Cost-Minimizing Dynamic Migration of Content Distribution Services into Hybrid Clouds

Xuanjia Qiu*, Hongxing Li*, Chuan Wu*, Zongpeng Li† and Francis C.M. Lau*
*Department of Computer Science, The University of Hong Kong, Hong Kong,
{xjqiu,hxli,cwu,fcmlau}@cs.hku.hk
†Department of Computer Science, University of Calgary, Canada, zongpeng@ucalgary.ca

Abstract—With the recent advent of cloud computing technologies, a growing number of content distribution applications are contemplating a switch to cloud-based services, for better scalability and lower cost. Two key tasks are involved for such a move: to migrate the contents to cloud storage, and to distribute the web service load to cloud-based web services. The main issue is to best utilize the cloud as well as the application provider's existing private cloud, to serve volatile requests with service response time guarantee at all times, while incurring the minimum operational cost. While it may not be too difficult to design a simple heuristic, proposing one with guaranteed cost optimality over a long run of the system constitutes an intimidating challenge. Employing Lyapunov optimization techniques, we design a dynamic control algorithm to optimally place contents and dispatch requests in a hybrid cloud infrastructure spanning geo-distributed data centers, which minimizes overall operational cost over time, subject to service response time constraints. Rigorous analysis shows that the algorithm nicely bounds the response times within the preset QoS target, and guarantees that the overall cost is within a small constant gap from the optimum achieved by a T-slot lookahead mechanism with known future information. We verify the performance of our dynamic algorithm with prototype-based evaluation.

Index Terms—Hybrid Cloud, Content Distribution, Dynamic Migration, Lyapunov Optimization

1 Introduction

Cloud computing technologies have enabled rapid provisioning and release of server utilities (CPU, storage, bandwidth) to users anywhere, anytime. To exploit the diversity of electricity costs and to provide service proximity to users in different geographic regions, a cloud service often spans multiple data centers over the globe, *e.g.*, Amazon CloudFront [1], Microsoft Azure [2], Google App Engine [3]. The elastic and on-demand nature of resource provisioning has made cloud computing attractive to providers of various applications. More and more new applications are being created on the cloud platform [4][5][6], while many existing applications are also considering the cloud-ward move [7][8], including content distribution applications [9][10].

As an important category of popular Internet services, content distribution applications, *e.g.*, video streaming, web hosting and file sharing, feature large volumes of contents and demands that are highly dynamic in the temporal domain. A cloud platform with multiple, distributed data centers is ideal to host such a service, with substantial advantages over a traditional private or public content distribution network (CDN) based solution, in terms of more agility and significant cost reduction with respect to machines, bandwidth, and management. In this way, the application providers can focus their business more on content provisioning, rather than IT infrastructure maintenance.

Two major components exist in a typical content distribution application, namely back-end storage for keeping the contents, and front-end web services to serve the requests. Both can be migrated to the cloud: contents can be stored in storage servers in the cloud, and requests can be distributed to cloud-based web services. Therefore, the key challenge for cloud-ward move of a content distribution application is how to efficiently replicate contents and dispatch requests across multiple cloud data centers, as well as the provider's existing private cloud, such that good service response time is guaranteed and only modest operational expenditure is incurred.

It may not be too hard to design a simple heuristic for dynamic content placement and load distribution in the hybrid cloud; however, proposing one with guarantee of cost optimality over a long run of the system, is an intriguing yet intimidating challenge, especially when arbitrary arrival rates of requests are considered. Some existing work [7][8][9][10] have advocated optimal application migration into clouds, but none focus on guaranteeing over-time cost minimization with a dynamic algorithm.

In this paper, we present a generic optimization framework for dynamic, cost-minimizing migration of content distribution services into a hybrid cloud (*i.e.*, private and public clouds combined), and design a joint content placement and load distribution algorithm that minimizes overall operational cost over time, subject to service response time constraints. Our design is rooted in Lyapunov optimization theory [11][12], where cost minimization and response time guarantee are achieved simultaneously by efficient scheduling of content mi-

gration and request dispatching among data centers. Lyapunov optimization provides a framework for designing algorithms with performance arbitrarily close to the optimal performance over a long run of the system, without the need for any future information. It has been extensively used in routing and channel allocation in wireless networks [11][13], and has only recently been introduced to address resource allocation problems in a few other types of networks [14][15]. We tailor Lyapunov optimization techniques in the setting of a hybrid cloud, to dynamically and jointly resolve the optimal content replication and load distribution problems.

The contribution of this work can be summarized as follows:

- ▷ We propose a generic optimization framework for dynamic, optimal migration of a content distribution service to a hybrid cloud consisting of a private cloud and public geo-distributed cloud services.
- ▷ We design a joint content placement and load distribution algorithm for dynamic content distribution service deployment in the hybrid cloud. Providers of content distribution services can practically apply it to guide their service migration, with confidence in cost minimization and performance guarantee, regardless of the request arrival pattern.
- ▷ We demonstrate optimality of our algorithm with rigorous theoretical analysis and prototype-based evaluation. The algorithm nicely bounds the response times (including queueing and round-trip delays) within the preset QoS target in cases of arbitrary request arrivals, and guarantees that the overall cost is within a small constant gap from the optimum achieved by a T-slot lookahead mechanism with information into the future.

In the rest of the paper, we discuss related work in Sec. 2, present the optimization framework in Sec. 3, design a joint content placement and load distribution algorithm in Sec. 4, and rigorously analyze its performance in Sec. 5. We evaluate the algorithm with prototype-based evaluation in Sec. 6, and conclude the paper in Sec. 7.

2 Related work

Migration of applications into clouds: A number of research projects have emerged in recent years that explore the migration of services into a cloud platform. Hajjat *et al.* [7] develop an optimization model for migrating enterprise IT applications onto a hybrid cloud. Their model takes into account enterprise-specific constraints, such as transaction delays and security policies. One-time optimal service deployment is considered, while our work investigates optimal dynamic migration over time, to achieve the long-term optimality. Zhang *et al.* [8] propose an intelligent algorithm to factor workload and dynamically determine the service placement across the public cloud and the private cloud. Their focus is on designing an algorithm for distinguishing base workload and trespassing workload.

Migration of content delivery services into clouds: Some research efforts have been put into migrating generic content delivery services onto clouds. MetaCDN by Pathan *et al.* [16] is a proof-of-concept testbed, experiments on which show that deploying content delivery based on storage clouds can improve utility, based on primitive content placement and request routing mechanisms. Chen *et al.* [17] propose to build CDNs in the cloud in order to minimize cost under the constraints of QoS requirement, but they only propose greedy-strategy based heuristics without provable properties. In contrast, we target an optimization framework which renders optimal migration solutions for long run of the system.

Some work focuses on migrating specific types of content delivery services onto clouds, *e.g.*, social networking service, or video streaming service. Cheng *et al.* [10] study the partition of social data and their storage onto a number of cloud servers, to migrate a social networking application into the cloud. It focuses on balancing the data access load, by considering social relationships and user access patterns in the data storage. Li *et al.* [9] advocate cost saving by partial migration of a VoD service to a content cloud. Heuristic strategies are proposed to decide the update of cloud contents, which are verified by trace-driven evaluations. Our work focuses on cost minimization in migration of a generic content distribution application, based on differentiated charging models of different data centers.

Application of Lyapunov optimization theory: Lyapunov optimization was developed from the stochastic network optimization theory [11][12], and has been applied in routing and channel allocation in wireless networks [18][11][13], as well as in a few other types of networks including peer-to-peer networks [15]. Maguluri *et al.* [19] propose various VM configuration scheduling algorithms for cloud computing platforms, that achieve arbitrary fraction of the capacity region of the cloud. But their model does not take into consideration delay guarantee, which is an important component in our optimization framework. The work of Ren *et al.* [20] also considers an online scheduler that dispatches workloads across multiple geographically distributed data centers subject to delay requirements. It assumes where each job's data is stored is fixed and known. However in our work we further incorporate the decision on data migration into the scheduling. The work of Amble *et al.* [14] is close to ours in that it also utilizes Lyapunov function to study request routing and content caching, but in the setting of CDNs with capacitated caches and links. They investigate the optimality of different caching policies. Given a workload within the capacity region, they prove that several types of caching and content eviction methods can each provide a throughput equal to the workload. Instead, our study focuses on optimal migration of content distribution services onto a hybrid cloud, such that the operational cost is minimized while

service delay bound is guaranteed.

Related service placement problems: Placement of services onto different sites has been investigated [21][22] based on the theories of Facility Location Problems (FLP) [23], including the *k*-Median Problem (*k*MP) [24] and *k*-Component Multi-Site Placement Problem (*k*-CMSP) [22]. Such a problem typically involves an NP-hard integer program, and can only be solved by approximation algorithms; it focuses on one-time optimization with fixed service demands, rather than online optimization over a long run of the system. Our work is significantly different, which applies Lyanopuv optimization theory to pursue the global optimality with dynamic request arrivals over time.

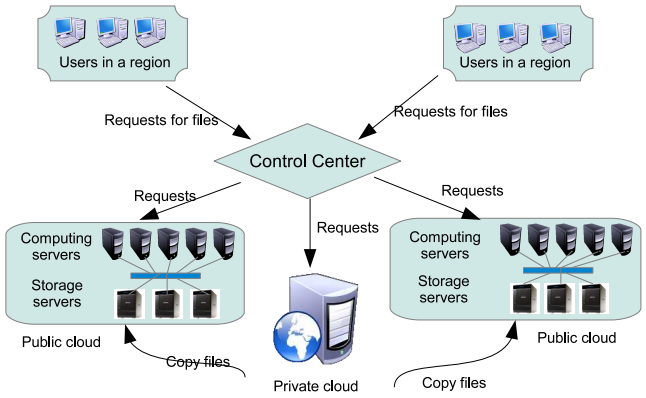

Fig. 1. The system architecture.

3 The Service Migration Problem

3.1 System Model

We consider a typical content distribution application, which provides a collection of contents (files), denoted as set $\mathcal{M}$, to users spreading over multiple geographical regions. There is a private cloud owned by the provider of the content distribution application, which stores the original copies of all the contents. The private cloud has an overall upload bandwidth of b units for serving contents to users.

There is a public cloud consisting of data centers located in multiple geographical regions, denoted as set $\mathcal{N}$. One data center resides in each region. There are two types of inter-connected servers in each data center: storage servers for data storage, and computing servers that support the running and provisioning of virtual machines (VMs). Servers inside the same data center can access each other via a certain DCN (Data Center Network).

The provider of the content distribution application (*application provider*) wishes to provision its service by exploiting a *hybrid cloud* architecture, which includes the geo-distributed public cloud and its private cloud. The major components of the content distribution application include: (i) back-end storage of the contents and (ii) front-end web service that serves users' requests for contents. The application provider may migrate both service components into the public cloud: contents can be replicated in storage servers in the cloud, while requests can be dispatched to web services installed on VMs on the computing servers. An illustration of the system architecture is given in Fig. 1.

Our objective in this paper is to design a dynamic, optimal algorithm for the application provider to strategically make the following decisions for service migration into the hybrid cloud architecture: (i) *content replication*: which content should be replicated in which data center at each time? (ii) *request distribution*: How many requests for a content should be directed to the private cloud and to each of the data centers that store this content at the time? The goal is to pursue the minimum operational cost for the application provider over time, while ensuring the service quality of content distribution.

We next develop an optimization framework to characterize the optimal content distribution service migration problem. Important notations are summarized in Table 1 for ease of reference.

