# Data Center Server Provision: Distributed Asynchronous Control for Coupled Renewal Systems

Xiaohan Wei and Michael J. Neely, *Senior Member, IEEE*

***Abstract*—This paper considers a cost minimization problem for data centers with $N$ servers and randomly arriving service requests. A central router decides which server to use for each new request. Each server has three types of states (active, idle, setup) with different costs and time durations. The servers operate asynchronously over their own states and can choose one of multiple sleep modes when idle. We develop an online distributed control algorithm so that each server makes its own decisions, the request queues are bounded and the overall time average cost is near optimal with probability 1. The algorithm does not need probability information for the arrival rate or job sizes. Next, an improved algorithm that uses a single queue is developed via a "virtualization" technique which is shown to provide the same (near optimal) costs. Simulation experiments on a real data center traffic trace demonstrate the efficiency of our algorithm compared to other existing algorithms.**

***Index Terms*—Data center, opportunistic scheduling, distributed optimization, renewal optimization**

## I. Introduction

### A. Overview

Consider a data center that consists of a central controller and $N$ servers that serve randomly arriving requests. The system operates in slotted time with time slots $t \in \{0, 1, 2, \ldots\}$. Each server $n \in \{1, \ldots, N\}$ has three basic states:

- Active: The server is available to serve requests. Server $n$ incurs a cost of $e_n \geq 0$ on every active slot, regardless of whether or not requests are available to serve. In data center applications, such cost often represents the power consumption of each individual server.
- Idle: A low cost sleep state where no requests can be served. The idle state is actually comprised of a choice of multiple sleep modes with different per-slot costs. The specific sleep mode also affects the setup time required to transition from the idle state to the active state. For the rest of the paper, we use "idle" and "sleep" exchangeably.
- Setup: A transition period from idle to active during which no requests can be served. The setup cost and duration depend on the preceding sleep mode. The setup duration is typically more than one slot, and can be a random variable that depends on the server $n$ and on the preceding sleep mode.

An active server can choose to transition to the idle state at any time. When it does so, it chooses the specific sleep mode to use and the amount of time to sleep. For example, deeper sleep modes can shut down more electronics and thereby save on per-slot idling costs. However, a deeper sleep incurs a longer setup time when transitioning back to the active state. Each server makes separate decisions about when to transition and what sleep mode to use. The resulting transition times for each server are asynchronous. On top of this, a central controller makes slot-wise decisions for routing requests to servers. It can also reject requests (with a certain amount of cost) if it decides they cannot be supported. The goal is to minimize the overall time average cost.

This problem is challenging mainly for two reasons: First, since each setup state generates cost but serves no request, it is not clear whether or not transitioning to idle from the active state indeed saves power. It is also not clear which sleep mode the server should switch to. Second, if one server is currently in a setup state, it cannot make another decision until it reaches the active state (which typically takes more than one slot), whereas other active servers can make decisions during this time. Thus, this problem can be viewed as a system with coupled Markov decision processes (MDPs) making decisions asynchronously. Classical methods for MDPs, such as dynamic programming and linear programming [1][2][3], can be impractical for two reasons: First, the state space has dimension that depends on the number of servers, making solutions difficult when the number of servers is large. Second, some statistics of the system, such as the arrival probabilities, are unknown.

### B. Related works

In this paper, we use renewal-reward theory (e.g. [4]) together with Lyapunov optimization (e.g. [5] and [6]) to develop a simple implementable algorithm for this problem. Our work is not alone in approaching the problem this way. Work in [7] uses Lyapunov theory for a system with one renewal server, while work in [8] considers a multi-server system but in a deterministic context. The work in [9] develops a two stage algorithm for stochastic multi-server systems, but the first stage must be solved offline. This paper is distinct in that we overcome the open challenges of previous work and develop a near optimal fully online algorithm for a stochastic system with multiple servers.

Several alternative approaches to multi-server systems use queueing theory. Specifically, the work in [10] treats the multi-server system as an $M/M/k/setup$ queue, whereas the work in [11] considers modeling a single server system as a

The authors are with the Electrical Engineering department at the University of Southern California, Los Angeles, CA.

multi-class $M/G/1$ queue. These approaches require Poisson traffic and do not treat the asynchronous control problem with multiple sleep options considered in the current paper.

Experimental work on power and delay minimization is treated in [12], which proposes to turn each server ON and OFF according to the rule of an $M/M/k/setup$ queue. The work in [13] applies Lyapunov optimization to optimize power in virtualized data centers. However, it assumes each server has negligible setup time and that ON/OFF decisions are made synchronously at each server. The works [14], [15] focus on power-aware provisioning over a time scale large enough so that the whole data center can adjust its service capacity. Specifically, [14] considers load balancing across geographically distributed data centers, and [15] considers provisioning over a finite time interval and introduces an online 3-approximation algorithm.

Prior works [16], [17], [18] consider servers with multiple hypothetical sleep states with different levels of power consumption and setup times. Although empirical evaluations in these works show significant power saving by introducing sleep states, they are restricted to the scenario where the setup time from sleep to active is on the order of milliseconds, which is not realistic for today's data center. Realistic sleep states with setup time on the order of seconds are considered in [19], where effective heuristic algorithms are proposed and evaluated via extensive testbed simulations. However, little is known about the theoretical performance bound regarding these algorithms. In this paper, we propose a new algorithm which incorporates servers with different sleep modes while having a provable performance guarantee.

### C. Contributions

On the theoretical side, the current paper is the first to consider the stochastic control problem with heterogeneous servers and multiple idle and setup states at each server. This gives rise to a nonstandard asynchronous problem with coupled Markov decision systems. A novel aspect of the solution is the construction of a process with super-martingale properties by piecing together the asynchronous processes at each server. This is interesting because neither the individual server processes nor their asynchronous sum are super-martingales. The technique yields a simple distributed algorithm that can likely be applied more broadly for other coupled stochastic MDPs.

On the practical side, we run our algorithm on a real data center traffic trace with server setup time on the order of seconds. Simulation experiments show that our algorithm is effective compared to several existing algorithms.

## II. Problem formulation

Consider a slotted time system with $N$ servers, denoted by set $\mathcal{N}$, that serve randomly incoming requests.

### A. Front-end load balancing

At each time slot $t \in \{0, 1, 2, \ldots\}$, $\lambda(t)$ new requests arrive at the system. We assume $\lambda(t)$ takes values in a finite set $\Lambda$. Let $R_n(t),\ n \in \mathcal{N}$ denote the number of requests routed into server $n$ at time $t$. In addition, the system is allowed to reject requests. Let $d(t)$ be the number of requests that are rejected on slot $t$, and let $c(t)$ be the corresponding per-request cost for such rejection. Assume $c(t)$ takes values in a finite state space $\mathcal{C}$. The $R_n(t)$ and $d(t)$ decision variables on slot $t$ must be nonnegative integers that satisfy:

$$\sum_{n=1}^{N} R_n(t) + d(t) = \lambda(t)$$
$$\sum_{n=1}^{N} R_n(t) \leq R_{\max}$$

for a given integer $R_{max} > 0$. The vector process $(\lambda(t), c(t))$ takes values in $\Lambda \times \mathcal{C}$ and is assumed to be an independent and identically distributed (i.i.d.) vector over slots $t \in \{0, 1, 2, \ldots\}$ with an unknown probability mass function.

Each server $n$ maintains a request queue $Q_n(t)$ that stores the requests that are routed to it. Requests are served in a FIFO manner with queueing dynamics as follows:

$$Q_n(t+1) = \max\left\{Q_n(t) + R_n(t) - \mu_n(t)H_n(t),\ 0\right\}. \quad (1)$$

where $H_n(t)$ is an indicator variable that is 1 if server $n$ is active on slot $t$, and 0 else, and $\mu_n(t)$ is a random variable that represents the number of requests can be served on slot $t$. Each queue is initialized to $Q_n(0) = 0$. Assume that, every slot in which server $n$ is active, $\mu_n(t)$ is independent and identically distributed with a known mean $\mu_n$. This randomness can model variation in job sizes.

**Assumption 1.** *The process $\{(\lambda(t), c(t))\}_{t=0}^{\infty}$ is observable, i.e. the router can observe the $(\lambda(t), c(t))$ realization each time slot $t$ before making decisions. In contrast, the process $\{\mu_n(t)\}_{t=0}^{\infty}$ is not observable, i.e. given that $H_n(t) = 1$, at the beginning of slot $t$ server $n$ knows a random service will take place, but it does not know the realization of $\mu_n(t)$ until the end of slot $t$. Moreover, $\lambda(t),\ c(t)$ and $\mu_n(t)$ are all bounded.*

### B. Server model

Each server $n \in \mathcal{N}$ has three types of states: active, idle, and setup. The idle state of each server $n$ is further decomposed into a collection of distinct sleep modes. Each server $n \in \mathcal{N}$ makes decisions over its own *renewal frames*. Define the renewal frame for server $n$ as the time period between successive visits to active state (with each renewal period ending in an active state). Let $T_n[f]$ denote the frame size of the $f$-th renewal frame for server $n$, for $f \in \{0, 1, 2, \ldots\}$. Let $t_f^{(n)}$ denote the start of frame $f$, so that $T_n[f] = t_{f+1}^{(n)} - t_f^{(n)}$. Assume that $t_0^{(n)} = 0$ for all $n \in \mathcal{N}$, so that time slot 0 is the start of the first renewal frame (labeled frame $f = 0$) for all servers. For simplicity, assume all servers are "active" on slot $t = -1$. Thus, the slot just before each renewal frame is an active slot. Fig. 1 illustrates this renewal frame construction.

Fix a server $n \in \mathcal{N}$ and a frame index $f \in \{0, 1, 2, \ldots\}$. Time $t_f^{(n)}$ marks the start of renewal frame $f$. At this time, server $n$ must decide whether to remain active or to go idle. If it remains active then the renewal frame lasts for one slot, so

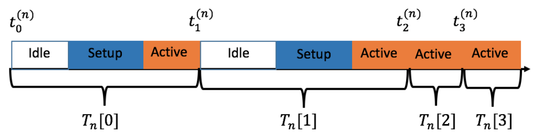


Fig. 1. Illustration of a typical renewal frame construction, where $T_n[i]$ is the length of frame $i$ and $t_i^{(n)}$ is the start slot of frame $i$.

that $T_n[f] = 1$. If it goes idle, it chooses an idle mode from a finite set $\mathcal{L}_n$, representing the set of idle mode options. Let $\alpha_n[f]$ represent this initial decision for server $n$ at the start of frame $f$, so that:

$$\alpha_n[f] \in \{active\} \cup \mathcal{L}_n$$

where $\alpha_n[f] = active$ means the server chooses to remain active. If the server chooses to go idle, so that $\alpha_n[f] \in \mathcal{L}_n$, it then chooses a variable $I_n[f]$ that represents *how much time it remains idle*. The decision variable $I_n[f]$ is chosen as an integer in the set $\{1, \ldots, I_{max}\}$ for some given integer $I_{max} > 0$. The consequences of these decisions are described below.

