OPTIMAL RESOURCE ALLOCATION ALGORITHMS FOR CLOUD COMPUTING

BY

SIVA THEJA MAGULURI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

ABSTRACT

Cloud computing is emerging as an important platform for business, personal and mobile computing applications. We consider a stochastic model of a cloud computing cluster, where jobs arrive according to a random process and request virtual machines (VMs), which are specified in terms of resources such as CPU, memory and storage space. 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.

There are many design issues associated with such systems. One important issue is the resource allocation problem, i.e., the design of algorithms for load balancing among servers, and algorithms for scheduling VM configurations. Given our model of a cloud, we define its capacity, i.e., the maximum rates at which jobs can be processed in such a system. An algorithm is said to be throughput-optimal if it can stabilize the system whenever the load is within the capacity region. We show that the widely-used Best-Fit scheduling algorithm is not throughput-optimal.

We first consider the problem where the jobs need to be scheduled nonpreemptively on servers. Under the assumptions that the job sizes are known and bounded, we present algorithms that achieve any arbitrary fraction of the capacity region of the cloud. We then relax these assumptions and present a load balancing and scheduling algorithm that is throughput optimal when job sizes are unknown. In this case, job sizes (durations) are modeled as random variables with possibly unbounded support.

Delay is a more important metric then throughput optimality in practice. However, analysis of delay of resource allocation algorithms is difficult, so we study the system in the asymptotic limit as the load approaches the boundary of the capacity region. This limit is called the heavy traffic regime. Assuming that the jobs can be preempted once after several time slots, we present delay optimal resource allocation algorithms in the heavy traffic regime. We study delay performance of our algorithms through simulations.

To Bhagavan Sri Ramakrishna

Oh Lord! You are my mother, father, relative and friend. You are my knowledge and my wealth. You are my all in all.

‘What is that which, being known, everything else is known?’

- Mundaka Upanishad

ACKNOWLEDGMENTS

When I look back at my past self at the time when I arrived here at UIUC as a fresh college graduate, I see how much my experience at UIUC has helped in my growth and how much I have learned here. I have been very fortunate to have had an opportunity to work in CSL. The one person who played the most important role in my experience here is obviously my advisor, Prof. Srikant. He is not only an amazing teacher but also a great mentor. I thank him for his constant guidance and support all through my grad school. He taught me the basics of how research is done, how it is communicated and above all how to think about research problems. He has always been there to support me. My heart-felt thanks for his support, care, and concern for my well-being.

I thank Prof. Lei Ying from Arizona State University, who has been my coauthor for much of the work that went into this dissertation. His suggestions and insights have been very important in this work. I thank Prof Bruce Hajek who was co-advisor during my master's. I also thank Prof Bruce Hajek, Prof. Lei Ying, Prof. Yi Lu and Prof. Pramod Viswanath for serving on my committee. I thank the anonymous reviewers who have taken time to review my papers and have given their valuable feedback to improve the results that went into this dissertation. Thanks are also due to Weina Wang for her important comments on one of the results.

I thank UIUC for providing me with one of the best possible learning environments. In addition to an excellent research environment in CSL, the university has provided me an opportunity to learn through the various courses that I took from ECE, CS and Maths. These courses had some of the best teachers and I have thoroughly enjoyed taking them.

I thank my group members Rui, Chong, Javad and postdocs Chandramani and Joohwan who have always been ready for discussions. Thanks are due to Sreeram who has been a great partner in course projects and for the discussions we had on a variety of topics. I acknowledge the advice given by Vineet, Sreeram and Siva Kumar on research, career and life. Conversations with Sachin on research problems, life, academia and everything else have been a highlight of my life at CSL.

Administrative staff in ECE and CSL have made life a lot smoother. Special thanks to Peggy Wells, who has been our best office manager. She radiates cheerfulness and positive attitude in CSL. I thank Jamie Hutchinson for carefully editing my thesis.

I spent a very fruitful semester as an intern at Qualcomm, Bridgewater. I am very grateful to my mentors Xinzhou Wu and Sundar Subramanian. They provided me with a great experience which was crucial in my decision to take up industry research as my career. Special thanks to Nilesh for his warmth and friendship and most importantly for teaching me driving and thus empowering me.

