac{E\{|\widehat{Q}(t)|\}}{t} = 0,$$

where $|\widehat{Q}(t)|$ denotes the backlog of queue $\widehat{Q}(t)$.

We then have the following penalty-minimization formulation.

$$\min \quad \overline{Pen} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \{\beta \cdot \zeta(t) - \phi(t)\} \tag{3}$$

$$\text{s.t.} \quad \sum_{k\in\mathbb{K}_r} a_{rki} \le \alpha_r, \ \forall r \in R(t), i \in I_{y(r)}. \tag{4}$$

$$g(p_{ij}^{rk}(t), c(t)) \ge \frac{f(r)}{\delta \overline{T}_r}, \ _{\forall (r,k)\in\widehat{Q}_i(t),(i,j)\in E(t),i\in S.} \tag{5}$$

$$\sum_{(i,j)\in E(t)} \mathbf{1}_{\{\Pi\}} \le 1, \ \ \forall (r, k) \in \widehat{Q}_i(t), i \in S. \tag{6}$$

$$\sum_{(r,k)\in\widehat{Q}_i(t)} \sum_{(i,j)\in E(t)} \mathbf{1}_{\{\Pi\}} \le \sigma_i, \ \forall i \in S. \tag{7}$$

$$\sum_{(r,k)\in\widehat{Q}_i(t)} \sum_{(i,j)\in E(t)} p_{ij}^{rk}(t) \le p_i^{bgt}, \ \forall i \in S. \tag{8}$$

$$\widehat{Q}_i(t) \text{ is mean-rate stable}, \ \forall i \in S.$$
$$\text{Variable: } \alpha_r \in \{0, 1\}, \ a_{rki} \in \{0, 1\}, \ p_{ij}^{rk}(t) \in \overrightarrow{P}, \tag{9}$$
$$\forall r \in R(t), k \in \mathbb{K}_r, \forall (i, j) \in E(t), \forall t = 1, \ldots, T,$$

where the coefficient β in Eq. (3) represents the weight (or price) of each unit of energy consumption. Tuning β also implies changing the relative weighting of the energy consumption against the amount of data admitted.

Constraint (4) specifies the aforementioned *coflow-like* policy, i.e., the number of chunk-downloading jobs dispatched to each of its associated satellite $i \in I_{y(r)}$ is either 1, if $\alpha_r = 1$; or 0 otherwise.

To enforce the SLA described in system model, Constraint (5) claims that the download rate at each time slot for job $(r, k) \in \widehat{Q}_i(t)$ needs to be guaranteed, by the required average downloading rate represented in its right-hand-side, where a time slot is referred to as a valid one to a job (r, k) only if the downlink $(i, j) \in E(t)$ for job (r, k) is allocated with power, i.e., $p_{ij}^{rk}(t) > 0$ and $g(p_{ij}^{rk}(t), c(t)) > 0$. Further, Constraint (5) also includes the following important implications. Since the number of downlinks is limited in each contact window between the satellites and the ground stations, each downlink channel should be fully exploited by allocating sufficient power. Thus, an extremely slow speed to download a chunk in a time slot is not allowed. Constraint (5) is also to enforce the fact that the system needs to download the desired file size of each admitted request $r \in R(t)$ in a long run. Notice that, the second arguments $c(t)$ of function $g(.)$ represents the current channel condition of downlink $(i, j) \in E(t)$.

Constraint (6) implies that the number of downlink serving the job (r, k) dispatched to $\widehat{Q}_i(t)$ should be no more than 1. Here, $\mathbf{1}_{\{\Pi\}}$ is a binary indicator that returns 1 if condition

Π is met; or 0 otherwise. Constraint (7) specifies that the total number of exploited downlinks at each satellite $i \in S$ should be limited by its total number of channels σ_i. Let p_i^{bgt} denote the total power budget of LEO satellite $i \in S$ at any time slot, constraint (8) indicates that the total power capacity should be conserved at each satellite while transmitting data to ground stations. Finally, Constraint (9) specifies that all queues residing in the LEO satellites need keeping the mean-rate stable.

4 Online Scheduling Framework

In this section, we first transform the original problem by applying the *drift-plus-penalty* optimization technique [9] to construct an online scheduling framework, which will deliver a near-optimal solution to the problem. We then analyze the optimality of the proposed approach and its achieved system stability rigorously.

4.1 Problem Transformation

4.1.1 Dynamics of Actual Queues

Denote by $Q_i(t)$ the *queue backlog* as the total size measured in bits that has not been yet transmitted for jobs in $\widehat{Q}_i(t)$. Initially, $Q_i(1) = 0, \forall i \in S$. The *queue dynamics* over time for each satellite is expressed as follows.

$$Q_i(t+1) = \max[Q_i(t) - b_i(t), 0] + A_i(t), \forall i \in S, \tag{10}$$

where

$$b_i(t) = \sum_{(i,j)\in E(t)} \sum_{(r,k)\in \widehat{Q}_i(t)} \delta g(p_{ij}^{rk}(t), c(t)). \tag{11}$$

Here $b_i(t)$ represents the total diminishing bits of the backlog in $\widehat{Q}_i$, and

$$A_i(t) = \sum_{r\in R(t)} \sum_{k\in \mathbb{K}_r} f(a_{rki}), \tag{12}$$

denotes the total size of all arrived jobs at time slot t when dispatching jobs to queue $i \in S$. Note that, $f(a_{rki})$ is a function returning the size of the job (r, k) if a_{rki}= 1; 0 otherwise. That is, $f(a_{rki}) = Z_{y(r)} \cdot a_{rki}$.

4.1.2 Virtual Queues

Next, we transform the original optimization objective $\min \overline{Pen}$ with its constraints into a queue-stability problem [9].

To enforce all constraints on the optimization objective $\min \overline{Pen}$, we define the following virtual queues: $M_{ri}(t)$, $\forall r \in R(t), i \in I_{y(r)}$; $H_{ij}^{rk}(t)$, $\forall r \in R(t)$; $U_{rki}(t)$, $\forall (r,k) \in \widehat{Q}_i(t)$, $i \in S$; $D_i(t)$ and $X_i(t)$, $\forall i \in S$ corresponding to constraints (4), (5), (6), (7), and (8), respectively. Particularly, the virtual queues need to be updated in the end of each time slot, and the update equations are defined as follows.

$$\Lambda(t+1) = \max[\Lambda(t) + \lambda(t), 0],\ t = 1, \ldots, T, \tag{13}$$

where Λ represents virtual queues M_{ri}, H_r, U_{rki}, D_i and X_i, respectively. And λ denotes m_{ri}, h_r, u_{rki}, d_i and x_i, respectively. For brevity, we still use Π to represent the condition $g(p_{ij}^{rk}(t), c(t)) > 0$ when using function $\mathbf{1}_{\{g(p_{ij}^{rk}(t),c(t))>0\}}$ in the following. In particular, $\forall t = 1, \ldots, T$,

$$m_{ri}(t) = \sum_{k\in\mathbb{K}_r} a_{rki} - \alpha_r, \forall r \in R(t), i \in I_{y(r)}, \tag{14}$$

$$h_{ij}^{rk}(t) = \frac{f(r)}{\delta \overline{T}_r} - g(p_{ij}^{rk}(t), c(t)), \forall (r,k)\in \widehat{Q}_i(t), (i,j)\in E(t), i\in S, \tag{15}$$

$$u_{rki}(t) = \sum_{(i,j)\in E(t)} \mathbf{1}_{\{\Pi\}} - 1, \forall (r,k) \in \widehat{Q}_i(t), i \in S, \tag{16}$$

$$d_i(t) = \sum_{(r,k)\in \widehat{Q}_i(t)} \sum_{(i,j)\in E(t)} \mathbf{1}_{\{\Pi\}} - \sigma_i, \forall i \in S, \tag{17}$$

and

$$x_i(t) = \sum_{(r,k)\in \widehat{Q}_i(t)} \sum_{(i,j)\in E(t)} p_{ij}^{rk}(t) - p_i^{bgt}, \forall i \in S. \tag{18}$$

We consider the initial conditions satisfying $\Lambda(1) = 0$ for all virtual queues.

Insight. The virtual queues $\Lambda(t)$ are used to enforce the constraint $\lambda(t) \le 0$, i.e., inequalities (4), (5), (6), (7), and (8). By summing $\Lambda(t)$ over time slots $t = 1, \ldots, T$, we have $\frac{\Lambda(T)}{T} - \frac{\Lambda(1)}{T} \ge \frac{1}{T}\sum_1^T \lambda(t)$. With $\Lambda(1) = 0$, taking expectations on both sides and letting $T \to \infty$, it yields: $\lim_{T\to\infty} \sup \frac{\mathbb{E}\{\Lambda(T)\}}{T} \ge \lim_{T\to\infty} \sup \overline{\lambda}(t)$, where $\overline{\lambda}(t)$ is the time-average expectation of $\lambda(t)$ over $t = 1, \ldots, T$. If $\Lambda(t)$ is mean-rate stable, according to Definition 1, we have $\lim_{T\to\infty} \sup \frac{\mathbb{E}\{\Lambda(T)\}}{T} = 0$, which implies that $\lim_{T\to\infty} \sup \overline{\lambda}(t) \le 0$. This means that the desired constraints for $\lambda(t)$ are satisfied.

We then define $\mathbf{Q}(t) = \{Q_i(t), \forall i \in S\}$, $\mathbf{M}(t) = \{M_{ri}(t), \forall r \in R(t), i \in I_{y(r)}\}$, $\mathbf{H}(t) = \{H_{ij}^{rk}(t),\ \forall (r,k) \in \widehat{Q}_i(t), (i,j) \in E(t), i \in S\}$, $\mathbf{U}(t) = \{U_{rki}(t),\ \forall (r,k) \in \widehat{Q}_i(t), i \in S\}$, $\mathbf{D}(t) = \{D_i(t), \forall i \in S\}$ and $\mathbf{X}(t) = \{X_i(t), \forall i \in S\}$ as the set of all actual and virtual queues. Let $\mathbf{\Theta}(t) = [\mathbf{Q}(t), \mathbf{M}(t), \mathbf{H}(t), \mathbf{U}(t), \mathbf{D}(t), \mathbf{X}(t)]$ represent a concatenated vector of all actual and virtual queues under update Equations (10) and (13), we define a Lyapunov function of the LEO based datacenter system as follows.

$$L(\mathbf{\Theta}(t)) \triangleq \frac{1}{2}\left[\sum_{i\in S} Q_i(t)^2 + \sum_{r\in R(t), i\in I_{y(r)}} M_{ri}(t)^2 + \sum_{i\in S} \sum_{(r,k)\in \widehat{Q}_i(t),(i,j)\in E(t)} H_{ij}^{rk}(t)^2 + \sum_{(r,k)\in \widehat{Q}_i(t), i\in S} U_{rki}(t)^2 + \sum_{i\in S} D_i(t)^2 + \sum_{i\in S} X_i(t)^2\right]. \tag{19}$$

In fact, $L(\mathbf{\Theta}(t))$ calculates a scalar volume of queue congestion in the LEO based datacenters. Intuitively, a small value of the Lyapunov function implies small backlogs of both actual queues and virtual queues, and the holistic system tends to be stable consequently.