TABLE 1
Notations

$\mathcal{M}$	File set
$\mathcal{N}$	Region set
$v^{(m)}$	Size of file m, in bytes
$a_j^{(m)}(t)$	No. of requests for file m from region j at time slot t
$Q_j^{(m)}(t)$	Size of request queue for file m in region j at t
A_{max}	Max. no. of requests for file m from region j in a time slot
$s_j^{(m)}(t)$	No. of requests dispatched from $Q_j^{(m)}$ to private cloud at t
$c_{ji}^{(m)}(t)$	No. of requests dispatched from $Q_j^{(m)}$ to data center i at t
$y_i^{(m)}(t)$	Binary var: store file m on data center i or not at t.
b	Max. no. of requests the private cloud can serve in a time slot
μ_{max}	Max. no. of requests dispatched from each request queue to a data center in a time slot, i.e., $c_{ji}^{(m)}(t) \leq \mu_{max}$
g_i	Charge for uploading a byte from data center i
o_i	Charge for downloading a byte into data center i
f_i	Charge for renting one VM instance in data center i
r_i	No. of requests a VM in data center i can serve in a time slot
p_i	Charge for storing a byte on data center i
$q_i^{(m)}$	Charge for uploading file m from data center i
h	Time-averaged charge for uploading a byte from the private cloud
$w_i^{(m)}$	Charge for migrating file m to data center i
$W_j^{(m)}$	Bound of queueing delay of requests in queue $Q_j^{(m)}$
$\epsilon_j^{(m)}$	Pre-set constant for controlling queueing delay in $Q_j^{(m)}$
d_j	Round-trip delay between region j and the private cloud
e_{ji}	Round-trip delay between region j and data center i
α	Bound of time-averaged round-trip delay
$G(t)$	Virtual queue for bounding time-averaged round-trip delay
$Z_j^{(m)}(t)$	Virtual queue for bounding queueing delay in $Q_j^{(m)}$
V	A non-negative parameter to control the tradeoff between the operational cost and request response delays

3.2 Cost-Minimizing Service Migration Problem

We suppose that the system runs in a time-slotted fashion. Each time slot is a unit time which is enough for uploading any file $m \in \mathcal{M}$ with size $v^{(m)}$ (bytes) at the unit bandwidth. In time slot t, $a_j^{(m)}(t)$ requests are generated for downloading file $m \in \mathcal{M}$, from users in region j. We assume that the request arrival is an arbitrary process over time, and the number of requests arising from one region for a file in each time slot is upper-bounded by A_{max}.

The cost of uploading a byte from the private cloud is h. The charge for storage at data center i is p_i per byte per unit time. g_i and o_i per byte are charged for uploading from and downloading into data center i, respectively. The cost for renting a VM instance in data center i is f_i per unit time. These charges follow the charging model of leading commercial cloud providers, such as Amazon EC2 [25] and S3[26]. We assume that the storage capacity in each data center is sufficient for storing contents from this content distribution application. We also assume that each request is served at one unit bandwidth, and the number of requests that a VM in data center i can serve per unit time is r_i.

Decision variables. The decision variables in our optimization framework are formulated as follows:

(1) For *content replication*, binary variable $y_i^{(m)}(t)$ indicates whether file m is stored in data center i in time slot t or not. If $y_i^{(m)}(t-1) = 0$ and $y_i^{(m)}(t) = 1$, file m is copied from the private cloud to the data center i at t; if $y_i^{(m)}(t-1) = 1$ and $y_i^{(m)}(t) = 0$, file m is removed from data center i. In other cases, the storage status remain the same. In case of migration, we assume that the video is always copied from the private cloud to the destination data center.

(2) For *dispatching requests* from region j for file m, let $s_j^{(m)}(t)$ be the number of requests to be served by the private cloud in time slot t, and $c_{ji}^{(m)}(t)$ denote the number of requests dispatched to data center i in time slot t, with an upper bound of μ_{max}. Based on the elasticity of clouds, we reasonably assume that $A_{max} < \mu_{max}$. Requests for file m can only be dispatched to data center i when it stores the file, *i.e.*, $c_{ji}^{(m)}(t) > 0$ only if $y_i^{(m)}(t) = 1$. We assume that a data center can serve a file to users in the time slot when the file is being copied to the data center, since replicating the file from the private cloud and serving chunks of the file can be carried out in parallel: after receiving a small portion of the file, a data center can already start to serve the received chunks of the file to users. We assume that upload bandwidth is reserved for replicating files to data centers from the private cloud, and this bandwidth is not counted in b, the maximum units of bandwidth that the private cloud can use to upload contents to users.

Not all requests arising in one time slot are dispatched in the same time slot, subject to capacity constraints. A queue $Q_j^{(m)}$ is maintained to buffer requests for file m generated from users in region j over time, $\forall j \in \mathcal{N}, m \in \mathcal{M}$. The backlog size of queue $Q_j^{(m)}$ at time t, *i.e.*, the number of requests generated in region j for file m but not dispatched yet by t, is denoted by $Q_j^{(m)}(t)$. The update of the request queue size is given as the following queueing law [12]:

$$Q_j^{(m)}(t+1) = \max[Q_j^{(m)}(t) - s_j^{(m)}(t) - \sum_{i \in \mathcal{N}} c_{ji}^{(m)}(t), 0] + a_j^{(m)}(t). \tag{1}$$

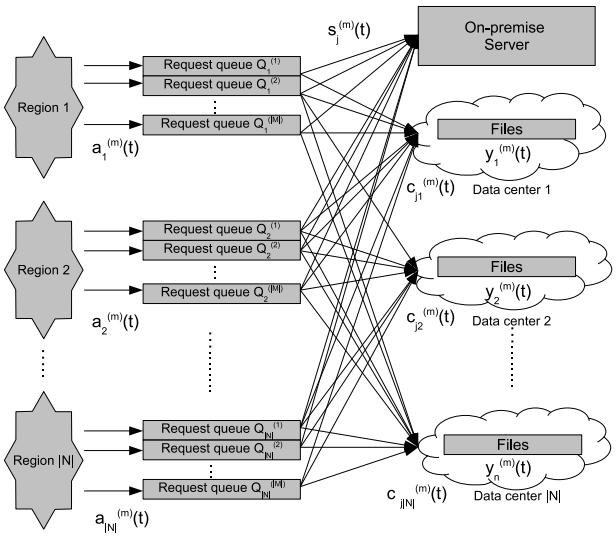

Fig. 2. The queueing model.

Fig. 2 illustrates the queueing model in our system.

Service quality. The service quality experienced by users is evaluated by request response delay, consisting of two major components: queueing delay in the request queue, and round-trip delay from when the request is dispatched from the queue to the time the first byte of the requested file is received. We ignore the processing delay inside a data center, due to the high inter-connection bandwidth and CPU capacities inside a data center. Let d_j and e_{ji} denote the round-trip delay between region j and the private cloud, and between region j and data center i, respectively. Let α be the upper-bound of the average round-trip delay per request, which the application provider wishes to enforce in this content distribution application. We reasonably assume $\alpha > e_{ii}, \forall i \in \mathcal{N}$, *i.e.*, this bound is larger than the round-trip delay between a user and the data center in the same region. We will show that our dynamic optimal service migration algorithm can bound both the average round-trip delay and queueing delay experienced by users.

Operational cost. Our algorithm focuses on minimizing recurring operational cost of the content distribution system, not one-time costs such as the purchase of machines in the private cloud and contents. The recurring costs in each time slot t include the following categories:

i) Bandwidth charge at the private cloud for uploading contents to users, at the total amount of

$\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} v^{(m)}s_j^{(m)}(t)h$.

ii) Storage cost at data center $i, \forall i \in \mathcal{N}$, for caching replicated contents, at the total amount of $\sum_{m\in\mathcal{M}} v^{(m)}y_i^{(m)}(t)p_i$.

iii) Request service cost at data center i for uploading replicated files to users. The cost for serving file m includes VM rental cost $\frac{\sum_{j\in\mathcal{N}} c_{ji}^{(m)}(t)}{r_i}\times f_i$ and upload bandwidth cost $\sum_{j\in\mathcal{N}} v^{(m)}c_{ji}^{(m)}(t)g_i$. Let $q_i^{(m)} = \frac{f_i}{r_i} + v^{(m)}g_i$ denote the unit cost to serve each request for file m on data center i. The total cost of serving requests at data center i is $\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} c_{ji}^{(m)}(t)q_i^{(m)}$.

iv) Migration cost for copying files from the private cloud to data center i. Let $w_i^{(m)}$ denote the migration cost to copy file m into data center i, which includes costs of upload and download bandwidths from the private cloud to data center i, *i.e.*, $w_i^{(m)} = v^{(m)}(h+o_i)$. The total migration cost incurred at data center i is $\sum_{m\in\mathcal{M}}[y_i^{(m)}(t) - y_i^{(m)}(t-1)]^+ w_i^{(m)}$, where notation $[x]^+ = x$ if $x \geq 0$ and $[x]^+ = 0$ if $x < 0$.

We will not consider any recurring storage cost on the private cloud (the purchase of storage disks by the application provider is a one-time investment). In addition, the removal of contents from a data center is cost-free.

Therefore, the overall operational cost to the application provider in time slot t is

$$\begin{aligned} M(t) \;=\; & \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} v^{(m)}s_j^{(m)}(t)h + \sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}} v^{(m)}y_i^{(m)}(t)p_i \\ & + \sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} c_{ji}^{(m)}(t)q_i^{(m)} \\ & + \sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}} [y_i^{(m)}(t) - y_i^{(m)}(t-1)]^+ w_i^{(m)}. \end{aligned} \quad (2)$$

Optimization formulation. The optimization pursued by our dynamic algorithm is formulated as follows, which minimizes the time-averaged operational cost while guaranteeing service quality. We use $\overline{x(t)} = \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T} x(t)$ to represent the time-averaged value of $x(t)$.

$$\min \overline{M(t)} \quad (3)$$

subject to:

$$\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} s_j^{(m)}(t) \leq b, \forall t, \quad (4)$$

$$c_{ji}^{(m)}(t) \leq \mu_{max} y_i^{(m)}(t), \forall j\in\mathcal{N}, i\in\mathcal{N}, m\in\mathcal{M}, \forall t, \quad (5)$$

$$\overline{a_j^{(m)}(t)} \leq \overline{s_j^{(m)}(t)} + \overline{\sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)}, \forall j\in\mathcal{N}, \forall m\in\mathcal{M}, \quad (6)$$

$$\overline{\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}} (s_j^{(m)}(t)d_j + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)e_{ji})} \leq \alpha\overline{\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}} (s_j^{(m)}(t) + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t))}, \quad (7)$$

$$s_j^{(m)}(t) \in Z^+\cup\{0\}, \forall j\in\mathcal{N}, m\in\mathcal{M}, \forall t, \quad (8)$$

$$c_{ji}^{(m)}(t) \in Z^+\cup\{0\}, \forall j\in\mathcal{N}, i\in\mathcal{N}, m\in\mathcal{M}, \forall t, \quad (9)$$

$$y_i^{(m)}(t) \in \{0,1\}, \forall i\in\mathcal{N}, m\in\mathcal{M}, \forall t. \quad (10)$$

Recall that each request is served by a unit bandwidth. (4) corresponds to the upload bandwidth limit at the private cloud. (5) states that requests for a file are only dispatched to data centers storing this file, and the maximum number of requests dispatched from each request queue to a data center in each time slot is no larger than μ_{max}. (6) represents each request queue is rate stable. In (7), $\Gamma_1 = \overline{\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t) + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t))}$ is the average number of overall requests in the system per unit time, and $\Gamma_2 = \overline{\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t)d_j + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)e_{ji})}$ is the total round-trip delay experienced by requests in the system per unit time. Therefore, this constraint specifies that the average round-trip delay per request, $\frac{\Gamma_2}{\Gamma_1}$, should be bounded by α.