I thank my alma-mater IIT Madras, where I spent my formative years. My interactions with the professors there motivated me to pursue the PhD. At IITM, I made lifelong friends, Vamsi, Ramnath, Aditya, Gaurav and Balu to name a few. Our regular phone calls and annual reunions have been great fun and were very refreshing.

I consider myself extremely fortunate to have known Swami Baneshananda, head of Vedanta-Gesellschaft, Frankfurt and Swami Atmashraddhananda, editor of Vedanta Kesari. Swami Atmashraddhananda has played a key role in defining my outlook towards life through his weekly lectures, personal conversations, regular pilgrimages, and above all, the example set by his life. I always look forward to the annual visits of Swami Baneshananda to the US, every moment of which are filled with great fun and laughter. I can always go to them with any questions or dilemmas I face and can be assured of an answer. They have been my constant guiding force and I am grateful for their time in spite of their extremely busy schedule. I feel blessed to have their unconditional love.

The Vivekananda Vedanta Society of Chicago and the community there has been my home away from home. I thank Swami Chidananda, Swami Ishatmananda and Swami Varadananda, who have always welcomed me at the society. Anjali, Pooja, Maya, Mithilesh, Jyoti Uncle, Manju Aunty, Manjusha Aunty, Rajini Aunty, Manish, Sachin and others have been like a family to me. The town/city where I lived the longest so far has been

Champaign-Urbana. If not for this family, my life here, with the harsh winter, would have been very challenging. They have also provided me a smooth transition from life in India to the US.

Over the last year of my stay in Chambana, Vedanta Study Circle and my friends there have become the defining aspect of my life. I thank Chaitanya, Ramsai, Srikanthan, Raghavendra, Suraj, Sasidhar and Kalyan for their company, our weekly meetings, philosophical discussions, outings and relaxing dinners.

In the last year of my grad school, I had to undergo a major surgery. It was one of the biggest challenges in my life and I could face it only with the support of so many of my friends and well-wishers. I express my heartfelt gratitude to Pooja, Chaitanya, Manju Aunty, Pushpa Aunty and Manish. Since my parents could not be around, my advisor, Prof Srikant assumed that role. He drove in harsh winter weather to visit me and he constantly inquired about my well-being and progress. My sister, Hima Bindu, took a week off in the very first month of her new job to attend to my needs. I should specially mention my friend Ramsai in this context. He has been my constant companion, caretaker and has attended to my every need. He has shown in practice, the philosophy of 'Karma Yoga' (Unselfish Action). I do not have words to express my gratitude to him and so I quote Swami Vivekananda, the famous Indian monk. 'In happiness, in misery, in famine, in pain, in the grave, in heaven, or in hell who never gives me up is my friend. Is such friendship a joke? A man may have salvation through such friendship. That brings salvation if we can love like that.' I am fortunate to have found such a friend in Ramsai.

I thank my parents for their unconditional love, constant support and encouragement in all my endeavors. They have made several sacrifices to give me the best in everything. Since my childhood, my mother has motivated me to excel and has taken pride in my success. She has supported my decision to pursue the PhD in a far off country instead of a lucrative career in management closer to home, despite not completely appreciating why. I thank my sister for her love and support.

Most importantly, I thank the Lord who, out of His grace, has more than provided for my every need. I thank him for all the experiences he has put me through and for giving me the strength to go through them. He has given me the grad school experience which was a wonderful learning opportunity not only academically and professionally, but also personally and spiritually. He put me in touch with great friends, teachers and well-wishers. I humbly dedicate this dissertation to Him.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Cloud computing services are becoming the primary source of computing power for both enterprises and personal computing applications. A cloud computing platform can provide a variety of resources, including infrastructure, software, and services, to users in an on-demand fashion. To access these resources a cloud user submits a request for resources. The cloud provider then provides the requested resources from a common resource pool (e.g., a cluster of servers) and allows the user to use these resources for a required time period. Compared to traditional "own-and-use" approaches, cloud computing services eliminate the costs of purchasing and maintaining the infrastructures for cloud users and allow the users to dynamically scale up and down computing resources in real time based on their needs. Several cloud computing systems are now commercially available, including Amazon EC2 system [1], Google's AppEngine [2], and Microsoft's Azure [3]. Comprehensive surveys on cloud computing can be found in [4, 5, 6].

