Scheduling Jobs with Unknown Duration in Clouds

Siva Theja Maguluri, *Student Member, IEEE,* and R. Srikant, *Fellow, IEEE,*

***Abstract*—We consider a stochastic model of jobs arriving at a cloud data center. Each job requests a certain amount of CPU, memory, disk space, etc. Job sizes (durations) are also modeled as random variables, with possibly unbounded support. These jobs need to be scheduled non preemptively on servers. The jobs are first routed to one of the servers when they arrive and are queued at the servers. Each server then chooses a set of jobs from its queues so that it has enough resources to serve all of them simultaneously. This problem has been studied previously under the assumption that job sizes are known and upper bounded, and an algorithm was proposed which stabilizes traffic load in a diminished capacity region. Here, we present a load balancing and scheduling algorithm that is throughput optimal, without assuming that job sizes are known or are upper bounded.**

I. Introduction

Cloud computing has emerged as an important source of computing infrastructure to meet the needs of both corporate and personal computing users. There are several cloud computing paradigms. We will consider an Infrastructure as a Service (IaaS) system where users request Virtual Machines (VMs) to be hosted on the cloud. A user can choose from a class of VMs, each with different amounts of processing capacity, memory and disk space. We call each request a 'job'. The amount of time each VM or job is to be hosted is called its size.

Each server in the data center has certain amount of resources. This imposes a constraint on the number of jobs of different types that can be served simultaneously. The primary focus in this paper is to study the following resource allocation problems: When a job of a given type arrives, which server should it be sent to? We will call this the routing or load balancing problem. At each server, among the jobs that are waiting for service, which subset of the jobs should be scheduled? Jobs have to be scheduled in a nonpreemptive manner. We will call this the scheduling problem. We want to do this without knowledge of system parameters like arrival rates.

The resource allocation problem in cloud data centers has been well studied [1], [2]. Best Fit policy [3], [4] is a popular policy that is used in practice. A stochastic model of the IaaS cloud data center was studied in [5] where the capacity region of such a system was characterized in terms of the arrival rates. It was also shown in [5] that the Best Fit policy is not stable for all the arrival rates in the capacity region, i.e., is not throughput optimal. A simple preemptive and a more realistic nonpreemptive model were studied. A joint routing (or load balancing) and scheduling algorithm was proposed that is almost throughput optimal. That is, for any $\epsilon > 0$, a fraction $(1-\epsilon)$ of the capacity region is stabilizable in the nonpreemptive case. In the preemptive case, the complete capacity region is stabilizable. However, this algorithm assumes that the size of each job is known when the job arrives into the system. This assumption is not realistic in some settings.

The scheduling algorithm in [5] is inspired by MaxWeight scheduling algorithm in wireless networks that has been well studied [6]. MaxWeight scheduling is known to have good delay performance and has been studied by extensive simulations, as well as optimality results in various asymptotic regimes. However, one drawback of MaxWeight scheduling in wireless networks is that its complexity increases exponentially with the number of wireless nodes. Moreover, MaxWeight is a centralized policy.

It was shown in [5] that if each server chooses a MaxWeight schedule, it is same as choosing a MaxWeight schedule for the whole cloud system. This is a very useful result in practice because this gives a distributed MaxWeight policy with much lower complexity. Consider the following example. If there are L servers and each server has S allowed configurations. When each server computes a separate MaxWeight allocation, it has to find a schedule from S allowed configurations. Since there are L servers, this is equivalent to finding a schedule from LS possibilities. However, for a centralized MaxWeight schedule, one has to find a schedule from S^L schedules. Moreover, the complexity of each server's scheduling problem depends only on its own set of allowed configurations, which is independent of the total number of servers. Typically the data center is scaled by adding more servers rather than adding more allowable configurations.

It was shown in [7] that the preemptive algorithm of [5] optimizes a function of the backlog in the asymptotic regime when the arrival rates are close to the boundary of the capacity region. A study of the nonpreemptive algorithm in this setting was not easy because the exact stability region of the nonpreemptive algorithm was not known. Only an inner bound was known. Reference [8] studies a resource allocation algorithm in the many server asymptotic limit.

In this work, we study a nonpreemptive algorithm when the job sizes are not known. Nonpreemptive algorithms are more challenging to study because the state of the system in different time slots is coupled. For example, a MaxWeight schedule cannot be chosen in each time slot nonpreemptively. Suppose that there are certain unfinished jobs that are being served at the beginning of a time slot. These jobs cannot be paused in the new time slot. So, the new schedule should be chosen to include these jobs. A Maxweight schedule may not

The authors are with the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana Champaign, Urbana, IL 61801 USA (e-mail: siva.theja@gmail.com).

Research was supported by NSF Grant ECCS-1202065 and Army MURI W911NF-12-1-0385

This paper is a longer version of a paper which will appear in the Proc. of IEEE INFOCOM 2013.

include these jobs.

Nonpreemptive algorithms were studied in literature in the context of input queued switches with variable packet sizes. One such algorithm was studied in [9]. This algorithm, however uses the special structure of a switch and so, it is not clear as to how it can be generalized for the case of a cloud system. Reference [10] presents another algorithm that is inspired by CSMA type algorithm in wireless networks. One needs to prove a time scale separation result to prove optimality of this algorithm. This was done in [10] by appealing to prior work [11]. However, the result in [11] is applicable only when the Markov chain has finite number of states. However, since the Markov chain in [10] depends on the job sizes, it could have infinite states even in the special case when the job sizes are geometrically distributed. So, the results in [10] do not seem to be immediately applicable to our problem.

A similar problem was studied in [12]. Since a MaxWeight schedule cannot be chosen in every time slot without disturbing the packets in service, a MaxWeight schedule is chosen only at every refresh time. A time slot is called a *refresh time* if no packets are in service at the beginning of the time slot. Between the refresh times, either the schedule can be left unchanged or a 'greedy' MaxWeight schedule can be chosen. It was argued that such a scheduling algorithm is throughput optimal in a switch.

The proof of throughput optimality in [12] is based on first showing that the duration between consecutive refresh times is bounded so that a MaxWeight schedule is chosen often enough. Blackwell's renewal theorem was used to show this result. Since Blackwell's renewal theorem is applicable only in steady state, we were unable to verify the correctness of the proof.

Furthermore, to bound the refresh times of the system, it was claimed in [12] that the refresh time for a system with infinite queues provides an upper bound over the system with arrivals. This is not true for every sample path. For a set of jobs with given sizes, the arrivals could be timed in such a way that the system with arrivals has a longer refresh time than an infinitely backlogged system.

For example consider the following scenario. Let the the original system be called System 1 and the system with infinite queues, System 2. System 1 could have empty queues while system 2 never has empty queues. Say T_0 is a time when all jobs finish service for system 2. This does not guarantee that all jobs finish service for system 1. This is because system 1 could be serving just one job at time $T_0 - 1$, when there could be an arrival of a job of two time slots long. Let us say that it can be scheduled simultaneously with the job in service. This job then will not finish its service at time T_0, and so T_0 is not a refresh time for system 1.

The result in [12] does not impose any conditions on job size distribution. However, this insensitivity to job size distribution seems to be a consequence of the relationship between the infinite queue system and the finite queue system which is assumed there, but which we do not believe is true in general.

In particular, the examples presented in [5], [13] show that the policy presented in [12] is not throughput optimal when the job sizes are deterministic.

Here, we develop a coupling technique to bound the expected time between two refresh times. With this technique, we do not need to use Blackwell's renewal theorem. The coupling argument is also used to precisely state how the system with infinitely backlogged queue provides an upper bound on the mean duration between refresh times.

The main contributions in this work are the following.

1) We propose a throughput optimal scheduling and load balancing algorithm for a cloud data center, when the job sizes are unknown. Job sizes are assumed to be unknown not only at arrival but also at the beginning of service. This algorithm is based on using queue lengths (number of jobs in the queue) for weights in MaxWeight schedule instead of the workload as in [5]. The scheduling part of our algorithm is based on [12], but includes an additional routing component. Further, our proof of throughput-optimality is different from the one in [12] due to the earlier mentioned reasons.
2) Even if the job sizes are known, this algorithm does not waste any resources unlike the algorithm in [5] which forces a refresh time every T time slots potentially wasting resources during the process. In particular, when the job sizes have high variability, the amount of wastage can be high. However, the algorithm in this paper works even when the job sizes are not bounded, for instance, when the job sizes are geometrically distributed.

In terms of proof techniques, we make the following contributions.

1) We use a coupling technique to show that the mean duration between refresh times is bounded. We then use Wald's identity to bound the drift of a Lyapunov function between the refresh times.
2) Our algorithm can be used with a large class of weight functions to compute the MaxWeight schedule (for example, the ones considered in [14]). However, we present it here for log-weight functions. Log-weight functions are known to have good performance properties [15] and are also amenable to low-complexity implementations using randomized algorithms [16], [17].
3) Since we allow general job-size distributions, it is difficult to find a Lyapunov function whose drift is negative outside a finite set, as required by the Foster-Lyapunov theorem which is typically used to prove stability results. Instead, we use a theorem in [18] to prove our stability result, but this theorem requires that the drift of the Lyapunov function be (stochastically) bounded. We present a novel modification of the typical Lyapunov function used to establish the stability of MaxWeight algorithms to verify the conditions of the theorem in [18].

In an earlier version of this paper [19], we primarily considered the case of geometric job sizes and simply mentioned the extension to general job sizes without a proof. Here, we provide complete proofs for both cases.

The paper is organized as follows. In the next section, we describe the system and traffic model and present the scheduling and load balancing algorithm. In section III, we present the coupling technique and argue that the refresh times are bounded. We illustrate the use of this result by first proving throughput optimality in the the simple case when the job sizes are geometrically distributed in section IV. In section V, we present the proof for the case of general job size distributions. In section VI, we present some simulations and finally conclude in section VII.

II. Model Description and Algorithm

We first present the system and traffic model. Then, we present the algorithm and queueing model.

A. System and Traffic Model

The cloud data center consists of L servers or machines. There are K different resources. Server i has C_{ik} amount of resources of type k. There are M different types of VMs that the users can request from the cloud service provider. Each type of VM is specified by the amount of different resources (such as CPU, disk space, memory, etc) that it requests. Type m VM requests R_{mk} amount of resources of type k.

Given a server, an M-dimensional vector N is said to be a feasible VM-configuration if the given server can *simultaneously* host N_1 type-1 VMs, N_2 type-2 VMs, $\ldots$, and N_M type-M VMs. In other words, N is feasible at server i if and only if

$$\sum_{m=1}^{M} N_m R_{mk} \leq C_{ik}$$

for all k. We let $N_{\max}$ denote the maximum number of VMs of any type that can be served on any server.

In this paper, we consider a cloud system which hosts VMs for clients. A VM request from a client specifies the type of VM the client needs. We call a VM request a "job." A job is said to be a type-m job if a type-m VM is requested. We assume that time is slotted. We say that the size of the job is S if the VM needs to be hosted for S time slots. We assume that S is unknown when a VM arrives. We next define the concept of *capacity* for a cloud.

Let $\mathcal{A}_m(t)$ denote the set of type-m jobs that arrive at the beginning of time slot t, and let $A_m(t) = |\mathcal{A}_m(t)|$, i.e., the number of type-m jobs that arrive at the beginning of time slot t. $A_m(t)$ is assumed to be a stochastic process which is i.i.d across time and independent across different types. We also assume that $A_m(t) \leq A_{\max}$. Some of these assumptions can be relaxed, but we consider a simple model for ease of exposition.

For each job j, let S_j denote its size, i.e., the number of time slots required to serve the job. For each j, S_j is assumed to be a (positive) integer valued random variable independent of the arrival process and the sizes of all other jobs in the system. The distribution of S_j is assumed to be identical for all jobs of same type. In other words, for each type m, the job sizes are i.i.d. Let $\mathcal{S}$ be the support of the random variable S, i.e., $\mathcal{S} = \{n \in \mathbb{N} : P(S = n) > 0\}$. The job size distribution is assumed to satisfy the following assumption.

Assumption 1: If $l_1 \in \mathcal{S}$ is in the support of the distribution, then any $l_2 \in \mathbb{N}$ such that $1 \leq l_2 < l_1$ is also in the support of the distribution, i.e., $l_2 \in \mathcal{S}$. For each job type m, let $C_m \triangleq \inf_{l \in \mathcal{S}} P(S_m = l | S_m > l-1)$. Then, there exists a $C > 0$ such that for each server m, $C_m \geq C > 0$. In the case when the support is finite, this just means that the conditional probabilities $P(S_k = l | S_k > l-1)$ are non-zero for any l in the support.