- Case $\alpha_n[f] = active$. The frame starts at time $t_f^{(n)}$ and has size $T_n[f] = 1$. The active variable becomes $H_n(t_f^{(n)}) = 1$ and an activation cost of $e_n$ is incurred on this slot $t_f^{(n)}$. A random service variable $\mu_n(t_f^{(n)})$ is generated and requests are served according to the queue update (1). Recall that, under Assumption 1, the value of $\mu_n(t)$ is not known until the end of the slot.
- Case $\alpha_n[f] \in \mathcal{L}_n$. In this case, the server chooses to go idle and $\alpha_n[f]$ represents the specific sleep mode chosen. The idle duration $I_n[f]$ is also chosen as an integer in the set $[1, I_{max}]$. After the idle duration completes, the setup duration starts and has an independent and random duration $\tau_n[f] = \hat{\tau}(\alpha_n[f])$, where $\hat{\tau}(\alpha_n[f])$ is an integer random variable with a known mean and variance that depends on the sleep mode $\alpha_n[f]$. At the end of the setup time the system goes active and serves with a random $\mu_n(t)$ as before. The active variable is $H_n(t) = 0$ for all slots $t$ in the idle and setup times, and is 1 at the very last slot of the frame. Further:
  - Idle cost: Every slot $t$ of the idle time of frame $f$, an idle cost of $g_n(t) = \hat{g}_n(\alpha_n[f])$ is incurred (so that the idle cost depends on the sleep mode). We have $g_n(t) = 0$ if server $n$ is not idle on slot $t$. The idle cost can be zero, but can also be a small but positive value if some electronics are still running in the sleep mode chosen.
  - Setup cost: Every slot $t$ of the setup time of frame $f$, a cost of $W_n(t) = \hat{W}_n(\alpha_n[f])$ is incurred. We have $W_n(t) = 0$ if server $n$ is not in a setup duration on slot $t$.

Thus, the length of frame $f$ for server $n$ is:

$$T_n[f] = \begin{cases} 1, & \text{if } \alpha_n[f] = active; \\ I_n[f] + \tau_n[f] + 1, & \text{if } \alpha_n[f] \in \mathcal{L}_n. \end{cases} \quad (2)$$

TABLE I
PARAMETERS

| Control parameters | Control objectives |
|---|---|
| $R_n(t)$ | Requests routed to server $n$ at slot $t$ |
| $d(t)$ | Requests rejected at slot $t$ |
| $\alpha_n[f]$ | The option (active/idle) server $n$ takes in frame $f$ |
| $I_n[f]$ | Number of slots server $n$ stays idle in frame $f$ |
| Other parameters | Meaning |
| $\lambda(t)$ | Number of arrivals at time $t$ |
| $c(t)$ | Per request rejection cost at time $t$ |
| $e_n$ | Per slot active service cost for server $n$ |
| $T_n[f]$ | The length of frame $f$ for server $n$ |
| $t^{(n)}[f]$ | Starting slot of frame $f$ for server $n$ |
| $\tau_n[f]$ | Setup duration in frame $f$ |
| $\mu_n(t)$ | Number of requests served on server $n$ at time $t$ |
| $H_n(t)$ | Server active indicator (equal to 1 if active, 0 if not) |
| $g_n(t)$ | Idle cost of server $n$ at time $t$ |
| $W_n(t)$ | Setup cost of server $n$ at time $t$ |

In summary, the costs $\hat{g}_n(\alpha_n)$, $\hat{W}_n(\alpha_n)$ and the setup time $\hat{\tau}_n(\alpha_n)$ are functions of $\alpha_n \in \mathcal{L}_n$. We further make the following assumption regarding $\hat{\tau}_n(\alpha_n)$:

**Assumption 2.** *For any $\alpha_n \in \mathcal{L}_n$, the function $\hat{\tau}_n(\alpha_n)$ is an integer random variable with known mean and variance, as well as bounded first four moments. Denote $\mathbb{E}[\tau_n(\alpha_n)] = m_{\alpha_n}$ and Var$[\tau_n(\alpha_n)] = \sigma^2_{\alpha_n}$.*

Note that this is a very mild assumption in view of the fact that the setup time of a real server is always bounded. The motivation behind emphasizing the fourth moment here instead of simply proceeding with boundedness assumption is more of theoretical interest than practical importance.

Table I summarizes the parameters introduced in this section. The data center architecture is shown is Fig. 2. Since different servers might make different decisions, the renewal frames are not necessarily aligned. A typical asynchronous timeline for a system of three servers is illustrated in Fig. 3.

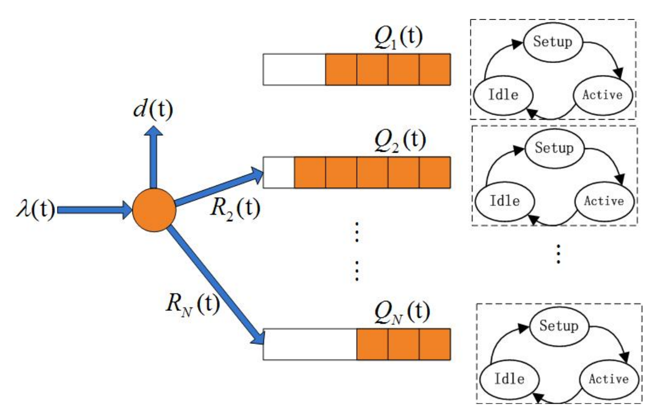


Fig. 2. **Illustration of a data center structure which contains a front-end load balancer, $N$ application servers with $N$ request queues and a backend database (omitted here for brevity).**

## C. Performance Objective

For each $n \in \mathcal{N}$, let $\overline{C}$, $\overline{W}_n$, $\overline{E}_n$, $\overline{G}_n$ be the time average costs resulting from rejection, setup, service and idle, respectively. They are defined as

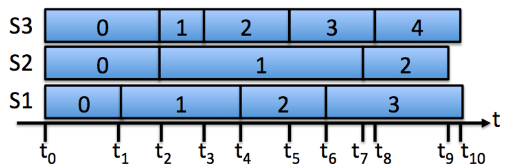


Fig. 3. Illustration of a typical asynchronous timeline for a system of three servers, where we count the number of renewal frames per single server.

follows: $\overline{C} = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} d(t)c(t)$, $\overline{W}_n = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} W_n(t)$, $\overline{E}_n = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} e_nH_n(t)$, $\overline{G}_n = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} g_n(t)$.

The goal is to design a joint routing and service policy so that the time average overall cost is minimized and all queues are stable, i.e.

$$\min\ \overline{C} + \sum_{n=1}^{N}\left(\overline{W}_n + \overline{E}_n + \overline{G}_n\right),\ \text{s.t. } Q_n(t) \text{ stable } \forall n. \quad (3)$$

Notice that the constraint in (3) is not easy to work with. In order to get an optimization problem one can deal with, we further define the time average request rate, rejection rate, routing rate and service rate as $\overline{\lambda}$, $\overline{d}$, $\overline{R}_n$, and $\overline{\mu}_n$ respectively: $\overline{\lambda} = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\lambda(t)$, $\overline{d} = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} d(t)$, $\overline{R}_n = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} R_n(t)$, $\overline{\mu}_n = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mu_n(t)H_n(t)$.

Then, rewrite the problem (3) as follows

$$\min\quad \overline{C} + \sum_{n=1}^{N}\left(\overline{W}_n + \overline{E}_n + \overline{G}_n\right) \quad (4)$$

$$\text{s.t.}\quad \overline{R}_n \le \overline{\mu}_n,\ \forall n \in \mathcal{N} \quad (5)$$

$$\sum_{n=1}^{N} R_n(t) \le R_{\max},\quad \sum_{n=1}^{N} R_n(t) + d(t) = \lambda(t)\ \forall t \quad (6)$$

Constraint (5) requires the time average arrival rate to server $n$ to be less than the time average service rate. We aim to develop an algorithm so that each server can make its own decision (without looking at the workload or service decision of any other server) and prove its near optimality.

## III. An online control algorithm

We now present an online control algorithm which makes the joint request routing and service decisions as functions of $(\lambda(t), c(t), \mathbf{Q}(t))$, where $\mathbf{Q}(t) = (Q_1(t), \dots, Q_N(t))$. The algorithm is inherited from the Lyapunov optimization technique and features a trade-off parameter $V > 0$. In view of the fact that the traditional Lyapunov optimization technique does not require the statistics of the arrivals in a stochastic network optimization problem (see [5], [6] for details), our proposed algorithm does not need the statistics of $\lambda(t)$, $c(t)$ either and yields a simple distributed implementation.

### A. Intuition on the algorithm design

Recall that our goal is to make routing and service decisions so as to solve the optimization problem (4)-(6). First of all, from the queueing model described in the last section and Fig. 2, it is intuitive that an efficient algorithm would have each server make decisions regarding its own queue state $Q_n(t)$, whereas the front-end load-balancer make routing and rejection decisions slot-wise based on the global information $(\lambda(t), c(t), \mathbf{Q}(t))$.

Next, to get an idea on what exactly the decision should be, by virtue of Lyapunov optimization, one would introduce a trade-off parameter $V > 0$ and penalize the time average constraint (5) via $\mathbf{Q}(t)$ to solve the following slotwise optimization problem

$$\min\quad V\left(c(t)d(t) + \sum_{n=1}^{N}\left(W_n(t) + e_nH_n(t) + g_n(t)\right)\right) \quad (7)$$
$$+ \sum_{n=1}^{N} Q_n(t)(R_n(t) - \mu_n(t))\quad \text{s.t. constraint (6)},$$

which is naturally separable regarding the load-balancing decision ($d(t)$, $R_n(t)$), and the service decision ($W_n(t)$, $H_n(t)$, $g_n(t)$, $\mu_n(t)$). However, because of the existence of a setup state (on which no decision could be made), the server does not have an identical decision set every slot and furthermore, the decision set itself depends on previous decisions. This poses a significant difficulty analyzing the above optimization (7).

In order to resolve this difficulty, we try to find the smallest "identical time unit" for each individual server in lieu of slots. This motivates the notion of renewal frame in the previous section (see Fig. 1). Specifically, from Fig. 1 and the related renewal frame construction, at the starting slot of each renewal, the server faces the identical decision set (remain active or go to idle with certain slots) regardless of previous decisions. Following this idea, we modify (7) as follows:

- For the front-end load balancer, we observe $(\lambda(t), c(t), \mathbf{Q}(t))$ and solve $min\ \ Vc(t)d(t) + \sum_{n=1}^{N} Q_n(t)R_n(t)$, $s.t.$ (6), which is detailed in Section III-B.
- For each server, instead of per slot optimization $\min\ V(W_n(t) + e_nH_n(t) + g_n(t)) - Q_n(t)\mu_n(t)$, we propose to minimize the time average of this quantity per renewal frame $T_n[f]$, which is (12) in Section III-C.

### B. Thresholding algorithm on admission and routing

During each time slot, the data center chooses integers $R_n(t)$ and $d(t)$ to solve the following optimization problem:

$$\min\quad Vd(t)c(t) + \sum_{n=1}^{N} Q_n(t)R_n(t) \quad (8)$$
$$\text{s.t.}\quad \sum_{n=1}^{N} R_n(t) + d(t) = \lambda(t),\ 0 \le \sum_{n=1}^{N} R_n(t) \le R_{\max}.$$

Notice that the solution of this problem admits a simple thresholding rule (with shortest queue ties broken arbitrarily):

$$d(t) = \begin{cases} \max\{\lambda(t) - R_{\max},\ 0\}, & \text{if } \exists n \in \mathcal{N} \text{ s.t.} \\ & \quad Q_n(t) \le Vc(t); \\ \lambda(t), & \text{otherwise.} \end{cases} \quad (9)$$

$$R_n(t) = \begin{cases} \min\{\lambda(t), R_{\max}\}, & \text{if } Q_n(t) \text{ is the shortest queue and } Q_n(t) \leq Vc(t); \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

### C. Ratio of expectation minimization on service decisions

Each server makes its own frame-based service decisions. Specifically, for server $n$, at the beginning of its $f$-th renewal frame $t_f^{(n)}$, it observes its current queue state $Q(t_f^{(n)})$ and makes decisions on $\alpha_n[f] \in \{active\} \cup \mathcal{L}_n$ and $I_n[f]$ so as to solve the minimization of ratio of expectations in (12), where $B_0 = \frac{1}{2}(R_{\max} + \mu_{\max})\mu_{\max}$. Recall the definition of $T_n[f]$, if the server chooses to remain active, then the frame length is exactly 1, otherwise, the server is allowed to choose how long it stays in idle. It can be easily shown that over all randomized decisions between staying active and going to different idle states, it is optimal to make a pure decision which either stays active or goes to one of the idle states with probability 1. Thus, we are able to simplify the problem by computing $D_n[f]$ for active and idle options separately.