While cloud computing services in practice provide many different services, we consider cloud computing platforms that provide infrastructure as a service (IaaS), in the form of virtual machines (VMs), to users. We assume cloud users request VMs, which are specified in terms of resources such as CPU, memory and storage space. Each request is called a "job." The type of a job refers to the type of VM the user wants. The amount of time each VM or job is to be hosted is called its size. After receiving these requests, the cloud provider will schedule the VMs on physical machines, called "servers."

As an example, Table 1.1 lists three types of VMs (called instances) available in Amazon EC2.

A cloud system consists of a number of networked servers. Each server in the data center has certain amount of resources. This imposes a constraint on the number of VMs of different types that can be served simultaneously depending on the amount of resources requested by each VM. This is illustrated

Table 1.1: Three representative instances in Amazon EC2

Instance Type	Memory	CPU	Storage
Standard Extra Large	15 GB	8 EC2 units	1,690 GB
High-Memory Extra Large	17.1 GB	6.5 EC2 units	420 GB
High-CPU Extra Large	7 GB	20 EC2 units	1,690 GB

in the following example.

Example 1.1. Consider a server with 30 GB memory, 30 EC2 computing units and $4,000$ GB storage space. Then $N = (2, 0, 0)$ and $N = (0, 1, 1)$ are two feasible VM-configurations on the server, where N_1 is the number of standard extra-large VMs, N_2 is the number of high-memory extra-large VMs, and N_3 is the number of high-CPU extra-large VMs. $N = (0, 2, 1)$ is not a feasible VM configuration on this server because it does not have enough memory and computing units.

Jobs with variable sizes arrive according to a stochastic process. These jobs need to be hosted on the servers for a requested amount of time, after which they depart. We assume jobs are queued in the system when the servers are busy.

There are many design issues associated with such systems [7, 8, 9, 10, 11, 12]. One important issue is the resource allocation problem: 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? Typically jobs have to be scheduled in a nonpreemptive manner. However, preemption once in a while is sometimes allowable. We will call this the scheduling problem.

We are interested in resource allocation algorithms with certain optimality properties. The simplest notion of optimality is throughput optimality. We say that an algorithm is throughput optimal if it can stabilize the system when any other algorithm can. Loosely speaking, a throughput optimal algorithm can sustain the maximum possible rate at which jobs can be processed. Another notion of optimality of interest is delay optimality. delay optimality means that the mean delay experienced by the jobs is minimized. We will study Delay optimality in the asymptotic limit when the load is close to the boundary of the capacity region. This is called the heavy traffic limit.

1.1 Best Fit Is Not Throughput Optimal

The resource allocation problem in cloud data centers has been well studied [7, 8]. Best Fit policy [13, 14] is a popular policy that is used in practice. According to this policy, whenever resources become available, the job which uses the largest amount of resources, among all jobs that can be served, is selected for service. Such a definition has to be made more precise when a VM requests multiple types of multiple resources. In the case of multiple types of resources, we can select one type of resource as "reference resource," and define best fit with respect to this resource. If there is a tie, then best fit with respect to another resource is considered, and so on. Alternatively, one can consider a particular linear or nonlinear combination of the resources as a meta-resource and define best fit with respect to the meta-resource.

We now show that best fit is not throughput optimal. Consider a simple example where we have two servers, one type of resource and two types of jobs. A type-1 job requests half of the resource and four time slots of service, and a type-2 job requests the whole resource and one time slot of service. Now assume that initially, the server 1 hosts one type-1 job and server 2 is empty; two type-1 jobs arrive once every three time slots starting from time slot 3, and type-2 jobs arrive according to some arrival process with arrival rate ϵ starting at time slot 5. Under the best-fit policy, type-1 jobs are scheduled forever since type-2 jobs cannot be scheduled when a type-1 job is in a server. So the backlogged workload due to type-2 jobs will blow up to infinity for any $\epsilon > 0$. The system, however, is clearly stabilizable for $\epsilon < 2/3$. Suppose we schedule type-1 jobs only in time slots 1, 7, 13, 19, $\ldots$, i.e., once every six time slots. Then time slots 5, 6, 11, 12, 17, 18, ... are available for type-2 jobs. So if $\epsilon < 2/3$, both queues can be stabilized under this periodic scheduler.