Assumption (1) means that when the job sizes are not bounded, they have geometric tails. For example, truncated heavy-tailed distributions would be allowed but purely heavy-tailed distributions would not be allowed under our model.

B. Algorithm and Queueing Model

We assume that each server maintains M different queues for different types of jobs. It then uses this queue length information in making scheduling decisions. Let $\mathbf{Q}$ denote the vector of these queue lengths where $\mathbf{Q}_{mi}$ is the number of type m jobs at server i.

Algorithm 1 performs load balancing to route jobs to servers (Step 1) and uses a MaxWeight algorithm to schedule jobs on each server (Step 2) with an appropriately chosen function $g(.)$. It is important to note that, unlike the algorithm in [5], Algorithm 1 does not require the cloud system to know the job sizes nor does it require the job sizes to be upper bounded.

Let $\mathbf{D}_{mi}(t)$ denote the number of type-m jobs that finish service at server i in time slot t. Then the queue lengths evolve as follows:

$$\mathbf{Q}_{mi}(t+1) = \mathbf{Q}_{mi}(t) + \mathbf{A}_{mi}(t) - \mathbf{D}_{mi}(t).$$

The cloud system is said to be *stable* if the expected total queue length is bounded, i.e.,

$$\limsup_{t\to\infty} E[\sum_i \sum_m \mathbf{Q}_{mi}(t)] < \infty$$

A vector of arriving loads λ and mean job sizes $\overline{S}$ is said to be supportable if there exists a resource allocation mechanism under which the cloud system is stable. Let $\overline{S}_{\max} = \max_m\{\overline{S}_m\}$ and $\overline{S}_{\min} = \min_m\{\overline{S}_m\}$.

In the following, we first identify the set of supportable $(\lambda, \overline{S})$ pairs. Let $\mathcal{N}_i$ be the set of feasible VM-configurations on a server i. We define sets $\widehat{\mathcal{C}}$ and $\mathcal{C}$ as follows.

$$\widehat{\mathcal{C}} = \left\{ N \in \mathbb{R}_+^M : N = \sum_{i=1}^{L} N^{(i)} \text{ and } N^{(i)} \in \text{Conv}(\mathcal{N}_i) \right\},$$

where Conv denotes the convex hull. Now define

$$\mathcal{C} = \left\{ (\lambda, \overline{S}) \in \mathbb{R}_+^M \times \mathbb{R}_+^M : (\lambda \circ \overline{S}) \in \widehat{\mathcal{C}} \right\},$$

where $(\lambda \circ \overline{S})$ denotes the Hadamard product or entrywise product of the vectors λ and $\overline{S}$ and is defined as $(\lambda \circ \overline{S})_m = \lambda_m \overline{S}_m$. We use $int(.)$ to denote interior of a set.

We will show that a pair $(\lambda, \overline{S})$ is supportable if and only if $(\lambda, \overline{S}) \in \mathcal{C}$. As in [6], it is easy to show the following result.

Proposition 1: For any pair $(\lambda, \overline{S})$ such that $(\lambda, \overline{S}) \notin \mathcal{C}$, $\lim_{t\to\infty} E\left[\sum_m Q_m(t)\right] = \infty$, i.e., the pair $(\lambda, \overline{S})$ is not supportable.

Algorithm 1 JSQ Routing and Myopic MaxWeight Scheduling

1) *Routing Algorithm (JSQ Routing)*: All the type m jobs that arrive in time slot t are routed to the server with the shortest queue for type m jobs i.e., the server $i^*_m(t) = \arg\min_{i\in\{1,2,,,L\}} \mathbf{Q}_{mi}(t)$. Therefore, the arrivals to $\mathbf{Q}_{mi}$ in time slot t are given by

$$\mathbf{A}_{mi}(t) = \begin{cases} A_m(t) & \text{if } i = i^*_m(t) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

2) *Scheduling Algorithm (MaxWeight Scheduling) for each server i*: Let $\widetilde{N}^{(i)}_m(t)$ denote a configuration chosen in each time slot. If the time slot is a refresh time (i.e., if none of the servers are serving any jobs at the beginning of the time slot), $\widetilde{N}^{(i)}_m(t)$ is chosen according to the MaxWeight policy, i.e.,

$$\widetilde{N}^{(i)}(t) \in \arg\max_{N\in\mathcal{N}_i} \sum_m g(\mathbf{Q}_{mi}(t))N_m. \tag{2}$$

If it is not a refresh time, $\widetilde{N}^{(i)}_m(t) = \widetilde{N}^{(i)}_m(t-1)$. However, $\widetilde{N}^{(i)}_m(t)$ jobs of type m may not be present at server i, in which case all the jobs in the queue that are not yet being served will be included in the new configuration. If $\overline{N}^{(i)}_m(t)$ denotes the actual number of type m jobs selected at server i, then the configuration at time t is $N^{(i)}(t) = \overline{N}^{(i)}(t)$. Otherwise, i.e., if there are enough number of jobs at server i, $N^{(i)}(t) = \widetilde{N}^{(i)}_m(t)$.

III. Refresh Times

Recall that a time slot is called a *refresh time* if none of the servers are serving any jobs at the beginning of the time slot. Note that a time slot is refresh time if, in the previous time slot, either all jobs in service departed the system or the system was completely empty.

Refresh times are important for our stability proof later, due to the fact that a new MaxWeight schedule can be chosen for all servers only at such time instants. At all other time instants, an entirely new schedule cannot be chosen for all servers simultaneously since this would require job preemption which we assume is not allowed.

Let us denote the nth refresh time by t_n. Let $z_n = t_{n+1} - t_n$ be the duration (in slots) between the nth and $(n+1)$th refresh times.

The following fact about refresh times is needed to study the throughput of the system.

Lemma 1: There exists constants $\mathcal{K}_1 > 0$ and $\mathcal{K}_2 > 0$ such that $\mathbb{E}[z_n] < \mathcal{K}_1$ and $\mathbb{E}[z_n^2] < \mathcal{K}_2$.

Proof: Let $R(t)$ be a binary valued random process that takes a value 1 if and only if time t is a refresh time. Consider a job of type m that is being served at a server. Say it was scheduled l time slots ago. The conditional probability that it finishes its service in the next time slot is

$$P(S_m = l+1 | S_m > l) \geq C_m \geq C$$

from the assumption on the job size distribution. Thus, the probability that a job of type m that is being served finishes its service at any time slot is atleast C. So, the probability that all the jobs scheduled at a server finish their service at any time slot is no less than $C^{MN_{\max}}$ and the probability that all the jobs scheduled in the system finish their service is atleast $\overline{C} \triangleq C^{LMN_{\max}} > 0$. If all the jobs that are being served at all the servers finish their service during a time slot, it is a refresh time. Thus probability that a given time slot is a refresh time is at least $\overline{C}$. In other words, for any time t, if $p(t)$ is the probability that $R(t) = 1$ conditioned on the history of the system (i.e., arrivals, departures, scheduling decisions made and the finished service of the jobs that are in service), then $p(t) \geq \overline{C} > 0$.

Define $R_n(z) = R(t_n + z)$ for $z \geq 0$. Then z_n is the first time $R_n(z)$ takes a value of 1. Now consider a Bernoulli process $\overline{R_n}(z)$ with probability of success $\overline{C}$ that is coupled to the refresh time process $R_n(z)$ as follows. Whenever $\overline{R_n}(z) = 1$, we also have $R_n(z) = 1$. One can construct such a pair $(R_n(t), \overline{R_n}(z))$ as follows. Consider an i.i.d random process $\widehat{R_n}(z)$ uniformly distributed between 0 and 1. Then the pair $(R_n(z), \overline{R_n}(z))$ can be modeled as $R_n(z) = 1$ if and only if $\widehat{R_n}(z) < p(t_n + z)$ and $\overline{R}(t)) = 1$ if and only if $\widehat{R_n}(z) < \overline{C}$.

Let $\overline{z}_n$ be the first time $\overline{R_n}(z)$ takes a value of 1. Then, by the construction of $\overline{R_n}(z)$, $z_n \leq \overline{z}_n$ and since $\overline{R_n}(z)$ is a Bernoulli process, there exists constants $\mathcal{K}_1 > 0$ and $\mathcal{K}_2 > 0$ such that $\mathbb{E}[\overline{z}_n] < \mathcal{K}_1$ and $\mathbb{E}[\overline{z}_n^2] < \mathcal{K}_2$ proving the Lemma. ■

IV. Throughput Optimality - Geometric Packet Sizes

In this section, we will characterize the throughput performance of Algorithm 1 in the special case when the job sizes are geometrically distributed. We will consider a more general case in the next section.

We will need Wald's identity [20, Chap 6, Cor 3.1 and Sec 4(a)] to bound the drift between two refresh times. We state it here for convenience.

Theorem 1 (Wald's Identity): Let $\{X_n : n \in \mathbb{N}\}$ be a sequence of real-valued, random variables such that all $\{X_n : n \in \mathbb{N}\}$ have same expectation and there exists a constant C such that $E[|X_n|] \leq C$ for all $n \in \mathbb{N}$. Assume that there exists a filtration $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ such that X_n and $\mathcal{F}_{n-1}$ are independent for every $n \in \mathbb{N}$. Then, if N is a finite mean stopping time with respect to the filtration $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$, $E[\sum_{n=1}^{N} X_n] = E[X_n]E[N]$.

In the case of geometric packet sizes, a wide class of functions $g(y)$ can be used to obtain a stable MaxWeight policy [14]. Typically, $V((Q)) = \sum_{i,m} \int g((Q)_{mi})dy$ is used as a Lyapunov function to prove stability of a MaxWeight policy using $g(y)$. To avoid excessive notation, we will illustrate the proof of throughput optimality using $g(y) = y$ in this section.

Proposition 2: When the job sizes are geometrically distributed with mean job size vector $\overline{S}$, any job load vector that satisfies $(\lambda, \overline{S}) \in int(\mathcal{C})$ is supportable under the JSQ routing and myopic MaxWeight allocation as described in Algorithm 1 with $g(y) = y$.

Proof: Since the job sizes are geometrically distributed, it is easy to see that $\mathbf{X}(t) = (\mathbf{Q}(t), N(t))$ is a Markov chain under Algorithm 1.

Obtain a new process, $\widetilde{\mathbf{X}}(n)$ by sampling the Markov Chain $\mathbf{X}(t)$ at the refresh times, i.e., $\widetilde{\mathbf{X}}(n) = \mathbf{X}(t_n)$. Note that $\widetilde{\mathbf{X}}(n)$ is also a Markov Chain because, conditioned on $\widetilde{\mathbf{Q}}(n) = \mathbf{Q}(t_n) = \mathbf{q_0}$ (and so $N(t_n)$), the future of evolution of $\mathbf{X}(t)$ and so $\widetilde{\mathbf{X}}(n)$ is independent of the past.

Using $V(\mathbf{X}) = V(\mathbf{Q}) = \sum_m \sum_i \overline{S}_m \mathbf{Q}_{mi}^2$ as the Lyapunov function, we will first show that the drift of the Markov Chain is negative outside a bounded set. This gives positive recurrence of the sampled Markov Chain from Foster-Lyapunov Theorem. We will then use this to prove the positive recurrence of the original Markov Chain.

First consider the following one step drift of $V(t)$. Let $t = t_n + \tau$ for $0 \leq \tau < z_n$.

$$\begin{aligned}
&(V(t+1) - V(t))\\
=&\sum_i \sum_m \overline{S}_m \left(\mathbf{Q}_{mi}(t) + \mathbf{A}_{mi}(t) - \mathbf{D}_{mi}(t)\right)^2 - \overline{S}_m \mathbf{Q}_{mi}^2(t)\\
=&2\sum_i \sum_m \overline{S}_m \mathbf{Q}_{mi}(t)\left(\mathbf{A}_{mi}(t) - \mathbf{D}_{mi}(t)\right)\\
&+\sum_i \sum_m \overline{S}_m \left(\mathbf{A}_{mi}(t) - \mathbf{D}_{mi}(t)\right)^2\\
\leq&2\sum_m \sum_i \overline{S}_m \mathbf{Q}_{mi}(t)\left(\mathbf{A}_{mi}(t) - \mathbf{D}_{mi}(t)\right) + K_1
\end{aligned}$$

where $K_1 = L(A_{\max} + N_{\max})^2(\sum_m \overline{S}_m)$.