4.1.3 Drift-Plus-Penalty Expression

To make the system stable, the Lyapunov function (19) needs to be controlled at each time step to maintain lower congestion in queues. Denoted by $\Delta(\mathbf{\Theta}(t))$ the *one-slot conditional Lyapunov drift* [9], which is defined as

$$\Delta(\mathbf{\Theta}(t)) = \mathbb{E}\{L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t)) | \mathbf{\Theta}(t)\}. \tag{20}$$

Given the current state $\mathbf{\Theta}(t)$, this drift is the expectation of changes in the Lyapunov function (19) over one time slot.

Under the Lyapunov optimization framework [9], the supremum bound of *drift-plus-penalty* expression is expected to be minimized at each time slot to achieve the optimal solution to the proposed original optimization problem. That is,

$$\min \Delta(\Theta(t)) + V\mathbb{E}\{\beta\zeta(t) - \phi(t)|\Theta(t)\}, \tag{21}$$

where V is a tunable knob denoting the weight of penalty. We can observe that a positive V in the objective function (21) is to minimize the energy consumption and maximize the amount of data admitted, while maintaining the stability of holistic system simultaneously. A large positive V implies that the system operator desires a small penalty, i.e., a small energy consumption and a large amount of data admitted. We then have the following Lemma.

Lemma 1. *Given that the arrival request set $R(t)$, the available downlink channel set $E(t)$, all the backlogs of both actual and virtual queues, as well as the job queue $\widehat{Q}_i(t)$ ($i \in S$) are observable at each slot t, for arbitrary $\Theta(t)$, the Lyapunov drift $\Delta(\Theta(t))$ of a storage system in the LEO based datacenters under arbitrary control policies satisfies the following result:*

$$\begin{aligned}
\Delta(\Theta(t)) \le\ & B(t) + \sum_{i\in S} Q_i(t)\mathbb{E}\{A_i(t) - b_i(t)|\Theta(t)\} \\
& + \sum_{i\in I_{y(r)}} \sum_{r\in R(t)} M_{ri}(t)\mathbb{E}\{m_{ri}(t)|\Theta(t)\} \\
& + \sum_{i\in S} \sum_{(r,k)\in\widehat{Q}_i(t),(i,j)\in E(t)} H_{ij}^{rk}(t)\mathbb{E}\{h_{ij}^{rk}(t)|\Theta(t)\} \\
& + \sum_{(r,k)\in\widehat{Q}_i(t),i\in S} U_{rki}(t)\mathbb{E}\{u_{rki}(t)|\Theta(t)\} \\
& + \sum_{i\in S}(D_i(t)\mathbb{E}\{d_i(t)|\Theta(t)\} + X_i(t)\mathbb{E}\{x_i(t)|\Theta(t)\}),
\end{aligned} \tag{22}$$

where $B(t) \triangleq \frac{1}{2}\sum_{i\in S}\{(\sum_{r\in R(t)} Z_{y(r)})^2 + |\widehat{Q}_i(t)|^2\sigma_i^2(\delta^2\mu_{\max}^2 + p_{\max}^2 + 1) + |\widehat{Q}_i(t)|((\frac{f(r)}{\delta\overline{T}_r})^2 - \sigma_i^2 - 2\sigma_i(1 + p_{\max}\cdot p_i^{bgt}) + 1) + \sigma_i^2 + (p_i^{bgt})^2\} + \frac{1}{2}\sum_{r\in R(t)} N_{y(r)}K_{y(r)}^2$ is a time-varying positive constant in each time slot. Note that, $|.|$ returns the size of a set.

The proof of Lemma 1 can be seen in Appendix of [34].

We then show the upper bound on the *drift-plus-penalty* expression of the storage system, by combining objective function (21) and inequality (22).

$$\begin{aligned}
& \Delta(\Theta(t)) + V\mathbb{E}\{\beta\zeta(t) - \phi(t)|\Theta(t)\} \\
& \le B(t) + \sum_{i\in S}\Bigg[\sum_{(r,k)\in\widehat{Q}_i(t),(i,j)\in E(t)} H_{ij}^{rk}(t)\cdot\frac{f(r)}{\overline{T}_r\delta} - \sum_{(r,k)\in\widehat{Q}_i(t)} U_{rki}(t) \\
& \quad - D_i(t)\sigma_i - X_i(t)p_i^{bgt}\Bigg] && (23) \\
& \quad + \sum_{i\in S} Q_i(t)\mathbb{E}\{A_i(t)|\Theta(t)\} && (24) \\
& \quad + \sum_{r\in R(t)} \alpha_r\mathbb{E}\left\{-Vf(r) - \sum_{i\in I_{y(r)}} M_{ri}(t)|\Theta(t)\right\} && (25) \\
& \quad + \sum_{r\in R(t)}\sum_{k\in\mathbb{K}_r}\sum_{i\in I_{y(r)}} \mathbb{E}\{M_{ri}(t)a_{rki}|\Theta(t)\} && (26) \\
& \quad + \sum_{i\in S}\sum_{(r,k)\in\widehat{Q}_i(t),(i,j)\in E(t)} \mathbb{E}\{(V\beta\delta + X_i(t))p_{ij}^{rk}(t) \\
& \qquad - [H_{ij}^{rk}(t) + \delta Q_i(t)] \\
& \qquad \times g(p_{ij}^{rk}(t), c(t)) + (U_{rki}(t) + D_i(t))\mathbf{1}_{\{\Pi\}}|\Theta(t)\}. && (27)
\end{aligned}$$

4.2 Online Scheduling Algorithm

We notice that, it is impractical to assume the arrival rate of requests is known in a realistic system setting, because it is difficult to predict the precise arriving time of a request. Therefore, existing offline solutions based on known arriving rates of requests are not applicable to the problem here. Using an emerging technology such as Software-defined Satellite Networks [35], [36], [37] based centralized control mechanism, we can design a near-optimal online scheduling algorithm that yields a solution for the system controller without a-priori statistical knowledge of arrival rates, while maintaining low backlogs in all queues, and thus stabilizes the system in a long run.

Through observing the upper bound of *drift-plus-penalty* expression shown from term (23) to term (27), we found that term (23) yields a constant at time slot t. Thus, minimizing the objective function (21) is equivalent to minimizing from terms (24), (25), (26), and (27) meanwhile.

In particular, we first analyze the $A_i(t)$ appeared in the term (24). Let $A_i^r(t)$ denote the job size for request $r \in R(t)$ that is dispatched to queue $i \in S$, we have

$$A_i(t) = \sum_{r\in R(t)} A_i^r(t), \forall i \in S. \tag{28}$$

According to the location of original file indicated by each request,

$$A_i^r(t) = \begin{cases} \sum_{k\in\mathbb{K}_r} f(a_{rki}) = \sum_{k\in\mathbb{K}_r} a_{rki}Z_{y(r)}, \text{if } i \in I_{y(r)}; \\ 0, \qquad \text{otherwise if } i \in S\backslash I_{y(r)}. \end{cases} \tag{29}$$

And taking into account constraint (4), we then have

$$\begin{aligned}
\sum_{i\in S} Q_i(t)A_i(t) &= \sum_{i\in S} Q_i(t)\sum_{r\in R(t)} A_i^r(t) \\
&\le \sum_{r\in R(t)}\sum_{i\in I_{y(r)}} Q_i(t)\sum_{k\in\mathbb{K}_r} a_{rki}Z_{y(r)} && (30) \\
&\le \sum_{r\in R(t)}\sum_{i\in I_{y(r)}} Q_i(t)Z_{y(r)}\cdot\alpha_r. && (31)
\end{aligned}$$

Through terms (30) and (31), it can be seen that term (24) involves both a_{rki} and α_r, respectively. Furthermore, the variables α_r, a_{rki} and $p_{ij}^{rk}(t)$ appear in separate terms (25), (26) and (27) in the right-hand-side of the *drift-plus-penalty* expression, respectively. Therefore, when minimizing terms (25) and (26), the term (24) can be viewed as an auxiliary.

We then decouple the minimization over *drift-plus-penalty* function (21) into a four-phase online scheduling framework:

(1) admission control over arrival requests;
(2) job dispatch control;
(3) power allocation to downlink channels;
(4) and queue update.

Recall that as mentioned in Lemma 1, the online scheduling algorithm needs online observations of all actual and virtual queues. In the following, we start with the first phase of online scheduling framework, i.e., admission control of requests, given an arriving request set $R(t)$ at time slot t.

4.2.1 Admission Control

Notice that, the admission decision variables α_r ($\forall r \in R(t)$) are independent of each other among the arrival requests. Thus, we have the following subproblem (32) based on the joint minimization over the terms (24) and (25)

$$\min \alpha_r \left[\sum_{i \in I_{y(r)}} (Q_i(t) Z_{y(r)} - M_{ri}(t)) - Vf(r) \right] \tag{32}$$

$$\text{s.t.} \qquad \alpha_r \in \{0,1\}, \forall r \in R(t).$$

Differentiating the objective function (32) with respect to α_r yields the following simple *threshold-based* admission control strategy

$$\alpha_r = \begin{cases} 1 : \sum_{i \in I_{y(r)}} (Q_i(t) Z_{y(r)} - M_{ri}(t)) \le Vf(r); \\ 0 : \text{otherwise.} \end{cases} \tag{33}$$

When $V > 0$, it can be observed that a large V will benefit to admission ratio.

4.2.2 Coflow-Like JKQ Based Job Dispatch

If a request is admitted, system generates its associated jobs (r,k), which should be dispatched to the selected queues. We decide the *job-dispatch* decisions by reducing the minimization of term (26) to the following subproblem (34). Since the job-dispatch decisions a_{rki} ($\forall i \in I_{y(r)}, k \in \mathbb{K}_r, r \in R(t)$) are independent among different requests, the job dispatch can be conducted for different requests concurrently in a complete distributed manner.

$$\min \qquad a_{rki}[Q_i(t) Z_{y(r)} + M_{ri}(t)] \tag{34}$$

$$\text{s.t.} \qquad a_{rki} \in \{0,1\}, \forall r \in R(t), k \in \mathbb{K}_{y(r)}, i \in I_{y(r)}.$$

Recall that the *coflow-like* parallel download policy is implied in the definition of virtual queue $M_{ri}(t)$. Thus, subproblem (34) results in a *JKQ* scheme for the *job dispatch* phase, which is described as Algorithm 1.