Though queueing delay is not explicitly modeled in the constraints, we will show in Sec. 5 that our algorithm can simultaneously solve this optimization and bound the queueing delay of each request within a pre-set threshold as well.

4 Dynamic Migration Algorithm

In this section, we design a dynamic control algorithm using Lyapunov optimization techniques, which solves the optimal migration problem in (3) and bounds the time-averaged round-trip delays and queueing delays for each request. We also discuss its practical implementation.

4.1 Bounding Delays

The optimization problem in (3) includes a constraint on time-averaged variable values, *i.e.*, inequality (7). Our dynamic algorithm will only be able to adjust variables in each time slot. How can we guarantee this inequality by controlling the variable values over time?

To satisfy constraint (7), we resort to the virtual queue techniques in Lyapunov optimization [12]. We introduce a virtual queue G, with arrival rate of $\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t)d_j + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)e_{ji})$, *i.e.*, the overall round-trip delay experienced by all requests in t, and departure rate of $\alpha\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t) + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t))$, *i.e.*, the total number of requests in t multiplied by the round-trip delay bound α. G is updated as follows:

$$\begin{aligned} G(t+1) = \max[G(t) &+ \sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t)d_j + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)e_{ji}) \\ &- \alpha\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t) + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)), 0]. \end{aligned} \quad (11)$$

If queue G is stable, its time-averaged arrival rate should not exceed its time-averaged departure rate (according to Theorem 2.5 in [12]), *i.e.*, constraint (7) is satisfied. Therefore, we can adjust the request distribution strategies $s_j^{(m)}(t)$'s and $c_{ji}^{(m)}(t)$'s in each time slot to guarantee that this virtual queue is always stable, in order to satisfy constraint (7). Intuitively, when the size of G is large, *i.e.*, when a risk arises for constraint (7) to be violated,

requests should be dispatched more to the private cloud or data centers with small round-trip delay from users; when the queue size is small, requests can be distributed based more on cost considerations.

Recall that our dynamic migration algorithm also seeks to bound queueing delays in the request queues Q_j^m, $\forall j \in \mathcal{N}, m \in \mathcal{M}$. To bound the worst-case queueing delay of each request in all queues $Q_j^{(m)}, \forall j \in \mathcal{N}, m \in \mathcal{M}$, the *$\epsilon$-persistent service queue* technique [27] can be applied. This technique features carefully designing a set of virtual queues. Any scheduling algorithm keeping that set of virtual queues bounded ensures worst-case queueing delay of each request bounded. In particular, we design a set of virtual queues by associating $Q_j^{(m)}$ with a virtual queue $Z_j^{(m)}$, updated by:

$$Z_j^{(m)}(t+1) = \max[Z_j^{(m)}(t) + 1_{\{Q_j^{(m)}(t)>0\}}(\epsilon_j^{(m)} - s_j^{(m)}(t) - \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)) - 1_{\{Q_j^{(m)}(t)=0\}}\mu_{max}, 0], \quad (12)$$

where $\epsilon_j^{(m)} (0 < \epsilon_j^{(m)} < \mu_{max})$ is a constant that can be gauged to control the queueing delay bound, which further renders a tradeoff between the queueing delay bound and the cost optimality achieved by our algorithm (to be discussed in Sec. 5). The rationale behind (12) can be explained intuitively: If request queue $Q_j^{(m)}$ is not empty in time slot t (*i.e.*, $Q_j^{(m)}(t) > 0$), then a constant number of arrivals $\epsilon_j^{(m)}$ are added into virtual queue $Z_j^{(m)}$, while the departure rate from the virtual queue, $s_j^{(m)}(t) + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)$, is the same as the departure rate from request queue $Q_j^{(m)}$. If the request queue $Q_j^{(m)}$ is empty in t (*i.e.*, $Q_j^{(m)}(t) = 0$), the length of the virtual queue $Z_j^{(m)}$ decreases by μ_{max}. In the algorithm design given in Sec. 4.2, we will strategically decide $s_j^{(m)}(t)$'s and $c_{ji}^{(m)}(t)$'s to keep the virtual queue bounded. In this way, requests are expediently dispatched from the request queue, resulting in limited queueing delay per request. Detailed analysis is given in Theorem 2 in Sec. 5.

4.2 Dynamic Algorithm Design via Drift-Plus-Penalty Minimization Method

Next we design a dynamic algorithm which stabilizes all queues and solves optimization (3).

In our dynamic algorithm for the cost minimizing problem, three types of queues are needed, *i.e.*, request queues $Q_j^{(m)}$ ($\forall j \in \mathcal{N}, \forall m \in \mathcal{M}$), virtual queue G, and virtual queues $Z_j^{(m)}$ ($\forall j \in \mathcal{N}, \forall m \in \mathcal{M}$). Let $\mathbf{\Theta}(t) = [\mathbf{Q}(t), \mathbf{G}(t), \mathbf{Z}(t)]$ be the vector of all queues in the system. Define our Lyapunov function as

$$L(\mathbf{\Theta}(t)) = \frac{1}{2}[\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}(Q_j^{(m)}(t)^2 + Z_j^{(m)}(t))^2 + G(t)^2]. \quad (13)$$

The one-slot conditional Lyapunov drift is

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

Following the *drift-plus-penalty* framework in Lyapunov optimization (Chapter 5 in [12]), we can make the time-averaged operational cost $\overline{M(t)}$ within an upper bound of optimality and stabilize all queues, by minimizing an upper bound of the following item in each time slot:

$$\Delta(\mathbf{\Theta}(t)) + VM(t),$$

where V is a non-negative parameter set by the application provider to control the tradeoff between the operational cost and request response delays. The rationale is as follows: if in every time slot greedily minimizing *Lyapunov drift* $\Delta(\Theta(t))$, backlogs are consistently pushed towards smaller, which intuitively maintains queues stable. Adding $VM(t)$ (a weighted term of incurred operational cost at time slot t) onto $\Delta(\Theta(t))$ allows a tradeoff between backlog reduction and operational cost minimization at time slot t. Although we do not minimize $\Delta(\Theta(t)) + VM(t)$ directly, minimizing one of its upper bound can have similar effects. In the long term with this carefully designed local optimizing objective we stabilize all queues and make the time-averaged operational cost $\overline{M(t)}$ within an upper bound of optimality.

Now we derive an upper bound of $\Delta(\mathbf{\Theta}(t)) + VM(t)$ by squaring the queueing laws (1), (11) and (12) as follows (the details are in Appendix D):

$$\begin{aligned}
&\Delta(\mathbf{\Theta}(t)) + VM(t)\\
&= B - \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} s_j^{(m)}(t)[Q_j^{(m)}(t) + (\alpha - d_j)G(t)\\
&\quad + 1_{\{Q_j^{(m)}(t)>0\}}Z_j^{(m)}(t) - v^{(m)}Vh] - \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)\\
&\quad [Q_j^{(m)}(t) + (\alpha - e_{ji})G(t) + 1_{\{Q_j^{(m)}(t)>0\}}Z_j^{(m)}(t) - Vq_i^{(m)}]\\
&\quad + V\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{N}}[v^{(m)}y_i^{(m)}(t)p_i + [y_i^{(m)}(t) - y_i^{(m)}(t-1)]^+ w_i^{(m)}]\\
&\quad + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} Z_j^{(m)}(t)[1_{\{Q_j^{(m)}(t)>0\}}\epsilon_j^{(m)} - 1_{\{Q_j^{(m)}=0\}}\mu_{max}]\\
&\quad + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} Q_j^{(m)}(t)a_j^{(m)}(t),
\end{aligned} \quad (14)$$

where $B = \frac{1}{2}|\mathcal{M}||\mathcal{N}|[A_{max}^2 + \epsilon_{max}^2 + 2(b + N\mu_{max})^2] + +\frac{1}{2}(|\mathcal{M}||\mathcal{N}|^2\mu_{max}e_{max} + bd_{max})^2 + \frac{1}{2}\alpha^2(|\mathcal{M}||\mathcal{N}|^2\mu_{max} + b)^2$ is a constant, with $d_{max} = \max\{d_j | j \in \mathcal{N}\}$, $e_{max} = \max\{e_{ji} | j \in \mathcal{N}, i \in \mathcal{N}\}$, and $\epsilon_{max} = \max\{\epsilon_j^{(m)} | j \in \mathcal{N}, m \in \mathcal{M}\}$. The impact of constant B will be shown in Theorem 3.

In the following we design an algorithm that minimizes the the right-hand-side of inequality (14), and will discuss queue stability and cost optimality in Sec. 5. To minimize the right-hand-side of inequality (14), the algorithm observes the queues $Q_j^{(m)}(t)$, $G(t)$ and $Z_j^{(m)}(t)$ in each time slot t, and decides optimal values of $s_j^{(m)}(t)$ and $c_{ji}^{(m)}(t)$, $\forall j \in \mathcal{N}, i \in \mathcal{N}, m \in \mathcal{M}$.

We simplify the notation by defining

$$\gamma_j^{(m)}(t) = Q_j^{(m)}(t) + 1_{\{Q_j^{(m)}(t)>0\}}Z_j^{(m)}(t) - Vv^{(m)}h + (\alpha - d_j)G(t),$$

which is a constant in time slot t, and

$$\eta_{ji}^{(m)}(t) = Q_j^{(m)}(t) + 1_{\{Q_j^{(m)}(t)>0\}}Z_j^{(m)}(t) - Vq_i^{(m)} + (\alpha - e_{ji})G(t),$$

which is also a constant in t, and

$$\phi_i^{(m)}(t) = V(v^{(m)}p_i + 1_{\{y_i^{(m)}(t-1)=0\}}w_i^{(m)}),$$

which is a constant in t as well, when $y_i^{(m)}(t-1)$ is given.

Therefore, minimizing the right-hand-side of (14) is equivalent to:

$$\max \quad F(t) = \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} s_j^{(m)}(t)\gamma_j^{(m)}(t) + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)\eta_{ji}^{(m)}(t) - \sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{N}}\phi_i^{(m)}(t)y_i^{(m)}(t) \quad (15)$$

$$\text{subject to: constraints (4) (5) (8) (9) (10).}$$

It is equivalent to solve the following problems separately:

$$\max \quad \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} s_j^{(m)}(t)\gamma_j^{(m)}(t) \quad (16)$$
$$s.t.: \quad \text{constraint } (8)$$

and, for each $m \in \mathcal{M}$ and each $i \in \mathcal{N}$,

$$\max \quad \sum_{j\in\mathcal{N}} c_{ji}^{(m)}(t)\eta_{ji}^{(m)}(t) - \phi_i^{(m)}(t)y_i^{(m)} \quad (17)$$
$$s.t.: \quad c_{ji}^{(m)}(t) \le \mu_{max}y_i^{(m)}(t), \forall j \in \mathcal{N}$$
$$\text{constraints } (9)(10)$$

The solution of (16) is:

$$s_j^{(m)*}(t) = \begin{cases} b & \text{if } (m,j) = \arg\max_{(m'\in\mathcal{M},j'\in\mathcal{N})}\{\phi_{j'}^{(m')}(t)\} \\ & \quad \text{and } \gamma_j^{(m)}(t) \ge 0 \\ 0 & \text{otherwise} \end{cases}$$

(17) can be solved by choosing the larger solution value from the following two cases:

- *Case 1*: $y_i^{(m)}(t) = 0$. ${c_{ji}^{(m)}}^0 = 0, \forall j \in \mathcal{N}$. The optimal value of the objective function (17) is 0.
- *Case 2*: $y_i^{(m)}(t) = 1$. (17) is equivalent to:

$$\max \quad \sum_{j\in\mathcal{N}} c_{ji}^{(m)}(t)\eta_{ji}^{(m)}(t) - \phi_i^{(m)} \quad (18)$$
$$s.t.: \quad c_{ji}^{(m)}(t) \le \mu_{max}, \forall j \in \mathcal{N}$$
$$c \in Z^+ \cup \{0\}.$$

The optimal solution of (18) is

$${c_{ji}^{(m)}}^1(t) = \begin{cases} \mu_{max} & \text{if } \eta_{ji}^{(m)}(t) \ge 0 \\ 0 & \text{otherwise} \end{cases}$$

Now we compare the optimal values of the objective function (17) in Case 1 and Case 2: If $\sum_{j\in\mathcal{N}} {c_{ji}^{(m)}}^1(t)\eta_{ji}^{(m)}(t) - \phi_i^{(m)} > 0$, the optimal solution is $y_i^{(m)} = 0$ and ${c_{ji}^{(m)}}^* = {c_{ji}^{(m)}}^0, \forall j \in \mathcal{N}$. Otherwise, the optimal solution is $y_i^{(m)} = 1$ and ${c_{ji}^{(m)}}^* = {c_{ji}^{(m)}}^1, \forall j \in \mathcal{N}$.