Now using this relation in the drift of the sampled system, we get the following. With a slight abuse of notation, we denote $E\left[(.)|\mathbf{Q}(t_n) = \mathbf{q}\right]$ by $E_{\mathbf{q}}\left[(.)\right]$.

$$\begin{aligned}
&E[V(\widetilde{\mathbf{Q}}(n+1)) - V(\widetilde{\mathbf{Q}}(n))|\widetilde{\mathbf{Q}}(n) = \mathbf{q}]\\
=&E[V(t_{n+1}) - V(t_n)|\mathbf{Q}(t_n) = \mathbf{q}]\\
=&E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1} V(t_n+\tau+1) - V(t_n+\tau)\right]\\
\leq&E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1} 2\sum_{m,i}\big(\overline{S}_m \mathbf{Q}_{mi}(t_n+\tau)\mathbf{A}_{mi}(t_n+\tau)\right.\\
&\left.-\overline{S}_m \mathbf{Q}_{mi}(t_n+\tau)\mathbf{D}_{mi}(t_n+\tau)\big) + K_1\right] \qquad (3)
\end{aligned}$$

The last term above is bounded by $K_1\mathcal{K}_1$ from Lemma 1. We will now bound the first term in (3). From the definition of $\mathbf{A}_{mi}$ in the routing algorithm in (1), we have

$$\begin{aligned}
&2E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_m\sum_i \overline{S}_m \mathbf{Q}_{mi}(t_n+\tau)\mathbf{A}_{mi}(t_n+\tau)\right]\\
=&2E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_m \overline{S}_m \mathbf{Q}_{mi_m^*(t_n+\tau)}(t_n+\tau)A_m(t_n+\tau)\right]\\
\leq&2E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_m \overline{S}_m\big(\mathbf{Q}_{mi_m^*(t_n)}(t_n)A_m(t_n+\tau)+\tau A_{\max}^2\overline{S}_m\big)\right]\\
\leq&2\sum_m \overline{S}_m \mathbf{q}_{mi_m^*} E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1} A_m(t_n+\tau)\right]+\sum_m A_{\max}^2\overline{S}_m E_{\mathbf{q}}\left[z_n^2\right]\\
\leq&A_{\max}^2\mathcal{K}_2\sum_m \overline{S}_m + 2E_{\mathbf{q}}\left[z_n\right]\sum_m \overline{S}_m \mathbf{q}_{mi_m^*}\lambda_m \qquad (4)
\end{aligned}$$

where $i_m^*(t) = \underset{i\in\{1,2,,,L\}}{\arg\min}\, \mathbf{Q}_{mi}(t)$ and $i_m^* = i_m^*(t_n)$. Since $\mathbf{Q}_{mi_m^*(t_n+\tau)}(t_n+\tau) \leq \mathbf{Q}_{mi_m^*(t_n)}(t_n+\tau) \leq \mathbf{Q}_{mi_m^*}(t_n) + A_{\max}\tau$ because the load at each queue cannot increase by more than $A_{\max}$ in each time slot, we get the first inequality.

Let $\mathbf{Y}(t) = \{\mathbf{Y}_{mi}(t)\}_{m,i}$ denote the state of jobs of type-m at server i. When $\mathbf{Q}_{mi}(t) \neq 0$, $\mathbf{Y}_{mi}(t)$ is a vector of size $N_m^{(i)}(t)$ and $\mathbf{Y}_{mi}^j(t)$ is the amount of time the j^{th} type-m job that is in service at server i has been scheduled. Note that the departures $\mathbf{D}(t)$ can be inferred from $\mathbf{Y}(t)$. Let $\mathcal{F}_\tau^{(n)}$ be the filtration generated by the process $\mathbf{Y}(t_n+\tau)$. Then, $A(t_n+\tau+1)$ is independent of $\mathcal{F}_\tau^{(n)}$ and z_n is a stopping time for $\mathcal{F}_\tau^{(n)}$. Then, from Wald's identity (Theorem 1) and Lemma 1, we have (4).

Now we will bound the second term in (3). To do this, consider the following system. For every job of type m that is in the configuration $\widetilde{N}_m^{(i)}(t_n)$, consider an independent geometric random variable of mean $\overline{S}_m$ to simulate potential departures or job completions. Let $\mathcal{I}_{j,m}^i(t)$ be an indicator function denoting if the j^{th} job of type m at server i in configuration $\widetilde{N}_m^{(i)}(t_n)$ has a potential departure at time t. Because of memoryless property of geometric distribution, $\mathcal{I}_{j,m}^i(t)$ are i.i.d Bernoulli with mean $1/\overline{S}_m$.

If the j^{th} job was scheduled, $\mathcal{I}_{j,m}^i(t)$ corresponds to an actual departure. If not (i.e., if there was unused service), there is no actual departure corresponding to this. Let $\widehat{\mathbf{D}}_{mi}(t) = \sum_{j=1}^{\widetilde{N}_m^{(i)}(t_n)} \mathcal{I}_{j,m}^i(t)$ denote the number of potential departures of type m at server i. Note that if $\mathbf{Q}_{mi}(t) \geq N_{\max}$, $\widehat{\mathbf{D}}_{mi}(t) = \mathbf{D}_{mi}(t)$ since there is no unused service in this case. Also, $\widehat{\mathbf{D}}_{mi}(t) - \mathbf{D}_{mi}(t) \leq \widehat{\mathbf{D}}_{mi}(t) \leq N_{\max}$. Thus, we have,

$$\begin{aligned}
\mathbf{Q}_{mi}(t)\mathbf{D}_{mi}(t) =& \left(\mathbf{Q}_{mi}(t)\mathbf{D}_{mi}(t)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t)\geq N_{\max}}\\
&+\left(\mathbf{Q}_{mi}(t)\mathbf{D}_{mi}(t)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t)< N_{\max}}\\
\geq& \left(\mathbf{Q}_{mi}(t)\widehat{\mathbf{D}}_{mi}(t)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t)\geq N_{\max}}\\
&+\Big(\mathbf{Q}_{mi}(t)\Big(\widehat{\mathbf{D}}_{mi}(t)-N_{\max}\Big)\Big)\mathbb{I}_{\mathbf{Q}_{mi}(t)< N_{\max}}\\
\geq&\mathbf{Q}_{mi}(t)\widehat{\mathbf{D}}_{mi}(t) - N_{\max}^2 \qquad (5)
\end{aligned}$$

Note that $\mathbf{Q}_{mi}(t) \geq \mathbf{Q}_{mi}(t-1) - N_{\max}$, since $N_{\max}$ is the maximum possible departures in each time slot. So, we have

$$\begin{aligned}
\mathbf{Q}_{mi}(t_n+\tau)\widehat{\mathbf{D}}_{mi}(t_n+\tau) \geq& \left(\mathbf{Q}_{mi}(t_n)-\tau N_{\max}\right)\widehat{\mathbf{D}}_{mi}(t_n+\tau)\\
\geq& \mathbf{Q}_{mi}(t_n)\widehat{\mathbf{D}}_{mi}(t_n+\tau) - \tau N_{\max}^2
\end{aligned}$$

Using this with (5), we can bound the second term in (3) as follows

$$\begin{aligned}
&2E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_{i,m}\overline{S}_m \mathbf{Q}_{mi}(t_n+\tau)\mathbf{D}_{mi}(t_n+\tau)\right]\\
\geq&2E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_{i,m}\overline{S}_m \mathbf{Q}_{mi}(t_n+\tau)\widehat{\mathbf{D}}_{mi}(t_n+\tau)\right]\\
&-LN_{\max}^2\sum_m \overline{S}_m 2E_{\mathbf{q}}\left[z_n\right]
\end{aligned}$$

$$\geq 2E_{\mathbf{q}}\left[\sum_{i,m}\overline{S}_m\mathbf{Q}_{mi}(t_n)\sum_{\tau=0}^{z_n-1}\widehat{\mathbf{D}}_{mi}(t_n+\tau)\right]-K_2$$
$$=2E_{\mathbf{q}}[z_n]\sum_{i,m}\mathbf{q}_{mi}\widetilde{N}_m^{(i)}-K_2 \qquad (6)$$

where $K_2 = LN_{\max}^2\sum_m\overline{S}_m(2\mathcal{K}_1+\mathcal{K}_2)$. Let $\widehat{\mathcal{F}}_\tau^{(n)}$ denote the filtration generated by $\{\mathbf{Y}(t_n+\tau),\widehat{D}(t_n+\tau)\}$. Then, $\mathcal{F}_\tau^{(n)}\subseteq\widehat{\mathcal{F}}_\tau^{(n)}$. Since z_n is a stopping time with respect to the filtration $\mathcal{F}_\tau^{(n)}$, it is also a stopping time with respect to the filtration $\widehat{\mathcal{F}}_\tau^{(n)}$. Since $\widehat{D}(t_n+\tau+1)$ is independent of $\widehat{\mathcal{F}}_\tau^{(n)}$, Wald's identity can be applied here. $\widehat{D}(t_n+\tau)$ is sum of $\widetilde{N}_m^{(i)}(t_n)$ independent Bernoulli random variables each with mean $1/\overline{S}_m$. Thus, we have $E[\widehat{\mathbf{D}}_{mi}(t)]=\widetilde{N}_m^{(i)}(t_n)/\overline{S}_m$. Using this in Wald's identity (Theorem 1), we get (6).

Since $(\lambda,\overline{S})\in int(\mathcal{C})$, there exists $\epsilon>0$ such that $((1+\epsilon)\lambda,\overline{S})\in\mathcal{C}$, there exists $\{(1+\epsilon)\lambda^i\}_i$ such that $\lambda^i\circ\overline{S}\in\text{Conv}(\mathcal{N}_i)$ for all i and $\lambda=\sum_i\lambda^i$. According to the scheduling algorithm (2), for each server i, we have that

$$\sum_m\mathbf{Q}_{mi}(t_n)(1+\epsilon)\lambda_m^i\overline{S}_m\leq\sum_m\mathbf{Q}_{mi}(t_n)\widetilde{N}_m^{(i)}(t_n). \qquad (7)$$

Then, from (4), (3) and (6), we get

$$E[V(\widetilde{\mathbf{X}}(n+1))-V(\widetilde{\mathbf{X}}(n))|\widetilde{\mathbf{Q}}(n)=\mathbf{q},\widetilde{\mathbf{Y}}(n)]$$
$$\leq K_3+2E_{\mathbf{q}}[z_n]\sum_m\overline{S}_m\mathbf{q}_{mi_m^*}\lambda_m-2E_{\mathbf{q}}[z_n]\sum_{i,m}\mathbf{q}_{mi}\widetilde{N}_m^{(i)}$$
$$\overset{(a)}{\leq}K_3-2\epsilon E_{\mathbf{q}}[z_n]\sum_i\sum_m\mathbf{q}_{mi}\lambda_m^i\overline{S}_m$$
$$\leq K_3-2\epsilon\sum_i\sum_m\mathbf{q}_{mi}\lambda_m^i\overline{S}_m$$

where $K_3=A_{\max}^2\mathcal{K}_2\sum_m\overline{S}_m+K_2$. Inequality (a) follows from $\lambda=\sum_i\lambda^i$ and (7). The last inequality follows from $z_n\geq 1$.

Then, from the Foster-Lyapunov theorem [21], [22], we have that the sampled Markov Chain $\widetilde{\mathbf{X}}(n)$ is positive recurrent. So, there exists a constant $\mathcal{K}_3>0$ such that $\lim_{n\to\infty}\sum_m\sum_i E[\mathbf{Q}_{mi}(t)]\leq\mathcal{K}_3$.

For any $t>0$, let t_n be the last refresh time before t. Then,

$$\sum_{m,i}E[\mathbf{Q}_{mi}(t)]\leq\sum_{m,i}E[(\mathbf{Q}_{mi}(t_n)+z_n(A_{\max}+N_{\max}))]$$

As $t\to\infty$, we get

$$\limsup_{t\to\infty}\sum_m\sum_i E[\mathbf{Q}_{mi}(t)]$$
$$\leq\limsup_{n\to\infty}\sum_m\sum_i E[(\mathbf{Q}_{mi}(t_n)+z_n(A_{\max}+N_{\max}))]$$
$$\leq\mathcal{K}_3+\mathcal{K}_1LM(A_{\max}+N_{\max})$$

■

V. Throughput Optimality - General Packet Size Distribution

In this section, we will consider a general packet size distribution that satisfies Assumption 1.

We will show that Algorithm 1 is throughput optimal in this case with appropriately chosen $g(.)$. Unlike the Geometric job size case, for a job that is scheduled, the expected number of departures in each time slot is not constant here.