- If the server chooses to go active, i.e. $\alpha_n[f] = active$, then,
$$D_n[f] = Ve_n - Q_n(t_f^{(n)})\mu_n. \tag{11}$$
- If the server chooses to go idle, i.e. $\alpha_n[f] \in \mathcal{L}_n$, then, $D_n[f]$ is shown in (13), which follows from the fact that if the server goes idle, then, $H_n(t)$ are all zero during the frame except for the last slot. Now we try to compute the optimal idle option $\alpha_n[f] \in \mathcal{L}_n$ and idle time length $I_n[f]$ given the server chooses to go idle. The following lemma illustrates that the decision on $I_n[f]$ can also be reduced to pure decision.

  **Lemma 1.** *The best decision minimizing* (13) *is a pure decision which takes one $\alpha_n[f] \in \mathcal{L}_n$ and one integer value $I_n[f] \in \{1, \cdots, I_{\max}\}$ minimizing the deterministic function* (14).

  The proof of above lemma is given in appendix A.

Then, the server computes the minimum of (14), which is nothing but a deterministic optimization problem. It goes in the following two steps:

1) For each $\alpha_n \in \mathcal{L}_n$, first differentiating (14) with respect to $I[f]$ to get a real minimizer. Then, choosing $I[f]$ as one of the two integer values bracketing the real minimizer which achieves a smaller value on (14).
2) Compare (14) for different $\alpha_n \in \mathcal{L}_n$ and choose the one achieving the minimum.

Thus, the server compares (11) with the minimum of (14). If (11) is less than the minimum of (14), then, the server chooses to go active. Otherwise, the server chooses to go idle and stay idle for $I_n[f]$ time slots.

Above all, our algorithm is summarized in Algorithm 1.

## IV. Performance Analysis

In this section, we prove the online algorithm introduced in the last section makes all request queues $Q_n(t)$ bounded (on the order of $V$) and achieves the near optimality with sub-optimality gap on the order of $1/V$.

**Algorithm 1.**

- *At each time slot $t$, the data center observes $\lambda(t)$, $c(t)$, and $\mathbf{Q}(t)$ chooses rejection decision $d(t)$ according to* (9) *and chooses routing decision $R_n(t)$ according to* (10).
- *For each server $n \in \mathcal{N}$, at the beginning of its $f$-th frame $t_f^{(n)}$, observe its queue state $Q_n(t_f^{(n)})$ and compute* (11) *and the minimum of* (14). *If* (11) *is less than the minimum of* (14)*, then the server still stays active. Otherwise, the server switches to the idle state minimizing* (14) *and stays idle for $I_n[f]$ achieving the minimum of* (14).
- *Update $Q_n(t)$, $\forall n \in \mathcal{N}$ according to* (1).

### A. Bounded request queues

**Lemma 2.** *If $Q_n(0) = 0$, $\forall n \in \mathcal{N}$, then, each request queue $Q_n(t)$ is deterministically bounded with bound: $Q_n(t) \leq Vc_{\max} + R_{\max}$, $\forall t$, $\forall n \in \mathcal{N}$, where $c_{\max} \triangleq \max_{c \in \mathcal{C}} c$.*

*Proof.* We use induction to prove the claim. Base case is trivial since $Q_n(0) = 0 \leq Vc_{\max} + R_{\max}$. Suppose the claim holds at the beginning of $t = i$ for $i > 0$, so that $Q_n(i) \leq Vc_{\max} + R_{\max}$. Then,

1) If $Q_n(i) \leq Vc_{\max}$, then, it is possible for the queue to increase during slot $i$. However, the increase of the queue within one slot is bounded by $R_{\max}$. which implies at the beginning of slot $i+1$, $Q_n(i+1) \leq Vc_{\max} + R_{\max}$.
2) If $Vc_{\max} < Q_n(i) \leq Vc_{\max} + R_{\max}$, then, according to (10), it is impossible to route any request to server $n$ during slot $i$, and $R_n(i) = 0$ which results in $Q_n(i+1) \leq Vc_{\max} + R_{\max}$.

Above all, we finished the proof of lemma. □

**Lemma 3.** *The proposed algorithm meets the constraint* (5) *with probability 1.*

*Proof.* From the queue update rule (1), it follows, $Q_n(t+1) \geq Q_n(t) + R_n(t) - \mu_n H_n(t)$. Taking telescoping sums from 0 to $T-1$ gives $Q_n(T) \geq Q_n(0) + \sum_{t=0}^{T-1} R_n(t) - \sum_{t=0}^{T-1} \mu_n H_n(t)$. Since $Q_n(0) = 0$, dividing both sides by $T$ gives $\frac{Q_n(T)}{T} \geq \frac{1}{T}\sum_{t=0}^{T-1} R_n(t) - \frac{1}{T}\sum_{t=0}^{T-1} \mu_n H_n(t)$. Substitute the bound $Q_n(T) \leq Vc_{\max} + R_{\max}$ from lemma 2 into above inequality and take limit as $T \to \infty$ give the desired result. □

### B. Optimal randomized stationary policy

In this section, we introduce a class of algorithms which are theoretically helpful for doing analysis, but practically impossible to implement.

Since servers are coupled only through time average constraint (5), each server $n$ can be viewed as a separate renewal system, thus, it can be shown that any possible time average service rate $\overline{\mu}_n$ can be achieved through a frame based stationary randomized service decision, meaning that the decisions are i.i.d. over frames (the proof is similar to Lemma 4.2 of [20] and is omitted here for brevity). Furthermore, it can be shown that the optimality of (4)-(6) can be achieved over the following randomized stationary algorithms: At the beginning of each time slot $t$, the data center observes the

$$D_n[f] \triangleq \frac{\mathbb{E}\left[\sum_{t=t_f^{(n)}}^{t=t_{f+1}^{(n)}-1}\left(VW_n(t)+Ve_nH_n(t)+Vg_n(t)-Q_n(t_f^{(n)})\mu_n(t)H_n(t)\right)+\left(t-t_f^{(n)}\right)B_0 \;\middle|\; Q_n(t_f^{(n)})\right]}{\mathbb{E}\left[T_n[f]\;\middle|\;Q_n(t_f^{(n)})\right]} \tag{12}$$

$$D_n[f] = \frac{V\hat{W}_n(\alpha_n[f])m_{\alpha_n[f]}+Ve_n-Q_n(t_f^{(n)})\mu_n+\mathbb{E}\left[V\hat{g}(\alpha_n[f])I_n[f]+\frac{B_0}{2}T_n[f](T_n[f]-1)\;\middle|\;Q_n(t_f^{(n)})\right]}{\mathbb{E}\left[T_n[f]\;\middle|\;Q_n(t_f^{(n)})\right]} \tag{13}$$

$$D_n[f] = \frac{V\hat{W}_n(\alpha_n[f])m_{\alpha_n[f]}+Ve_n-Q_n(t_f^{(n)})\mu_n+\frac{B_0}{2}\sigma^2_{\alpha_n[f]}+V\hat{g}(\alpha_n[f])I_n[f]}{I_n[f]+m_{\alpha_n[f]}+1}+\frac{B_0}{2}(I_n[f]+m_{\alpha_n[f]}+1). \tag{14}$$

incoming requests $\lambda(t)$ and rejecting cost $c(t)$, then routes $R_n^*(t)$ incoming requests to server $n$ and rejects $d^*(t)$ requests, both of which are random functions of $(\lambda(t), c(t))$. They satisfy the same instantaneous relation as (6). Meanwhile, server $n$ chooses a frame based stationary randomized service decision $(\alpha_n^*[f], I_n^*[f])$, so that the optimal service rate is achieved.

If one knows the stationary distribution for $(\lambda(t), c(t))$, then, this optimal control algorithm can be computed using dynamic programming or linear programming. Moreover, the optimal setup cost $W_n^*(t)$, idle cost $g_n^*(t)$, and the active state indicator $H^*(t)$ can also be deduced. Since the algorithm is stationary, these three cost processes are all ergodic Markov processes. Let $T_n^*[f]$ be the frame length process under this algorithm. Thus, it follows from the renewal reward theorem that $\left\{\sum_{t=t_f^{(n)}}^{t_{f+1}^{(n)}-1} W_n^*(t)\right\}_{f=0}^{+\infty}$, $\left\{\sum_{t=t_f^{(n)}}^{t_{f+1}^{(n)}-1} g_n^*(t)\right\}_{f=0}^{+\infty}$, $\left\{\sum_{t=t_f^{(n)}}^{t_{f+1}^{(n)}-1} e_nH_n^*(t)\right\}_{f=0}^{+\infty}$, $\left\{\sum_{t=t_f^{(n)}}^{t_{f+1}^{(n)}-1} \mu_n(t)H_n^*(t)\right\}_{f=0}^{+\infty}$ and $\{T_n^*[f]\}_{f=0}^{+\infty}$ are all *i.i.d. random variables over frames*. Let $\overline{C}^*$, $\overline{W}_n^*$, $\overline{G}_n^*$ and $\overline{E}_n^*$ be the optimal time average costs. Let $\overline{R}_n^*$, $\overline{\mu}_n^*$ and $\overline{d}^*$ be the optimal time average routing rate, service rate and rejection rate respectively. Then, by the strong law of large numbers,

$$\overline{W}_n^* = \frac{\mathbb{E}\left[\sum_{t=t_f^{(n)}}^{t_f^{(n)}+T_n^*[f]-1} W_n^*(t)\right]}{\mathbb{E}\left[T_n^*[f]\right]} \tag{15}$$

$$\overline{E}_n^* = \frac{\mathbb{E}\left[\sum_{t=t_f^{(n)}}^{t_f^{(n)}+T_n^*[f]-1} e_nH_n^*(t)\right]}{\mathbb{E}\left[T_n^*[f]\right]} \tag{16}$$

$$\overline{G}_n^* = \frac{\mathbb{E}\left[\sum_{t=t_f^{(n)}}^{t_f^{(n)}+T_n^*[f]-1} g_n^*(t)\right]}{\mathbb{E}\left[T_n^*[f]\right]} \tag{17}$$

$$\overline{\mu}_n^* = \frac{\mathbb{E}\left[\sum_{t=t_f^{(n)}}^{t_f^{(n)}+T_n^*[f]-1} \mu_n(t)H_n^*(t)\right]}{\mathbb{E}\left[T_n^*[f]\right]}, \tag{18}$$

Also, notice that $R_n^*(t)$ and $d^*(t)$ depend only on the random variables $\lambda(t)$ and $c(t)$, which is i.i.d. over slots. Thus, $R_n^*(t)$ and $d^*(t)$ are also *i.i.d. random variables over slots*. By the law of large numbers,

$$\overline{R}_n^* = \mathbb{E}\left[R_n^*(t)\right], \tag{19}$$

$$\overline{C}^* = \mathbb{E}\left[c(t)d^*(t)\right]. \tag{20}$$

**Remark 1.** *Since the idle time $I_n^*[f] \in [1, I_{\max}]$ and the first two moments of the setup time are bounded, it follows the first two moments of $T_n^*[f]$ are bounded.*

### C. Key features of thresholding algorithm

In this part, we compare the algorithm deduced from the two optimization problems (8) and (12) to that of the best stationary algorithm in section IV-B, illustrating the key features of the proposed online algorithm. Define $\mathcal{F}(t)$ as the system history up till slot $t$, which includes all the decisions taken and all the random events before slot $t$. We first consider (8). For simplicity of notations, define two random processes $\{X_n[f]\}_{f=0}^{\infty}$ and $\{Z[t]\}_{t=0}^{\infty}$ as follows

$$\begin{aligned} X_n[f] = \sum_{t=t_f^{(n)}}^{t=t_{f+1}^{(n)}-1} &\left(V\left(W_n(t)-\overline{W}_n^*\right)+V\left(e_nH_n(t)-\overline{E}_n^*\right)\right. \\ &+V\left(g_n(t)-\overline{G}_n^*\right)-Q_n(t_f^{(n)})\left(\mu_nH_n(t)-\overline{\mu}^*\right) \\ &\left.+\left(t-t_f^{(n)}\right)B_0-\Psi_n\right), \end{aligned}$$

$$Z[t] = V\left(c(t)d(t)-\overline{C}^*\right)+\sum_{n=1}^{N} Q_n(t)\left(R_n(t)-\overline{R}_n^*\right),$$

where $\Psi_n = \frac{B_0}{2}\frac{\mathbb{E}[T_n^*[f](T_n^*[f]-1)]}{\mathbb{E}[T_n^*[f]]}$ and $B_0 = \frac{1}{2}(R_{\max}+\mu_{\max})\mu_{\max}$.