The solution means that for each data center, it will choose migrating the file and uploading the file to users if the corresponding queues is long enough ($\eta_{ji}^{(m)}(t) \ge 0$ implies that the request queue and associated virtual queue is long) and the related weight surpasses the cost of migrating the file and/or keeping the file in storage ($\sum_{j\in\mathcal{N}} c_{ji}^{(m)}(t)\eta_{ji}^{(m)}(t) - \phi_i^{(m)} > 0$).

4.3 Discussions on Practical Implementation

Our dynamic algorithm is to be deployed by the application provider to optimally distribute its content distribution service onto the hybrid cloud. The application provider deploys one or multiple web servers providing portal service of the content distribution application, in a centralized or distributed fashion. The portal aggregates user requests and sends the collected request information to a *control center*, which executes our algorithm periodically.

The control center maintains a content placement table with entries $y_i^{(m)}, \forall j \in \mathcal{N}, m \in \mathcal{M}$, indicating whether file m is currently replicated on data center i. The entries are initialized to be 0 at the system initialization stage. In each time slot, received requests for file m originated from region j are placed in request queue $Q_j^{(m)}$. Virtual queues $Z_j^{(m)}$ and G are maintained simply as counters. The control center observes lengths of the queues and request arrival rates, and calculates the optimal content placement and load distribution strategies by solving (15).

Based on the derived content placement strategies, it updates the placement table, and compares the optimal solution $y_i^{(m)}(t)$ against the current value of $y_i^{(m)}(t-1)$ in the table, $\forall j \in \mathcal{N}, m \in \mathcal{M}$: If $y_i^{(m)}(t-1) = 0$ and $y_i^{(m)}(t) = 1$, the control center instructs data center i to request a copy of file m from the private cloud; if $y_i^{(m)}(t-1) = 1$ and $y_i^{(m)}(t) = 0$, it signals data center i to remove file m from its storage.

Based on the request distribution decisions, the control center dispatches $s_j^{(m)}(t)$ requests from queue $Q_j^{(m)}$ to the private cloud, and $c_{ji}^{(m)}(t)$ requests from the queue $Q_j^{(m)}$ to data center i, $\forall j, i \in \mathcal{N}, m \in \mathcal{M}$. Virtual queue $Z_j^{(m)}$ and G are updated accordingly.

The sketch of our complete dynamic, joint content placement and load distribution algorithm is presented in Algorithm 1.

As an engineering parameter, the length of intervals between two executions of the algorithm can be set by the application provider, based on update frequencies of contents, sizes of the files, as well as its targeted performance optimality.

5 Performance Analysis

We next analyze the performance guarantee provided by our dynamic algorithm, with respect to bounded queueing delay and optimality in cost minimization.

5.1 Bound of Queueing Delay

Theorem 1: (Bound of Queue Length) Define

$$Q_j^{(m)max} = V(v^{(m)}p_{\tilde{i}} + w_{\tilde{i}}^{(m)} + q_{\tilde{i}}^{(m)}) + A_{max}, \quad (19)$$
$$Z_j^{(m)max} = V(v^{(m)}p_{\tilde{i}} + w_{\tilde{i}}^{(m)} + q_{\tilde{i}}^{(m)}) + \epsilon_j^m, \quad (20)$$

where

Algorithm 1: Control Algorithm on the Control Center

Initialization:

Set up request queue $Q_j^{(m)}$, virtual queues G and $Z_j^{(m)}, \forall j \in \mathcal{N}, m \in \mathcal{M}$, and initialize their backlogs to 0;

In every time slot t:

1. Enqueue received requests to request queues ($Q_j^{(m)}$'s);
2. Solve optimization (15) to obtain optimal content placement and load distribution strategies $c_{ji}^{(m)}(t)$, $s_j^{(m)}(t)$, $y_i^{(m)}(t)$, $\forall j, i \in \mathcal{N}, m \in \mathcal{M}$;
3. Update content placement table with $y_i^{(m)}(t)$'s, and migrate files as follows:

for $i \in \mathcal{N}, m \in \mathcal{M}$ **do**

  if $y_j^{(m)}(t-1) = 0$ *and* $y_j^{(m)}(t) = 1$ **then**

    instruct data center i to request file m from private cloud;

  if $y_j^{(m)}(t-1) = 1$ *and* $y_j^{(m)}(t) = 0$ **then**

    signal data center i to remove file m;

4. Dispatch $s_j^{(m)}(t)$ requests from queue $Q_j^{(m)}$ to private cloud, $c_{ji}^{(m)}(t)$ requests to data center i, $\forall j, i \in \mathcal{N}, m \in \mathcal{M}$;
5. Update virtual queue $Z_{ji}^{(m)}$ and G according to Eqn. (12) and (11);

$$\tilde{i} = \text{argmin}_i \{v^{(m)} p_i + w_i^{(m)} + q_i^{(m)} | \alpha - e_{ji} > 0, \forall i \in \mathcal{N}\}. \quad (21)$$

Then we have $Q_j^{(m)}(t) \leq {Q_j^{(m)}}^{max}$ and $Z_j^{(m)}(t) \leq {Z_j^{(m)}}^{max}$, $\forall j \in \mathcal{N}, \forall m \in \mathcal{M}$, *i.e.*, ${Q_j^{(m)}}^{max}$ and ${Z_j^{(m)}}^{max}$ is the maximum size of queue $Q_j^{(m)}$ and $Z_j^{(m)}$ at any time t respectively.

The proof is given in Appendix A. Based on Theorem 1, we have

Theorem 2: (Bounded Queueing Delay): For each request queue $Q_j^{(m)}$, $\forall j \in \mathcal{N}, \forall m \in \mathcal{M}$, define

$$W_j^{(m)} = \lceil \frac{{Q_j^{(m)}}^{max} + {Z_j^{(m)}}^{max}}{\epsilon_j^{(m)}} \rceil$$

where $\tilde{i}$ is defined as in (21). The queueing delay of each request in $Q_j^{(m)}$ is bounded by $W_j^{(m)}$.

The proof is given in Appendix B.

5.2 Optimality against the T-Slot Lookahead Mechanism

Since request arrival rates are arbitrary in our system, it is difficult to find the global cost optimum, with which to compare the time-averaged cost $\overline{M(t)}$ achieved by our algorithm. Therefore we utilize a local optimum target, which is the optimal (objective function) value of a similar cost minimization problem within known information (*e.g.*, request arrivals) for T time slots into the future, *i.e.*, a *T-slot lookahead* mechanism [12]. We will show that the optimal value obtained by our algorithm is close to that of the T-slot lookahead mechanism, even if our algorithm does not assume any future information.

In the T-slot lookahead mechanism, time is divided into successive frames, each consisting of T time slots. Denote each frame as $F_k = \{kT+1, kT+2, \ldots, kT+T\}$, where $k = 0, 1, \ldots$. In each time frame, consider the following optimization problem on variables $c_{ji}^{(m)}(t), s_j^{(m)}(t), y_i^{(m)}(t)$, $\forall j \in \mathcal{N}, i \in \mathcal{N}, m \in \mathcal{M}$, $t \in F_k$:

$$\min \frac{1}{T} \sum_{t=kT+1}^{kT+T} M(t) \quad (22)$$

subject to:

$$\sum_{m \in \mathcal{M}} \sum_{j \in \mathcal{N}} s_j^{(m)}(t) \leq b, \forall t \in F_k, \quad (23)$$

$$c_{ji}^{(m)}(t) \leq \mu_{max} y_i^{(m)}(t), \quad \forall j \in \mathcal{N}, \forall i \in \mathcal{N}, m \in \mathcal{M}, t \in F_k, \quad (24)$$

$$\sum_{t=kT+1}^{kT+T} [a_j^{(m)}(t) - s_j^{(m)}(t) - \sum_{i \in \mathcal{N}} c_{ji}^{(m)}(t)] \leq 0, \quad \forall j \in \mathcal{N}, m \in \mathcal{M}, \quad (25)$$

$$\frac{1}{T} \sum_{t=kT+1}^{kT+T} [s_j^{(m)}(t) + \sum_{i \in \mathcal{N}} c_{ji}^{(m)}(t)] \geq \epsilon_j^{(m)}, \quad \forall j \in \mathcal{N}, m \in \mathcal{M} \quad (26)$$

$$\sum_{t=kT+1}^{kT+T} \sum_{j \in \mathcal{N}} \sum_{m \in \mathcal{M}} (\sum_{i \in \mathcal{N}} c_{ji}^{(m)}(t) e_{ji} + s_j^{(m)}(t) d_j) \leq \alpha \sum_{t=kT+1}^{kT+T} \sum_{j \in \mathcal{N}} \sum_{m \in \mathcal{M}} (\sum_{i \in \mathcal{N}} c_{ji}^{(m)}(t) + s_j^{(m)}(t)), \quad (27)$$

$$s_j^{(m)}(t) \in Z^+ \{0\}, \forall j \in \mathcal{N}, m \in \mathcal{M}, t \in F_k, \quad (28)$$

$$c_{ji}^{(m)}(t) \in Z^+ \{0\}, \forall j \in \mathcal{N}, i \in \mathcal{N}, m \in \mathcal{M}, t \in F_k \quad (29)$$

$$y_i^{(m)}(t) \in \{0, 1\}, \forall i \in \mathcal{N}, m \in \mathcal{M}, t \in F_k. \quad (30)$$

Assuming full knowledge of request arrivals in the T time slots in F_k, this optimization derives the optimal content placement and load distribution decisions in each of the T time slots, which minimize the average cost per time slot in the objective function. We show the time-averaged cost $\overline{M(t)}$ achieved by our algorithm is within a constant gap $\frac{BT}{V}$ from that achieved by solving the above optimization:

Theorem 3: (Optimality of Cost) Let $\widehat{M_k}$ denote the optimal objective function value in the T-slot Lookahead problem (22) in time frame F_k. The minimum operational cost derived with our algorithm is $M(t)$ in time slot t. Suppose the time lasts for KT time slots, where K is a constant. We have

$$\frac{1}{KT} \sum_{t=0}^{KT-1} M(t) \leq \frac{1}{K} \sum_{k=0}^{K-1} \widehat{M_k} + \frac{BT}{V}, \quad (31)$$

i.e., our algorithm achieves a time-averaged cost within constant gap $\frac{BT}{V}$ from that by assuming full knowledge in T time slots in the future.