The process $\mathbf{X}(t)=(\mathbf{Q}(t),\mathbf{Y}(t))$ is a Markov Chain, where $\mathbf{Y}(t)$ is defined in the section IV. Let $W_m(l)$ be the expected remaining service time of a job of type m given that it has already been served for l time slots. In other words, $W_m(l)=E[S_m-l|S_m>l]$. Note that $W_m(0)=\overline{S}_m$. Then, we denote the *expected backlogged workload* at each queue by $\overline{\mathbf{Q}}_{mi}(t)$. Thus,

$$\overline{\mathbf{Q}}_{mi}(t)=\sum_{j=1}^{Q_{mi}}W_m(l_j)$$

where l_j is the duration of completed service for the j^{th} job in the queue. Note that $l_j=0$ if the job was never served.

The expected backlog evolves as follows.

$$\overline{\mathbf{Q}}_{mi}(t+1)=\overline{\mathbf{Q}}_{mi}(t)+\overline{\mathbf{A}}_{mi}(t)-\overline{\mathbf{D}}_{mi}(t).$$

where $\overline{\mathbf{A}}_{mi}(t)=\mathbf{A}_{mi}(t)\overline{S}_m$ since each arrival of type m brings in an expected load of $\overline{S}_m$. $\overline{\mathbf{D}}_{mi}(t)$ is the departure of the load.

Let $\widehat{p}_{ml}=P(S_m=l+1|S_m>l)$. A job of type m that is scheduled for l amount of time, has a backlogged workload of $W_m(l)$. It departs in the next time slot with a probability $\widehat{p}_{ml}$. With a probability $1-\widehat{p}_{ml}$, the job does not depart, and the expected remaining load changes to $W_m(l+1)$. So, the departure in this case is $W_m(l)-W_m(l+1)$. In effect, we have

$$\overline{\mathbf{D}}_{mi}(t)=\begin{cases}W_m(l) & \text{with prob }\widehat{p}_{ml}\\ W_m(l)-W_m(l+1) & \text{with prob }1-\widehat{p}_{ml}.\end{cases} \qquad (8)$$

This means that the $\overline{\mathbf{D}}_{mi}(t)$ could be negative sometimes, which means the expected backlog could increase even if there are no arrivals. Since the job size distribution is lower bounded by a geometric distribution by Assumption 1, the expected remaining workload is upper bounded by that of a geometric distribution. We will now show this formally.

From Assumption 1 on the job size distribution, we have

$$P(S_m=l+1|S_m>l)\geq C$$
$$P(S_m>l+1|S_m>l)\leq(1-C)$$

Then, using the relation $P(S_m>l+k+1|S_m>l)=P(S_m>l+k+1|S_m>l+k)P(S_m>l+k|S_m>l)$, one can inductively show that $P(S_m>l+k|S_m>l)\leq(1-C)^k$ for $k\geq 1$. Then,

$$\begin{aligned}W_m(l)&=E[S_m-l|S_m>l]\\&=\sum_{k=0}^{\infty}P(S_m>l+k|S_m>l)\\&\leq\sum_{k=0}^{\infty}(1-C)^k\\&\leq 1/C.\end{aligned} \qquad (9)$$

Then from (8), the increase in backlog of workload due to 'departure' for each scheduled job can increase by at most

$W_m(l+1)$, which is bounded $1/C$. There are at most $N_{\max}$ jobs of each type that are scheduled. The arrival in backlog queue is at most $A_{\max}\overline{S}_{\max}$. Thus, we have

$$\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t) \leq A_{\max}\overline{S}_{\max} + \frac{N_{\max}}{C} \tag{10}$$

Similarly, since the maximum departure in work load for each scheduled job is $1/C$, we have

$$\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t) \geq -\frac{N_{\max}}{C} \tag{11}$$

Since every job in the queue has at least one more time slot of service left, $\mathbf{Q}_{mi}(t) \leq \overline{\mathbf{Q}}_{mi}(t)$. Since $W_m(l) \leq 1/C$, we have the following lemma.

Lemma 2: There exists a constant $\widetilde{C} \geq 1$ such that $\mathbf{Q}_{mi}(t) \leq \overline{\mathbf{Q}}_{mi}(t) \leq \widetilde{C}\mathbf{Q}_{mi}(t)$ for all i, m and t.

Unlike the case of geometric job sizes, the actual departures in each time slot depend on the amount of finished service. However, the expected departure of workload in each time slot, is constant even for a general job size distribution. This is the reason we use a Lyapunov function that depends on the workload. We prove this result in the following lemma. This is a key result that we need for the proof.

Lemma 3: If a job of type m has been scheduled for l time slots, then the expected departure in the backlogged workload is $E[\overline{D}_m|l] = 1$. Therefore, we have $E[\overline{D}_m] = 1$

Proof: Define $p_{ml} = P(S_m = l)$. Then,

$$\begin{aligned} W_m(l) =& E[S_m - l | S_m > l] \\ =& \widehat{p}_{ml} \cdot 1 + (1-\widehat{p}_{ml})\left(1 + E[S_m - (l+1) | S_m > l+1]\right) \\ =& 1 + W_m(l+1)\left(1 - \widehat{p}_{ml}\right) \end{aligned}$$

Thus, we have

$$W_m(l) - W_m(l+1) = 1 - W_m(l+1)\left(\widehat{p}_{ml}\right) \tag{12}$$

Then, from (8),

$$\begin{aligned} E[\overline{D}_m|l] &= W_m(l) - (1-\widehat{p}_{ml})W_m(l+1) \\ &= W_m(l) - W_m(l+1) + (\widehat{p}_{ml})W_m(l+1) \quad = 1 \end{aligned}$$

from (12).

Since $E[\overline{D}_m|l] = 1$ for all l, we have $E[\overline{D}_m] = \sum_l E[\overline{D}_m|l]P(l) = 1$. ■

As in the case of Geometric packet sizes, we will show stability by first showing that the system obtained by sampling at refresh times has negative drift. For reasons mentioned in the introduction, here we will use $g(y) = \log(1+y)$ and the corresponding Lyapunov function

$$V(\overline{\mathbf{Q}}) = \sum_{i,m} G(\overline{\mathbf{Q}}_{mi})$$

where $G(.) : [0,\infty) \to [0,\infty)$ is defined as

$$\begin{aligned} G(q) =& \int_0^q g(y)dy \\ =& \int_0^q \log(1+y)dy \\ =& (1+q)\log(1+q) - q \end{aligned}$$

To use Foster-Lyapunov Theorem to prove stability, one needs to show that the drift of the Lyapunov function is negative outside a finite set. However in the general case when the packet sizes are not bounded, this set may not be finite and so Foster-Lyapunov Theorem is not applicable. We will instead use the following result by Hajek [18, Thm 2.3, Lemma 2.1], which can be thought of as a generalization of Foster-Lyapunov Theorem for nonmarkovian random processes.

Theorem 2: Let $\{Z_n\}_{n\geq=0}$ be a sequence of random variables adapted to a filtration $\{\mathcal{F}_n\}_{n\geq=0}$, which satisfies the following conditions:

C1 For some M and ϵ_0,

$$E[Z_{n+1} - Z_n|\mathcal{F}_n] \leq -\epsilon_0 \text{whenever } Z_n > M$$

C2 $(|Z_{n+1} - Z_n||\mathcal{F}_n) < \widetilde{Z}$ for all $n \geq 0$ and $E[e^{\theta\widetilde{Z}}]$ is finite for some $\theta > 0$.

Then, there exists $\theta^* > 0$ and C^* such that $\limsup_{n\to\infty} E[e^{\theta^* Z_n}] \leq C^*$.

We will use this theorem with the filtration generated by the process $\mathbf{X}(t)$ and consider the drift of a Lyapunov function. However, the Lyapunov function corresponding to the logarithmic $g(.)$ does not satisfy the Lipschitz like bounded drift condition C1 even though the queue lengths have bounded increments.

Typically, if α-MaxWeight algorithm is used (i.e., one where the weight for the queue of type m jobs at server i is $\overline{\mathbf{Q}}_{mi}^{\alpha}$ with $\alpha > 1$) corresponding to the Lyapunov function $V_\alpha(\overline{\mathbf{Q}}) = \sum_{i,m}(\overline{\mathbf{Q}}_{mi})^{(1+\alpha)}$, one can modify this and use the corresponding $(1+\alpha)$ norm by considering the new Lyapunov function $U_\alpha(\overline{\mathbf{Q}}) = (\sum_{i,m}(\overline{\mathbf{Q}}_{mi})^{(1+\alpha)})^{\frac{1}{1+\alpha}}$ [23]. Since this is a norm on $\mathbb{R}^{LM}$, this has the bounded drift property. One can then obtain the drift of $U_\alpha(.)$ in terms of the drift of $V_\alpha(.)$.

Here, we don't have a norm corresponding to the logarithmic Lyapunov function. So, we define a new Lyapunov function $U(.)$ as follows. Note that $G(.)$ is a strictly increasing function on the domain $[0,\infty)$, $G(0) = 0$ and $G(q) \to \infty$ as $q \to \infty$. So, $G(.)$ is a bijection and its inverse, $G^{-1}(.) : [0,\infty) \to [0,\infty)$ is well-defined.

$$U(\overline{\mathbf{Q}}) = G^{-1}(V(\overline{\mathbf{Q}})) = G^{-1}(\sum_{i,m} G(\overline{\mathbf{Q}}_{mi}))$$

This is related to the Lambart W function which is defined as the inverse of xe^x that is studied in literature.

We will need the following Lemma relating the drift of the Lyapunov functions $U(.)$ and $V(.)$.

Lemma 4: For any two nonnegative and nonzero vectors $\overline{\mathbf{Q}}^{(1)}$ and $\overline{\mathbf{Q}}^{(2)}$,

$$U(\overline{\mathbf{Q}}^{(2)}) - U(\overline{\mathbf{Q}}^{(1)}) \leq \frac{V(\overline{\mathbf{Q}}^{(2)}) - V(\overline{\mathbf{Q}}^{(1)})}{\log(1 + U(\overline{\mathbf{Q}}^{(1)}))}$$

The proof of this Lemma is based on concavity of $U(.)$ and is similar to the one in [23]. The proof is presented in Appendix A.

We will need the following Lemma to verify the conditions C1 and C2 in Theorem 2.

Lemma 5: For any nonnegative queue length vector $\overline{\mathbf{Q}}$,

$$\frac{1}{LM}\sum_{i,m}\log(1+\overline{\mathbf{Q}}_{mi}) \leq \log(1 + G^{-1}(V(\overline{\mathbf{Q}})))$$

$$\leq 1 + \sum_{i,m} \log(1 + \overline{\mathbf{Q}}_{mi})$$

The proof of this Lemma is presented in the Appendix B.

We will also need the following general form of Wald's identity.

Theorem 3 (Generalized Wald's Identity): Let $\{X_n : n \in \mathbb{N}\}$ be a sequence of real-valued random variables and let N be a nonnegative integer-valued random variable. Assume that

D1 $\{X_n\}_{n\in\mathbb{N}}$ are all integrable (finite-mean) random variables

D2 $E[X_n \mathbb{I}_{\{N\geq n\}}] = E[X_n]P(N \geq n)$ for every natural number n, and

D3 $\sum_{n=1}^{\infty} E[|X_n|\mathbb{I}_{\{N\geq n\}}] < \infty$.

Then the random sums $S_N \triangleq \sum_{n=1}^{N} X_n$ and $T_N \triangleq \sum_{n=1}^{N} E[X_n]$ are integrable and $E[S_N] = E[T_N]$.

We will state and prove the main proposition of this section, which establishes the throughput optimality of Algorithm 1 when $g(q) = \log(1+q)$.

Proposition 3: Assume that the job size distribution satisfies Assumption 1. Then, any job load vector that satisfies $(\lambda, \overline{S}) \in int(\mathcal{C})$ is supportable under JSQ routing and myopic MaxWeight allocation as described in Algorithm 1 with $g(q) = \log(1+q)$.

Proof: When the queue length vector is $\mathbf{Q}_{mi}(t)$, let $\mathbf{Y}(t) = \{\mathbf{Y}_{mi}(t)\}_{m,i}$ denote the state of jobs of type-m at server i. When $\mathbf{Q}_{mi}(t) \neq 0$, $\mathbf{Y}_{mi}(t)$ is a vector of size $N_m^{(i)}(t)$ and $\mathbf{Y}_{mi}^{j}(t)$ is the amount of time the j^{th} type-m job that is in service at server i has been scheduled.