Given the system information $\mathcal{F}(t)$, the random events $c(t)$ and $\lambda(t)$, the solutions (9) and (10) take rejecting and routing decisions so as to minimize (8) over all possible routing and rejecting decisions at time slot $t$. Thus, the proposed algorithm achieves smaller value on (8) compared to that of the best stationary algorithm in section IV-B. Formally, this idea can be stated as the following inequality: $\mathbb{E}\left[Vc(t)d(t)+\sum_{n=1}^{N}Q_n(t)R_n(t)\;\middle|\;c(t),\lambda(t),\mathcal{F}(t)\right] \leq \mathbb{E}\left[Vc(t)d^*(t)+\sum_{n=1}^{N}Q_n(t)R_n^*(t)\;\middle|\;c(t),\lambda(t),\mathcal{F}(t)\right]$. Taking expectation regarding $c(t)$ and $\lambda(t)$ using the fact that the best stationary algorithm on $R_n^*(t)$ and $d^*(t)$ are i.i.d. over slots (independent of $\mathcal{F}(t)$), together with (19) and (20), we get

$$\mathbb{E}\left[Z(t) \mid \mathcal{F}(t)\right] \leq 0. \tag{21}$$

Similarly, for (12), the proposed service decisions within frame $f$ minimize $D_n[f]$ in (12), thus, compared to the best stationary policy, the inequality (22) holds. Again, using the fact that the optimal stationary algorithm gives i.i.d. $W_n^*(t)$, $g_n^*(t)$, $H_n^*(t)$ and $T_n^*[f]$ over frames (independent of $\mathcal{F}(t_f^{(n)})$), as well as (15), (16) and (18), we get

$$\mathbb{E}\left[X_n[f] \;\middle|\; \mathcal{F}(t_f^{(n)})\right] \Big/ \mathbb{E}\left[T_n[f] \;\middle|\; \mathcal{F}(t_f^{(n)})\right] \leq 0 \tag{23}$$

### *D. Bounded average of supermartingale difference sequeces*

The key feature inequalities (21) and (23) provide us with bounds on the expectations. The following lemma serves as a stepping stone passing from expectation bounds to probability 1 bounds. We need the following basic definition of supermartingale to start with:

**Definition 1** (Supermartingale)**.** *Let $\{\mathcal{F}_t\}_{t=0}^{\infty}$ be a filtration, i.e. an increasing sequence of $\sigma$-fields. Let $\{Y_t\}_{t=0}^{\infty}$ be a sequence of random variables. Then, $\{Y_t\}_{t=0}^{\infty}$ is said to be a supermartingale if (i) $\mathbb{E}\left[|Y_t|\right] < \infty,\ \forall t$, (ii) $Y_t$ is measurable with respect to $\mathcal{F}_t$ for all $t$, (iii) $\mathbb{E}\left[Y_{t+1}|\mathcal{F}_t\right] \leq Y_t,\ \forall t$. The corresponding difference process $X_t = Y_{t+1} - Y_t$ is called the supermartingale difference sequence.*

We also need the following strong law of large numbers for supermartingale difference sequences:

**Lemma 4** (Corollary 4.2 of [21])**.** *Let $\{X_t\}_{t=0}^{\infty}$ be a supermartingale difference sequence. If*

$$\sum_{t=1}^{\infty} \mathbb{E}\left[X_t^2\right] \big/ t^2 < \infty,$$

*then,*

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} X_t \leq 0,$$

*with probability 1.*

With this lemma, we are ready to prove the following result:

**Lemma 5.** *Under the proposed algorithm, the following hold with probability 1,*

$$\limsup_{F\to\infty} \frac{1}{F} \sum_{f=0}^{F-1} X_n[f] \leq 0, \tag{24}$$

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} Z[t] \leq 0. \tag{25}$$

*Proof.* The key to the proof is treating these two sequences as supermartingale difference sequences and applying law of large numbers for supermartingale difference sequences (theorem 4.1 and corollary 4.2 in (24)).

We first look at the sequence $\{X_n[f]\}_{f=0}^{\infty}$. Let $Y_n[F] = \sum_{f=0}^{F-1} X_n[f]$. We first prove that $Y_n[F]$ is a supermartingale. Notice that $Y_n[F] \in \mathcal{F}\left(t_n^{(F)}\right)$, i.e. it is measurable given all the information before frame $t_n^{(F)}$, and $|Y_n[F]| < \infty,\ \forall F < \infty$. Furthermore, $\mathbb{E}\left[Y_n[F+1] - Y_n[F] \;\middle|\; \mathcal{F}\left(t_F^{(n)}\right)\right] = \mathbb{E}\left[X_n[F] \;\middle|\; \mathcal{F}\left(t_F^{(n)}\right)\right] \leq 0 \cdot \mathbb{E}\left[T_n[F] \;\middle|\; \mathcal{F}\left(t_F^{(n)}\right)\right] = 0$, where the only inequality follows from (23). Thus, it follows $Y_n[F]$ is a supermartingale. Next, we show that the second moment of supermartingale differences, i.e. $\mathbb{E}\left[X_n[f]^2\right]$, is deterministically bounded by a fixed constant for any $f$. This part of proof is given in Appendix B. Thus, the following holds: $\sum_{f=1}^{\infty} \mathbb{E}\left[X_n[f]^2\right] \big/ f^2 < \infty$. Now, applying Lemma 4 immediately gives (24).

Similarly, we can prove (25) by proving $M[t] = \sum_{t=0}^{T-1} Z[t]$ is a supermartingale with bounded second moment on differences using (19), (20) and (21). The procedure is almost the same as above and we omitted the details here for brevity. □

**Corollary 1.** *The following ratio of time averages is upper bounded with probability 1,*
$\limsup_{F\to\infty} \sum_{f=0}^{F-1} X_n[f] \Big/ \sum_{f=0}^{F-1} T_n[f] \leq 0.$

*Proof.* From (24), it follows for any $\epsilon > 0$, there exists an $F_0(\epsilon)$ such that $F \geq F_0(\epsilon)$ implies $\sum_{f=0}^{F-1} X_n[f] \Big/ \sum_{f=0}^{F-1} T_n[f] \leq \epsilon \Big/ \frac{1}{F}\sum_{f=0}^{F-1} T_n[f] \leq \epsilon$. Thus, $\limsup_{F\to\infty} \sum_{f=0}^{F-1} X_n[f] \Big/ \sum_{f=0}^{F-1} T_n[f] \leq \epsilon$. Since $\epsilon$ is arbitrary, take $\epsilon \to 0$ gives the result. □

### *E. Near optimal time average cost*

The ratio of time averages in corollary 1 and the true time average share the same bound, which is proved by the following lemma:

**Lemma 6.** *The following time average is bounded,*

$$\begin{aligned}
&\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \Big( V\left(W_n(t) + e_n H_n(t) + g_n(t)\right) - Q_n(t_f^{(n)}) \\
&(\mu_n H_n(t) - \overline{\mu}_n^*) + \left(t - t_f^{(n)}\right) B_0 \Big) \\
&\leq V(\overline{W}_n^* + \overline{E}_n^* + \overline{G}_n^*) + \Psi_n,
\end{aligned} \tag{26}$$

*where $\Psi_n = \frac{B_0}{2} \frac{\mathbb{E}[T_n^*[f](T_n^*[f]-1)]}{\mathbb{E}[T_n^*[f]]}$ and $B_0 = \frac{1}{2}(R_{\max} + \mu_{\max})\mu_{\max}$.*

The idea of the proof is similar to that of basic renewal theory, which derives upper and lower bounds for each $T$ within any frame $F$ using corollary 1, thereby showing that as $T \to \infty$, the upper and lower bounds meet. See appendix C for details. With the help of this lemma, we are able to prove the following near optimal performance theorem:

**Theorem 1.** *If $Q_n(0) = 0, \forall n \in \mathcal{N}$, then the time average total cost under the algorithm is near optimal on the order of $\mathcal{O}(1/V)$, i.e.*

$$\begin{aligned}
&\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \left( c(t)d(t) + \sum_{n=1}^{N} \left(W_n(t) + e_n H_n(t) + g_n(t)\right) \right) \\
&\leq \underbrace{\overline{C}^* + \sum_{n=1}^{N} \left(\overline{W}_n^* + \overline{E}_n^* + \overline{G}_n^*\right)}_{\textit{Optimal cost}} + \frac{\sum_{n=1}^{N} \Psi_n + B_3}{V},
\end{aligned} \tag{27}$$

$$\frac{\mathbb{E}\left[\sum_{t=t_f^{(n)}}^{t=t_{f+1}^{(n)}-1}\left(V(W_n(t)+e_nH_n(t)+g_n(t))-Q_n(t_f^{(n)})\mu_n(t)H_n(t)\right)+\left(t-t_f^{(n)}\right)B_0\;\middle|\;\mathcal{F}(t_f^{(n)})\right]}{\mathbb{E}\left[T_n[f]\;\middle|\;\mathcal{F}(t_f^{(n)})\right]}$$
$$\leq\frac{\mathbb{E}\left[\sum_{t=t_f^{(n)}}^{t=t_f^{(n)}+T_n^*[f]-1}\left(V\left(W_n^*(t)+e_nH_n^*(t)+g_n^*(t)\right)-Q_n(t_f^{(n)})\mu_nH_n^*(t)\right)+\frac{B_0}{2}T_n^*[f](T_n^*[f]-1)\;\middle|\;\mathcal{F}(t_f^{(n)})\right]}{\mathbb{E}\left[T_n^*[f]\;\middle|\;\mathcal{F}(t_f^{(n)})\right]} \tag{22}$$

*where* $B_3 \triangleq \frac{1}{2}\sum_{n=1}^N(R_{\max}+\mu_n)^2$, $\Psi_n = \frac{B_0}{2}\frac{\mathbb{E}[T_n^*[f](T_n^*[f]-1)]}{\mathbb{E}[T_n^*[f]]}$ *and* $B_0 = \frac{1}{2}(R_{\max}+\mu_{\max})\mu_{\max}$.

The proof uses Lyapunov drift analysis. See appendix D for details of proof.