Algorithm 1. Join in the First K-Shortest Queues (JKQ)

Input : $r \in R(t)$, $\mathbb{K}_r$ and α_r (obtained from (33))
Output : a_{rki} ($\forall k \in \mathbb{K}_r, i \in I_{y(r)}$)

if $\alpha_r = 1$ **then**
  $\Psi_r \leftarrow \emptyset$
  for $i \in I_{y(r)}$ **do**
    $\psi_{ri} \leftarrow Q_i(t) Z_{y(r)} + M_{ri}(t)$
    $\Psi_r \leftarrow \Psi_r \cup \{\psi_{ri}\}$
  initialize $a_{rki} \leftarrow 0, \forall k \in \mathbb{K}_r, i \in I_{y(r)}$
  for $k = 1$ *to* $K_{y(r)}$ **do**
    $\psi_{ri^*} \leftarrow \arg\min(\Psi_r)$
    $\Psi_r \leftarrow \Psi_r - \{\psi_{ri^*}\}$
    $a_{rki^*} \leftarrow 1$, where $i^* \leftarrow \arg \psi_{ri^*}$
    $\widehat{Q}_{i^*}(t) \leftarrow \widehat{Q}_{i^*}(t) \cup \{(r,k)\}$

The basic idea is to find the first $K_{y(r)}$ shortest queues in terms of their backlogs out of the candidate queue set $I_{y(r)}$ for the admitted request r, and dispatch $K_{y(r)}$ jobs to $K_{y(r)}$ different chosen queues eventually.

4.2.3 Power Allocation on Downlinks

In each time slot, the overall energy consumption of channels on all satellites $i \in S$ can be determined by minimizing term (27). Because the decision of power allocation on downlink channels is independent among the satellites, the energy-minimization can be also realized by the system controller for each satellite in a distributed manner: the power allocation at each satellite is implemented individually without knowing anything from other satellites. Let (p,c) represent $(p_{ij}^{rk}(t), c(t))$, we then have the following subproblem (35).

$$\min \quad \Gamma(p,c) \text{ s.t.} \quad p_{ij}^{rk}(t) \in \overrightarrow{P}, (r,k) \in \widehat{Q}_i(t), (i,j) \in E(t), i \in S, \tag{35}$$

where $\Gamma(p,c) = (V\beta\delta + X_i(t))p_{ij}^{rk}(t)$ - [$H_{ij}^{rk}(t) + \delta Q_i(t)$] $g(p,c)$ + [$U_{rki}(t) + D_i(t)$] $\mathbf{1}_{\{g(p,c)>0\}}$.

Problem (35) is a simple linear programming. By partially differentiating $\Gamma(p,c)$ with respect to p and rearranging terms, it yields

$$\frac{\partial \Gamma(p,c)}{\partial p} = V\beta\delta + X_i(t) - (H_{ij}^{rk}(t) + \delta Q_i(t)) \frac{\partial g(p,c)}{\partial p}. \tag{36}$$

Recall that, the rate-power curve $g(p,c)$ is a given function over power supply levels and channel conditions. Thus, the values of the term $\frac{\partial g(p,c)}{\partial p}$ in each piecewise power supply level $p \in \overrightarrow{P}$ can be calculated under the observed channel condition c. Specifically, let p vary within $\overrightarrow{P} = [p_1, p_2, \ldots, p_{\max}]$, and by Eq. (36), we obtain a vector of derivative values

$$\overrightarrow{\mathbb{D}} = \left[\frac{\partial \Gamma(p,c)}{\partial p_1}, \frac{\partial \Gamma(p,c)}{\partial p_2}, \ldots, \frac{\partial \Gamma(p,c)}{\partial p_{\max}} \right].$$

Furthermore, the concave function $g(p,c)$ implies that $\Gamma(p,c)$ is a convex function. Based on Eq. (36), we have the valley point $(p^*, c(t))$ of $\Gamma(p,c)$ such that

$$\frac{\partial g(p^*, c(t))}{\partial p^*} = \frac{V\beta\delta + X_i(t)}{(H_{ij}^{rk}(t) + \delta Q_i(t))}. \tag{37}$$

From Eq. (37), we can see that a large V implies a large slope in the rate-power curve $g(p,c)$, i.e., a large $\frac{\partial g(p^*,c(t))}{\partial p^*}$. On the other hand, through observing from the curve $g(p,c)$ shown in Fig. 1, a large slope indicates a small power level. Therefore, Equation (37) tells that a large V leads to a small optimal power level p^*.

Now, we can make the power-allocation decisions by discussing the condition of elements (ele.) in $\overrightarrow{\mathbb{D}}$

$$p_{ij}^{rk}(t) = \begin{cases} p_{\min}, & \text{if ele. in } \overrightarrow{\mathbb{D}} \text{ are non-negative;} \\ p_{\max}, & \text{if ele. in } \overrightarrow{\mathbb{D}} \text{ are non-positive;} \\ p^- \text{or } p^+: & \arg\min\{\Gamma(p^-, c(t)), \Gamma(p^+, c(t))\}, \text{if ele.} \\ & \text{in } \overrightarrow{\mathbb{D}} \text{ vary from negative to positive,} \end{cases}$$

where p^- and p^+ are two successive discrete power levels such that $p^- \leq p^* \leq p^+$, where $p^-, p^+ \in \overrightarrow{P}$, and p^* is the optimal power level denoted by the valley point $(p^*, c(t))$ mentioned above.

4.2.4 Queue Update

In the end of each time slot, the actual queues in $\mathbf{Q}(t)$ should be updated by Eq. (10) based on the optimal solutions a_{rki} and $p_{ij}^{rk}(t)$. Similarly, the virtual queues $\mathbf{M}(t)$, $\mathbf{H}(t)$, $\mathbf{U}(t)$, $\mathbf{D}(t)$ and $\mathbf{X}(t)$ need to be updated according to Eq. (13) using all the optimal solutions derived.

4.3 Analysis on Optimality and Queue Stability

We now show the optimality and stability of the proposed online scheduling algorithm. Notice that, the*optimality* means the performance gap between the proposed online algorithm and the theoretical optimum, while the system*stability* implies that all the actual and virtual queues residing the system are stable.

Theorem 1. *Given that $V > 0$, for arbitrary arrival requests $r \in R(t)$, $\forall t$, the proposed online scheduling framework yields a solution ensuring that:*

(a) the gap between the achieved time-average penalty and the optimal one Pen^{opt} is within $\frac{\widetilde{B}}{V}$, i.e.,

$$\lim_{T\to\infty} \sup \frac{1}{T}\sum_{t=1}^{T}\{\beta\zeta(t) - \phi(t)\} - Pen^{opt} \leq \frac{\widetilde{B}}{V}, \tag{38}$$

where $Pen^{opt} = \lim_{T\to\infty} \inf \frac{1}{T}\sum_{t=1}^{T}\{\beta\zeta^*(t) - \phi^*(t)\}$, $\zeta^*(t)$ and $\phi^*(t)$ are the resulted energy consumption and mount of data admitted of the admitted requests indicated by the optimal solution to the original optimization (3), and $\widetilde{B} = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} B(t)$;

(b) all actual and virtual queues are mean-rate stable.

Recall that V denotes the weight of penalty in the objective function (21). From inequality (38), we can see that a large positive V can lead to a small gap between the achieved time-average penalty and the optimal one Pen^{opt}.

Before we proceed to prove Theorem 1, we have the following Lemma.

Lemma 2 (Existence of an optimal randomized stationary policy). *For any arrival requests $r \in R(t)$, $\forall t$, and for any $\epsilon > 0$, there is a randomized stationary control policy ω that indicates feasible control decisions α_r^*, a_{rki}^* and $p_{ij}^{rk^*}$ for $\forall i, j, r, k, t$, independent of the current queue backlogs, and gives the following steady state values:*

$$\mathbb{E}\{\widehat{Pen}(\langle\alpha_r^*, a_{rki}^*\rangle, \omega(t))\} \leq Pen^{opt} + \epsilon \tag{39}$$

$$\mathbb{E}\{\widehat{m}_{ri}(\langle\alpha_r^*, a_{rki}^*\rangle, \omega(t))\} \leq \epsilon, _{\forall r\in R(t), i\in I_{y(r)}} \tag{40}$$

$$\mathbb{E}\{\widehat{h}_{ij}^{rk}(p_{ij}^{rk^*}, \omega(t))\} \leq \epsilon, _{\forall (r,k)\in\widehat{Q}_i(t), \forall (i,j)\in E(t)} \tag{41}$$

$$\mathbb{E}\{\widehat{u}_{rki}(p_{ij}^{rk^*}, \omega(t))\} \leq \epsilon, _{\forall (r,k)\in\widehat{Q}_i(t), \forall (i,j)\in E(t)} \tag{42}$$

$$\mathbb{E}\{\widehat{d}_i(p_{ij}^{rk^*}, \omega(t))\} \leq \epsilon, _{\forall i\in S} \tag{43}$$

$$\mathbb{E}\{\widehat{x}_i(p_{ij}^{rk^*}, \omega(t))\} \leq \epsilon, _{\forall i\in S} \tag{44}$$

$$\mathbb{E}\{\widehat{A}_i(a_{rki}^*, \omega(t))\} \leq \mathbb{E}\{\widehat{b}_i(p_{ij}^{rk^*}, \omega(t))\} + \epsilon, _{\forall i\in S}, \tag{45}$$

where $\widehat{Pen}$, $\widehat{A}_i$, $\widehat{b}_i$, $\widehat{m}_{ri}$, $\widehat{h}_{ij}^{rk}$, $\widehat{u}_{rki}$, $\widehat{d}_i$ and $\widehat{x}_i$ are the resulting penalty, arrival rate, service capability, and other attributes under policy ω.

Lemma 2 can be proved by adopting the similar techniques in the proof body of Theorem 4.5 in [9]. We now prove Theorem 1 as the follows.

Proof. The proposed online scheduling algorithm finds a solution that can minimize the right-hand-side of the inequation (22) over all feasible control decisions, including the policy ω, in each time slot, we thus have

$$\begin{aligned}
&\Delta\left(\Theta(t)\right) + V\mathbb{E}\{Pen(t)|\Theta(t)\} \\
&\leq B(t) + V\mathbb{E}\{Pen^*(t)|\Theta(t)\} \\
&\quad + \sum_{r\in R(t)}\sum_{i\in I_{y(r)}} M_{ri}(t)\mathbb{E}\{m_{ri}^*(t)|\Theta(t)\} \\
&\quad + \sum_{i\in S}\mathbb{E}\{[Q_i(t)(A_i^*(t) - b_i^*(t)) + \sum_{(r,k)\in\widehat{Q}_i(t),(i,j)\in E(t)} H_{ij}^{rk}(t)\cdot h_{ij}^{rk*}(t) + \\
&\quad \sum_{(r,k)\in\widehat{Q}_i(t)} U_{rki}(t)u_{rki}^*(t) + D_i(t)d_i^*(t) + X_i(t)x_i^*(t)]|\Theta(t)\},
\end{aligned} \tag{46}$$

where $Pen^*(t) \triangleq \widehat{Pen}\ (\langle\alpha_r^*, a_{rki}^*\rangle, \omega(t))$, $A_i^*(t) \triangleq \widehat{A}_i(a_{rki}^*, \omega(t))$, $b_i^*(t) \triangleq \widehat{b}_i(p_{ij}^{rk^*}, \omega(t))$, $m_{ri}^*(t) \triangleq \widehat{m}_{ri}(\langle\alpha_r^*, a_{rki}^*\rangle, \omega(t))$, $h_{ij}^{rk*}(t) \triangleq \widehat{h}_{ij}^{rk}(p_{ij}^{rk^*}, \omega(t))$, $u_{rki}^*(t) \triangleq \widehat{u}_{rki}(p_{ij}^{rk^*}, \omega(t))$, $d_i^*(t) \triangleq \widehat{d}_i(p_{ij}^{rk^*}, \omega(t))$, $x_i^*(t) \triangleq \widehat{x}_i(p_{ij}^{rk^*}, \omega(t))$.