It is easy to see that $\mathbf{X}(t) = (\mathbf{Q}(t), \mathbf{Y}(t))$ is a Markov chain under Algorithm 1.

We will show stability of $\mathbf{X}(t)$ by first showing that the Markov Chain $\widetilde{\mathbf{X}}(n)$ corresponding to the sampled system is stable, as in the proof of Geometric case.

With slight abuse of notation, we will use $V(t)$ for $V(\overline{\mathbf{Q}}(t))$. Similarly, $V(n)$, $U(t)$, $U(n)$ and $U(\widetilde{\mathbf{X}}(n))$. We will establish this result by showing that the Lyapunov function $U(n)$ satisfies both the conditions of Theorem 2. We will study the drift of $U(n)$ in terms of drift of $V(n)$ using Lemma 4. First consider the following one step drift of $V(t)$.

$$\begin{aligned}
&(V(t+1) - V(t))\\
=&\sum_{m,i} \left(G\left(\overline{\mathbf{Q}}_{mi}(t+1)\right) - G\left(\overline{\mathbf{Q}}_{mi}(t)\right)\right)\\
\leq&\sum_{m,i} \left(\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t)\right) g(\overline{\mathbf{Q}}_{mi}(t+1)) \qquad (13)\\
=&\sum_{m,i} \left(\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t)\right)\left(g(\overline{\mathbf{Q}}_{mi}(t+1)) - g(\overline{\mathbf{Q}}_{mi}(t))\right)\\
&+\sum_{m,i} \left(\overline{\mathbf{A}}_{mi}(t) - \overline{\mathbf{D}}_{mi}(t)\right) g(\overline{\mathbf{Q}}_{mi}(t)) \qquad (14)
\end{aligned}$$

where (13) follows from the convexity of $G(.)$. To bound the first term in (14), first consider the case when $\overline{\mathbf{Q}}_{mi}(t+1) \geq \overline{\mathbf{Q}}_{mi}(t)$. Since $g(.)$ is strictly increasing and concave, we have

$$\begin{aligned}
&\left|g(\overline{\mathbf{Q}}_{mi}(t+1)) - g(\overline{\mathbf{Q}}_{mi}(t))\right|\\
&= g(\overline{\mathbf{Q}}_{mi}(t+1)) - g(\overline{\mathbf{Q}}_{mi}(t))\\
&\leq g'(\overline{\mathbf{Q}}_{mi}(t))(\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t))\\
&\leq (\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t))\\
&= \left|\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t)\right|
\end{aligned}$$

where the second inequality follows from $g'(.) \leq 1$. Similarly, we get the same relation even when $\overline{\mathbf{Q}}_{mi}(t) > \overline{\mathbf{Q}}_{mi}(t+1)$ as follows.

$$\begin{aligned}
&\left|g(\overline{\mathbf{Q}}_{mi}(t+1)) - g(\overline{\mathbf{Q}}_{mi}(t))\right|\\
&= g(\overline{\mathbf{Q}}_{mi}(t)) - g(\overline{\mathbf{Q}}_{mi}(t+1))\\
&\leq g'(\overline{\mathbf{Q}}_{mi}(t+1))(\overline{\mathbf{Q}}_{mi}(t) - \overline{\mathbf{Q}}_{mi}(t+1))\\
&\leq (\overline{\mathbf{Q}}_{mi}(t) - \overline{\mathbf{Q}}_{mi}(t+1))\\
&= \left|\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t)\right|.
\end{aligned}$$

So the first term in (14) can be bounded as

$$\begin{aligned}
&\sum_{m,i} \left(\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t)\right)\left(g(\overline{\mathbf{Q}}_{mi}(t+1)) - g(\overline{\mathbf{Q}}_{mi}(t))\right)\\
\leq&\sum_{m,i} \left|\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t)\right| \left|g(\overline{\mathbf{Q}}_{mi}(t+1)) - g(\overline{\mathbf{Q}}_{mi}(t))\right|\\
\leq&\sum_{m,i} \left|\left(\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t)\right)\right|^2\\
\leq&K_4.
\end{aligned}$$

where $K_4 = LM(A_{\max}\overline{S}_{\max} + \frac{N_{\max}}{C})^2$. The last inequality follows from (10) and (11). Thus, we have

$$V(t+1) - V(t) \leq K_4 + \sum_{m,i} (\overline{\mathbf{A}}_{mi}(t) - \overline{\mathbf{D}}_{mi}(t)) g(\overline{\mathbf{Q}}_{mi}(t)) \qquad (15)$$

Similarly, it can be shown that

$$V(t) - V(t+1) \leq K_4 + \sum_{m,i} \left(\overline{\mathbf{D}}_{mi}(t) - \overline{\mathbf{A}}_{mi}(t)\right) g(\overline{\mathbf{Q}}_{mi}(t+1)) \qquad (16)$$

Let t_n denote the last refresh time up to t. Let $t = t_n + \tau$ for $0 \leq \tau < z_n$. Again, we use $E_{\mathbf{q}}[(.)]$ to denote $E[(.)|\mathbf{Q}(t_n) = \mathbf{q}, \mathbf{Y}(t_n)]$. Now using (15) in the drift of the sampled system, we get

$$\begin{aligned}
&E[V(\widetilde{\mathbf{X}}(n+1)) - V(\widetilde{\mathbf{X}}(n))|\widetilde{\mathbf{Q}}(n) = \mathbf{q}, \widetilde{\mathbf{Y}}(n)]\\
=&E[V(t_{n+1}) - V(t_n)|\mathbf{Q}(t_n) = \mathbf{q}, \mathbf{Y}(t_n)]\\
=&E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1} V(t_n+\tau+1) - V(t_n+\tau)\right]\\
\leq&E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\left(\sum_{m,i}\left(g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{A}}_{mi}(t_n+\tau)\right.\right.\right.\\
&\left.\left.\left.-g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\right)\right) + K_4\right] \qquad (17)
\end{aligned}$$

The last term above is bounded by $K_4\mathcal{K}_1$ from Lemma 1. We will now bound the first term in (17).

$$\begin{aligned}
&E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_{m}\sum_{i} g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{A}}_{mi}(t_n+\tau)\right]\\
=&E_{\mathbf{q}}[\sum_{\tau=0}^{z_n-1}\sum_{m} g(\overline{\mathbf{Q}}_{mi^*_m(t_n+\tau)}(t_n+\tau))A_m(t_n+\tau)\overline{S}_m]
\end{aligned}$$

$$\overset{(a)}{\leq} E_{\mathbf{q}}[\sum_{\tau=0}^{z_n-1}\sum_m g(\overline{\mathbf{Q}}_{mi^*_m}(t_n)+\tau A_{\max}\overline{S}_m)A_m(t_n+\tau)\overline{S}_m]$$
$$\overset{(b)}{\leq} E_{\mathbf{q}}[\sum_{\tau=0}^{z_n-1}\sum_m g(\overline{\mathbf{Q}}_{mi^*_m}(t_n))A_m(t_n+\tau)\overline{S}_m+\tau A^2_{\max}\overline{S}^2_m]$$
$$\leq \sum_m \overline{S}_m g(\overline{\mathbf{q}}_{mi^*_m})E[\sum_{\tau=0}^{z_n-1}A_m(t_n+\tau)]+\sum_m A^2_{\max}\overline{S}^2_m E[z_n^2]$$
$$\overset{(c)}{\leq} A^2_{\max}\mathcal{K}_2\sum_m \overline{S}^2_m + E[z_n]\sum_m \overline{S}_m g(\overline{\mathbf{q}}_{mi^*_m})\lambda_m$$
$$\leq K_5 + E[z_n]\sum_m \overline{S}_m g(\overline{\mathbf{q}}_{m\hat{i}_m})\lambda_m \quad (18)$$

where $i^*_m(t) = \arg\min_{i\in\{1,2,,,L\}} \mathbf{Q}_{mi}(t)$, $i^*_m = i^*_m(t_n)$, $\hat{i}_m(t) = \arg\min_{i\in\{1,2,,,L\}} \overline{\mathbf{Q}}_{mi}(t)$, $\hat{i}_m = \hat{i}_m(t_n)$ and $K_5 = A^2_{\max}\mathcal{K}_2\sum_m \overline{S}^2_m + \mathcal{K}_1\sum_m \overline{S}_m \log(\widetilde{C})\lambda_m$. The first equality follows from the definition of $\mathbf{A}_{mi}$ in the routing algorithm in (1). Since $\overline{\mathbf{Q}}_{mi^*_m(t_n+\tau)}(t_n+\tau) \leq \overline{\mathbf{Q}}_{mi^*_m(t_n)}(t_n+\tau) \leq \overline{\mathbf{Q}}_{mi^*_m}(t_n) + \overline{S}_m A_{\max}\tau$ because the load at each queue cannot increase by more than $A_{\max}\overline{S}_m$ in each time slot, we get (a). Inequality (b) follows from concavity of $g(.)$ and $g'(.) \leq 1$.

Inequality (c) follows from Wald's identity (Theorem 1)and Lemma 1. For Wald's Identity, we let $\mathcal{F}_t$ be the filtration generated by the process $\mathbf{Y}(t)$. Then, $A(t+1)$ is independent of $\mathcal{F}_t$ and z_n is a stopping time for $\mathcal{F}_t$. Note that Lemma 2 gives $\overline{\mathbf{q}}_{mi^*_m} \leq \mathbf{q}_{mi^*_m}\widetilde{C} \leq \mathbf{q}_{m\hat{i}_m}\widetilde{C} \leq \overline{\mathbf{q}}_{m\hat{i}_m}\widetilde{C}$. This gives (18).

Now we will bound the second term in (17). Though we use a fixed configuration between two refresh times, there may be some unused service when the corresponding queue length is small. We will first bound the unused service. Let $\overline{\mathcal{D}}^{(j)}_{mi}(t)$ be the departure in workload at queue $\overline{\mathbf{Q}}_{mi}(t)$ due to the j^{th} job of type m in the configuration $\widetilde{N}^{(i)}_m(t_n)$ so that

$$\overline{\mathbf{D}}_{mi}(t) = \sum_{j=1}^{\widetilde{N}^{(i)}_m(t_n)} \overline{\mathcal{D}}^{(j)}_{mi}(t)$$

Define a fictitious departure process to account for the unused service as follows.

$$\widehat{\mathcal{D}}^{(j)}_{mi}(t)=\begin{cases}\overline{\mathcal{D}}^{(j)}_{mi}(t) & \text{if the } j^{\text{th}} \text{ job in } \widetilde{N}^{(i)}_m(t_n)\text{was scheduled}\\ 1 & \text{otherwise, i.e., if the } j^{\text{th}} \text{ job was unused.}\end{cases} \quad (19)$$

$$\widehat{\mathbf{D}}_{mi}(t) = \sum_{j=1}^{\widetilde{N}^{(i)}_m(t_n)} \widehat{\mathcal{D}}^{(j)}_{mi}(t) \quad (20)$$

Using $\widehat{\mathbf{D}}_{mi}(t) - \overline{\mathbf{D}}_{mi}(t) \leq N_{\max}$, we get

$$\begin{aligned} &g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\\ &= \left(g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t_n+\tau)<N_{\max}}\\ &\quad + \left(g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t_n+\tau)\geq N_{\max}}\\ &\overset{(a)}{\geq}\left(g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\left(\widehat{\mathbf{D}}_{mi}(t_n+\tau)-N_{\max}\right)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t_n+\tau)<N_{\max}}\\ &\quad + \left(g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\widehat{\mathbf{D}}_{mi}(t_n+\tau)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t_n+\tau)\geq N_{\max}}\\ &\overset{(b)}{\geq} g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\widehat{\mathbf{D}}_{mi}(t_n+\tau) - N_{\max}g(\widetilde{C}N_{\max}) \end{aligned} \quad (21)$$

Since that there is no unused service if $\mathbf{Q}_{mi}(t) \geq N_{\max}$, we have (a). Inequality (b) follows from the fact that $\overline{\mathbf{Q}}_{mi}(t) \leq \mathbf{Q}_{mi}(t)\widetilde{C}$ from Lemma 2 and $N^{(i)}_m(t) \leq N_{\max}$.