The specific arrival process we constructed is not key to the instability of best-fit. Assume type-1 and type-2 jobs arrive according to independent Poisson processes with rates λ_1 and λ_2, respectively. Figure 1.1 is a simulation result which shows that the number of backlogged jobs blows up under best-fit with $\lambda_1 = 0.7$ and $\lambda_2 = 0.1$, but is stable under a MaxWeight-based policy with $\lambda_1 = 0.7$ and $\lambda_2 = 0.5$.

This example raises the question as to whether there are any throughput-optimal policies. To answer this question, we will first propose a stochastic

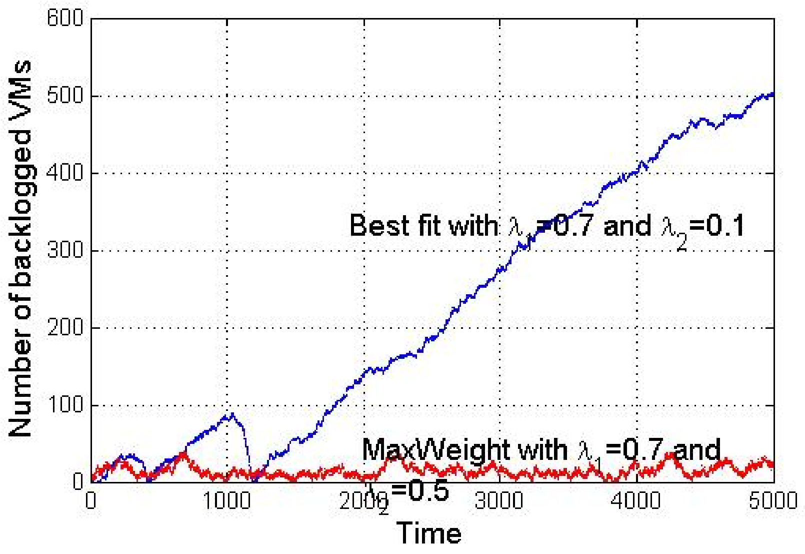


Figure 1.1: The number of backlogged jobs under the best-fit policy and a MaxWeight policy

model to study resource allocation problems in cloud computing and then pose this question in a precise manner.

1.2 A Stochastic Model for Cloud Computing

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.

For server i, 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, ..., 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.

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 next define the concept of *capacity* for a cloud.

First, as an example, consider the three servers defined in Example 1.1. Clearly this system has an aggregate capacity of 90 GB of memory, 90 EC2 compute units and 12,000 GB of storage space. However, such a crude definition of capacity fails to reflect the system's ability to host VMs. For example, while

$$\begin{aligned} 4 \times 17.1 + 3 \times 7 = 89.4 &\leq 90, \\ 4 \times 6.5 + 3 \times 20 = 86 &\leq 90, \\ 4 \times 420 + 3 \times 1690 = 6750 &\leq 12000, \end{aligned}$$

it is easy to verify that the system cannot host 4 high-memory extra-large VMs and 3 high-CPU extra-large VMs at the same time. Therefore, we have to introduce a VM-centric definition of capacity.

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. Let $\lambda_m = \mathbb{E}[A_m(t)]$ denote the arrival rate of type-m jobs. Assume $P(A_m(t) = 0) > \epsilon_A$ for some $\epsilon_A > 0$ for all m and t.

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.

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.

Jobs are routed to the servers according to a load balancing algorithm. Let $\mathbf{A}_{mi}(t)$ denote the number of type m jobs that are routed to server i. Since

$A_m(t)$ denotes the total number of type m job arrivals at time t, routing is done so that $\sum_i \mathbf{A}_{mi}(t) = A_m(t)$.

In each time slot, jobs are served at each server according to a scheduling algorithm. 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.,

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

A vector of arrival rates 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 $\overline{S}_{\min} = \min_m\{\overline{S}_m\}$.

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 $\mathcal{C}$ and $\widehat{\mathcal{C}}$ as follows:

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

where Conv denotes the convex hull. Now define

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

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 $\check{\lambda}_m$ to denote $\lambda_m \overline{S}_m$ so $\check{\lambda} \in \mathcal{C}$ is same as $(\lambda \circ \overline{S}) \in \widehat{\mathcal{C}}$ We use *int*(.) to denote interior of a set.