Letting $\epsilon > 0$, and having Lemma 2, the resulting values of $Pen^*(t)$, $A_i^*(t)$, $b_i^*(t)$, $m_{ri}^*(t)$, $h_{ij}^{rk*}(t)$, $u_{rki}^*(t)$, $d_i^*(t)$ and $x_i^*(t)$ are independent of the current queue backlogs $\Theta(t)$. We then have

$$\mathbb{E}\{Pen(t)|\Theta(t)\} = \mathbb{E}\{Pen(t)\} \leq Pen^{opt} + \epsilon \tag{47}$$

$$\mathbb{E}\{A_i^*(t) - b_i^*(t)|\Theta(t)\} = \mathbb{E}\{A_i^*(t) - b_i^*(t)\} \leq \epsilon, _{\forall i\in S} \tag{48}$$

$$\mathbb{E}\{m_{ri}^*(t)|\Theta(t)\} = \mathbb{E}\{m_{ri}^*(t)\} \leq \epsilon, _{\forall i\in S, r\in R(t)} \tag{49}$$

$$\mathbb{E}\{h_{ij}^{rk*}(t)|\Theta(t)\} = \mathbb{E}\{h_{ij}^{rk*}(t)\} \leq \epsilon, _{(r,k)\in\widehat{Q}_i(t),(i,j)\in E(t)} \tag{50}$$

$$\mathbb{E}\{u_{rki}^*(t)|\Theta(t)\} = \mathbb{E}\{u_{rki}^*(t)\} \leq \epsilon, _{\forall (r,k)\in\widehat{Q}_i(t)} \tag{51}$$

$$\mathbb{E}\{d_i^*(t)|\Theta(t)\} = \mathbb{E}\{d_i^*(t)\} \leq \epsilon, _{\forall i\in S} \tag{52}$$

$$\mathbb{E}\{x_i^*(t)|\Theta(t)\} = \mathbb{E}\{x_i^*(t)\} \leq \epsilon, _{\forall i\in S}. \tag{53}$$

Plugging (47), (48), (49), (50), (51), (52), and (53) into the right-hand-side of (46) and taking $\epsilon \to 0$, we get

$$\Delta(\Theta(t)) + V\mathbb{E}\{Pen(t)|\Theta(t)\} \leq B(t) + V\cdot Pen^{opt}. \tag{54}$$

Taking expectations of both sides of (54) and using the law of iterated expectations,

$$\begin{aligned}
\mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t))\} + V\mathbb{E}\{Pen(t)\} \\
\leq B(t) + V\cdot Pen^{opt}.
\end{aligned} \tag{55}$$

Summing over $t \in \{1, 2, \ldots, T\}$ for $T > 1$ and using the law of telescoping sums, we have

$$\mathbb{E}\{L(\Theta(T)) - L(\Theta(1)\} + V\sum_{t=1}^{T}\mathbb{E}\{Pen(t)\} \leq \sum_{t=1}^{T} B(t) + VT \cdot Pen^{opt}. \tag{56}$$

Rearranging terms and neglecting $-L(\Theta(T))$ in the right-hand-side, we can obtain the inequality for $\forall T > 1$

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\{Pen(t)\} \leq Pen^{opt} + \frac{1}{VT}\sum_{t=1}^{T} B(t) + \frac{1}{VT}\mathbb{E}\{L(\Theta(1))\}. \tag{57}$$

Then, taking limits of (57) by $T \to \infty$ proves the part (a). To prove part (b), we have the following inequation from (56), for all time slots $T > 1$:

$$\mathbb{E}\{L(\Theta(T))\} - \mathbb{E}\{L(\Theta(1)\} \leq \sum_{t=1}^{T} B(t) + VTPen^{opt}. \tag{58}$$

Referring the definition of $L(\Theta(T))$, we know

$$\frac{1}{2}\mathbb{E}\left\{\sum_{i\in S} Q_i(T)^2\right\} \leq \mathbb{E}\{L(\Theta(1)\} + \sum_{t=1}^{T} B(t) + VTPen^{opt}. \tag{59}$$

Then, for $\forall i \in S$, we have

$$\mathbb{E}\{Q_i(T)^2\} \leq 2\mathbb{E}\{L(\Theta(1)\} + 2\sum_{t=1}^{T} B(t) + 2VTPen^{opt}. \tag{60}$$

Because $|Q_i(T)|$ is non-negative, we have $\mathbb{E}\{Q_i(T)^2\} \geq \mathbb{E}\{|Q_i(T)|\}^2$. Thus, for all slots $T > 1$

$$\mathbb{E}\{|Q_i(T)|\} \leq \left[2\mathbb{E}\{L(\Theta(1)\} + 2\sum_{t=1}^{T} B(t) + 2VTPen^{opt}\right]^{\frac{1}{2}}. \tag{61}$$

Dividing by T and taking a limit as $T \to \infty$, we have

$$\begin{aligned}&\lim_{T\to\infty}\frac{\mathbb{E}\{|Q_i(T)|\}}{T}\\ &\leq \lim_{T\to\infty}\left[\frac{2\mathbb{E}\{L(\Theta(1)\}}{T^2} + \frac{2}{T^2}\sum_{t=1}^{T} B(t) + \frac{2}{T}V\ Pen^{opt}\right]^{\frac{1}{2}}\\ &= 0.\end{aligned} \tag{62}$$

Thus, all actual queues Q_i ($\forall i \in S$) are mean-rate stable according to Definition 1. Similarly, the mean-rate stability of all virtual queues can be proved following the same routine. This concludes part (b). □

5 Performance Evaluation

In this section, we first elaborate the methodology of performance evaluation, including the emulator, trace, metrics and benchmark algorithms used in simulations. We then demonstrate the simulation results that can prove the high efficiency of the proposed algorithm.

5.1 Settings

5.1.1 Emulator of LEO Satellites

We conduct the trace-driven simulations based on the commercial emulator Satellite Tool Kit (STK) to evaluate the performance of the proposed online algorithm. The STK [38] designed by Analytical Graphics, Inc. (AGI) is an analytical tool enabling engineers and scientists to analyze the complex land, sea, air and space assets. Our simulations are performed on the Globalstar LEO constellation (a typical system of Walker delta constellation [39]), in which 48 satellites are organized into 8 equal-separated orbital planes with 6 satellites in each. Satellites in the same plane are separated from each other with equal angular distance. The orbital period is 114 minutes.

Through selecting 108 locations globally, we first deploy ground stations under the Globalstar LEO constellation. Then, we generate the synthetic channel-state traces with three states (i.e., good, medium and bad), according to the weather conditions in all locations obtained from the Internet. The LEO constellation mission starts from 12 July 2017 00:00:00 Gregorian Coordinated Universal Time (UTCG) and terminates 24 hours later. The length of each time slot, i.e., δ, is set as 10 seconds. The contact window between any satellite and any ground station is recorded and analyzed in each time slot.

5.1.2 Request Trace

Without loss of generality, we adopt YouTube traffic traces in [40] as user request traces. To integrate the traces with the Globalstar LEO constellation, we extract a subset of traces with 48 data-storage servers, ranging in a 24-hour duration. This set of traces includes 731 file-downloading requests originated from 430 users. The servers are equally attached to the 48 LEO satellites. The features of the YouTube traffic traces can be observed in Fig. 4 that user requests arrive with drastically different volumes at different time in one day. The size of files in small-scale ranges between 11,680 bytes and 26,511,812 bytes. In particular, each traffic has a duration attribute, which is viewed as the download deadline, i.e., $\overline{T_r}$ in our system model. On the other hand, the duration of file-downloading task ranges from 0.16 seconds to 625 seconds, and 90 percent of them are lower than 200 seconds. Notice that, when the number of the user requests is too few, it can be observed from Fig. 4a that the number of requests on average in each time slot is less than 20. Thus, the original traces are not suitable to evaluate the job-allocation performance of algorithms. We then generate a large-scale of synthesized request traces based on the original one. For example, the file size and downloading duration are randomly generated according to the corresponding features of the original traces. We thus obtain a synthesized trace set, which includes 7,310 requests originating from 430 users. In the following, we show the performance evaluations using the synthesized traces.

5.1.3 Channel Attenuation Coefficient

The number of downlink channels in each satellite is fixed at 100. The bandwidth of each downlink channel is set at 1 Megahertz (MHz). To calculate the transmission rate of downlinks, we adopt the rate-power function $g(p, c) = 1\ \text{MHz} \cdot \log(1 + v(c) \cdot p)$ presented in [7], where $v(c)$ represents the channel attenuation coefficients determined by the channel state c. Viewing that the three-state condition model [29] is adopted for a single downlink, we vary $v(c)$

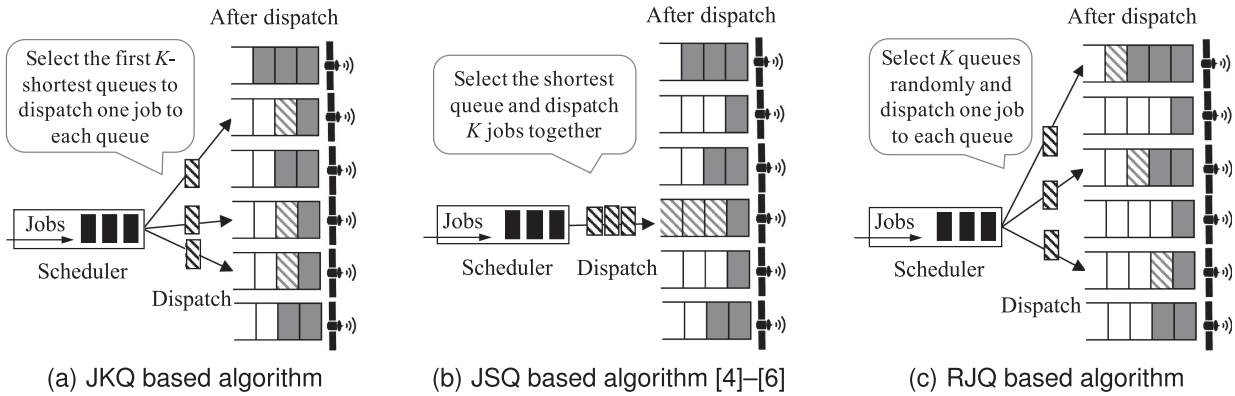


Fig. 3. Comparison of three algorithms while dispatching three chunk-downloading jobs for an admitted request r, with $K_{y(r)} = 3$, $N_{y(r)} = 6$. Notice that, all the six candidate LEO satellites in the right hand side of each illustrative figure hold the required chunks associated with this request.

within three representative combinations, i.e., {5.03, 3.46, 1.0}, {10, 2, 0.1} and {16, 1, 0.01}, to evaluate how the different *varying scales of channel attenuation rate* impact the system performance. The three values in each combination correspond to the channel condition c when it is observed as "good", "medium" and "bad", respectively. Note that, the *varying-scale of channel attenuation rate* describes the changing degree of transmission rate on downlinks due to the "medium" and "bad" weather conditions. We can see that the *varying-scale of channel attenuation rate* become from weak to strong when $v(c)$ is set from {5.03, 3.46, 1.0} to {16, 1, 0.01}.