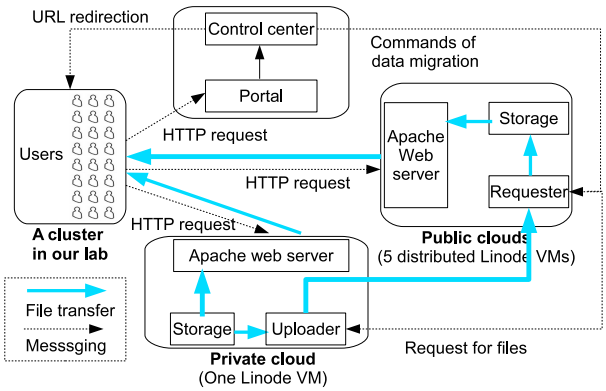

Fig. 3. The key modules of our prototype.

Theorem 3 is proved in details in Appendix C.

Theorems 2 and 3 show that when V increases, worst-case queueing delay $W_j^{(m)}$ increases, while the gap between the operational cost of our algorithm and that of the T-Slot lookahead mechanism is reduced. $\epsilon_j^{(m)}$ has a similar effect: when $\epsilon_j^{(m)}$ increases, the worst-case queueing delay $W_j^{(m)}$ decreases, and B increases such that the gap to optimality increases. We will investigate proper values to assign to these tradeoff control parameters in our evaluation in Sec. 6.

6 Performance Evaluation

6.1 Experiment Setup

We evaluate the performance of the dynamic algorithm with a prototype deployed on six VMs that are located in six data centers (in the cities of Dallas, Fremont, Atlanta, Newark, London, Tokyo) of Linode Cloud [28] and a cluster residing in our lab in the University of Hong Kong. We deploy the web portal and the control center on one Linode VM and use it to emulate the private cloud at the same time, while we use the remaining five Linode VMs to emulate five data centers of the public cloud. When instructed by the control center, the Requester module at the public cloud is responsible for requesting a copy of a file from the Uploader module at the private cloud. The communication among the control center, the private cloud and the public cloud is via our customized protocol encapsulated in TCP. We emulate the users of the content distribution service using our lab cluster. Each user is emulated by an independent thread, which communicates with the portal, the public cloud and the private cloud via HTTP protocol. We implement a web-based file download application. Users issue requests in the form of HTTP requests for the files. A user will be responded with a URL redirection command pointing to the data center from which the user can download a copy of the requested file. The key modules of the prototype are illustrated in Fig. 3.

6.1.1 Workload

The distribution of the file sizes follows the measurement result of YouTube videos in [29], with a mean size of 7.6 MB and an upper bound of 25 MB (because 99.1% of YouTube videos are less than 25 MB in size). The total number of files is 1000. The total number of requests for all files follows a Poisson distribution over time, with a mean of 0.15 request per file per time slot. All the concurrent file uploadings share the dynamical overall bandwidth of a VM, which is generally larger than $150Mbps$. The number of files and the mean number of requests for a file are so set due to the bandwidth capacity of Linode VMs available to us. Especially, the scale of experiment is approximately restricted by the following:

$$\frac{\text{Mean file size}\times\text{Mean \# of requests per file per time slot}\times\text{\# of files}}{\text{\# of seconds per time slot}} \leq \text{Bandwidth per VM} \times \text{\# of VMs}$$

Nevertheless, we believe that the system can scale up smoothly without degraded performance when the system is deployed on real elastic clouds. We briefly discuss how the system will behave when the system is scaled up: When the average file size becomes larger, the application provider adjusts the system parameters b (maximum number of requests that the private cloud can serve in a time slot) and r_i (number of requests a VM in data center i can serve in a time slot), or use VMs that have higher bandwidth capacity, to make sure each file can be uploaded with a time slot. When the workload is more intensive, more VMs will be rented from the public cloud to serve more requests. Since our control center is implemented in a distributed fashion, it would not become the bottleneck of the system. We will emperically study the case when the number of files increases in Sec. 6.4.

We split the total number of requests onto different files in different regions, in each time slot. The relative access frequency of the files follows a Zipf-like distribution [30] with parameter 0.8. We emulate the geographical diversity of requests by splitting the requests among regions following a binomial distribution. We assign the probability that a request is dispatched to region i ($i = 0, 1, 2, 3, 4, 5$) to be the probability of obtaining exactly i successes out of 6 Bernoulli trials. For distributing the requests for each file, we assign each of the 6 regions an index by arranging them in a random order. The maximum number of requests arising from each region for each file in one time slot, A_{max}, is 2, and the maximum number of requests dispatched from a queue to a data center per time slot, μ_{max}, is 4. The impact of the setting of A_{max} and μ_{max} is negligible as long as they are significantly larger than the mean number of requests for a file in each time slot (0.15 as set above).

6.1.2 Cost

We emulate a charging mechanism in the prototype as follows, instead of relying on native charging method of Linode Cloud (Linode allows us to transfer $4TB$ of data per month for free based on our two-year rental contract). All charges below are in the units of US dollars.

The cost of uploading data from the private cloud is $\$1 \times 10^{-10}$ per byte, which is the average price for Internet access provided by typical hosting service providers [31][32].

The charges by the public cloud are extracted from real settings of Amazon EC2 and S3 [25][26]. We estimate VM rental cost per file according to the following: VM rental cost per file $= \frac{\text{Rental cost of a VM instance}}{\text{Upload bandwidth per VM/Mean size of a file}}$. Since the rental cost of individual VM instances is not available in the Linode charging model, we set it according to comparable charges in Amazon EC2, as the rental cost of a typical Amazon EC2 VM with upload bandwidth of 250 Mbps [33], which is \$0.7 per hour. The charge of storing a byte on a data center is in the range $[\$4.6\times10^{-16}, \$1.0\times10^{-15}]$. The cost of uploading from a data center in the public cloud is randomly selected from the set $\{\$0.5 \times 10^{-10}, \$0.7 \times 10^{-10}, \$0.9 \times 10^{-10}, \$1.2 \times 10^{-10}\}$ (per byte), which are prices of the data upload service offered by Amazon EC2 for different scales of purchases [25]. According to the current cloud business model [26], there is no charge for downloading data into the cloud, i.e., $o_i = 0, \forall i \in \mathcal{N}$.

6.1.3 Delays

In the control center we set the round-trip delay between users in a region and a data center in another region as the real latency we obtained by pinging the respective Linode VMs. We set the round-trip delay between a user and the data center in the same region e_{jj} to be 5 ms for all regions j. The average round-trip delay bound set by the application provider, *i.e.*, α, is 200 ms, since a RTT more than 200ms will bring users poor experience [34]. $\epsilon_j^{(m)}$'s are set proportional to $Q_j^{(m)^{max}} + Z_j^{(m)^{max}}$ to make the queueing delay bounds $W_j^{(m)}$'s the same for each request. By default the target $W_j^{(m)}$'s are 20 and the impact of its other values will be evaluated in Sec. 6.3.2.

6.1.4 Other Parameters

The duration of a time slot is 10 seconds. The duration of a time slot is set based on the following practical considerations: On one hand, running the optimization solver too frequently is too costly, and since file migration is involved, it is unlikely to be done in a time scale smaller than a few seconds; on the other hand, the duration of a time slot should not be too long, as otherwise queueing delays experienced by requests tend to be too long. After some trials, we find 10-seconds is an appropriate value. The default value of V is 100000, and its impact on the system performance will be evaluated in Sec. 6.3.1.

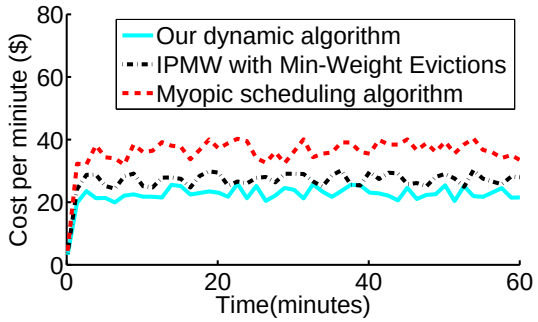

Fig. 4. Cost comparison among our dynamic algorithm, IPMW with Min-Weight Evictions, and the Myopic Scheduling Algorithm (average costs are 22.5, 26.8, 35.6 dollars respectively).

Initially, the files are not deployed in the data centers, *i.e.*, $y_i^{(m)}(0) = 0, \forall i \in \mathcal{N}, \forall m \in \mathcal{M}$.

Under each configuration, we run the prototype system for 60 minutes, for multiple times. The data presented in Sec. 6.2 and Sec. 6.3, which show temporal dynamics of the system, are collected in a representative run of the experiment, because we observe that the results of multiple runs reveal the same pattern, while the data presented in Sec. 6.4 are the average of the results in 10 runs of the experiments.

6.2 Cost Optimality

6.2.1 Comparison with Existing Algorithms

We first compare our dynamic algorithms against a simple Myopic Scheduling Algorithm and another existing algorithm for content placement and request routing for traditional CDN [14], named *Iterative Periodic Max-Weight Scheduling with Min-Weight Evictions* (abbreviated as *IPMW with MWE*).

The Myopic Scheduling Algorithm processes all requests in the time slot when they arrive without buffering them in queues, and decides content replication and request distribution by minimizing overall operational cost (3) under constraints (4)(5)(7)(8)(9)(10) by changing all time-average expressions to that of the current single time slot.

Similar to our work, IPMW with MWE Algorithm models the system of frontend source nodes and backend cache (*backend cache in [14] is a synonym for space of the storage server in this paper*) of the CDN as the input and output nodes of a switch. It builds queues for requests for different files at the source nodes, makes decisions on request routing in every time slot, and refreshes contents of backend cache periodically. But different from our algorithm, the size of the cache is static, which renders a trade-off between the storage cost and queueing delays. To make the comparison fair, we do a binary search for the optimal size of its backend cache which leads to the smallest cost, under the constraint that the queueing delays of more than 90% of requests are within the

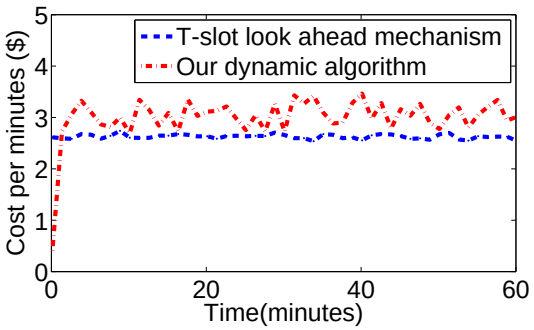

Fig. 5. Cost comparison among our dynamic algorithm and the T-slot lookahead mechanism ($|\mathcal{M}| = 100$) (average costs are 2.6 and 3.0 dollars respectively).

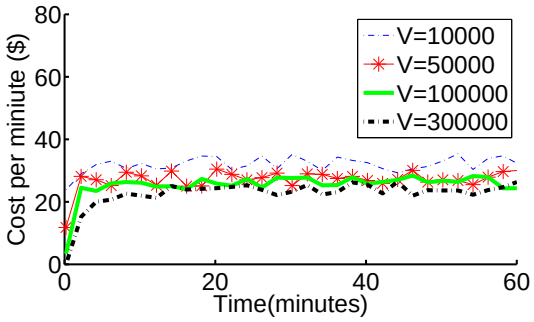

Fig. 6. Cost with different V (from $V = 10000$ to 300000, average costs are 31.6, 27.4, 25.3, 23.2 dollars respectively).

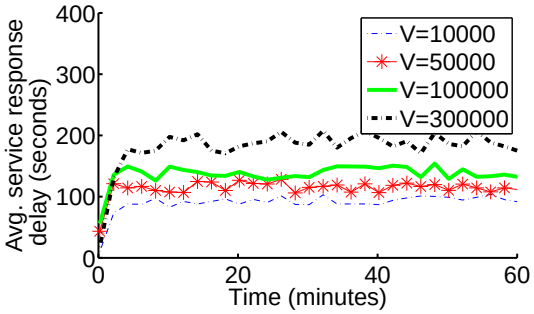

Fig. 7. Average service response delay with different V (from $V = 10000$ to 300000, average delays are 91.5, 116.6, 137.9, 183.6 seconds respectively).

specific target. Another parameter that needs to be set for IPMW with MWE is the periodicity when the content of the backend cache is allowed to be refreshed. To compare fairly, we set the periodicity to be 1, the same as what is allowed in running our dynamic algorithm.