We next use a simple example to illustrate the definition of $\mathcal{C}$.

Example 1.2. Consider a simple cloud system consisting of three servers. Servers 1 and 2 are of the same type (i.e., they have the same amount of resources), and server 3 is of a different type. Assume there are two types of VMs. The set of feasible VM configurations on servers 1 and 2 is assumed to be $\mathcal{N}_1 = \mathcal{N}_2 = \{(0,0), (1,0), (0,1)\}$, i.e., each of these servers can at most host either one type-1 VM or one type-2 VM. The set of feasible configurations on

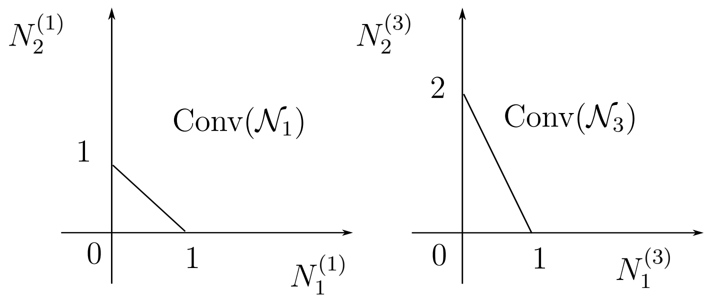

$N_2^{(1)}$
1
$\mathrm{Conv}(\mathcal{N}_1)$
0
1
$N_1^{(1)}$
$N_2^{(3)}$
2
$\mathrm{Conv}(\mathcal{N}_3)$
0
1
$N_1^{(3)}$

Figure 1.2: Regions $\mathrm{Conv}(\mathcal{N}_1)$ and $\mathrm{Conv}(\mathcal{N}_3)$

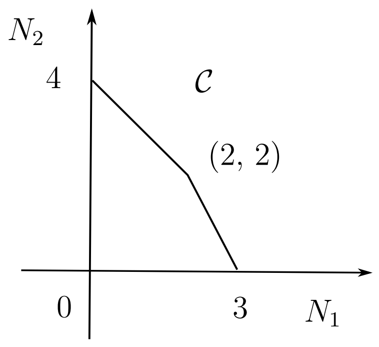


Figure 1.3: The region $\mathcal{C}$

server 3 is assumed to be $\mathcal{N}_3 = \{(0,0),(1,0),(2,0),(0,1)\}$, i.e., the server can at most host either two type-1 VMs or one type-2 VM. The regions $\mathrm{Conv}(\mathcal{N}_1)$ and $\mathrm{Conv}(\mathcal{N}_3)$ are plotted in Figure 1.2. Note that vector $(0.75, 0.25)$ is in the region $\mathrm{Conv}(\mathcal{N}_1)$. While a type-1 server cannot host "0.75" type-1 VMs and "0.25" type-2 VM, we can host a type-1 VM on server 1 for 3/4 of the time, and a type-2 VM on the server for 1/4 of the time to support load $(0.75, 0.25)$. Figure 1.3 shows the region $calC$. Capacity region for this simple cloud system is then the set of all λ and $\overline{S}$ such that the total load $(\lambda \circ \overline{S})$ is in the region $\mathcal{C}$.

This definition of the capacity of a cloud is motivated by similar definitions in [15]. As in [15], it is easy to show the following result.

Proposition 1.1. *For any pair* $(\lambda, \overline{S})$ *such that* $(\lambda, \overline{S}) \notin \widehat{\mathcal{C}}$, $\lim_{t\to\infty} E\left[\sum_m Q_{mi}(t)\right] = \infty$*, i.e., the pair* $(\lambda, \overline{S})$ *is not supportable.*

In the next two chapters we will present algorithms that stabilize the systems as long as the arrival loads are within the region $\widehat{\mathcal{C}}$. This shows that $\widehat{\mathcal{C}}$ is the capacity of the cloud. Such algorithms that stabilize the system for any arrival load in the capacity region are said to be throughput optimal.

Moreover, these algorithms do not require knowledge of the actual arrival rates.