## V. Delay improvement via virtualization

### A. Delay improvement

The algorithm in previous sections optimizes time average cost. However, it can route requests to idle queues, which increases system delay. This section considers an improvement in the algorithm that maintains the same average cost guarantees, but reduces delay. This is done by a "virtualization" technique that reduces from $N$ server request queues to only one request queue $Q(t)$. Specifically, the same Algorithm 1 is run, with queue updates (1) for each of the $N$ queues $Q_n(t)$. However, the $Q_n(t)$ processes are now virtual queues rather than actual queues: Their values are only kept in software. Every slot $t$, the data center observes the incoming requests $\lambda(t)$, rejection cost $c(t)$ and virtual queue values, making rejection decision according to (9) as before. The admitted requests are queued in $Q(t)$. Meanwhile, each server $n$ makes active/idle decisions observing its own virtual queue $Q_n(t)$ same as before. Whenever a server is active, it grabs the requests from request queue $Q(t)$ and serves them. This results in an actual queue updating for the system:

$$Q(t+1) = \max\left\{Q(t)+\lambda(t)-d(t)-\sum_{n=1}^N\mu_n(t)H_n(t),\ 0\right\}. \tag{28}$$

Fig. 4 shows this data center architecture.

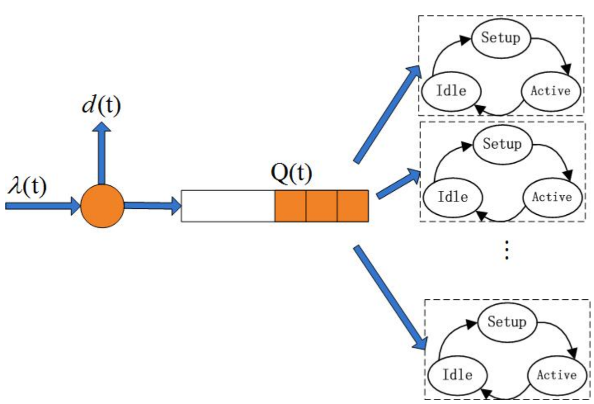


Fig. 4. Illustration of basic data center architecture.

### B. Performance guarantee

Since this algorithm does not look at the actual queue $Q(t)$, it is not clear whether or not the actual request queue would be stabilized under the proposed algorithm. The following lemma answers the question. For simplicity, we call the system with $N$ queues, where our algorithm applies, the virtual system, and call the system with only one queue the actual system.

**Lemma 7.** *If* $Q(0) = 0$ *and* $Q_n(0) = 0$, $\forall n \in \mathcal{N}$, *then the virtualization technique stabilizes the queue* $Q(t)$ *with the bound:* $Q(t) \leq N(Vc_{\max}+R_{\max})$.

*Proof.* Notice that this bound is $N$ times the individual queue bound in lemma 2, we prove the lemma by showing that the sum-up weights $\sum_{n=1}^N Q_n(t)$ in the virtual system always dominates the queue length $Q(t)$. We prove this by induction. The base case is obvious since $Q(0) = \sum_{n=1}^N Q_n(0) = 0$. Suppose at the beginning of time $t$, $Q(t) \leq \sum_{n=1}^N Q_n(t)$, then, during time $t$, we distinguish between the following two cases:

1) Not all active servers in actual system have requests to serve. This case happens if and only if there are not enough requests in $Q(t)$ to be served, i.e. $\lambda(t)-d(t)+Q(t) < \sum_{n=1}^N \mu_n(t)H_n(t)$. Thus, according to queue updating rule (28), at the beginning of time slot $t+1$, there will be no request sitting in the actual queue, i.e. $Q(t+1) = 0$. Hence, it is guaranteed that $Q(t+1) \leq \sum_{n=1}^N Q_n(t+1)$.
2) All active servers in actual system have requests to serve. Notice that the virtual system and the actual system have exactly the same arrivals, rejections and server active/idle states. Thus, the following holds, $Q(t+1) = Q(t)+\lambda(t)-d(t)-\sum_{n=1}^N\mu_n(t)H_n(t) \leq \sum_{n=1}^N Q_n(t)+\sum_{n=1}^N R_n(t)-\sum_{n=1}^N\mu_n(t)H_n(t) \leq \sum_{n=1}^N \max\{Q_n(t)+R_n(t)-\mu_n(t)H_n(t),\ 0\} = \sum_{n=1}^N Q_n(t+1)$, where the first inequality follows from induction hypothesis as well as the fact that $\sum_{n=1}^N R_n(t) = \lambda(t)-d(t)$.

Above all, we proved $Q(t) \leq \sum_{n=1}^N Q_n(t),\ \forall t$. Since each $Q_n(t) \leq Vc_{\max}+R_{\max},\ \forall t$, the lemma follows. □

Since the virtual system and the actual system have exactly the same cost, and it can be shown that the optimal cost in one queue system is lower bounded by the optimal cost in $N$ queue system, thus, the near optimal performance is still guaranteed.

TABLE II
PROBLEM PARAMETERS

| Server | $\mu_n$ | $e_n$ | $\hat{W}_n(\alpha_n)$ | $\mathbb{E}[\hat{\tau}(\alpha_n)]$ |
|---|---|---|---|---|
| 1 | $\{2,3,4,5,6\}$ | 4 | 2 | 5.893 |
| 2 | $\{2,3,4\}$ | 2 | 3 | 4.342 |
| 3 | $\{2,3,4\}$ | 3 | 3 | 27.397 |
| 4 | $\{1,2,3\}$ | 4 | 2 | 5.817 |
| 5 | $\{2,3,4\}$ | 2 | 4 | 6.211 |

## VI. SIMULATION

In this section, we demonstrate the performance of our proposed algorithm via extensive simulations. The first simulation runs over i.i.d. traffic which fits into the assumption made in Section II. We show that our algorithm indeed achieves $\mathcal{O}(1/V)$ near optimality with $\mathcal{O}(V)$ delay ($[\mathcal{O}(1/V), \mathcal{O}(V)]$ trade-off), which is predicted by Lemma 2 and Theorem 1. We then apply our algorithm to a real data center traffic trace with realistic scale, setup time and cost being the power consumption. We compare the performance of the proposed algorithm with several other heuristic algorithms and show that our algorithm indeed delivers lower delay and saves power.

### A. Near optimality in $N$ queues system

In the first simulation, we consider a relative small scale problem with i.i.d. generated traffic. We set the number of servers $N = 5$. The incoming requests $\lambda(t)$ are integers following a uniform distribution in $[10, 30]$. The request rejecting cost $c(t)$ are also integers following a uniform distribution in $[1, 6]$. The maximum admission amount $R_{\max} = 40$ and the maximum idle time $I_{\max} = 1000$. There is only one idle option $\alpha_n$ for each server where the idle cost $\hat{g}(\alpha_n) = 0$. The setup time follows a geometric distribution with mean $\mathbb{E}[\hat{\tau}(\alpha_n)]$, setup cost $\hat{W}_n(\alpha_n)$ per slot, service cost $e_n$ per slot, and the service amount $\mu_n$ follows a uniform distribution over integers. The values $1/\mathbb{E}[\hat{\tau}(\alpha_n)]$ are generated uniform at random within $[0, 1]$ and specified in table II.

The algorithm is run for 1 million slots in each trial and each plot takes the average of these 1 million slots. We compare our algorithm to the optimal stationary algorithm. The optimal stationary algorithm is computed using linear program [23] with the full knowledge of the statistics of requests and rejecting costs.

In Fig. 5, we show that as our tradeoff parameter $V$ gets larger, the average cost approaches the optimal value and achieves a near optimal performance. Furthermore, the cost curve drops rapidly when $V$ is small and becomes relatively flat when $V$ gets large, thereby demonstrating our $\mathcal{O}(1/V)$ optimality gap in Theorem 1. Fig. 6 plots the average sum-up queue size $\sum_{n=1}^{5} Q_n(t)$ and shows as $V$ gets larger, the average sum-up queue size becomes larger. We also plot the sum of individual queue bound from Lemma 2 for comparison. We can see that the real queue size grows linearly with $V$ (although the constant in Lemma 2 is not tight due to the much better delay we obtain here), which demonstrates the $\mathcal{O}(V)$ delay bound.

We then tune the requests $\lambda(t)$ to be uniform in $[20, 40]$ and keep other parameters unchanged. In Fig. 7, we see that since the request rate gets larger, we need $V$ to be larger in order to obtain the near optimality, but still, the near optimality gap scales roughly $\mathcal{O}(1/V)$. Fig. 8 gives the sum-up average queue length in this case. The average queue length is larger than that of Fig. 6 with linear growth with respect to $V$.

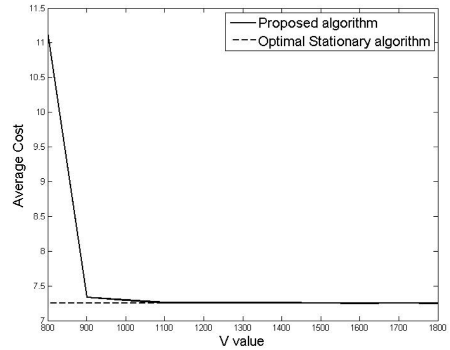


Fig. 5. Time average cost verses $V$ parameter over 1 millon slots.

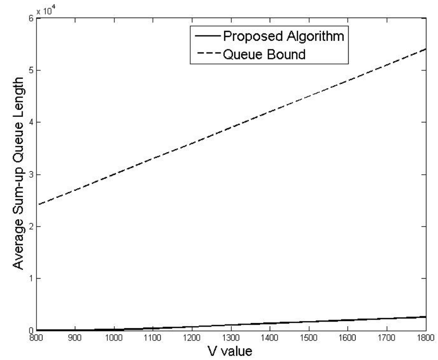


Fig. 6. Time average sum-up request queue length verses $V$ parameter over 1 millon slots.

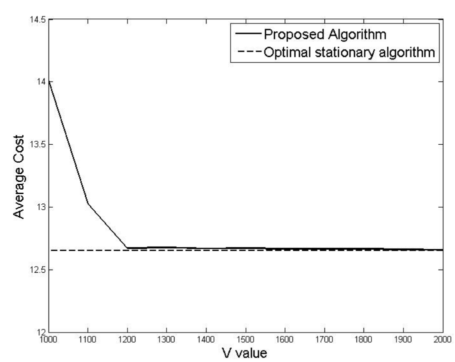


Fig. 7. Time average cost verses $V$ parameter over 1 millon slots.

### B. Real data center traffic trace and performance evaluation

This second considers a simulation on a real data center traffic obtained from the open source data sets([25]) of the paper [24]. The trace is plotted in Fig. 9. We synthesize different data chunks from the source so that the trace contains

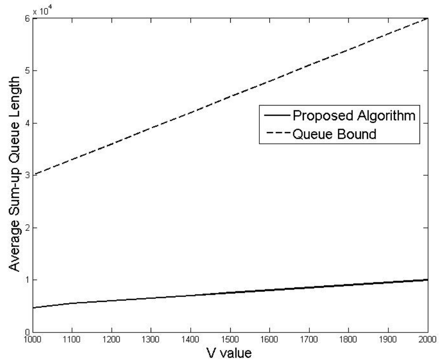


Fig. 8. Time average cost verses $V$ parameter over 1 millon slots.

both the steady phase and increasing phase. The total time duration is 2800 seconds with each slot equal to 20ms. The peak traffic is 2120 requests per 20 ms, and the time average traffic over this whole time interval is 654 requests per 20 ms.

We consider a data center consisting of 1060 homogeneous servers. We assume each server has only one sleep state and the service quantity of each server at each slot follows a Zipf's law[1] with parameter $K = 10$ and $p = 1.9$. This gives the service rate of each server equal to $1.9933 \approx 2$ requests per 20ms. So the full capacity of the data center is able to support the peak traffic. Zipf's law is previously introduced to model a wide scope of physics, biology, computer science and social science phenomenon ([26]), and is adopted in various literatures to simulate the empirical data center service rate ([12], [19]). The setup time of each server is geometrically distributed with success probability equal to $0.001$. This gives the mean setup time 1000 slots (20 seconds). This setup time is previously shown in [19] to be a typical time duration for a desktop to recover from the suspend or hibernate state.