Since g is concave and $0 \leq g'(.) \leq 1$, we have

$$\begin{aligned} g(\overline{\mathbf{Q}}_{mi}(t_n)) \leq& g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\\ &+ g'(\overline{\mathbf{Q}}_{mi}(t_n+\tau))(\overline{\mathbf{Q}}_{mi}(t_n) - \overline{\mathbf{Q}}_{mi}(t_n+\tau))\\ \leq& g(\overline{\mathbf{Q}}_{mi}(t_n+\tau)) + |\overline{\mathbf{Q}}_{mi}(t_n) - \overline{\mathbf{Q}}_{mi}(t_n+\tau)|\\ \leq& g(\overline{\mathbf{Q}}_{mi}(t_n+\tau)) + \tau(A_{\max}\overline{S}_{\max} + N_{\max}/C) \end{aligned}$$

where the last in follows from (10) and (11). Then, using $\widehat{\mathbf{D}}_{mi}(t_n+\tau) \leq N_{\max}/C$ from (11) in (21), we get

$$\begin{aligned} &g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\\ \geq& g(\overline{\mathbf{Q}}_{mi}(t_n))\widehat{\mathbf{D}}_{mi}(t_n+\tau) - K_6\tau - N_{\max}g(\widetilde{C}N_{\max}) \end{aligned}$$

where $K_6 = (N_{\max}/C)((A_{\max}\overline{S}_{\max}+N_{\max}/C)$. Then, using Lemma 1, the second term in (17) can be bounded as follows

$$\begin{aligned} &E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_{i,m} g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\right]\\ \geq& E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_{i,m} g(\overline{\mathbf{Q}}_{mi}(t_n))\widehat{\mathbf{D}}_{mi}(t_n+\tau)\right] - K_7\\ =& \sum_{i,m} g(\overline{\mathbf{q}}_{mi})E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\widehat{\mathbf{D}}_{mi}(t_n+\tau)\right] - K_7 \end{aligned} \quad (22)$$

where $K_7 = LM(K_6\mathcal{K}_2 + N_{\max}g(N_{\max})\mathcal{K}_1)$. We will now use the generalized Wald's Identity (Theorem 3), verifying conditions (D1), (D2) and (D3). Clearly, (D1) is true because $\widehat{\mathbf{D}}_{mi}(t_n+\tau)$ have finite mean by definition, and from Lemma 3.

From definition of $\widehat{\mathbf{D}}_{mi}(t_n+\tau)$, from (8) and (9), $|\widehat{\mathbf{D}}_{mi}(t_n+\tau)| \leq N_{\max}/C$. So,

$$\begin{aligned} \sum_{\tau=1}^{\infty} E_{\mathbf{q}}\left[|\widehat{\mathbf{D}}_{mi}(t_n+\tau)|\mathbb{I}_{\{z_n\geq\tau\}}\right] \leq& \frac{N_{\max}}{C}\sum_{\tau=1}^{\infty}E_{\mathbf{q}}\left[\mathbb{I}_{\{z_n\geq\tau\}}\right]\\ =& \frac{N_{\max}}{C}\sum_{\tau=1}^{\infty}P_{\mathbf{q}}(z_n\geq\tau)\\ =& \frac{N_{\max}}{C}E_{\mathbf{q}}[z_n]\\ \leq& \frac{N_{\max}\mathcal{K}_1}{C}. \end{aligned}$$

This verifies (D3). We verify (D2) by first proving the following claim.

Claim 1:

$$E_{\mathbf{q}}\left[\widehat{\mathbf{D}}_{mi}(t_n+\tau)|z_n\geq\tau\right] = E_{\mathbf{q}}\left[\widehat{\mathbf{D}}_{mi}(t_n+\tau)\right]$$

Proof: Consider the departures due to each job, $\widehat{\mathcal{D}}^{(j)}_{mi}(t)$ as defined in (19). Intuitively, conditioned on $\{z_n \geq \tau\}$, we have a conditional distribution on the amount of finished service for each of the jobs. However, from Lemma 3, the expected departure is 1 independent of finished service. Thus, the conditional workload departure due to each job is 1. This is the same as the unconditional departure, again from Lemma 3. We will now prove this more formally.

The event $\{z_n \geq \tau\}$ is a union of many (but finite) disjoint events $\{E_\alpha : \alpha = 1 \ldots \mathcal{A}\}$. Each of these events E_α is of the form $\{(\mathbf{q}(t_n), \mathbf{A}(t_n), \mathbf{D}(t_n)), (\mathbf{q}(t_n+1), \mathbf{A}(t_n+1), \mathbf{D}(t_n+1)), \ldots (\mathbf{q}(t_n+\tau-1), \mathbf{A}(t_n+\tau-1), \mathbf{D}(t_n+\tau-1))\}$. In other words, each event is a sample path of the system up to time $t_n+\tau$ and contains complete information about the evolution of the system from time t_n up to time $t_n+\tau$. Let $l_{mi}^{(j)}$ be the amount of finished service for the j^{th} job of type m at server i. $l_{mi}^{(j)}$ is completely determined by E_α. Conditioned on E_α, $\widehat{\mathcal{D}}_{mi}^{(j)}(t)$ depends only on $l_{mi}^{(j)}$. It is independent of the other jobs in the system, and is also independent of the past departures. Thus, we have

$$\begin{aligned}E_{\mathbf{q}}\left[\widehat{\mathcal{D}}_{mi}^{(j)}(t_n+\tau)|E_\alpha\right] =& E_{\mathbf{q}}\left[\widehat{\mathcal{D}}_{mi}^{(j)}(t_n+\tau)|l_{mi}^{(j)}\right]\\ =&1\end{aligned}$$

The last inequality follows from Lemma 3 and definition of $\widehat{\mathcal{D}}_{mi}^{(j)}(t)$ in terms of $\overline{\mathcal{D}}_{mi}^{(j)}(t)$ (19). Since E_α are disjoint, we have

$$\begin{aligned}&E_{\mathbf{q}}\left[\widehat{\mathcal{D}}_{mi}^{(j)}(t_n+\tau)|z_n \geq \tau\right]\\ =&\sum_\alpha P(E_\alpha|z_n \geq \tau) E_{\mathbf{q}}\left[\widehat{\mathcal{D}}_{mi}^{(j)}(t_n+\tau)|E_\alpha\right]\\ =&\sum_\alpha P(E_\alpha|z_n \geq \tau) \qquad = 1\end{aligned}$$

Similarly, from Lemma 3 and (19), we have $E_{\mathbf{q}}\left[\widehat{\mathcal{D}}_{mi}^{(j)}(t_n+\tau)\right] = 1$. Summing over j, from (20), we have the claim. ■

Since

$$\begin{aligned}E_{\mathbf{q}}\left[\widehat{\mathbf{D}}_{mi}(t_n+\tau)\mathbb{I}_{z_n\geq\tau}\right] =& E_{\mathbf{q}}\left[\widehat{\mathbf{D}}_{mi}(t_n+\tau)|z_n \geq \tau\right] P(z_n \geq \tau)\\ =& E_{\mathbf{q}}\left[\widehat{\mathbf{D}}_{mi}(t_n+\tau)\right] P(z_n \geq \tau),\end{aligned}$$

we have (D2). Therefore, using Generalized Wald's Identity (Theorem 3) in (22), we have

$$\begin{aligned}&E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_{i,m} g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\right]\\ &\qquad \geq \sum_{i,m} g(\overline{\mathbf{q}}_{mi}) E_{\mathbf{q}}[z_n] \widetilde{N}_m^{(i)}(t_n) - K_7 \quad (23)\end{aligned}$$

The key idea is to note that the expected departures of workload for each scheduled job is 1 from Lemma (3). We use this, along with the generalized Wald's theorem to bound the departures similar to the case of geometric job sizes.

Since $(\lambda, \overline{S}) \in \mathcal{C}$, there exists $\{\lambda^i\}_i$ such that $\lambda = \sum_i \lambda^i$ and $\lambda^i \circ \overline{S} \in int(\text{Conv}(\mathcal{N}_i))$ for all i . Then, there exists an an $\epsilon > 0$ such that $(\lambda^i + \epsilon) \circ \overline{S} \in \text{Conv}(\mathcal{N}_i)$ for all i. From Lemma 2, we have $g(\overline{\mathbf{q}}_{mi}) \leq g(\widetilde{C}\mathbf{q}_{mi}) \leq \log(\widetilde{C}(1+\mathbf{q}_{mi})) \leq g(\mathbf{q}_{mi}) + \log(\widetilde{C})$. The last inequality which is an immediate consequence of the log function has also been exploited in [24] [25]for a different problem. For each server i, we have

$$\begin{aligned}\sum_m (g(\overline{\mathbf{q}}_{mi}) - \log(\widetilde{C}))(\lambda_m^i+\epsilon)\overline{S}_m \leq& \sum_m g(\mathbf{q}_{mi})(\lambda_m^i+\epsilon)\overline{S}_m\\ \overset{(a)}{\leq}& \sum_m g(\mathbf{q}_{mi}) \widetilde{N}_m^{(i)}(t_n)\\ \leq& \sum_m g(\overline{\mathbf{q}}_{mi}) \widetilde{N}_m^{(i)}(t_n)\end{aligned}$$

where (a) follows from the Algorithm 1 since $\widetilde{N}_m^{(i)}(t_n)$ is chosen according to MaxWeight policy. The last inequality again follows from Lemma 2. Substituting this in (23) and from (18) and (17), we get

$$\begin{aligned}&E[V(\widetilde{\mathbf{X}}(n+1)) - V(\widetilde{\mathbf{X}}(n))|\widetilde{\mathbf{Q}}(n) = \mathbf{q}, \widetilde{\mathbf{Y}}(n)]\\ \leq& K_8 + E_{\mathbf{q}}[z_n]\sum_m\left(g(\overline{\mathbf{q}}_{m\hat{i}_m})\lambda_m\overline{S}_m - \sum_i g(\overline{\mathbf{q}}_{mi})(\lambda_m^i+\epsilon)\overline{S}_m\right)\\ \overset{(a)}{\leq}& K_8 - \epsilon\overline{S}_{\min}E_{\mathbf{q}}[z_n]\sum_i\sum_m g(\overline{\mathbf{q}}_{mi})\\ \leq& K_9 - \epsilon\overline{S}_{\min}\log(1+G^{-1}(V(\overline{\mathbf{q}})))\end{aligned}$$

where $K_8 = K_4\mathcal{K}_1 + K_5 + K_7 + \log(\widetilde{C})\sum_m(\lambda_m + L\epsilon)\overline{S}_m$ and $K_9 = K_8 + \epsilon\overline{S}_{\min}\mathcal{K}_1$. Inequality (a) follows from $\lambda = \sum_i \lambda^i$ and $\overline{\mathbf{q}}_{m\hat{i}_m} \leq \overline{\mathbf{q}}_{mi}$. The last inequality follows from Lemma 5 and since $z_n \geq 1$.

If the packet sizes were bounded, we can find a finite set of states $\mathcal{B} = \{\mathbf{x} : \sum_m\sum_i g(\overline{\mathbf{q}}_{mi}) < \mathcal{M}\}$ so that the drift is negative whenever $\mathbf{x} \in \mathcal{B}^c$. Then, similar to the proof in Section IV, Foster-Lyapunov theorem can be used to show that the sampled Markov Chain $\widetilde{\mathbf{X}}(n)$ is positive recurrent. We need the bounded packet size assumption here because, if not, the set $\mathcal{B}$ could then be infinite since for each $\mathbf{q}$ there are infinite possible values of state $\mathbf{x} = (\mathbf{q}, \mathbf{y})$ with different values of $\mathbf{y}$.