Fig. 4 shows the overall cost incurred at each time slot when each method is applied. We observe that the cost incurred by our dynamic algorithms is lower than that by the Myopic Scheduling Algorithm and IPMW with MWE at all times. Our dynamic algorithm outperforms IPMW with MWE not only in terms of cost reduction, but also in that our dynamic algorithm can guarantee the queueing delays of 100% of requests are within the specific QoS while under the same setup IPMW with MWE can only guarantee 90%. This is because (1) our algorithm is aware of worst-case queueing deadlines while IPMW with MWE is not; (2) our dynamic algorithm flexibly occupies caches of suitable sizes on the fly due to its deployment in an elastic cloud, while IPMW with MWE occupies caches of fixed sizes, which tends to incur more cost; (3) our dynamic algorithm aims to strike a good trade-off between delays and cost, while IPMW with MWE only shortens the queue lengths in the best effort fashion.

6.2.2 Comparison with T-slot Lookahead Mechanism

We next compare our algorithm against the T-slot lookahead mechanism. In order to solve (22) which contains non-linear items $[y_i^{(m)}(t) - y_i^{(m)}(t-1)]^+$, we convert the problem (22) to an equivalent problem as follows:

$$\min \ \frac{1}{T} \sum_{t=kT+1}^{kT+T} [\sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{N}} v^{(m)} s_j^{(m)}(t) h + \sum_{i\in\mathcal{N}} \sum_{m\in\mathcal{M}} v^{(m)} y_i^{(m)}(t) p_i + \sum_{i\in\mathcal{N}} \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{N}} c_{ji}^{(m)}(t) q_i^{(m)} + \sum_{i\in\mathcal{N}} \sum_{m\in\mathcal{M}} \Delta y_i^{(m)}(t) w_i^{(m)}] \quad (32)$$

$$\begin{aligned} \text{subject to:} \quad & (23)(24)(25)(26)(27)(28)(29)(30) \text{ and,} \\ & \text{for } \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t, \\ & \Delta y_i^{(m)}(t) \geq y_i^{(m)}(t) - y_i^{(m)}(t-1) \\ & \Delta y_i^{(m)}(t) \geq 0 \end{aligned}$$

To derive the optimal solution, we use an open source tool *MOSEK* [35]. We find that it is very time consuming to solve the T-slot lookahead problem in (32) using the MOSEK solver in our default system scale, even when we are merely optimizing the decisions in $T = 6$ time slots. Therefore, in this set of experiments, we reduce the total number of files to 100.

Fig. 5 shows that the gap between the costs achieved by our algorithm and those by the T-slot lookahead mechanism with known future information, is close.

6.3 Impact of Algorithm Parameters

We next study the impact of important parameters in our algorithm, on the tradeoff between cost optimality and service response delays.

6.3.1 Impact of V

Fig. 6 shows that when V increases, the overall operational cost becomes smaller. Fig. 7 reveals that the average service response delay per request (queueing delay+round-trip delay) increases with the increase of V, while Fig. 8 shows the increase of the average of maximum request queue lengths (*i.e.*, the average of the maximum lengths that the request queues have ever reached until each time slot) with the increase of V as well. These figures clearly show a tradeoff in V's setting. Selecting $V = [50000, 100000]$ can achieve a good tradeoff between cost optimality and service quality.

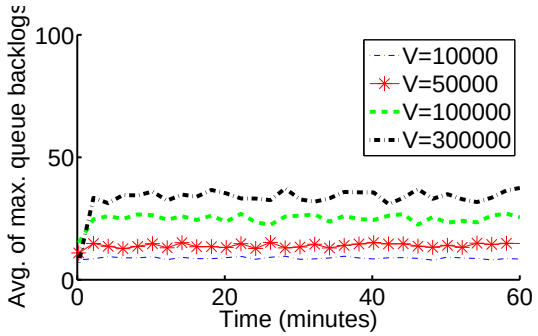

Fig. 8. Average of maximum queue lengths with different V (from $V = 10000$ to 300000, average lengths are 9.0, 13.9, 24.6, 34.1 respectively).

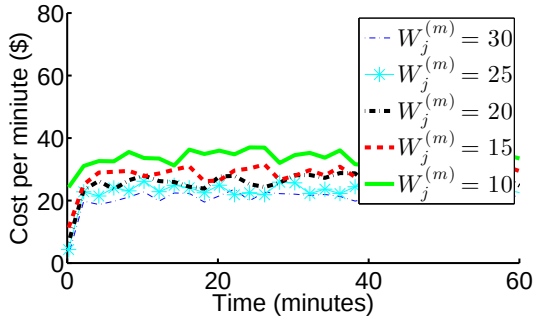

Fig. 9. Cost with different $W_j^{(m)}$'s (from $W_j^{(m)} = 30$ to 10, average costs are 21.3, 23.7, 26.1, 28.6, 33.5 dollars respectively).

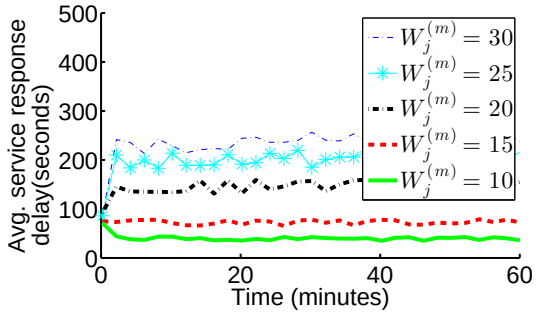

Fig. 10. Average service response delay with different $W_j^{(m)}$'s (from $W_j^{(m)} = 30$ to 10, average delays are 231.4, 201.5, 143.9, 72,9, 39.6 seconds respectively).

6.3.2 *Impact of $\epsilon_j^{(m)}$*

The parameters $\epsilon_j^{(m)}$ are controlled by preset target queueing delay bound $W_j^{(m)}$'s. In Fig. 9 and Fig. 10, we observe that when $W_j^{(m)}$ decreases, the overall operational cost increases while the average service response delay per request decreases. This shows that the value of $W_j^{(m)}$ also renders a tradeoff between cost optimality and service quality. When $W_j^{(m)}$ is smaller than 20, the cost increases significantly. Therefore selecting $\bar{\epsilon}$ around 20 would be a good choice.

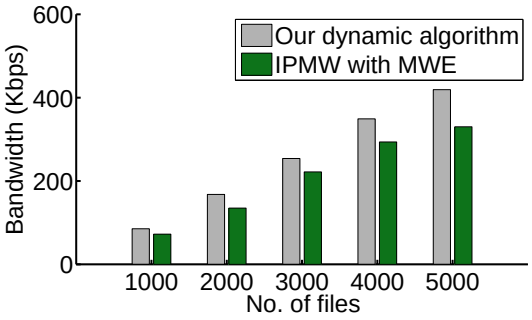

Fig. 11. Time-averaged control messaging bandwidth.

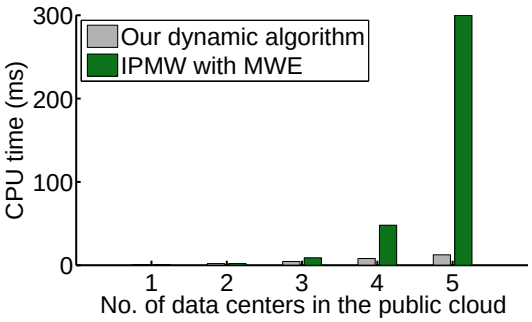

Fig. 12. Average computation time consumed by the control center.

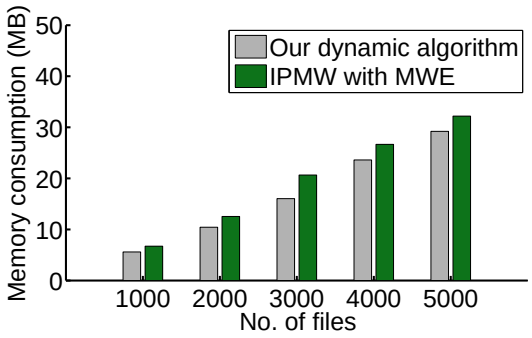

Fig. 13. Average memory consumption by the control center.

6.4 Overhead Comparison

Now we compare the overhead between our dynamic algorithm and IPMW with MWE, in terms of control messaging bandwidth, computation time, and memory consumption respectively. In this section, in order to see how the overhead grows when the number of files grows, we proportionally shrink down the size of each file, in order to accommodate more files with our bandwidth-limited Linode VMs.

6.4.1 *Control Messaging Bandwidth*

Fig. 11 shows that IPMW with MWE consumes less bandwidth for control messaging than our dynamic algorithm, and the overhead of both grows linearly with the number of files. We analyze the reason by decomposing the control messages into the following

categories: (1) HTTP requests from users to the portal, (2) URL redirection, (3) commands for file replication, and (4) requests for files from the public cloud to the private cloud.

In terms of (1) and (2), both algorithms generate similar numbers of messages. In terms of (3) and (4), our algorithm generates more messages, because our algorithm makes use of the elasticity of clouds to flexibly migrate files so as to minimize the operational cost. However, we notice that increased control messaging overhead only constitutes less than 0.05% of bandwidth consumption of the whole system, which is negligible.

6.4.2 Computation Time

Fig. 12 plots the CPU time consumed when the control center runs the respective scheduling algorithm once in each time slot. It shows that when the number of data centers increases, the CPU time by both algorithms grows, but our algorithm consumes less CPU time than IPMW with MWE. This is because the computation time of our dynamic algorithm is $O(|N|^2|M|)$ while the computation time of IPMW with MWE Algorithm is $O(|N|!|N||M|)$.

6.4.3 Memory Consumption

Fig. 13 illustrates the memory consumption of the control center. It shows that the memory consumed by both algorithms grows linearly with the number of files, and our algorithm consumes slightly less memory than IPMW with MWE. Memory consumption mainly consists of space for storing temporary results for the scheduling decisions, and space for the queues. Both algorithms have queues to buffer the requests. Although our dynamic algorithm has more queues (*except request queues, it has virtual queues*), each virtual queue is only represented by a single real number, with negligible memory consumption. Our algorithm incurs smaller average queue backlogs, and hence smaller overall memory consumption.

7 Conclusion

This paper investigates optimal migration of a content distribution service to a hybrid cloud consisting of a private cloud and public geo-distributed cloud services. We propose a generic optimization framework based on Lyapunov optimization theory, and design a dynamic, joint content placement and request distribution algorithm, which minimizes the operational cost of the application with QoS guarantees. We theoretically show that our algorithm approaches the optimality achieved by a mechanism with known information in the future T time slots by a small gap, no matter what the request arrival pattern is. Our prototype-based evaluation verifies our theoretical findings. We intend to extend the framework to specific content distribution services with detailed requirements, such as video-on-demand services or social media applications, in our ongoing work.

Acknowledgment

The research was supported in part by a grant from Hong Kong RGC under the contract HKU 717812E.

Appendix A
Proof of Theorem 1

A.1 Proving $Q_j^{(m)}(t) \leq {Q_j^{(m)}}^{max}, \forall j \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t$

Proof: We prove it by induction.

Induction Basis: According to the assumption of our model, we have $Q_j^{(m)}(0) = 0 \leq {Q_j^{(m)}}^{max}, \forall j \in \mathcal{N}, \forall m \in \mathcal{M}$.

Induction steps: We assume that $Q_j^{(m)}(t) \leq {Q_j^{(m)}}^{max}$ and then we show $Q_j^{(m)}(t+1) \leq {Q_j^{(m)}}^{max}$.