5.1.4 Other Parameters

Furthermore, referring to the parameter settings shown in [7], the power budget at each satellite is set to 100 Watts. The power values in $\overrightarrow{P}$ are averagely divided into 10 levels ranging from 0.1 Watt to 1 Watt. We then deploy redundant duplications of each file according to the storage parameters (N_y, K_y) randomly over all LEO servers. Before running each round of simulation, we randomly shuffle the file chunks for each file downloading task.

5.2 Metrics

We view the *energy consumption* and *queue backlog* as the major metrics to evaluate the performance of the proposed algorithm. The energy implies the cumulative energy consumption induced by all jobs. It is measured in kiloWatt·Second (KW·s). Notice that, the two metrics included in the objective function of the original minimization problem (3), i.e., amount of data admitted and energy consumption, are conflict with each other. Therefore, it is not fair to compare any single perspective of those two metrics. We then design an integrated performance metric that is associated with both of them. We call it *Watt·Sec per bit*, which denotes the energy consumed by each bit of file chunks downloaded. It is easy to find that the *Watt·Sec per bit* is reversely related to the *energy efficiency*. A low *Watt·Sec per bit* indicates a high energy efficiency of an algorithm. The length of queue backlogs is an indicator of the queue stability in system. Thus, a low backlog performance implies strong stability that an algorithm can bring to system.

In addition, by running at least 20 rounds of simulation under each parameter setting, we also explore the *job completion ratio* (JCR), packet loss, and *job waiting time* (JWT) in queues as important metrics to evaluate the performance of each algorithm.

5.3 Benchmark Algorithms

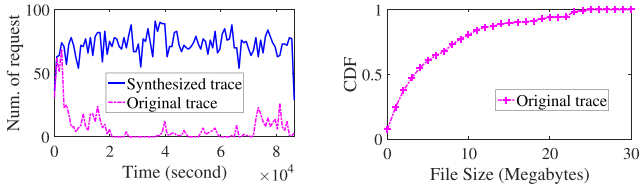
In addition to the aforementioned *batch based* algorithm [24], [25], another benchmark named "*Join in the Shortest Queue* (JSQ)" based algorithm is used to compare the performance with the proposed JKQ based algorithm. The JSQ based online algorithm [4], [5], [6] has been widely adopted on dynamic job scheduling in cloud computing and datacenter networks. Under our LEO data-acquisition framework, we implement a JSQ based algorithm, in which job allocation is conducted by the JSQ mechanism. As illustrated in Fig. 3b, we argue that the JSQ based algorithm potentially brings a sharp increase of backlog in the chosen queue for job dispatching under certain circumstances. In our simulation, we show this through the comparison of the queue backlogs yielded by both algorithms.

The other benchmark algorithm is called *Randomly Join K Queues* (RJQ) based algorithm. Although RJQ is a random selection based algorithm, it does*not arbitrarily* select any random satellite queues as the job-dispatch destinations for chunk-downloading jobs. As shown in Fig. 3c, based on the $(N_{y(r)}, K_{y(r)})$ file-storage strategy, only $K_{y(r)}$ out of the $N_{y(r)}$ satellites containing the encoded file chunks associated with a file-download request r can be selected as the job-dispatch destinations.

5.4 Simulation Results

5.4.1 Job Completion Ratio of Batch Based Algorithm

As mentioned in Section 2, the job-dispatch mechanism of the proposed JKQ based algorithm has significant differences from that of the *batch-based* algorithm [24], [25]. However, we still implement the job-dispatch approach of the original batch-based algorithm under the emphasized LEO based datacenter infrastructure. Fig. 5a illustrates the basic idea of the batch-based algorithm, which includes randomly sampling a specified number (i.e., the *batch size* [24], [25]) of candidate queues for each arrived job, and selecting the shortest queue among the sampled candidates as the job-dispatch destination. Through varying the *batch size* from 1 to 48, and setting the β, V, N_y and K_y to 1, 10, 6 and 3, respectively, we evaluate the *job completion ratio* performance of the batch-based algorithm. In addition, $v(c)$ is set to 10, 2, and 0.1



(a) Number of requests. (b) CDF of file size.

Fig. 4. The extracted original YouTube file-download request trace and its synthesized request trace, which involve 48 data-storage servers and last for 24 hours in (a). The cumulative distribution function (CDF) of the original file size requested by users is shown in (b).

corresponding to "good","medium" and "bad" weather conditions, respectively.

The JCR results of the batch-based algorithm are shown in Fig. 5b. We can see that the original batch-based algorithm exhibits a very low JCR performance while varying the *batch size* from 1 to 48. Notice that, once the batch size reaches to 48, all LEO servers will be sampled as the candidates to choose the least loaded server for job dispatch in the batch based algorithm. It means that even though all the LEO servers are examined for finding the queue-shortest destination server for each job, the JCR has only around 11 percent on average. This is because the queue-shortest server chosen from the sampled set may not contain the required file-chunk of a specified request.

Since the JCR of the original batch based algorithm is too low, we omit its performance comparison with other algorithms in the subsequent results.

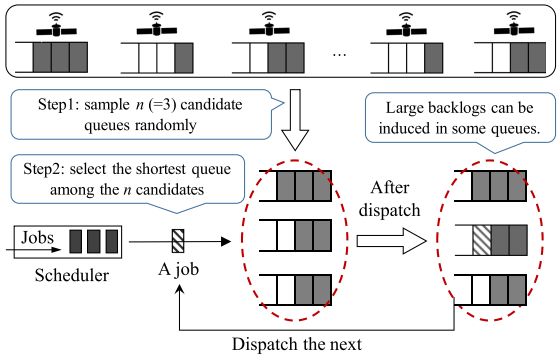
5.4.2 Time Varying Performance

In this group of simulations, under the same parameter settings as we did for Fig. 5, we examine the admission ratio and energy efficiency yielded by the three online algorithms illustrated in Fig. 3.

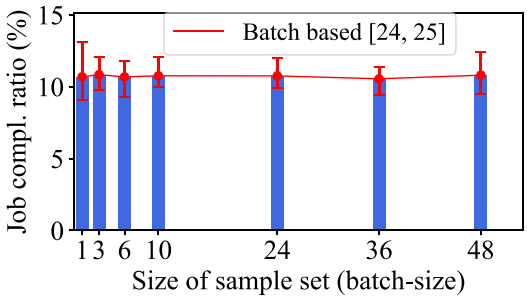
From Fig. 6a, we find that the admission ratios of all three algorithms are 100 percent in the end. This is because the current parameter settings match the *admission control* represented by equation (33). Thus, the amount of data admitted of all algorithms are same.

Fig. 6b demonstrates the cumulative time-varying energy consumption of the three algorithms while downloading data from satellites. It can be seen that the proposed JKQ based algorithm consumes the least energy comparing with the other two algorithms.

Fig. 6c shows the penalty performance, which is the objective of the original minimization problem (3). In the transformed problem (21), not only is the penalty needed to be minimized, but also the *Lyapunov drift* $\Delta(\Theta(t))$ must be minimized as well. Interestingly, on one hand, we observe that the panalties by the JSQ based algorithm grow over time, showing a contrary descending performance under the other two algorithms. The reason behind is as follows. Under the JSQ based algorithm, the chunk downloading jobs are always dispatched to the shortest queues, where the queue backlog increases dramatically. Thus, the JSQ based algorithm has to devote much great efforts to maintain the *Lyapunov drift* $\Delta(\Theta(t))$ in a low level and to sacrifice the penalty performance, even though the current weight of penalty V is set at 10 while the weight of *Lyapunov drift* $\Delta(\Theta(t))$ is only 1. On the other hand, the proposed JKQ based algorithm shows descending and much lower penalties than that of the other two algorithms.

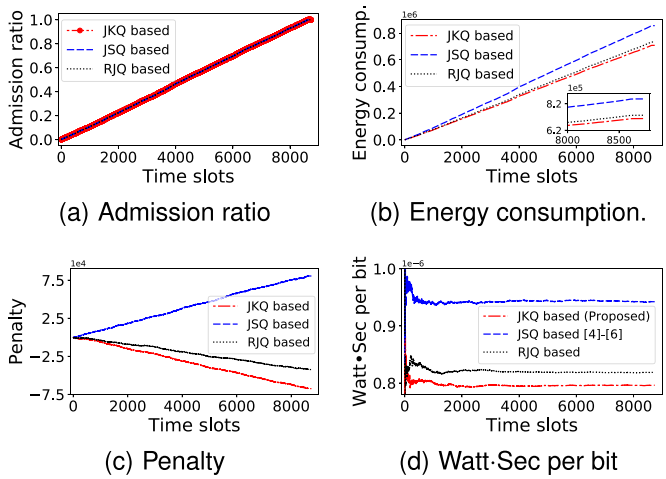


(a) Principle of each job-dispatch in the *batch-based* algorithm [24], [25]



(b) Job Completion Ratio (JCR) v.s. batch-size

Fig. 5. Principle and Job Completion Ratio performance of *batch based* algorithm [24], [25].



(a) Admission ratio (b) Energy consumption.

(c) Penalty (d) Watt·Sec per bit

Fig. 6. Time-varying performance comparison among three algorithms, with 7310 synthesized requests, 48 LEO servers, under $\beta = 1$, $V = 10$, $v(c) = \{10, 2, 0.1\}$ while c is {"Good", "Medium", "Bad"}, respectively.

Fig. 6d illustrates the energy efficiency of algorithms. Recall that energy efficiency is reversely related to *Watt·Sec per bit* . From which, we observe that JKQ based algorithm achieves lower time-varying *Watt·Sec per bit* than that of the JSQ and RJQ based algorithms. This observation implies that the proposed JKQ based algorithm has the highest energy efficiency against the other two benchmarks. The insight behind the high energy-efficiency of our proposed

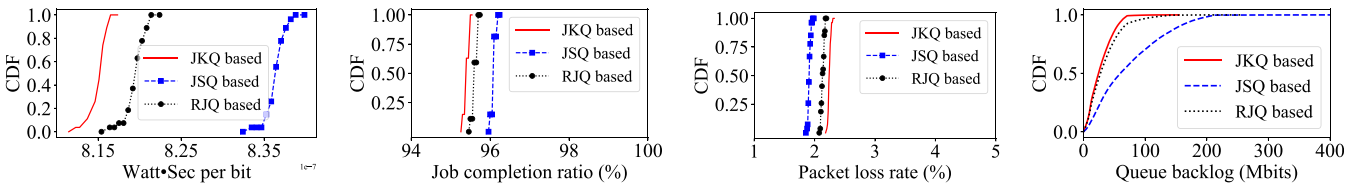


Fig. 7. Metrics of algorithms when β is fixed at 1, with 7,310 synthesized requests, 48 LEO servers, under $v(c)$ = {5.03, 3.46, 1.0} if c is {"Good", "Medium", "Bad"}, respectively.