Since the packet sizes are not bounded in general, we will use Theorem 2 to show stability of Algorithm 1 for the random process $U(n)$. From Lemma 4, we have

$$\begin{aligned}&E[U(\widetilde{\mathbf{X}}(n+1)) - U(\widetilde{\mathbf{X}}(n))|\widetilde{\mathbf{X}}(n) = \mathbf{x} = (\mathbf{q}, \mathbf{y})]\\ \leq& E\left[\left.\frac{V(\widetilde{\mathbf{X}}(n+1)) - V(\widetilde{\mathbf{X}}(n))}{\log(1+U(\widetilde{\mathbf{X}}(n))}\right|\widetilde{\mathbf{X}}(n) = \mathbf{x} = (\mathbf{q}, \mathbf{y})\right]\\ \leq& \frac{K_9}{\log(1+U(\overline{\mathbf{q}}))} - \epsilon\overline{S}_{\min}\mathcal{K}_1 \quad \leq -\frac{\epsilon\overline{S}_{\min}\mathcal{K}_1}{2}\end{aligned}$$

whenever $U(\overline{\mathbf{q}}) > e^{(2K_9/\epsilon\overline{S}_{\min}\mathcal{K}_1)}$. Thus, $U(n)$ satisfies condition C1 of Theorem 2 for the filtration generated by the $\{\widetilde{\mathbf{X}}(n)\}$. From Lemma 4, Lemma 5 and (15), we have

$$\begin{aligned}&(U(t_n+\tau+1) - U(t_n+\tau))\\ \leq& \frac{[V(t_n+\tau+1) - V(t_n+\tau)]}{\log(1+G^{-1}(V(\overline{\mathbf{Q}}(t_n+\tau))))}\\ \leq& \frac{K_4 + A_{\max}\sum_{m,i}\overline{S}_m g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))}{\log(1+G^{-1}(V(\overline{\mathbf{Q}}(t_n+\tau))))}\\ \leq& \frac{K_4}{\log(1+G^{-1}(V(\overline{\mathbf{Q}}(t_n+\tau))))} + \frac{A_{\max}\overline{S}_{\max}}{LM}\\ \overset{(a)}{\leq}& K_{10} \quad \text{ if } U(t_n+\tau) > 0\end{aligned}$$

where $K_{10} = \frac{K_4}{\log(2)} + \frac{A_{\max}\overline{S}_{\max}}{LM}$. Since $U(\overline{\mathbf{Q}}) > 0$ if and only if $V(\overline{\mathbf{Q}}) > 0$ if and only if $\overline{\mathbf{Q}} \neq 0$, there is at least one nonzero component of $\overline{\mathbf{Q}} = 0$ and so $V(t_n+\tau) > G(1)$. This gives the inequality (a). If $U(t_n+\tau) = 0$, from (15), we have $(U(t_n+\tau+1) - U(t_n+\tau)) \leq K_{11} \triangleq G^{-1}(K_4)$. Thus, we have

$$(U(t_n+\tau+1) - U(t_n+\tau)) \leq K_{12}$$

where $K_{12} = \max\{K_{10}, K_{11}\}$. Similarly, from (16) it can be shown that

$$(U(t_n+\tau) - U(t_n+\tau+1)) \leq K_{14}$$

where $K_{14} = \max\{K_{13}, K_{11}\}$ and $K_{13} = \frac{K_4}{\log(2)} + \frac{N_{\max}}{LM}$. Setting $K_{15} = \max\{K_{12}, K_{14}\}$, we have

$$\begin{aligned}(|U(t_n+\tau) - U(t_n+\tau+1)|) &\leq K_{15}\\ (\,|U(t_n+\tau) - U(t_n+\tau+1)|\,|\,\mathbf{X}(t_n)) &\leq K_{15}\\ \left(|U(\widetilde{\mathbf{X}}(n+1)) - U(\widetilde{\mathbf{X}}(n))|\,\middle|\,\widetilde{\mathbf{X}}(n)\right) &\leq K_{15}z_n\\ &\leq K_{15}\overline{z}_n\end{aligned}$$

where $\overline{z}_n$ is the coupled random variable constructed in the proof of Lemma 1. Since $\overline{z}_n$ is a geometric random variable by construction, it satisfies condition C1 in Theorem 2. Thus, we have that there are constants $\theta^* > 0$ and $\mathcal{K}_4 > 0$ such that, $\lim_{n\to\infty}\sum_m\sum_i E[e^{\theta^* U(\widetilde{\mathbf{X}}(n))}] \leq \mathcal{K}_4$. Since $G(.)$ is convex, from Jensen's inequality, we have

$$\begin{aligned}G\left(\frac{1}{LM}\sum_m\sum_i\overline{\mathbf{Q}}_{mi}(t_n)\right) \leq& \frac{1}{LM}\sum_m\sum_i G\left(\overline{\mathbf{Q}}_{mi}(t_n)\right)\\ \leq& V(\overline{\mathbf{Q}}(t_n)) \qquad (24)\end{aligned}$$

$$\begin{aligned}\sum_m\sum_i\mathbf{Q}_{mi}(t_n) \overset{(a)}{\leq}& \sum_m\sum_i\overline{\mathbf{Q}}_{mi}(t_n)\\ \overset{(b)}{\leq}& (LM)U(\overline{\mathbf{Q}}(t_n))\\ \leq& \frac{LM}{\theta^*}e^{\theta^* U(\widetilde{\mathbf{X}}(n))}\end{aligned}$$

where (a) follows from Lemma 2 and (b) follows from (24). Thus, we have $\lim_{n\to\infty}\sum_m\sum_i E[\mathbf{Q}_{mi}(t_n)] \leq \frac{LM}{\theta^*}\mathcal{K}_4$.

For any $t > 0$, if t_{n+1} is the next refresh time after t, from (11) we have

$$\begin{aligned}\mathbf{Q}_{mi}(t_{n+1}) \geq \mathbf{Q}_{mi}(t) - z_n\frac{N_{\max}}{C}\\ \sum_m\sum_i E[\mathbf{Q}_{mi}(t)] \leq \sum_m\sum_i E[(\mathbf{Q}_{mi}(t_{n+1}) + z_n N_{\max}/C)]\end{aligned}$$

As $t \to \infty$, we get

$$\begin{aligned}\limsup_{t\to\infty}\sum_{m,i}E[\mathbf{Q}_{mi}(t)] \leq& \limsup_{n\to\infty}\sum_{m,i}E\left[\mathbf{Q}_{mi}(t_n) + z_n\frac{N_{\max}}{C}\right]\\ \leq& \frac{LM}{\theta^*}\mathcal{K}_4 + \frac{\mathcal{K}_1 LMN_{\max}}{C}.\end{aligned}$$

■

A centralized queuing architecture was considered in [5]. In such a model, all the jobs are queued at a central location and all the servers serve jobs from the same queues. There are no queues at the servers. The scheduling algorithm in Algorithm 1 can be used in this case with each server using the central queue lengths for the MaxWeight policy. It can be shown that this algorithm is throughput optimal. The proof is similar to that of Proposition 3 and so we skip it.

VI. Simulations

According to Algorithm 1, each server performs MaxWeight scheduling only at refresh times. At other times, it uses the same schedule as before. In this section, we present two heuristic algorithms motivated by Algorithm 1. It is open whether these algorithms are throughput-optimal, therefore we study them through simulations.

At refresh times, a MaxWeight schedule is chosen at each server. At all other times, each server tries to choose a MaxWeight schedule greedily. It does not preempt the jobs that are in service. It adds new jobs to the existing configuration so as to maximize the weight using $g(\mathbf{Q}_{mi}(t))$ as weight. This algorithm tries to emulate a MaxWeight schedule in every time slot by greedily adding new jobs with higher priority to long queues. This algorithm has the advantage that, at refresh times an exact MaxWeight schedule is chosen automatically. Each server does not need to know departure information of other servers and so is independent of other servers. We call this Algorithm 2. It is not clear if this algorithm is throughput optimal. The proof in Section V is not applicable here, because one cannot use Wald's identity to bound the drift.

An alternate approach is to use *local refresh times*. For server m, a *local refresh time* is a time when all the jobs that are in service at server m finish their service simultaneously. Thus, if a time instant is a local refresh time for all the servers, it is a (global) refresh time for the system.

Consider the following Algorithm 3. Each server chooses a MaxWeight schedule only at local refresh times. Between the local refresh times, a server maintains the same configuration. Again, it is not clear if this is throughput optimal. Each server may have multiple local refresh times between two (global) refresh times. Since the schedule changes at these refresh times, again the approach in Section V is not applicable.

In this section, we use simulations to compare the performance of these two algorithms. Motivated by the Amazon EC2 example in [5], we consider a data center with 100 identical servers, and three types of jobs. The resource constraints are such that $(2,0,0)$, $(1,0,1)$, and $(0,1,1)$ are the three maximal VM configurations for each server. We consider two load vectors, $\lambda^{(1)} = (1, \frac{1}{3}, \frac{2}{3})$ and $\lambda^{(2)} = (1, \frac{1}{2}, \frac{1}{2})$ which are on the boundary of the capacity region of each server. $\lambda^{(1)}$ is a linear combination of all the three maximal schedules whereas $\lambda^{(2)}$ is a combination of two of the three maximal schedules.

We consider three different job size distributions. Distribution A is a bounded distribution which models the high variability in jobs sizes as follows: when a new job is generated, with probability 0.7, the size is an integer that is uniformly distributed between 1 and 50, with probability 0.15, it is an integer that is uniformly distributed between 251 and 300, and with probability 0.15, it is uniformly distributed between 451 and 500. Therefore, the average job size is 130.5.

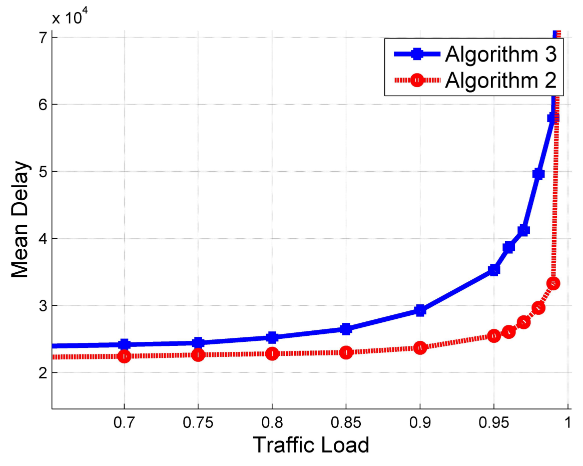

Fig. 1. Comparison of Mean delay using Algorithms 2 and 3 for load vector $\lambda^{(1)}$ and job size distribution A

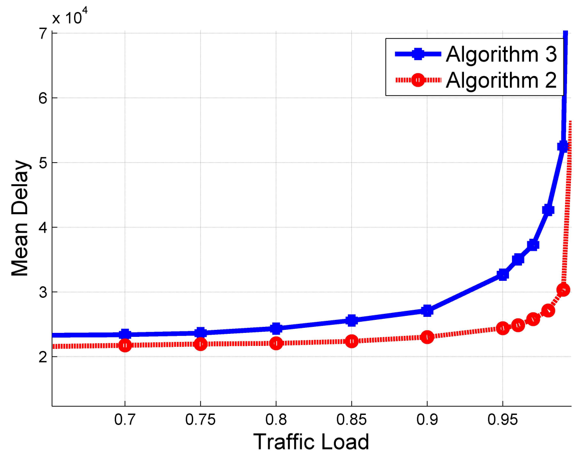

Fig. 2. Comparison of Mean delay using Algorithms 2 and 3 for load vector $\lambda^{(1)}$ and job size distribution B

Distribution B is a Geometric distribution with mean 130.5. Distribution C is a combination of Distributions A and B with equal probability, i.e., the size of a new job is sampled from Distribution A with probability $1/2$ and from Distribution B with probability $1/2$.

We further assume the number of type-i jobs arriving at each time slot follows a Binomial distribution with parameter $(\rho\frac{\lambda_i}{130.5}, 100)$.

Figures 1, 2 and 3 shows the mean delay of the jobs for both the algorithms with the three job size distributions using the load vector $\lambda^{(1)}$. We vary the parameter ρ to simulate different traffic intensities. Each simulation was run for one million time slots. Figures 4, 5 and 6 show the same results when the load vector $\lambda^{(2)}$ is used.

The simulations indicate that both the algorithms may be throughput optimal. For $\lambda^{(1)}$, Algorithm 2 has better delay performance at higher loads whereas for for $\lambda^{(2)}$, both the algorithms have similar delay performance. This indicates that using the greedy MaxWeight schedule is sometimes more efficient than using a static schedule between refresh times.