If $Q_j^{(m)}(t) \leq V(v^{(m)}p_{\tilde{i}} + w_{\tilde{i}}^{(m)} + q_{\tilde{i}}^{(m)})$, then $Q_j^{(m)}(t+1) \leq V(v^{(m)}p_{\tilde{i}} + w_{\tilde{i}}^{(m)} + q_{\tilde{i}}^{(m)}) + A_{max} = {Q_j^{(m)}}^{max}$.

If $Q_j^{(m)}(t) > V(v^{(m)}p_{\tilde{i}} + w_{\tilde{i}}^{(m)} + q_{\tilde{i}}^{(m)})$, then according the definition of $\tilde{i}$, we have $\alpha - e_{j\tilde{i}} > 0$. Therefore $\eta_{j\tilde{i}}^{(m)}(t) > \phi_{\tilde{i}}^{(m)}(t)$. According the solution to 17), we have $y_{\tilde{i}}^{(m)}(t) = 1$ and $c_{j\tilde{i}}^{(m)}(t) = \mu_{max}$.

$$
\begin{aligned}
Q_j^{(m)}(t+1) &= \max[Q_j^{(m)}(t) - s_j^{(m)}(t) - \sum_{i \in \mathcal{N}} c_{ji}^{(m)}(t), 0] + a_j^{(m)} \\
&\leq Q_j^{(m)}(t) - s_j^{(m)}(t) - \sum_{i \in \mathcal{N}} c_{ji}^{(m)}(t) + a_j^{(m)} \\
&\leq Q_j^{(m)}(t) - c_{j\tilde{i}}^{(m)}(t) + a_j^{(m)} \\
&= Q_j^{(m)}(t) - \mu_{max} + a_j^{(m)} \\
&\leq Q_j^{(m)}(t) - \mu_{max} + A_{max} \\
&< Q_j^{(m)}(t)
\end{aligned}
$$

This means that the length of $Q_j^{(m)}$ can not increase in time slot $t+1$. Therefore, $Q_j^{(m)}(t+1) < {Q_j^{(m)}}^{max}$. □

A.2 Proving $Z_j^{(m)}(t) \leq {Z_j^{(m)}}^{max}, \forall j \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t$

Proof: We prove it by induction.

Induction Basis: According to the assumption, we have $Z_j^{(m)}(0) = 0 \leq {Z_j^{(m)}}^{max}$.

Induction Steps: We assume that $Z_j^{(m)}(t) \leq {Z_j^{(m)}}^{max}$, and then we show $Z_j^{(m)}(t+1) \leq {Z_j^{(m)}}^{max}$.

If $Z_j^{(m)}(t) \leq V(v^{(m)}p_{\tilde{i}} + w_{\tilde{i}}^{(m)} + q_{\tilde{i}}^{(m)})$, then according to (12), $Z_j^{(m)}(t+1) \leq V(v^{(m)}p_{\tilde{i}} + w_{\tilde{i}}^{(m)} + q_{\tilde{i}}^{(m)}) + \epsilon_j^{(m)} = {Z_j^{(m)}}^{max}$.

If $Z_j^{(m)}(t) > V(v^{(m)}p_{\tilde{i}} + w_{\tilde{i}}^{(m)} + q_{\tilde{i}}^{(m)})$, there are two cases, which we are going to discuss respectively:

Case (1). When $Q_j^{(m)}(t) > 0$: We have $\eta_{j\tilde{i}}^{(m)}(t) > \phi_{\tilde{i}}^{(m)}(t)$. According the solution to (17), we have $c_{j\tilde{i}}^{(m)}(t) = \mu_{max}$. Then

$$
\begin{aligned}
Z_j^{(m)}(t+1) &= \max[Z_j^{(m)}(t) + \epsilon_j^{(m)} - s_j^{(m)}(t) - \sum_{i \in \mathcal{N}} c_{ji}^{(m)}(t), 0] \\
&\leq Z_j^{(m)}(t) + \epsilon_j^{(m)} - s_j^{(m)}(t) - \sum_{i \in \mathcal{N}} c_{ji}^{(m)}(t) \\
&\leq Z_j^{(m)}(t) + \epsilon_j^{(m)} - \mu_{max} \\
&< Z_j^{(m)}(t) \leq {Z_j^{(m)}}^{max}
\end{aligned}
$$

Case (2). When $Q_j^{(m)}(t)=0$: According (12), $Z_j^{(m)}(t+1)=\max[Z_j^{(m)}(t)-\mu_{max},0]\leq Z_j^{(m)}(t)\leq {Z_j^{(m)}}^{max}$.

□

APPENDIX B
PROOF OF THEOREM 2

Proof: Suppose $a_j^{(m)}(t_0)$ requests arrive at queue $Q_j^{(m)}$ at time slot t_0. We prove these requests can depart the queue by time $t_0+W_j^{(m)}$ by contradiction.

If these requests haven't left the queue by $t_0+W_j^{(m)}$, it must be that $Q_j^{(m)}(t)>0$ for all $t\in\{t_0+1,...,t_0+W_j^{(m)}\}$. Then for all $t\in\{t_0+1,...,t_0+W_j^{(m)}\}$ we have:

$$Z_j^{(m)}(t+1)=\max[Z_j^{(m)}(t)+\epsilon_j^{(m)}-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t),0].$$

Therefore we have

$$Z_j^{(m)}(t+1)\geq Z_j^{(m)}(t)+\epsilon_j^{(m)}-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t).$$

Summing up the above over $t\in\{t_0+1,...,t_0+W_j^{(m)}\}$ yields:

$$Z_j^{(m)}(t_0+W_j^{(m)}+1)-Z_j^{(m)}(t_0+1)\geq \epsilon_j^{(m)}W_j^{(m)}-\sum_{t=t_0+1}^{t_0+W_j^{(m)}}[s_j^{(m)}(t)+\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)].$$

Because $Z_j^{(m)}(t_0+W_j^{(m)}+1)\leq {Z_j^{(m)}}^{max}$ and $Z_j^{(m)}(t_0+1)\geq 0$, we have:

$$\sum_{t=t_0+1}^{t_0+W_j^{(m)}}[s_j^{(m)}(t)+\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)]\geq \epsilon_j^{(m)}W_j^{(m)}-{Z_j^{(m)}}^{max}$$

On the other side, because we assume that not all $a_j^{(m)}(t_0)$ requests depart by the time $t_0+W_j^{(m)}$, we have:

$$\sum_{t=t_0+1}^{t_0+W_j^{(m)}}[s_j^{(m)}(t)+\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)]<Q_j^{(m)}(t_0+1)\leq {Q_j^{(m)}}^{max}.$$

Therefore we have ${Q_j^{(m)}}^{max}>\epsilon_j^{(m)}W_j^{(m)}-{Z_j^{(m)}}^{max}$, *i.e.*, $W_j^{(m)}<\frac{{Q_j^{(m)}}^{max}+{Z_j^{(m)}}^{max}}{\epsilon_j^{(m)}}$. This contradicts the definition of $W_j^{(m)}$.

Therefore, any request that arrived at time t_0 will be dispatched by time slot $t_0+W_j^{(m)}$. □

APPENDIX C
PROOF OF THEOREM 3

Define *T-slot Drift* as $\Delta_T(\Theta(t))=L(\Theta(t+T))-L(\Theta(t))$.

Based on Lemma 4.11 in [12] , we have

Lemma 1: (T-slot Drift) With our dynamic algorithm, for all t, all $\Theta(t)$, and for any integer $T>0$ we have:

$$\begin{aligned}
&\Delta_T(\Theta(t))+V\sum_{\tau=t}^{t+T-1}M(\tau)\leq\\
&BT^2+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Q_j^{(m)}(t)\sum_{\tau=t}^{t+T-1}(a_j^{(m)}(\tau)-{s_j^{(m)}}^*(\tau)-\sum_i{c_{ji}^{(m)}}^*(\tau))\\
&+G(t)(\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}\sum_{\tau=t}^{t+T-1}(\sum_{i\in\mathcal{N}}{c_{ji}^{(m)}}^*(\tau)e_{ji}+{s_j^{(m)}}^*(\tau)d_j)\\
&-\alpha\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(\sum_{i\in\mathcal{N}}{c_{ji}^{(m)}}^*(\tau)+{s_j^{(m)}}^*(\tau)))\\
&+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Z_j^{(m)}(t)\sum_{\tau=t}^{t+T-1}(1_{\{Q_j^{(m)}(\tau)>0\}}(\epsilon_j^{(m)}-{s_j^{(m)}}^*(\tau)\\
&-\sum_{i\in\mathcal{N}}{c_{ji}^{(m)}}^*(\tau))-1_{\{Q_j^{(m)}(\tau)=0\}}\mu_{max})\\
&+V\sum_{\tau=t}^{t+T-1}(\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}}{c_{ji}^{(m)}}^*(\tau)q_i^{(m)}\\
&+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}}v^{(m)}{s_j^{(m)}}^*(\tau)h\\
&+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}v^{(m)}{y_i^{(m)}}^*(\tau)p_j\\
&+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}[{y_i^{(m)}}^*(\tau)-{y_i^{(m)}}^*(\tau-1)]^+w_j^{(m)}),
\end{aligned}$$

where ${s_j^{(m)}}^*(\tau)$, ${c_{ji}^{(m)}}^*(\tau)$, and ${y_i^{(m)}}^*(\tau)$, $\forall i,j\in\mathcal{N},m\in\mathcal{M}$, are any alternative decisions that can be made in time slot τ within the feasible set.

Proof of Theorem 3: Because ${s_j^{(m)}}^*(\tau)$, ${c_{ji}^{(m)}}^*(\tau)$, and ${y_i^{(m)}}^*(\tau)$, $\forall i,j\in\mathcal{N},m\in\mathcal{M}$, are any alternative decisions that can be made in time slot τ within the feasible set, apparently, ${s_j^{(m)}}^*(\tau)$, ${c_{ji}^{(m)}}^*(\tau)$, and ${y_i^{(m)}}^*(\tau)$, $\forall i,j\in\mathcal{N},m\in\mathcal{M}$, can be the optimal solution of the problem with information of T time slots into future that minimizes Eqn. (22). Then, combining with Lemma 1, we derive

$$\Delta_T(\Theta(t))+V\sum_{\tau=t}^{t+T-1}M(\tau)\leq BT^2+V\sum_{\tau=t}^{t+T-1}M^*(\tau).$$

Considering the total K frames and summing the above over $k\in\{0,...,K-1\}$ and then dividing the sum by VKT, we get

$$\frac{L(\Theta(KT))-L(\Theta(0))}{VKT}+\frac{1}{KT}\sum_{t=0}^{KT-1}M(t)\leq\frac{BT}{V}+\frac{1}{K}\sum_{k=0}^{K-1}\widehat{M_k}.$$

Rearranging the terms in the above inequality, and noting that $L(\Theta(KT))\geq 0$ and $L(\Theta(0))=0$, we derive

$$\frac{1}{KT}\sum_{\tau=0}^{KT-1}M(t)\leq\frac{1}{K}\sum_{k=0}^{K-1}\widehat{M_k}+\frac{BT}{V}.$$ □

APPENDIX D
DERIVATION OF (14)

$$
\begin{aligned}
&\Delta(\mathbf{\Theta}(t))\\
&=\mathbf{E}\{\frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}[Q_j^{(m)}(t+1)^2-Q_j^{(m)}(t)^2]+\frac{1}{2}[G(t+1)^2-G(t)^2]\\
&\quad+\frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}[Z_j^{(m)}(t+1)^2-Z_j^{(m)}(t)^2]|[\mathbf{Q}(t),\mathbf{G}(t),\mathbf{Z}(t)]\}\\
&=\frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}[Q_j^{(m)}(t+1)^2-Q_j^{(m)}(t)^2]+\frac{1}{2}[G(t+1)^2-G(t)^2]\\
&\quad+\frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}[Z_j^{(m)}(t+1)^2-Z_j^{(m)}(t)^2]
\end{aligned}
\tag{33}
$$