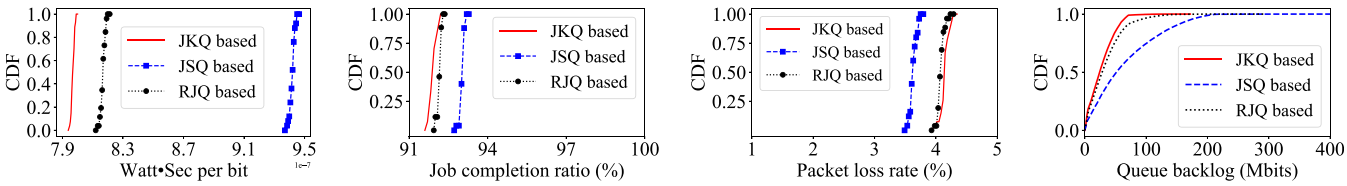


Fig. 8. Metrics of algorithms when β is fixed at 1, with 7,310 synthesized requests, 48 LEO servers, under $v(c)$ = {10, 2, 0.1} if c is {"Good", "Medium", "Bad"}, respectively.

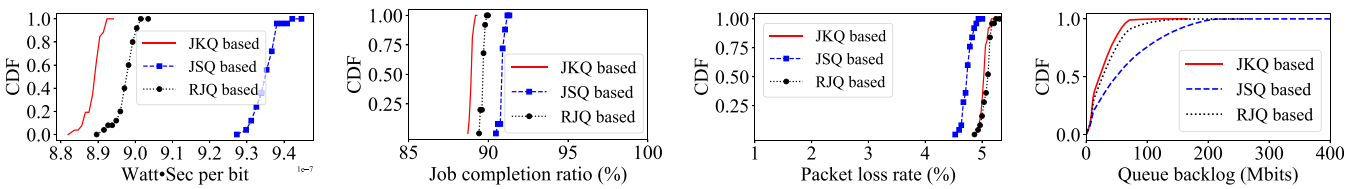


Fig. 9. Metrics of algorithms when β is fixed at 1, with 7,310 synthesized requests, 48 LEO servers, under $v(c)$ = {16, 1, 0.01} if c is {"Good", "Medium", "Bad"}, respectively.

algorithm is that the coflow-like JKQ based job-dispatch strategy dispatches jobs to a larger number of satellites that are with good weather conditions for its downlinks or with lower queue backlogs in a high probability than JSQ and RJQ do.

5.4.3 *Impact of Varying-Scale of Channel Attenuation Rate*

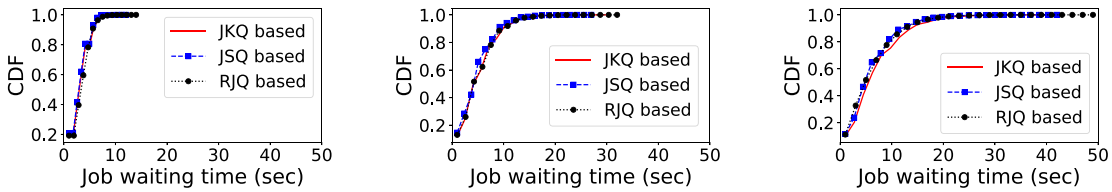
As aforementioned, we vary $v(c)$ within three combinations, i.e., {5.03, 3.46, 1.0}, {10, 2, 0.1} and {16, 1, 0.01}, to evaluate the impact of different varying scales of channel attenuation rate caused by three types of weather condition. As illustrated by Figs. 7, 8 and 9, we show the performance of three algorithms in terms of energy efficiency, JCR, Cumulative Distribution Function (CDF) of packets lost and CDF of backlog, under three attenuation rate combinations, respectively.

First, by comparing Figs. 7a, 8a and 9a, we can see that the difference of energy efficiency of three algorithms is small under low varying-scale of channel attenuation rate, i.e., when $v(c)$ = {5.03, 3.46, 1.0} if c is {"Good", "Medium", "Bad"}, respectively. However, under large varying-scale of channel attenuation rate combinations, i.e., when $v(c)$ = {10, 2, 0.1} and {16, 1, 0.01}, the energy efficiency of our proposed JKQ based algorithm shows growing benefits than the other two benchmarks. This feature proves the robustness of our JKQ based algorithm, because it has the best energy efficiency under the channel transmission rate is super sensitive to the weather conditions when $v(c)$ = {16, 1, 0.01} if c is {"Good", "Medium", "Bad"}, respectively.

Second, from Figs. 7b, 8b and 9b, we observe that the job completion ratios of all three algorithms perform very close to each other. On the other hand, when the varying-scale of channel attenuation rate becomes large, the JCR of algorithms degrades a little bit. The reason can be attributed to that the transmission rate becomes very low when a channel meets the bad weather condition under the large varying-scale of channel attenuation rate. Thus, there are more jobs, which cannot complete their chunk-downloading under this situation, than that under a milder varying-scale of channel attenuation rate.

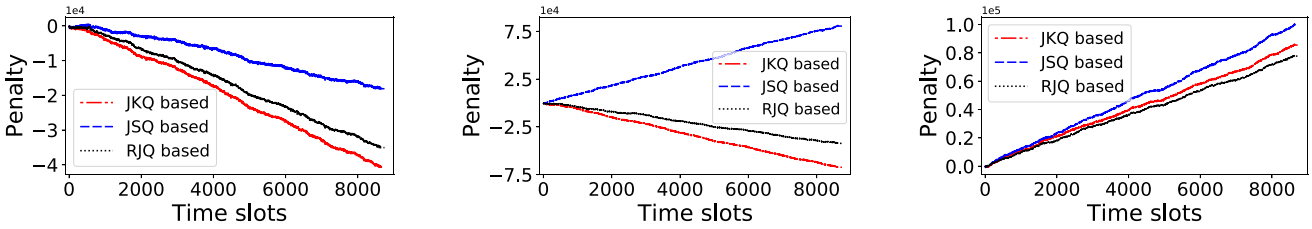
Third, Figs. 7c, 8c and 9c show the CDFs of packet-loss rates. It can be observed that: (1) the JSQ based algorithm has a lowest packet-loss performance among the three algorithms; (2) the packet-loss performance of the JKQ based algorithm is similar to that of the RJQ based algorithm. This is because the possibility of encountering bad transmission conditions under JKQ and RJQ based algorithms is higher than that under the JSQ based algorithm. Associated with the JCR performance, the packet-loss amount grows when the varying-scale of channel attenuation rate becomes large. The reason is same with that as analyzed for the JCR performance.

We show the CDFs of queue backlogs of algorithms in Figs. 7d, 8d and 9d, under three varying scales of channel attenuation rate, respectively. We see that there is almost no



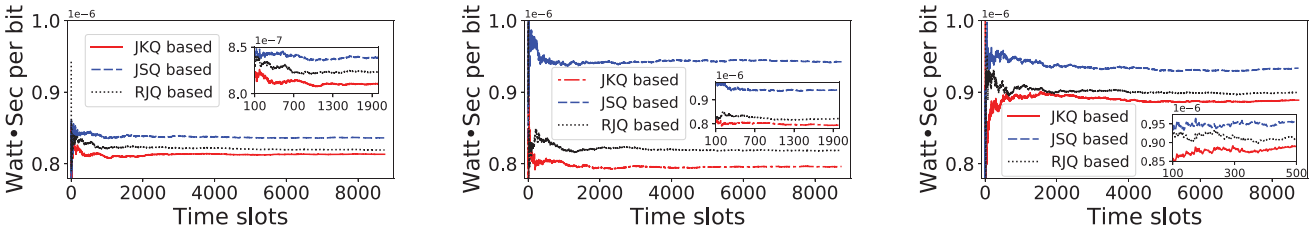
(a) $v(c)$ = {5.03, 3.46, 1} (b) $v(c)$ = {10, 2, 0.1} (c) $v(c)$ = {16, 1, 0.01}

Fig. 10. Job Waiting Time (JWT) of algorithms when β is fixed at 1 and V is fixed at 10, under different varying scales of channel attenuation rate.



(a) $v(c)$ = {5.03, 3.46, 1} (b) $v(c)$ = {10, 2, 0.1} (c) $v(c)$ = {16, 1, 0.01}

Fig. 11. *Penalty* performance of algorithms when β is fixed at 1 and V is fixed at 10, under different varying scales of channel attenuation rate.



(a) $v(c)$ = {5.03, 3.46, 1} (b) $v(c)$ = {10, 2, 0.1} (c) $v(c)$ = {16, 1, 0.01}

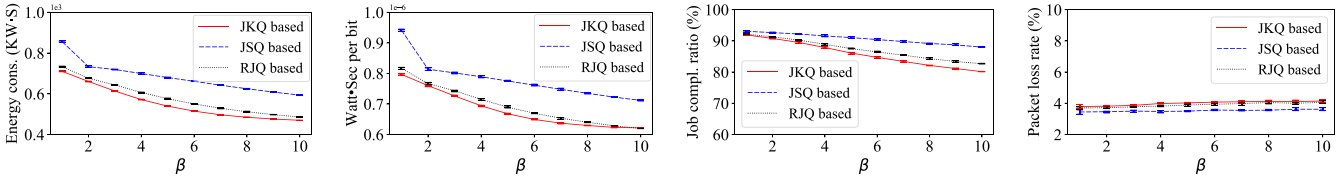
Fig. 12. *Watt·Sec per bit* performance of algorithms when β is fixed at 1 and V is fixed at 10, under different varying scales of channel attenuation rate.

difference among the three CDF figures, which indicate that the backlog performance yielded by the three algorithms is not affected by the varying scales of channel attenuation rate. It is also worth noting that the proposed JKQ based algorithm has the lowest backlog performance among the three algorithms. The reason is very clearly illustrated in Fig. 3 and omitted here.

Next, by recording all the *job waiting time* (JWT) of each job allocated to LEO queues, we compare the JWT performance of three algorithms under different combinations of channel attenuation rate in Fig. 10. These three CDF figures show that: (1) the JWT of all three algorithms performs the same under each group of simulation setting; (2) the overall JWT grows when the varying-scale of channel attenuation rate changes from small to large. The first observation indicates that the JWT performance is not affected by the job-dispatch schemes. The reason of the second observation is that the large varying-scale of channel attenuation rate, where $v(c)$ = {10, 2, 0.1} or {16, 1, 0.01}, makes the packet transmission in downlinks very difficult under the non-good weather conditions. Thus, the jobs waiting in queues need to stay for longer durations than that under a milder varying-scale of channel attenuation rate.