Furthermore, to make a fair comparison with several existing algorithms, we enforce the front end balancer to accept all requests at each time slot (so the rejection rate is always 0). The only cost in the system is then the power consumption. We assume that a server consumes 10 W each slot when active and 0 W each slot when idle. The setup cost is also 10 W per slot. Moreover, we apply the one queue model described in Section V for all the rest of the simulations. Following the problem formulation, the maximum idle time of a server for the proposed algorithm is $I_{\max} = 5000$, while no such limit is imposed for any other benchmark algorithms.

We first run our proposed algorithm over the trace with virtualization (in Section V) for different $V$ values. We set the initial virtual queue backlog $Q_n(0) = 2000\ \forall n$, and keep 20 servers always on. Fig. 10 and Fig. 11 plots the running average power consumption and corresponding queue length for $V = 400,\ 600,\ 800$ and $1200$, respectively. It can be seen that as $V$ gets large, the average power consumption does not improve too much but the queue length changes drastically.

[1]The pdf of Zipf's law with parameter $K, p$ is defined as: $f(n; K, p) = \frac{1/n^p}{\sum_{i=1}^{K} 1/i^p}$, $n = 1, 2, \cdots, K$. Thus, the mean of the distribution is $\frac{\sum_{i=1}^{K} 1/i^{p-1}}{\sum_{i=1}^{K} 1/i^p}$.

This phenomenon results from the $[\mathcal{O}(1/V), \mathcal{O}(V)]$ trade-off of our proposed algorithm. In view of this fact, we choose $V = 600$ which gives a reasonable delay performance in Fig. 11.

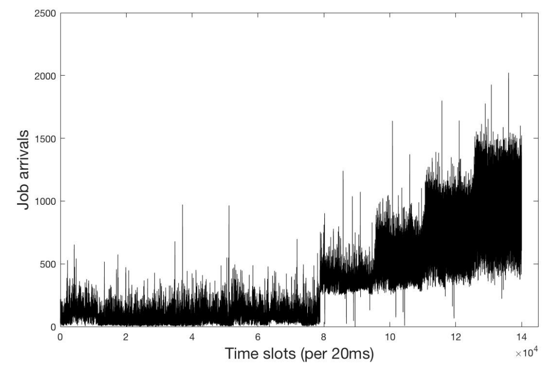


Fig. 9. Synthesized traffic trace from [25].

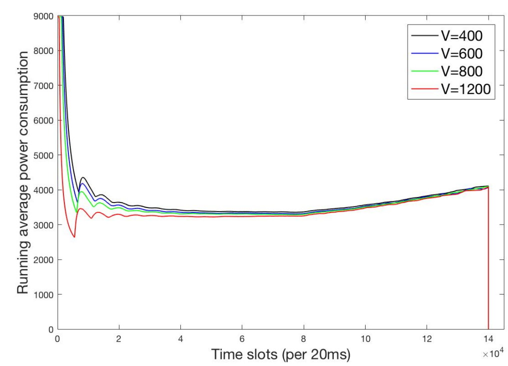


Fig. 10. Running average power consumption from slot 1 to the current slot for different $V$ value.

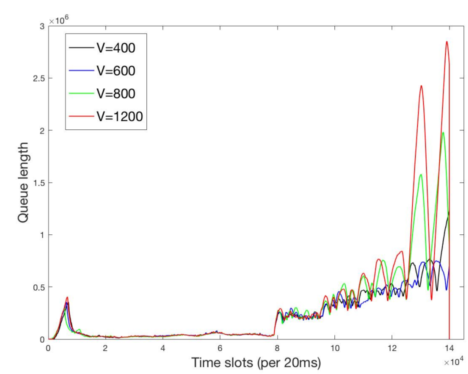


Fig. 11. Instantaneous queue length for different $V$ value.

Next, we compare our proposed algorithm with the same initial setup and $V = 600$ to the following algorithms:

- Always-on with $N = 327$ active servers and the rest servers staying on the sleep mode. Note that 327 servers can support the average traffic over the whole interval which is 654 requests per 20 ms.

- Always-on with full capacity. This corresponds to keeping all 1060 servers on at every slot.
- Reactive. This algorithm is developed in [19] which reacts to the current traffic $\overline{\lambda}(t)$ and maintains $k_{react}(t) = \lceil \overline{\lambda}(t)/2 \rceil$ servers on. In the simulation, we choose $\overline{\lambda}(t)$ to be the average of the traffic from the latest 10 slots. If the current active server $k(t) > k_{react}(t)$, then, we turn $k(t) - k_{react}(t)$ servers off, otherwise, we turn $k_{react}(t) - k(t)$ servers to the setup state.
- Reactive with extra capacity. This algorithm is similar to Reactive except that we introduce a virtual traffic flow of $p$ jobs per slot. So during each time slot $t$, the algorithm maintains $k_{react}(t) = \lceil (\overline{\lambda}(t) + p)/2 \rceil$ servers on.

Fig. 12-14 plots the average power consumption, queue length and the number of active servers, respectively. It can be seen that all algorithms perform pretty well during first half of the trace. For the second half of the trace, the traffic load is increasing. The Always-on algorithm with mean capacity does not adapt to the traffic so the queue length blows up quickly. Because of the long setup time, the number of active servers in the Reactive algorithm fails to catch up with the increasing traffic so the queue length also blows up. Our proposed algorithm minimizes the power consumption while stabilizing the queues, thereby outperforming both the Always-on and the Reactive algorithm. Note that the Reactive with extra 200 job capacity is able to achieve a similar delay performance as our proposed algorithm, but with significant extra power consumption.

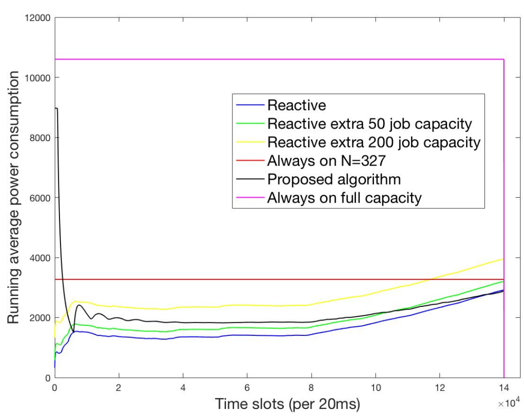


Fig. 12. Running average power consumption from slot 1 to the current slot for different algorithms.

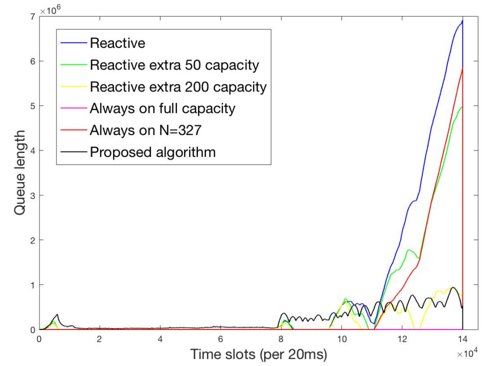


Fig. 13. Instantaneous queue length for different algorithms.

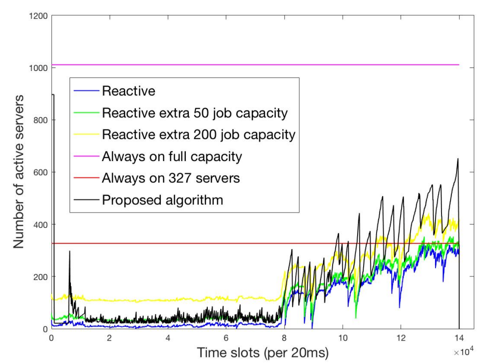


Fig. 14. Number of active servers over time.

Finally, we evaluate the influence of different sleep modes on the performance. We keep all the setups the same as before and consider the sleep modes with sleep power consumption equal to 2 W and 4 W per slot, respectively. Since the Always-on and the Reactive algorithm do not look at the sleep power consumption, their decisions remain the same as before, thus, we superpose the queue length of our proposed algorithm onto the previous Fig. 13 and get the queue length comparison in Fig. 15. We see from the plot that increasing the power consumption during the sleep mode only slightly increases the queue length of our proposed algorithm. Fig. 16 plots the running average power consumption under different sleep modes. Despite spending more power on the sleep mode, the proposed algorithm can still save considerable amount of power compared to other algorithms while keeping the request queue stable. This shows that our algorithm is empirically robust to the change of sleep mode.

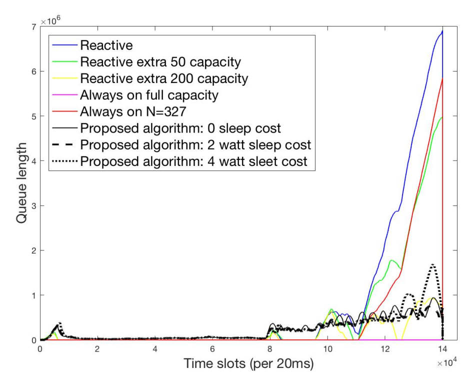


Fig. 15. Instantaneous queue length for different algorithms.

## VII. Conclusions

This paper proposes an efficient distributed asynchronous control algorithm reducing the cost in a data center, where the front-end load balancer makes slot-wise routing requests to the shortest queue and each server makes a frame-based

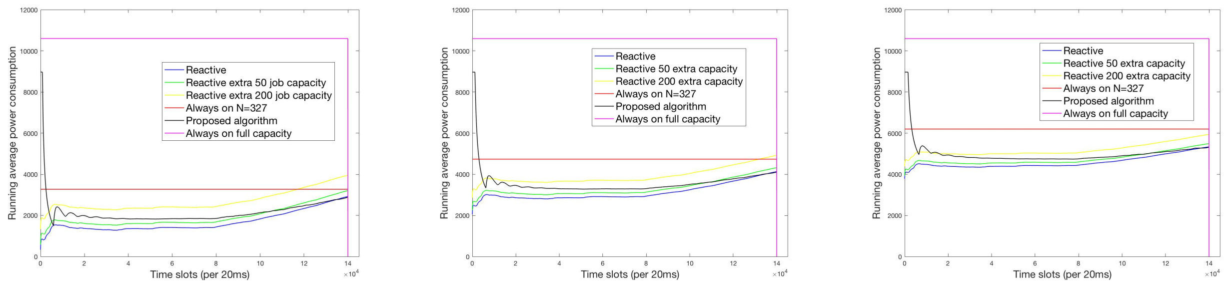


Fig. 16. Running average power consumption for 0 W sleep cost(left), 2 W sleep cost(middle) and 4 W sleep cost(right)

service decision by only looking at its own request queue. Theoretically, this algorithm is shown to achieve the near optimal cost while stabilizing the request queues. Simulation experiments on a real data center traffic trace demonstrate that our algorithm outperforms several other algorithms in reducing the power consumption as well as achieving lower delays.