We have

$$
\begin{aligned}
&\frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}[Q_j^{(m)}(t+1)^2-Q_j^{(m)}(t)^2]\\
&=\frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}[(\max[Q_j^{(m)}(t)-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t),0]+a_j^{(m)}(t))^2\\
&\quad-Q_j^{(m)}(t)^2]\\
&\le\frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}[(s_j^{(m)}(t)+\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t))^2+a_j^{(m)}(t)^2]\\
&\quad+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Q_j^{(m)}(t)[a_j^{(m)}(t)-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)]\\
&\le\frac{1}{2}[|\mathcal{M}||\mathcal{N}|(b+N\mu_{max})^2+A_{max}^2]\\
&\quad+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Q_j^{(m)}(t)[a_j^{(m)}(t)-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)],
\end{aligned}
\tag{34}
$$

and

$$
\begin{aligned}
&\frac{1}{2}[G(t+1)^2-G(t)^2]\\
&=\frac{1}{2}[(\max[G(t)+\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t)d_j+\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)e_{ji})\\
&\quad-\alpha\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t)+\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)),0])^2-G(t)^2]\\
&\le\frac{1}{2}[\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t)d_j+\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)e_{ji})]^2\\
&\quad+\frac{1}{2}\alpha^2[\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t)+\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t))]^2\\
&\quad+G(t)[\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)e_{ji}+s_j^{(m)}(t)d_j)\\
&\le\frac{1}{2}(|\mathcal{M}||\mathcal{N}|^2\mu_{max}e_{max}+bd_{max})^2+\frac{1}{2}\alpha^2(|\mathcal{M}||\mathcal{N}|^2\mu_{max}+b)^2\\
&\quad+G(t)[\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)e_{ji}+s_j^{(m)}(t)d_j),
\end{aligned}
\tag{35}
$$

and

$$
\begin{aligned}
&\frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}[Z_j^{(m)}(t+1)^2-Z_j^{(m)}(t)^2]\\
&=\frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}[(\max[Z_j^{(m)}(t)+1_{\{Q_j^{(m)}(t)>0\}}(\epsilon_j^{(m)}-s_j^{(m)}(t)\\
&\quad-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t))-1_{\{Q_j^{(m)}(t)=0\}}\mu_{max},0])^2-Z_j^{(m)}(t)^2]\\
&\le\frac{1}{2}[1_{\{Q_j^{(m)}(t)>0\}}(\epsilon_j^{(m)}-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t))-1_{\{Q_j^{(m)}(t)=0\}}\mu_{max}]^2\\
&\quad+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Z_j^{(m)}(t)[1_{\{Q_j^{(m)}(t)>0\}}(\epsilon_j^{(m)}-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t))\\
&\quad-1_{\{Q_j^{(m)}=0\}}\mu_{max}]\\
&\le\frac{1}{2}|\mathcal{M}||\mathcal{N}|[\epsilon_{max}^2+(b+N\mu_{max})^2]\\
&\quad+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Z_j^{(m)}(t)[1_{\{Q_j^{(m)}(t)>0\}}(\epsilon_j^{(m)}-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t))\\
&\quad-1_{\{Q_j^{(m)}=0\}}\mu_{max}].
\end{aligned}
\tag{36}
$$

We define a constant $B=\frac{1}{2}|\mathcal{M}||\mathcal{N}|[A_{max}^2+\epsilon_{max}^2+2(b+N\mu_{max})^2]+\frac{1}{2}(|\mathcal{M}||\mathcal{N}|^2\mu_{max}e_{max}+bd_{max})^2+\frac{1}{2}\alpha^2(|\mathcal{M}||\mathcal{N}|^2\mu_{max}+b)^2$, where $d_{max}=\max\{d_j|j\in\mathcal{N}\}$, $e_{max}=\max\{e_{ji}|j\in\mathcal{N},i\in\mathcal{N}\}$, and $\epsilon_{max}=\max\{\epsilon_j^{(m)}|j\in\mathcal{N},m\in\mathcal{M}\}$. Combining (34)(35)(36) together, we have

$$
\begin{aligned}
&\Delta(\mathbf{\Theta}(t))\\
&\le B+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Q_j^{(m)}(t)[a_j^{(m)}(t)-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)]\\
&\quad+G(t)[\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)e_{ji}+s_j^{(m)}(t)d_j)\\
&\quad-\alpha\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)+s_j^{(m)}(t))]+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Z_j^{(m)}(t)\\
&\quad[1_{\{Q_j^{(m)}(t)>0\}}(\epsilon_j^{(m)}-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t))-1_{\{Q_j^{(m)}=0\}}\mu_{max}]
\end{aligned}
$$

Therefore, we get

$$
\begin{aligned}
&\Delta(\mathbf{\Theta}(t))+VM(t)\\
&\le B+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Q_j^{(m)}(t)[a_j^{(m)}(t)-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)]\\
&\quad+G(t)[\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)e_{ji}+s_j^{(m)}(t)d_j)\\
&\quad-\alpha\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)+s_j^{(m)}(t))]+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Z_j^{(m)}(t)\\
&\quad[1_{\{Q_j^{(m)}(t)>0\}}(\epsilon_j^{(m)}-s_j^{(m)}(t)-\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t))-1_{\{Q_j^{(m)}=0\}}\mu_{max}]\\
&\quad+V[\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)q_i^{(m)}+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}v^{(m)}s_j^{(m)}(t)h\\
&\quad+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}v^{(m)}y_i^{(m)}(t)p_i\\
&\quad+\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{N}}[y_i^{(m)}(t)-y_i^{(m)}(t-1)]^+w_i^{(m)}]\\
&=B-\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}s_j^{(m)}(t)[Q_j^{(m)}(t)+(\alpha-d_j)G(t)\\
&\quad+1_{\{Q_j^{(m)}(t)>0\}}Z_j^{(m)}(t)-v^{(m)}Vh]-\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}}c_{ji}^{(m)}(t)\\
&\quad[Q_j^{(m)}(t)+(\alpha-e_{ji})G(t)+1_{\{Q_j^{(m)}(t)>0\}}Z_j^{(m)}(t)-Vq_i^{(m)}]\\
&\quad+V\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{N}}[v^{(m)}y_i^{(m)}(t)p_i+[y_i^{(m)}(t)-y_i^{(m)}(t-1)]^+w_i^{(m)}]\\
&\quad+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Z_j^{(m)}(t)[1_{\{Q_j^{(m)}(t)>0\}}\epsilon_j^{(m)}-1_{\{Q_j^{(m)}=0\}}\mu_{max}]\\
&\quad+\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}Q_j^{(m)}(t)a_j^{(m)}(t).
\end{aligned}
$$

REFERENCES

[1] *Amazon CloudFront*, http://aws.amazon.com/cloudfront/.
[2] *Microsoft Azure*, http://www.microsoft.com/windowsazure/.
[3] *Google App Engine*, http://code.google.com/appengine/.
[4] *Dropbox*, http://www.dropbox.com/.
[5] *Microsoft Office Web Apps*, http://office.microsoft.com/en-us/web-apps/.
[6] *Google docs*, http://docs.google.com/.
[7] M. Hajjat, X. Sun, Y. E. Sung, D. Maltz, and S. Rao, "Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud," in *Proc. of IEEE SIGCOMM*, August 2010.
[8] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, "Intelligent Workload Factoring for a Hybrid Cloud Computing Model," in *Proc. of the International Workshop on Cloud Services (IWCS 2009)*, June 2009.
[9] H. Li, L. Zhong, J. Liu, B. Li, and K. Xu, "Cost-effective Partial Migration of VoD Services to Content Clouds," in *Proc. of IEEE CLOUD*, July 2011.
[10] X. Cheng and J. Liu, "Load-Balanced Migration of Social Media to Content Clouds," in *Proc. of NOSSDAV*, June 2011.
[11] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1–149, 2006.
[12] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.
[13] ——, "Energy optimal control for time varying wireless networks," *IEEE Tran. on Information Theory*, no. 7, pp. 2915–2934, July 2006.
[14] M. M. Amble, P. Parag, S. Shakkottai, and L. Ying, "Content-Aware Caching and Traffic Management in Content Distribution Networks," in *Proc. of IEEE INFOCOM*, April 2011.
[15] M. J. Neely and L. Golubchik, "Utility Optimization for Dynamic Peer-to-Peer Networks with Tit-For-Tat Constraints," in *Proc. of IEEE INFOCOM*, April 2011.
[16] M. Pathan, J. Broberg, and R. Buyya, "Maximizing Utility for Content Delivery Clouds," in *Proc. of the 10th International Conference on Web Information Systems Engineering*, 2009.
[17] F. Chen, K. Guo, J. Lin, and T. L. Porta, "Intra-cloud Lightning: Building CDNs in the Cloud," in *Proc. of IEEE INFOCOM*, 2012.
[18] H. Li, W. Huang, C. W. abd Z. Li, and F. C. Lau, "Utility-Maximizing Data Dissemination in Socially Selfish Cognitive Radio Networks," in *Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2011)*, Oct 2011.
[19] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic Models of Load Balancing and Scheduling in Cloud Computing Clusters," in *Proc. of IEEE INFOCOM*, 2012.
[20] S. Ren, Y. He, and F. Xu, "Provably-Efficient Job Scheduling for Energy and Fairness in Geographically Distributed Data Centers," in *Proc. of IEEE ICDCS*, 2012.
[21] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, "Distributed Placement of Service Facilities in Large-Scale Networks," in *Proc. of IEEE INFOCOM*, 2007.
[22] J. Leblet, Z. Li, G. Simon, and D. Yuan, "Optimal Network Location in Distributed Virtualized Data-Centers," *Computer Communications*, no. 16, pp. 1968–1979, 2011.
[23] S. H. Owen and M. S. Daskin, "Strategic Facility Location: A Review," pp. 423–447, 1998.
[24] M. Charikar, S. Guha, E. Tardos, and D. B. Shmoys, "A constant-factor approximation algorithm for the k-median problem," in *Proc. of the 31st Annual ACM Symposium on Theory of Computing (STOC'99)*, 1999.
[25] *Amazon Elastic Compute Cloud*, http://aws.amazon.com/ec2/.
[26] *Amazon Simple Storage Service*, http://aws.amazon.com/s3/.
[27] M. J. Neely, "Opportunistic Scheduling with Worst Case Delay Guarantees in Single and Multi-Hop Networks," in *Proc. of IEEE INFOCOM*, 2011.
[28] *Linode,*, https://www.linode.com.
[29] X. Cheng, J. Liu, and C. Dale, "Understanding the Characteristics of Internet Short Video Sharing: A YouTube-Based Measurement Study," *IEEE Transactions on Multimedia*, vol. 15, no. 5, pp. 1184–1194, 2013.
[30] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," in *Prof. of IEEE INFOCOM*, 1999.
[31] *SoftLayer*, http://www.softlayer.com.
[32] *Layered Tech*, http://www.layeredtech.com/.
[33] X. Xing, J. Dang, S. Mishra, and X. Liu, "A Highly Scalable Bandwidth Estimation of Commercial Hotspot Access Points," in *Proc. of IEEE INFOCOM*, 2011.
[34] R. Kuschnig, I. Koer, and H. Hellwagner, "Improving Internet Video Streaming Performance by Parallel TCP-based Request-Response Streams," in *Proc. of CCNC*, Jan. 2010.
[35] *MOSEK*, http://www.mosek.com/.