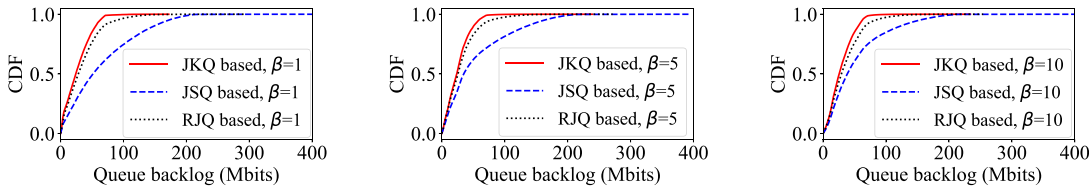
We then evaluate the penalty performance under the different varying scales of channel attenuation rate through Fig. 11, from which we see different performance curves under different varying scales of $V(c)$. For example, penaltiesintend to descend in Fig. 11a, ascend in Fig. 11c, and mixtured in Fig. 11b. As we have explained in Fig. 6c, the penalty is only a part of the transformed *penalty-plus-drift* minimization problem (21), where the *Lyapunov drift* $\Delta(\Theta(t))$ is minimized for short queues throughout the LEO satellites as well. Under different channel attenuation rates where the short queues are easily maintained when $V(c)$ = {5.03, 3.46, 1}, a low penalty (indicates a high overall data amount downloaded, or a low cumulative energy consumption, or both) is easier to acheive. In contrast, under large varying-scale of channel attenuation rate, e.g., when $V(c)$ = {10, 2, 0.1} and {16, 1, 0.01}, the data transmission via the downlinks will heavily be impacted by the bad weather conditions, and large queue backlogs are easily to be incurred. Thus, algorithms tend to ensure the system stability by sacrificing the penalty performance. That is why we see the penalty ascending trends in Figs. 11b and 11c.

Finally, we study the impact of varying-scale of channel attenuation rate on the performance of *Wat-Sec per bit* under three algorithmes. The comparison is shown in Fig. 12. From which, we can see that the proposed JKQ based algorithm significantly outperforms the other two on energy consumptions. On the other hand, it is can be seen that under the large varying-scale of channel attenuation rates, e.g., when $v(c)$ = {10, 2, 0.1} and {16, 1, 0.01}, the energy



(a) Energy consum. (KW·s). (b) Watt·Sec per bit. (c) Job completion ratio (%). (d) Packet loss rate (%).

Fig. 13. Metric examination among the three algorithms with 7,310 requests and 48 LEO servers, while varying β from 1 to 10, and the channel attenuation rate $v(c)$ is set to {10, 2, 0.1} while c is {"Good", "Medium", "Bad"}, respectively.



(a) β=1 (b) β=5 (c) β=10

Fig. 14. Backlog comparison of three algorithms while varying β within {1, 5, 10}, and setting the channel attenuation rate $v(c)$ to {10, 2, 0.1} if c is {"Good", "Medium", "Bad"}, respectively.

consumption of per bit of downloaded data becomes higher than that under a small varying-scale of channel attenuation rate when $v(c)$ = {5.03, 3.46, 1}.

5.4.4 *Impact of Varying β*

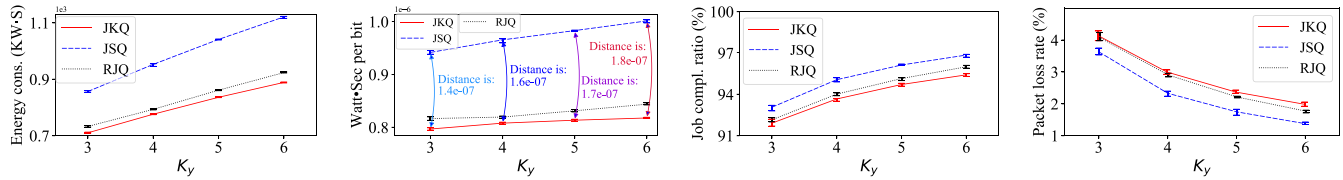
We then evaluate the impact of the weight of energy consumption by varying β from 1 to 10, using the synthesized trace file associated with 7,310 requests and 48 LEO servers. The channel attenuation rate $v(c)$ is set to {10, 2, 0.1} while c is {"Good", "Medium", "Bad"}, respectively. In this group of simulations, the admission ratios of all algorithms are always 100 percent and thus omitted here. We mainly examine the performance of algorithms in terms of the energy consumption, energy efficiency, job completion ratio and packet loss rate.

Figs. 13a and 13b illustrate the overall cumulative energy consumption and the energy efficiency versus varying β, respectively. We can see that: (1) the performance trends shown in these two figures are same; (2) the proposed JKQ based algorithm exhibits the best energy efficiency against the other two, especially much higher than that of the JSQ based algorithm; (3) the increasing β makes the cumulative energy consumption decreasing and the energy efficiency increasing. The reason of the third observation is that a large β implies a large weight of energy consumption in the objective function of the original penalty-minimization problem. Thus, the growing β degrades the energy consumption. When β is larger than 8, the benefit of large energy-consumption weight becomes saturated gradually. Furthermore, a coin always has two sides. A large weight of energy-consumption, i.e., a large β, also induces negative effects under our file acquisition system. Because once the weight of energy-consumption grows bigger, the file acquisition system tends to save energy. However, this may damage the job completion ratio and increase packet loss rate. These two concerns are verified by Figs. 13c and 13d, respectively. Fig. 13c shows that the *job completion ratio* performance of JKQ based algorithm is slightly lower than that of the JSQ based algorithm, while β increases to 10. In Fig. 13d, the packet loss rate of JKQ based algorithm is a little bit higher than that of the JSQ based algorithm. Therefore, setting an appropriate β needs to consider the nontrivial tradeoff between the energy-consumption and the job completion ratio.

Finally, under the same parameter settings with Fig. 13, Figs. 14a, 14b and 14c exhibit the backlog comparison of the three algorithms while setting β to 1, 5, and 10. We then have the following two observations. (1) Under each β, the proposed JKQ based algorithm outperforms the JSQ based algorithm overwhelmingly in terms of the queue backlog size. Furthermore, since the RJQ based algorithm randomly selects the target K queues to allocate K jobs for each request, its backlog performance is much better than that of the JSQ based, but still worse than that of the JKQ based. (2) The varying β has no effect on the distribution of queue backlog size for all three algorithms.

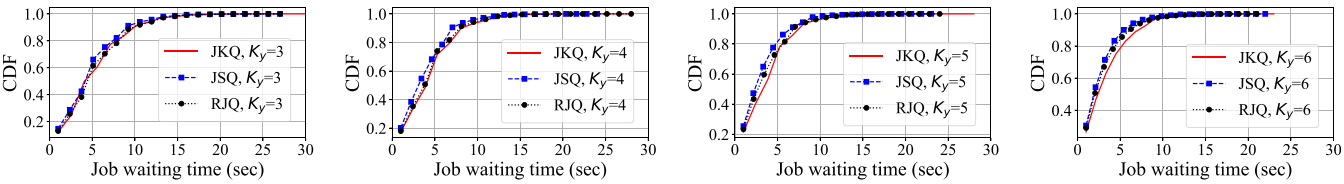
5.4.5 *Impact of Varying Storage Parameters K_y*

In this part, we study the impact of storage parameter K_y by varying it from 3 to 6, while fixing N_y at 6. From the four metrics shown in Fig. 15, we can observe that the energy consumption, Watt·Sec per bit, and the job completion ratio grow nearly linearly with the growth of K_y, whereas the packet loss rate drops. From Fig. 15b, we also see that the *Wat·Sec per bit* performance distance between the proposed JKQ and JSQ algorithms enlarges from 1.4e-7 to 1.8e-7 while K_y grows from 3 to 6. We attribute this result to the following reason. When K_y grows larger, the proposed JKQ algorithm will dispatch the K_y chunk-downloading jobs of a file-download request to K_y satellite queues, much more than the JSQ does. Thus the merit of JKQ will become more apparent comparing with that of JSQ. Fig. 16 shows that the job-waiting-time of all downloading jobs shrinks while K_y enlarges. This is because the size of each chunk-downloading job becomes smaller when each file was divided into more chunks. Due to the same reason, Fig. 17 illustrates that



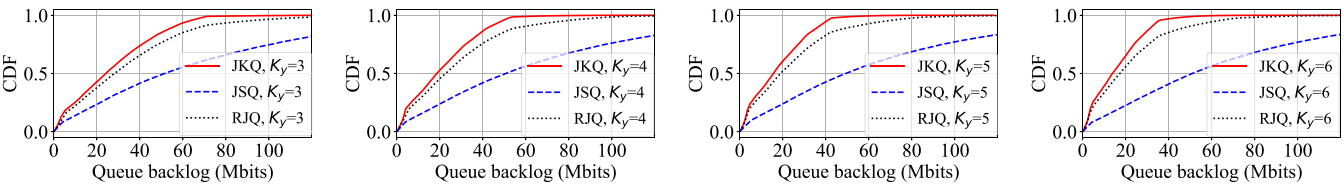
(a) Energy consum. (KW·s). (b) Watt·Sec per bit. (c) Job completion ratio (%). (d) Packet loss rate (%).

Fig. 15. Metric examination among the three algorithms with 7,310 requests and 48 LEO servers, while varying K_y from 3 to 6, fixing $N_y = 6$, and the channel attenuation rate $v(c)$ is set to {10, 2, 0.1} while c is {"Good", "Medium", "Bad"}, respectively.



(a) $K_y = 3$. (b) $K_y = 4$. (c) $K_y = 5$. (d) $K_y = 6$.

Fig. 16. Job-waiting-time (JWT) performance among the three algorithms with 7,310 requests and 48 LEO servers, while varying K_y from 3 to 6, fixing $N_y = 6$, and the channel attenuation rate $v(c)$ is set to {10, 2, 0.1} while c is {"Good", "Medium", "Bad"}, respectively.



(a) $K_y = 3$. (b) $K_y = 4$. (c) $K_y = 5$. (d) $K_y = 6$.

Fig. 17. Backlog comparison of three algorithms while varying $K_y$from 3 to 6, fixing $N_y = 6$, and setting the channel attenuation rate $v(c)$ to {10, 2, 0.1} if c is {"Good", "Medium", "Bad"}, respectively.

the queue backlogs of satellites keep decreasing while K_y grows from 3 to 6.

Through all trace-driven simulation results, we can summarize the following facts. The proposed JKQ algorithm outperforms the batch-based dispatch algorithm [24], [25] significantly in terms of job completion ratio. Under different varying scales of channel attenuation rate, the proposed JKQ based algorithm outperforms JSQ and RJQ in terms of energy efficiency. JKQ is also able to yield shorter queue backlogs than that under the other two algorithms.

6 Conclusion

In this paper, we studied how to maximize the overall amount of data admitted while minimizing the energy consumption when transmitting data from satellites to ground. We proposed an online scheduling framework. Particularly, we devised a novel coflow-like JKQ based algorithm, which can drastically reduce backlogs of queues in satellites. We also throughly analyzed the optimality gap between the solution delivered by the proposed algorithm and the theoretical optimal solution, and the queue stability. We finally evaluated the performance of the proposed algorithm through experimental simulations using the real-world traces. The evaluation results show that the proposed online algorithm exhibits its merits in terms of short queue backlogs and high energy efficiency.