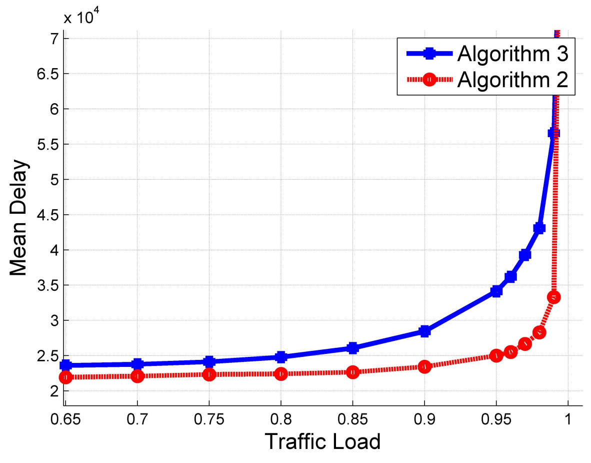

Fig. 3. Comparison of Mean delay using Algorithms 2 and 3 for load vector $\lambda^{(1)}$ and job size distribution C

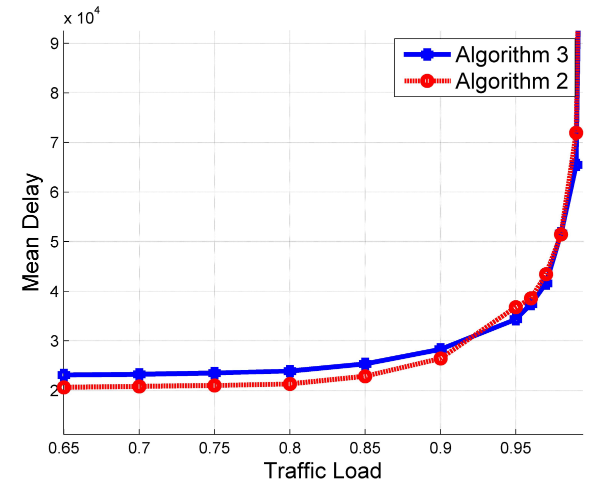

Fig. 4. Comparison of Mean delay using Algorithms 2 and 3 for load vector $\lambda^{(2)}$ and job size distribution A

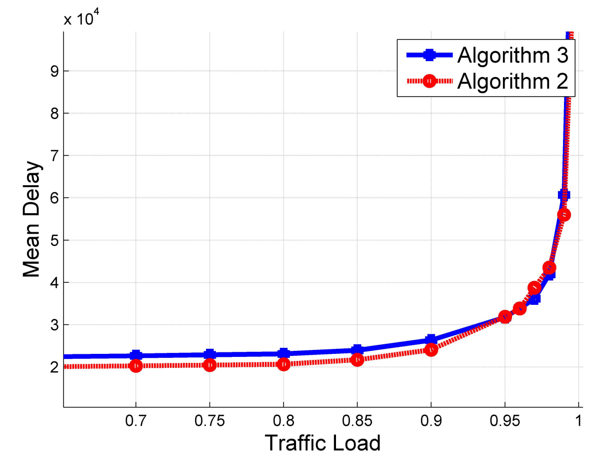

Fig. 5. Comparison of Mean delay using Algorithms 2 and 3 for load vector $\lambda^{(2)}$ and job size distribution B

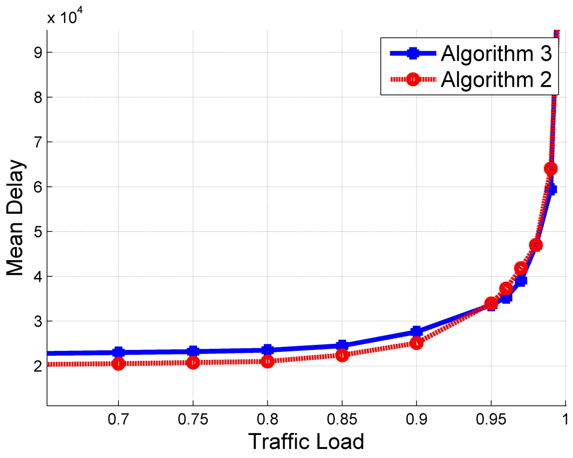

Fig. 6. Comparison of Mean delay using Algorithms 2 and 3 for load vector $\lambda^{(2)}$ and job size distribution C

VII. Conclusion

In this paper, we studied a MaxWeight scheduling and Join the Shortest Queue routing algorithm for a cloud computing data center. This is a nonpreemptive algorithm that can be implemented without knowing the job durations. A Maxweight schedule is chosen at every refresh time. The weights for the MaxWeight scheduling algorithm are chosen to be logarithmic functions of the queue lengths. We showed that the that the refresh times occur often enough. We used this to show that this algorithm is throughput optimal by showing that the drift of a lyapunov function is negative. We then presented two more natural algorithms and studied their performance using simulations.

VIII. Acknowledgements

We thank Weina Wang for her valuable comments on an earlier version of this paper.

References

[1] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. IEEE Infocom.*, 2010, pp. 1–9.
[2] Y. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady, "Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis," in *2010 IEEE 3rd International Conference on Cloud Computing*, 2010, pp. 91–98.
[3] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Transactions on Services Computing*, pp. 266–278, 2010.
[4] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 577–578.
[5] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *Proc. IEEE Infocom.*, 2012, pp. 702–710.
[6] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Contr.*, vol. 4, pp. 1936–1948, December 1992.
[7] S. T. Maguluri, R. Srikant, and L. Ying, "Heavy traffic optimal resource allocation algorithms for cloud computing clusters," in *International Teletraffic Congress*, 2012, pp. 1–8.
[8] A. Stolyar, "An infinite server system with general packing constraints," *Arxiv preprint arXiv:1205.4271*, 2012.
[9] T. Bonald and D. Cuda, "Rate-optimal scheduling schemes for asynchronous input-queued packet switches," in *ACM Sigmetrics MAMA Workshop*, 2012.
[10] Y. Shunyuan and S. Yanming Shenand Panwar, "An o(1) scheduling algorithm for variable-size packet switching systems," in *Proc. Ann. Allerton Conf. Communication, Control and Computing*, 2010.
[11] J. Ghaderi and R. Srikant, "On the design of efficient csma algorithms for wireless networks," in *Proc. Conf. on Decision and Control*. IEEE, 2010, pp. 954–959.
[12] M. A. Marsan, A. Bianco, P. Giaccone, S. Member, E. Leonardi, and F. Neri, "Packet-mode scheduling in input-queued cell-based switches," *IEEE/ACM Transactions on Networking*, vol. 10, pp. 666–678, 2002.
[13] Y. Ganjali, A. Keshavarzian, and D. Shah, "Cell switching versus packet switching in input-queued switches," *IEEE/ACM Trans. Networking*, vol. 13, pp. 782–789, 2005.
[14] A. Eryilmaz, R. Srikant, and J. R. Perkins, "Stable scheduling policies for fading wireless channels," *IEEE/ACM Trans. Network.*, vol. 13, no. 2, pp. 411–424, 2005.
[15] V. J. Venkataramanan and X. Lin, "On the queue-overflow probability of wireless systems: A new approach combining large deviations with lyapunov functions," preprint.
[16] D. Shah and J. Shin, "Randomized scheduling algorithm for queueing networks," *The Annals of Applied Probability*, vol. 22, no. 1, pp. 128–171, 2012.
[17] J. Ghaderi and R. Srikant, "Flow-level stability of multihop wireless networks using only MAC-layer information," in *WiOpt*, 2012.
[18] B. Hajek, "Hitting-time and occupation-time bounds implied by drift analysis with applications," *Advances in Applied Probability*, pp. 502–525, 1982.
[19] S. T. Maguluri and R. Srikant, "Scheduling jobs with unknown duration in clouds," to appear in the proceedings of IEEE INFOCOM 2013.
[20] S. Karlin and H. M. Taylor, *A First Course in Stochastic Processes*. Academic Press, 1975.
[21] S. Asmussen, *Applied Probability and Queues*. New York: Springer-Verlag, 2003.
[22] S. Meyn and R. L. Tweedie, *Markov chains and stochastic stability*. Cambridge University Press, 2009.
[23] A. Eryilmaz and R. Srikant, "Asymptotically tight steady-state queue length bounds implied by drift conditions," *Queueing Systems*, pp. 1–49, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11134-012-9305-y
[24] J. Ghaderi, T. Ji, and R. Srikant, "Connection-level scheduling in wireless networks using only MAC-layer information," in *INFOCOM*, 2012, pp. 2696–2700.
[25] T. Ji and R. Srikant, "Scheduling in wireless networks with connection arrivals and departures," in *Information Theory and Applications Workshop*, 2011.

Appendix A
Proof of Lemma 4

Since $G(.)$ is a strictly increasing bijective convex function on the open interval $(0,\infty)$, it is easy to see that $G^{-1}(.)$ is a strictly increasing concave function on $(0,\infty)$. Thus, for any two positive real numbers v_2 and v_1, $G^{-1}(v_2)-G^{-1}(v_1)\leq (v_2-v_1)\left(G^{-1}(v_1)\right)'$ where $(.)'$ denotes derivative.

Let $u=G^{-1}(v)$. Then,

$$\begin{aligned} v &= G(u)\\ \frac{dv}{du} &= G'(u)=g(u)=g(G^{-1}(v))\\ \frac{du}{dv} &= \frac{1}{g(G^{-1}(v))} \end{aligned}$$

Since $\frac{du}{dv}=\left(G^{-1}(v)\right)'$, we have $\left(G^{-1}(v)\right)'=\frac{1}{g(G^{-1}(v))}$. Thus, $G^{-1}(v_2)-G^{-1}(v_1)\leq\frac{(v_2-v_1)}{g(G^{-1}(v_1))}$. Using $V(\overline{\mathbf{Q}}^{(1)})$ and $V(\overline{\mathbf{Q}}^{(2)})$ for v_1 and v_2, we get the lemma.

APPENDIX B
PROOF OF LEMMA 5

Since the arithmetic mean is at least as as large as the geometric mean and since $G(.)$ is strictly increasing, we have

$$G\left(\prod_{i,m}(1+\overline{\mathbf{Q}}_{mi})^{\frac{1}{LM}}-1\right)\leq G\left(\frac{\sum_{i,m}(1+\overline{\mathbf{Q}}_{mi})}{LM}-1\right)$$
$$=\frac{\sum_{i,m}(1+\overline{\mathbf{Q}}_{mi})}{LM}\log\left(\frac{\sum_{i,m}(1+\overline{\mathbf{Q}}_{mi})}{LM}\right)-\frac{\sum_{i,m}(1+\overline{\mathbf{Q}}_{mi})}{LM}+1$$
$$\overset{(a)}{\leq}\frac{1}{LM}\sum_{i,m}(1+\overline{\mathbf{Q}}_{mi})\log\left((1+\overline{\mathbf{Q}}_{mi})\right)-\frac{\sum_{i,m}(\overline{\mathbf{Q}}_{mi})}{LM}$$
$$=\frac{V(\overline{\mathbf{Q}}))}{LM}\quad\leq V(\overline{\mathbf{Q}}))$$

where inequality (a) follows from log sum inequality. Now, since $G(.)$ and $\log(.)$ are strictly increasing, we have

$$e^{\left(\frac{1}{LM}\sum_{i,m}\log(1+\overline{\mathbf{Q}}_{mi})\right)}\leq 1+G^{-1}\left(V(\overline{\mathbf{Q}})\right)\Big)$$
$$\frac{1}{LM}\sum_{i,m}\log(1+\overline{\mathbf{Q}}_{mi})\leq\log\left((1+G^{-1}\left(V(\overline{\mathbf{Q}})\right)\Big)\right)\qquad(25)$$

Now to prove the second inequality, note that since $\overline{\mathbf{Q}}_{mi}$ is nonnegative for all i and m,

$$\sum_{i,m}\left(\log(1+\overline{\mathbf{Q}}_{mi})\prod_{i',m'}(1+\overline{\mathbf{Q}}_{m'i'})\right)$$
$$\geq\sum_{i,m}\left(\log(1+\overline{\mathbf{Q}}_{mi})(1+\overline{\mathbf{Q}}_{mi})\right)$$
$$\geq\frac{\sum_{i,m}\left((1+\overline{\mathbf{Q}}_{mi})\log(1+\overline{\mathbf{Q}}_{mi})\right)-\sum_{i,m}\overline{\mathbf{Q}}_{mi}-1}{e}$$

Shuffling the terms, we get,

$$\left(e\prod_{i,m}(1+\overline{\mathbf{Q}}_{mi})\right)\log\left(e\prod_{i,m}(1+\overline{\mathbf{Q}}_{mi})\right)-\left(e\prod_{i,m}(1+\overline{\mathbf{Q}}_{mi})\right)+1$$
$$\geq\sum_{i,m}\left((1+\overline{\mathbf{Q}}_{mi})\log(1+\overline{\mathbf{Q}}_{mi})\right)-\sum_{i,m}\overline{\mathbf{Q}}_{mi}$$

From the definition of $G(.)$ and $V(.)$, this is same as

$$G\left(e\prod_{i,m}(1+\overline{\mathbf{Q}}_{mi})-1\right)\geq V(\overline{\mathbf{Q}})$$
$$e^{\left(1+\sum_{i,m}\log(1+\overline{\mathbf{Q}}_{mi})\right)}\geq 1+G^{-1}(V(\overline{\mathbf{Q}}))$$
$$1+\sum_{i,m}\log(1+\overline{\mathbf{Q}}_{mi})\geq\log(1+G^{-1}(V(\overline{\mathbf{Q}})))$$

The last two inequalities again follow from the fact that $G(.)$ and $\log(.)$ are strictly increasing.