## APPENDIX A— PROOF OF LEMMA 1

We have (29), as shown at the bottom of this page, holds, where the first equality follows from the definition $T_n[f] = I_n[f]+\tau_n[f]+1$ and the second equality follows from iterated expectations conditioning on $I_n[f]$ and $\alpha_n[f]$. For simplicity of notations, let

$$\begin{aligned}F(\alpha_n[f], I_n[f]) =& V\hat{W}_n(\alpha_n[f])m_{\alpha_n[f]} + Ve_n - Q_n(t_f^{(n)})\mu_n\\&+ \frac{B_0}{2}(I_n[f] + m_{\alpha_n[f]} + 1)^2 + V\hat{g}(\alpha_n[f])I_n[f]\\&+ \frac{B_0}{2}\sigma^2_{\alpha_n[f]}\\G(\alpha_n[f], I_n[f]) =& I_n[f] + m_{\alpha_n[f]} + 1,\end{aligned}$$

then

$$D_n[f] = \frac{\mathbb{E}\left[F(\alpha_n[f], I_n[f]) \mid Q_n(t_f^{(n)})\right]}{\mathbb{E}\left[G(\alpha_n[f], I_n[f]) \mid Q_n(t_f^{(n)})\right]} - \frac{B_0}{2}.$$

Meanwhile, given the queue length $Q_n(t_f^{(n)})$ at frame $f$, denote the benchmark solution over pure decisions as

$$m \triangleq \min_{I_n[f]\in\mathbb{N},\ I_n[f]\in[1,I_{\max}],\alpha_n[f]\in\mathcal{L}_n} \frac{F(\alpha_n[f], I_n[f])}{G(\alpha_n[f], I_n[f])}. \quad (30)$$

Then, for any randomized decision on $\alpha_n[f]$ and $I_n[f]$, its realization within frame $f$ satisfies the following

$$\frac{F(\alpha_n[f], I_n[f])}{G(\alpha_n[f], I_n[f])} \geq m,$$

which implies

$$F(\alpha_n[f], I_n[f]) \geq mG(\alpha_n[f], I_n[f]).$$

Taking conditional expectation from both sides gives

$$\begin{aligned}&\mathbb{E}\left[F(\alpha_n[f], I_n[f]) \mid Q_n(t_f^{(n)})\right]\\&\geq m\mathbb{E}\left[G(\alpha_n[f], I_n[f]) \mid Q_n(t_f^{(n)})\right]\\&\Rightarrow \frac{\mathbb{E}\left[F(\alpha_n[f], I_n[f]) \mid Q_n(t_f^{(n)})\right]}{\mathbb{E}\left[G(\alpha_n[f], I_n[f]) \mid Q_n(t_f^{(n)})\right]} \geq m.\end{aligned}$$

Thus, it is enough to consider pure decisions only, which boils down to computing (30). This proves the lemma.

## APPENDIX B— PROOF OF LEMMA 5

This section is dedicated to prove that $\mathbb{E}\left[X_n[f]^2\right]$ is bounded. First of all, since the idle option set $\mathcal{L}_n$ is finite, denote

$$\begin{aligned}W_{\max} &= \max_{\alpha_n\in\mathcal{L}_n} W_n(\alpha_n)\\g_{\max} &= \max_{\alpha_n\in\mathcal{L}_n} g_n(\alpha_n)\end{aligned}$$

It is obvious that $|W_n(t)-\overline{W}_n^*| \leq W_{\max}$, $|g_n(t)-\overline{G}_n^*| \leq g_{\max}$, $|e_nH_n(t) - \overline{E}_n^*| \leq e_n$, and $|\mu_nH_n(t) - \overline{\mu}^*| \leq \mu_n$. Combining with the boundedness of queues in lemma 2, it follows

$$\begin{aligned}|X_n[f]| \leq& \sum_{t=t_f^{(n)}}^{t=t_{f+1}^{(n)}-1} \left(V\left(W_{\max} + e_n + g_{\max}\right) + \left(Vc_{\max}\right.\right.\\&\left.\left.+R_{\max}\right)\mu_n + \left(t - t_f^{(n)}\right)B_0 + \Psi_n\right)\\\leq&\left(V(W_{\max} + e_n + g_{\max}) + (Vc_{\max} + R_{\max})\mu_n\right.\\&\left.+\Psi_n\right)T_n[f] + \frac{T_n[f](T_n[f]-1)B_0}{2}\end{aligned}$$

Let $B_1 \triangleq V(W_{\max}+e_n+g_{\max})+(Vc_{\max}+R_{\max})\mu_n+\Psi_n+B_0/2$, it follows

$$|X_n[f]| \leq B_1T_n[f] + \frac{B_0}{2}T_n[f]^2.$$

Thus,

$$\mathbb{E}\left[X_n[f]^2\right] \leq B_1^2\mathbb{E}\left[T_n[f]^2\right]+B_1B_0\mathbb{E}\left[T_n[f]^3\right]+\frac{B_0^2}{4}\mathbb{E}\left[T_n[f]^4\right].$$

Notice that $T_n[f] \leq I_n[f] + \tau_n[f] + 1$ by (2), where $I_n[f]$ is upper bonded by $I_{\max}$ and $\tau_n[f]$ has first four moments bounded by assumption 2. Thus, $\mathbb{E}\left[X_n[f]^2\right]$ is bounded by a fixed constant.

## APPENDIX C— PROOF OF LEMMA 6

*Proof.* Let's first abbreviate the notation by defining

$$\begin{aligned}Y(t) =& V(W_n(t) + e_nH_n(t) + g_n(t)) - Q_n(t_f^{(n)})(\mu_nH_n(t) - \overline{\mu}_n^*)\\&+\left(t - t_f^{(n)}\right)B_0.\end{aligned}$$

$$
\begin{aligned}
D_n[f] &= \frac{V\hat{W}_n(\alpha_n[f])m_{\alpha_n[f]} + Ve_n - Q_n(t_f^{(n)})\mu_n + \mathbb{E}\left[\frac{B_0}{2}(I_n[f]+\tau_n[f]+1)^2 + V\hat{g}(\alpha_n[f])I_n[f] \,\middle|\, Q_n(t_f^{(n)})\right]}{\mathbb{E}\left[I_n[f]+\tau_n[f]+1 \,\middle|\, Q_n(t_f^{(n)})\right]} - \frac{B_0}{2} \\
&= \frac{V\hat{W}_n(\alpha_n[f])m_{\alpha_n[f]} + Ve_n - Q_n(t_f^{(n)})\mu_n + \mathbb{E}\left[\frac{B_0}{2}(I_n[f]+m_{\alpha_n}+1)^2 + \frac{B_0}{2}\sigma^2_{\alpha_n[f]} + V\hat{g}(\alpha_n[f])I_n[f] \,\middle|\, Q_n(t_f^{(n)})\right]}{\mathbb{E}\left[I_n[f]+m_{\alpha_n}+1 \,\middle|\, Q_n(t_f^{(n)})\right]} - \frac{B_0}{2}
\end{aligned} \tag{29}
$$

For any $T \in [t_F^{(n)},\ t_F^{(n+1)})$, we can bound the partial sums from above by the following

$$\sum_{t=0}^{T-1} Y(t) \le \sum_{t=0}^{t_F^{(n)}-1} Y(t) + B_2 T_n[F] + \frac{B_0}{2} T_n[F]^2,$$

where $B_0 = \frac{1}{2}(R_{\max} + \mu_{\max})\mu_{\max}$ is defined in (12), and $B_2 \triangleq VW_n + V\mu_n e_n + (Vc_{\max} + R_{\max})\mu_n + B_0/2$. Thus,

$$
\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1} Y(t) &\le \frac{1}{T}\sum_{t=0}^{t_F^{(n)}-1} Y(t) + \frac{1}{T}\left(B_2 T_n[F] + \frac{B_0}{2}T_n[F]^2\right) \\
&\le \max\{a[F],\ b[F]\},
\end{aligned}
$$

where

$$
\begin{aligned}
a[F] &\triangleq \frac{1}{t_F^{(n)}} \sum_{t=0}^{t_F^{(n)}-1} Y(t) + \frac{1}{t_F^{(n)}}\left(B_2 T_n[F] + \frac{B_0}{2}T_n[F]^2\right), \\
b[F] &\triangleq \frac{1}{t_{F+1}^{(n)}} \sum_{t=0}^{t_F^{(n)}-1} Y(t) + \frac{1}{t_{F+1}^{(n)}}\left(B_2 T_n[F] + \frac{B_0}{2}T_n[F]^2\right).
\end{aligned}
$$

Thus, this implies that

$$
\begin{aligned}
\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} Y(t) &\le \limsup_{F\to\infty} \max\{a[F],\ b[F]\} \\
&= \max\left\{\limsup_{F\to\infty} a[F],\ \ \limsup_{F\to\infty} b[F]\right\}.
\end{aligned}
$$

We then try to work out an upper bound for $\limsup_{F\to\infty} a[F]$ and $\limsup_{F\to\infty} b[F]$ respectively.

1) Bound for $\limsup_{F\to\infty} a[F]$:

$$
\begin{aligned}
\limsup_{F\to\infty} a[F] \le& \limsup_{F\to\infty} \frac{1}{t_F^{(n)}} \sum_{t=0}^{t_F^{(n)}-1} Y(t) \\
&+ \limsup_{F\to\infty} \frac{1}{t_F^{(n)}}\left(B_2 T_n[F] + \frac{B_0}{2}T_n[F]^2\right) \\
\le& V(\overline{W}_n^* + \overline{E}_n^* + \overline{G}_n^*) + \Psi_n \\
&+ \limsup_{F\to\infty} \frac{1}{t_F^{(n)}}\left(B_2 T_n[F] + \frac{B_0}{2}T_n[F]^2\right).
\end{aligned}
$$

where the second inequality follows from corollary 1. It remains to show that

$$\limsup_{F\to\infty} \frac{1}{t_F^{(n)}}\left(B_2 T_n[F] + \frac{B_0}{2}T_n[F]^2\right) \le 0. \tag{31}$$

Since $t_F^{(n)} \ge F$, it is enough to show that

$$\limsup_{F\to\infty} \frac{T_n[F]}{F} = 0, \tag{32}$$

$$\limsup_{F\to\infty} \frac{T_n[F]^2}{F} = 0. \tag{33}$$

We prove (33), and (32) is similar. Since each $T_n[F] = I_n[F] + \tau_n[F] + 1$, where $I_n[F] \le I_{\max}$ and $\tau_n[F]$ has bounded first four moments, the first four moments of $T_n[F]$ must also be bounded and there exists a constant $C > 0$ such that

$$\mathbb{E}\left[T_n[F]^4\right] \le C.$$

For any $\epsilon > 0$, define a sequence of events

$$A_F^\epsilon \triangleq \left\{T_n[F]^2 > \epsilon F\right\}.$$

According to Markov inequality,

$$Pr\left[A_F^\epsilon\right] \le \frac{\mathbb{E}\left[T_n[F]^4\right]}{\epsilon^2 F^2} \le \frac{C}{\epsilon^2 F^2}.$$

Thus,

$$\sum_{F=1}^{\infty} Pr\left[A_F^\epsilon\right] \le \frac{C}{\epsilon^2}\sum_{F=1}^{\infty}\frac{1}{F^2} \le \frac{2C}{\epsilon^2} < \infty.$$

By Borel-Cantelli lemma (lemma 1.6.1 in [22]),

$$Pr\left[A_F^\epsilon \text{ occurs infinitely often}\right] = 0,$$

which implies

$$Pr\left[\limsup_{F\to\infty} \frac{T_n[F]^2}{F} > \epsilon\right] = 0.$$

Since $\epsilon$ is arbitrary, this implies (33). Similarly, (32) can be proved. Thus, (31) holds and

$$\limsup_{F\to\infty} a[F] \le V(\overline{W}_n^* + \overline{E}_n^* + \overline{G}_n^*) + \Psi_n.$$

2) Bound for $\limsup_{F\to\infty} b[F]$:

$$\begin{aligned}\limsup_{F\to\infty} b[F] \leq& \limsup_{F\to\infty} \frac{1}{t_F^{(n)}} \sum_{t=0}^{t_F^{(n)}-1} Y(t) \cdot \frac{t_F^{(n)}}{t_{F+1}^{(n)}} \\ &+ \limsup_{F\to\infty} \frac{1}{t_{F+1}^{(n)}} \left( B_2 T_n[F] + \frac{B_0}{2} T_n[F]^2 \right). \\ \leq& \limsup_{F\to\infty} \left( \frac{1}{t_F^{(n)}} \sum_{t=0}^{t_F^{(n)}-1} Y(t) \right) \\ &\cdot \limsup_{F\to\infty} \frac{t_F^{(n)}}{t_{F+1}^{(n)}} \\ \leq& \left( V\left( \overline{W}_n^* + \overline{E}_n^* + \overline{G}_n^* \right) + \Psi_n \right) \\ &\cdot \limsup_{F\to\infty} \frac{t_F^{(n)}}{t_{F+1}^{(n)}} \\ \leq& V\left( \overline{W}_n^* + \overline{E}_n^* + \overline{G}_n^* \right) + \Psi_n,\end{aligned}$$

where the second inequality follows from (31), the third inequality follows from corollary 1 and the last inequality follows from the fact that $V\left( \overline{W}_n^* + \overline{E}_n^* + \overline{G}_n^* \right) + \Psi_n > 0$.