Acknowledgments

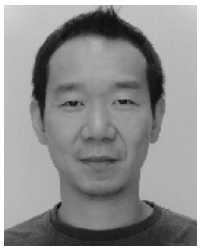
This work is partially supported by NSFC under research grant no. 61872310, 61872195, 61572262, the Kyoto University Research Fund for Young Scientists (Start-up) FY-2018, and the Start-up Research Fund (67000-18841220) from Sun Yat-Sen University, China.

References

[1] Spacebelt: The global cloud platform above all others, 2016. [Online]. Available: http://spacebelt.com/
[2] M. Strohbach, J. Daubert, H. Ravkin, and M. Lischka, *Big Data Storage*. Berlin, Germany: Springer International Publishing, 2016, pp. 119–141.
[3] The information ultra-highway for enterprises and governments, 2017. [Online]. Available: http://cloudconstellation.com/
[4] S. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *Proc. IEEE Conf. Comput. Commun.*, 2012, pp. 702–710.
[5] S. Maguluri and R. Srikant, "Scheduling jobs with unknown duration in clouds," *IEEE/ACM Trans. Netw.*, vol. 22, no. 6, pp. 1938–1951, Dec. 2014.
[6] F. Liu, Z. Zhou, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in SaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2648–2658, Oct. 2014.
[7] M. J. Neely, E. Modiano, and C. E. Rohrs, "Power allocation and routing in multibeam satellites with time-varying channels," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 138–152, Feb. 2003.
[8] Y. Yang, M. Xu, D. Wang, and Y. Wang, "Towards energy-efficient routing in satellite networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3869–3886, Dec. 2016.
[9] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.

[10] Q. Liang and E. Modiano, "Coflow scheduling in input-queued switches: Optimal delay scaling and algorithms," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2017, pp. 1–9.
[11] W. Wang, S. Ma, B. Li, and B. Li, "Coflex: Navigating the fairness-efficiency tradeoff for coflow scheduling," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2017, pp. 1–9.
[12] H. Wu, J. Li, H. Lu, and P. Hong, "A two-layer caching model for content delivery services in satellite-terrestrial networks," in *Proc. IEEE Global Commun. Conf.*, 2016, pp. 1–6.
[13] X. Jia, T. Lv, F. He, and H. Huang, "Collaborative data downloading by using inter-satellite links in leo satellite networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1523–1532, Mar. 2017.
[14] M. Cello, M. Marchese, and F. Patrone, "ColdSel: A selection Algorithm to mitigate congestion situations over nanosatellite networks," in *Proc. IEEE Global Commun. Conf.*, 2016, pp. 1–6.
[15] Oneweb satellite constellation, Mar. 2017. [Online]. Available: https://en.wikipedia.org/wiki/OneWeb_satellite_constellation
[16] Spacex satellite constellation, Aug. 2019. [Online]. Available: https://en.wikipedia.org/wiki/SpaceX_satellite_constellation
[17] Gisat-1, Dec. 2018. [Online]. Available: https://en.wikipedia.org/wiki/GiSAT-1
[18] H. Huang, S. Guo, and K. Wang, "Envisioned wireless big data storage for low-earth-orbit satellite-based cloud," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 26–31, Feb. 2018.
[19] W. Deng, F. Liu, H. Jin, C. Wu, and X. Liu, "MultiGreen: Cost-minimizing multi-source datacenter power supply with online control," in *Proc. 4th Int. Conf. Future Energy Syst.*, 2013, pp. 149–160.
[20] W. Deng, F. Liu, H. Jin, B. Li, and D. Li, "Harnessing renewable energy in cloud datacenters: Opportunities and challenges," *IEEE Netw.*, vol. 28, no. 1, pp. 48–55, Jan./Feb. 2014.
[21] Z. Zhou, F. Liu, B. Li, B. Li, H. Jin, R. Zou, and Z. Liu, "Fuel cell generation in geo-distributed cloud services: A quantitative study," in *Proc. IEEE 34th Int. Conf. Distrib. Comput. Syst.*, 2014, pp. 52–61.
[22] Z. Zhou, F. Liu, R. Zou, J. Liu, H. Xu, and H. Jin, "Carbon-aware online control of geo-distributed cloud services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2506–2519, Sep. 2016.
[23] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.
[24] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: Distributed, low latency scheduling," in *Proc. 24th ACM Symp. Operating Syst. Principles*, 2013, pp. 69–84.
[25] L. Ying, R. Srikant, and X. Kang, "The power of slightly more than one sample in randomized load balancing," *Math. Operations Res.*, vol. 42, pp. 692–722, 2017.
[26] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1378–1391, Oct. 2013.
[27] D. Xu and X. Liu, "Geographic trough filling for Internet datacenters," in *Proc. IEEE Conf. Comput. Commun.*, 2012, pp. 2881–2885.
[28] J. Castaing, "Scheduling downloads for multi-satellite, multi-ground station missions," in *Proc. Small Satellite Conf.*, 2014, pp. 1–12.
[29] J. Choi and V. Chan, "Predicting and adapting satellite channels with weather-induced impairments," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 38, no. 3, pp. 779–790, Jul. 2002.
[30] L. J. Ippolito and L. J. Ippolito Jr, *Satellite Communications Systems Engineering: Atmospheric Effects, Satellite Link Design and System Performance*. Hoboken, New Jersey, United States: Wiley, 2017.
[31] B. Li, A. Ramamoorthy, and R. Srikant, "Mean-field-analysis of coding versus replication in cloud storage systems," in *Proc. IEEE Conf. Comput. Commun.*, 2016, pp. 1–9.
[32] P. Patel, A. H. Ranabahu, and A. P. Sheth, "Service level agreement in cloud computing," in *Proc. Conf. Object Oriented Program. Syst. Lang. Appl.*, Oct. 2009, pp. 1–10.
[33] B. Yang, Y. Wu, X. Chu, and G. Song, "Seamless handover in software-defined satellite networking," *IEEE Commun. Lett.*, vol. 20, no. 9, pp. 1768–1771, Sep. 2016.
[34] Technical Report, Aug. 2019. [Online]. Available: https://drive.google.com/file/d/1-YtdLytVO2y2p_yvCF51cuFZ-po_LNtC/view?usp=sharing
[35] J. Bao, B. Zhao, W. Yu, Z. Feng, C. Wu, and Z. Gong, "OpenSAN: A software-defined satellite network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 347–348, 2014.
[36] L. Bertaux, S. Medjiah, P. Berthou, S. Abdellatif, A. Hakiri, P. Gelard, F. Planchou, and M. Bruyere, "Software defined networking and virtualization for broadband satellite networks," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 54–60, Mar. 2015.
[37] J. Nobre, D. Rosario, C. Both, E. Cerqueira, and M. Gerla, "Toward software-defined battlefield networking," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 152–157, Oct. 2016.
[38] Systems Tool Kit, July 2019. [Online]. Available: https://en.wikipedia.org/wiki/Systems_Tool_Kit/
[39] J. Wang, L. Li, and M. Zhou, "Topological dynamics characterization for leo satellite networks," *Comput. Netw.*, vol. 51, no. 1, pp. 43–53, 2007.
[40] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch global, cache local: Youtube network traffic at a campus network-measurements and implications," in *Proc. SPIE 6818, Multimedia Comput. Netw. 2008, 681805*, Jan. 28, 2008. [Online]. Available: https://doi.org/10.1117/12.774903

Huawei Huang (M'16) received the PhD degree in computer science and engineering from the University of Aizu, Japan, in 2016. He is currently an associate professor at the School of Data and Computer Science, Sun Yat-Sen University, China. His research interests mainly include SDN/NFV, edge computing, and blockchain. He received the best paper award of TrustCom-2016. He used to be a visiting scholar at Hong Kong Polytechnic University (2017-2018), a post-doctoral research fellow of JSPS (2016-2018), and an assistant professor at Kyoto University, Japan (2018-2019). He is a member of the IEEE.



Song Guo (M'02-SM'11) received the PhD degree in computer science from the University of Ottawa. He is a full professor with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. He has authored or coauthored more than 450 papers in major conferences and journals. His current research interests include big data, cloud and edge computing, mobile computing, and distributed systems. He was a recipient of the 2019 TCBD Best Conference Paper Award, the 2018 IEEE TCGCC Best Magazine Paper Award, the 2017 IEEE SYSTEMS JOURNAL Annual Best Paper Award, and six other Best Paper Awards from IEEE/ACM conferences. He was an IEEE Communications Society distinguished lecturer. He has served as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Cloud Computing*, the *IEEE Transactions on Emerging Topics in Computing*, etc. He also served as the general and program chair for numerous IEEE conferences. He currently serves in the Board of Governors of the IEEE Communications Society. He is a senior member of the IEEE.

Weifa Liang (M'99-SM'01) received the BSc degree from Wuhan University, China, in 1984, the ME degree from the University of Science and Technology of China, in 1989, and the PhD degree from the Australian National University, in 1998, all in computer science. He is currently a professor with the Research School of Computer Science, Australian National University. His research interests include design and analysis of energy-efficient routing protocols for wireless ad hoc and sensor networks, cloud computing, software-defined networking, virtualized network function services, design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.

Kun Wang (M'13-SM'17) received two PhD degrees in computer science from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2009, and from the University of Aizu, Aizuwakamatsu, Japan, in 2018. He was a post-doctoral fellow in UCLA from 2013 to 2015, where he is a senior research professor. He was a research fellow with the Hong Kong Polytechnic University, Hong Kong, from 2017 to 2018, and a professor at Nanjing University of Posts and Telecommunications. His current research interests are mainly in the area of big data, wireless communications and networking, energy Internet, and information security technologies. He is the recipient of IEEE GLOBECOM 2016 Best Paper Award, IEEE TCGCC Best Magazine Paper Award 2018, IEEE TCBD Best Conference Paper Award 2019, and IEEE ISJ Best Paper Award 2019. He serves/served as an associate editor of *IEEE Access*, an editor of the *Journal of Network and Computer Applications*, and a guest editor of *IEEE Network*, *IEEE Access*, *Future Generation Computer Systems*, *Peer-to-Peer Networking and Applications*, the *IEICE Transactions on Communications*, the *Journal of Internet Technology*, and *Future Internet*. He is a senior member of the IEEE.

Yasuo Okabe received the ME and PhD degrees in engineering from the Department of Information Science, Kyoto University, in 1988 and 1994, respectively. He joined Kyoto University as an instructor of Faculty of Engineering, in 1988. He advanced to an associate professor of the Data Processing Center, in 1994 and moved to the Graduate School of Informatics, in 1998. Since 2002, he has been a professor of division of networking research, Academic Center for Computing and Media Studies. His current research interest includes Internet architecture, distributed algorithms, and network security. He is a member of the IEICE, IPSJ. ISCIE, JSSST, IEEE, and ACM.