Above all, we proved the lemma. □

## Appendix D— Proof of Theorem 1

*Proof.* Define the drift-plus-penalty(DPP) expression $P(t)$ as follows

$$\begin{aligned}P(t) =& V\left( c(t)d(t) + \sum_{n=1}^{N} \left(W_n(t) + e_n H_n(t) + g_n(t)\right) \right) \\ &+ \frac{1}{2} \sum_{n=1}^{N} \left( Q_n(t+1)^2 - Q_n(t)^2 \right).\end{aligned}$$

By simple algebra using the queue updating rule (1), we can work out the upper bound for $P(t)$ as follows,

$$\begin{aligned}P(t) \leq& \frac{1}{2} \sum_{n=1}^{N} (R_n(t) + \mu_n)^2 + V\left( c(t)d(t) + \right. \\ &\left. \sum_{n=1}^{N} \left(W_n(t) + e_n H_n(t) + g_n(t)\right) \right) \\ &+ \sum_{n=1}^{N} Q_n(t)(R_n(t) - \mu_n H_n(t)) \\ \leq& B_3 + V\left( c(t)d(t) + \sum_{n=1}^{N} \left(W_n(t) + e_n H_n(t) + g_n(t)\right) \right) \\ &+ \sum_{n=1}^{N} Q_n(t)(R_n(t) - \mu_n H_n(t)) \\ \leq& B_3 + Vc(t)d(t) + \sum_{n=1}^{N} Q_n(t) \left( R_n(t) - \overline{R}_n^* \right) \\ &+ V \sum_{n=1}^{N} \left(W_n(t) + e_n H_n(t) + g_n(t)\right) \\ &+ \sum_{n=1}^{N} Q_n(t) \left( \overline{\mu}_n^* - \mu_n H_n(t) \right)\end{aligned}$$

where $B_3 = \frac{1}{2}\sum_{n=1}^{N}(R_{\max}+\mu_n)^2$, the last inequality follows from adding $\sum_{n=1}^{N} Q_n(t)\overline{\mu}_n^*$ and subtracting $\sum_{n=1}^{N} Q_n(t)\overline{R}^*$ with the fact that the best randomized stationary algorithm should also satisfy the constraint (5), i.e. $\overline{\mu}^* \geq \overline{R}_n^*$.

Now we take the partial average of $P(t)$ from 0 to $T-1$ and take $\limsup_{T\to\infty}$,

$$\begin{aligned}&\limsup_{T\to\infty} \frac{1}{T} \sum_{n=1}^{T-1} P(t) \\ \leq& B_3 + \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \left( Vc(t)d(t) + \sum_{n=1}^{N} Q_n(t) \left( R_n(t) - \overline{R}_n^* \right) \right) \\ &+ \sum_{n=1}^{N} \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \left( V\left(W_n(t) + e_n H_n(t) + g_n(t)\right) \right. \\ &\left. + Q_n(t) \left( \overline{\mu}_n^* - \mu_n H_n(t) \right) \right).\end{aligned} \tag{34}$$

According to (25),

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \left( Vc(t)d(t) + \sum_{n=1}^{N} Q_n(t) \left( R_n(t) - \overline{R}_n^* \right) \right) \leq V\overline{C}^*. \tag{35}$$

On the other hand,

$$\begin{aligned}&\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \left( V\left(W_n(t) + e_n H_n(t) + g_n(t)\right) \right. \\ &\left. + Q_n(t) \left( \overline{\mu}_n^* - \mu_n H_n(t) \right) \right) \\ \leq& \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \left( V\left(W_n(t) + e_n H_n(t) + g_n(t)\right) \right. \\ &\left. + Q_n(t_f^{(n)}) \left( \overline{\mu}_n^* - \mu_n H_n(t) \right) + (t - t_f^{(n)}) B_0 \right) \\ \leq& V\left( \overline{W}_n^* + \overline{E}_n^* + \overline{G}_n^* \right) + \Psi_n,\end{aligned} \tag{36}$$

where $B_0 = \frac{1}{2}(R_{\max} + \mu_{\max})\mu_{\max}$ as defined below (12), the first inequality follows from the fact that for any $t \in \left( t_f^{(n)},\ t_{f+1}^{(n)} \right)$,

$$\begin{aligned}&Q_n(t) \left( \overline{\mu}_n^* - \mu_n H_n(t) \right) \\ \leq& Q_n(t_f^{(n)}) \left( \overline{\mu}_n^* - \mu_n H_n(t) \right) \\ &+ \left( Q_n(t) - Q_n(t_f^{(n)}) \right) \left( \overline{\mu}_n^* - \mu_n H_n(t) \right) \\ \leq& Q_n(t_f^{(n)}) \left( \overline{\mu}_n^* - \mu_n H_n(t) \right) \\ &+ \sum_{t=t_f^{(n)}}^{t_{f+1}^{(n)}-1} \left( R_n(t) - \mu_n H_n(t) \right) \left( \overline{\mu}_n^* - \mu_n H_n(t) \right) \\ \leq& Q_n(t_f^{(n)}) \left( \overline{\mu}_n^* - \mu_n H_n(t) \right) + (t - t_f^{(n)}) B_0,\end{aligned}$$

and the second inequality follows from lemma 6. Substitute (35) and (36) into (34) gives

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}P(t)\leq V\left(\overline{C}^*+\sum_{n=1}^{N}\left(\overline{W}_n^*+\overline{E}_n^*+\overline{G}_n^*\right)\right) + B_3+\sum_{n=1}^{N}\Psi_n. \tag{37}$$

Finally, notice that by telescoping sums,

$$\begin{aligned}&\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}P(t)\\ =&\limsup_{T\to\infty}\left(\frac{V}{T}\sum_{t=0}^{T-1}(c(t)d(t)+\sum_{n=1}^{N}(W_n(t)+e_nH_n(t)\right.\\ &\left.+g_n(t)))+\frac{1}{2}\sum_{n=1}^{N}Q_n(T)^2\right)\\ \geq& V\cdot\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}(c(t)d(t)+\sum_{n=1}^{N}(W_n(t)+e_nH_n(t)\\ &+g_n(t)))\end{aligned}$$

Substitute above inequality into (37) and divide $V$ from both sides give the desired result. □

## References


[1] D. P. Bertsekas. *Dynamic Programming and Optimal Control, vols. 1 and 2*. Athena Scientific, Belmont, Mass, 1995.
[2] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, 2005.
[3] S. Ross. *Introduction to Probability Models.* Academic Press, 8th edition, 2002.
[4] R. Gallager. *Discrete Stochastic Processes. Kluwer Academic Publishers*, Boston, 1996.
[5] 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.
[6] M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.
[7] M. J. Neely, Dynamic Optimization and Learning for Renewal Systems, *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 32-46, Jan. 2013.
[8] M. J. Neely, Asynchronous Scheduling for Energy Optimality in Systems with Multiple Servers, *Proc. 46th Annual Conf. on Information Sciences and Systems (CISS)*, March 2012.
[9] M. J. Neely, Online Fractional Programming for Markov Decision Systems, *Proc. Allerton Conf. on Communication, Control, and Computing*, Sept. 2011.
[10] A. Gandhi, S. Doroudi, M. Harchol-Balter, A. Scheller-Wolf. Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward, *Proc. ACM Sigmetrics*, pp.153-166, 2013.
[11] D. D. Yao. Dynamic Scheduling via Polymatroid Optimization, *Proceeding Performance Evaluation of Complex Systems: Techniques and Tools*, pp. 89-113, 2002.
[12] A. Gandhi, *Dynamic Server Provisioning for Data Center Power Management.* PhD thesis, 2013.
[13] R. Urgaonkar, U. C. Kozat, K. Igarashi, and M. J. Neely, Dynamic Resource Allocation and Power Management in Virtualized Data Centers, *IEEE/IFIP Network Operations and Management Symposium 2010*, Osaka, Japan, April 2010.
[14] Y. Yao, L. Huang, A. Sharma, L. Golubchik, M. Neely, Data centers power reduction: a two time scale approach for delay tolerant workloads, *Proceedings of INFOCOM*, pp. 1431-1439, 2012.
[15] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, Dynamic right-sizing for power-proportional data centers, *Proceedings IEEE INFOCOM*, 2011.
[16] T. Horvath and K. Skadron, Multi-mode energy management for multitier server clusters, *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, Toronto, Ontario, Canada: ACM, 2008.
[17] D. Meisner, B. T. Gold, and T. F. Wenisch. Powernap: eliminating server idle power. *Proceedings of ASPLOS*, pp. 205-216, 2009.
[18] D. Meisner, C. M. Sadler, L. A. Barroso, W. Weber, and T. F. Wenisch. Power management of online data-intensive services. *Proceedings of ISCA*, pp. 319-330, 2011.
[19] A. Gandhi, M. Harchol-Balter, and M. A. Kozuch. Are sleep states effective in data centers?, *Proceedings of the 2012 International Green Computing Conference (IGCC) (IGCC '12)*, Washington, DC, USA, 2012.
[20] X. Wei and M. J. Neely, On the theory and application of distributed asynchronous optimization over weakly coupled renewal systems, *ArXiv preprint arXiv:1608.00195*, 2016.
[21] M. J. Neely, Stability and probability 1 convergence for queueing networks via Lyapunov optimization, *Journal of Applied Mathematics*, doi:10.1155/2012/831909, 2012.
[22] R. Durrett, *Probability: Theory and Examples*, Cambridge University Press, 2013.
[23] B. Fox, Markov renewal programming by linear fractional programming, *SIAM Journal on Applied Mathematics*, vol. 14, no. 6, pp. 1418-1432, Nov. 1966.
[24] T. Benson, A. Akella, and D. A. Maltz, Network traffic characteristics of data centers in the wild, *In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC '10),* New York, NY, USA, 2010.
[25] Data Set for IMC 2010 Data Center Measurement. http://pages.cs.wisc.edu/~tbenson/IMC10_Data.
[26] M. E. J. Newman. Power laws, pareto distributions and Zipf's law, *Contemporary Physics,* vol. 46, pp. 323-351, Dec. 2005.


**Xiaohan Wei** received B.S. degree in electrical engineering and information science from University of Science and Technology of China, Hefei, Anhui, China in 2012, and M.S. degree with honor in electrical engineering from University of Southern California, Los Angeles, in 2014. He is now working towards Ph.D. degree in electrical engineering. His research is in the area of stochastic optimization and statistical learning theory.

**Michael J. Neely** received B.S. degrees in both Electrical Engineering and Mathematics from the University of Maryland, College Park, in 1997. He was then awarded a 3 year Department of Defense NDSEG Fellowship for graduate study at the Massachusetts Institute of Technology, where he received an M.S. degree in 1999 and a Ph.D. in 2003, both in Electrical Engineering. He joined the faculty of Electrical Engineering at the University of Southern California in 2004, where he is currently an Associate Professor. His research is in the area of stochastic network optimization. Michael received the NSF Career award in 2008 and the Viterbi School of Engineering Junior Research Award in 2009. He is a member of Tau Beta Pi and Phi Beta Kappa.