In the next chapter, we will consider the case when the job sizes are bounded and are known at arrival. We will also assume that preemption is not allowed. We will make a connection to the scheduling problem in an ad hoc wireless network and propose an algorithm inspired by the MaxWeight algorithm for wireless networks. In Chapter 3, we will consider the case when the job sizes are not bounded and are known neither at arrival nor at the beginning of service and again present throughput optimal resource allocation. algorithm for this case. In Chapter 4, we will consider the case when jobs allowed to be preempted once in a while, and in Chapter 5 we will consider delay optimality in the heavy traffic limit.

CHAPTER 2

THROUGHPUT OPTIMALITY: KNOWN JOB SIZES

In this chapter, we consider the resource allocation problem when preemption is not allowed. We assume that the job sizes are known at arrival and are bounded. We will first draw an analogy with scheduling in an ad hoc wireless network. We will show that the algorithms for ad hoc wireless, such as MaxWeight scheduling can be directly used here for a simplified system.

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 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 [16]. This algorithm, however, uses the special structure of a switch and so it is not clear how it can be generalized for the case of a cloud system. Reference [17] 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 [17] by appealing to prior work [18]. However, the result in [18] is applicable only when the Markov chain has finite number of states. However, since the Markov chain in [17] 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 [17] do not seem to be immediately applicable to our problem.

A similar problem was studied in [19] for scheduling in an input queued switch. An algorithm claimed to be throughput optimal was presented. However, the proof of optimality is incorrect. We present more details about the algorithm and the errors in the proof in the next chapter.

Since the job sizes are known at arrival, when the jobs are queued, one knows the total backlogged workload in the queue, i.e., the total amount of time required to serve all the jobs in the queue. One can use this information in the resource allocation problem. Let $q_{mi}(t)$ denote the total backlogged workload of type m jobs at server i.

In this chapter, we will first draw an analogy with scheduling in an ad hoc wireless network. We will show that the algorithms for ad hoc wireless can be directly used here for a simplified system. Then, we present an almost throughput optimal resource allocation algorithm for the cloud computing system. The results in this chapter have been presented in [20] and [21].

2.1 A Centralized Queueing Approach

We now make certain simplifying assumptions to gain intuition. Though some of these are not practical, we first use them to present a very simple solution, which can then be generalized to the original cloud problem.

We assume that jobs are queued in a centralized manner. So, for each type of job, there is a single queue at a centralized location as opposed to a separate queue at each server. So, there are M queues in all, one for each type of job.

Recall that $\mathcal{A}_m(t)$ is the set of type-m jobs that arrive at the beginning of time slot t, and $A_m(t) = |\mathcal{A}_m(t)|$, is the number of such jobs.

We let $a_m(t) = \sum_{j\in\mathcal{A}_m(t)} S_j$ be the total number of time slots of service requested by the jobs that arrive at time t, i.e., the total arrival of workload of type-m jobs in time slot t. Then, $\mathbb{E}[a_m(t)] = \lambda_m \overline{S}_m = \check{\lambda}_m$. Let $var(a_m^2(t)) = \sigma_m^2$ for each m. Let $N_m(t)$ denote the number of type-m jobs that are served by the cloud at time slot t. Note that the job size of each of these $N_m(t)$ jobs reduces by one at the end of time slot t. We assume that a server can serve at most $N_{\max}$ jobs at the same time. The total backlogged workload due to type-m jobs is defined to be the sum of the remaining job sizes of all jobs of type-m in the system. We let $q_m(t)$ denote the backlogged workload of type-m jobs in the network at the beginning of time slot t, before any other job arrivals. Then the dynamics of $q_m(t)$ can be described as

$$q_m(t+1) = \left(q_m(t) + a_m(t) - N_m(t)\right). \tag{2.1}$$

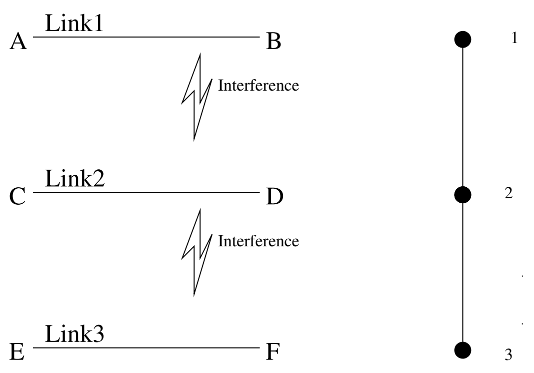


Figure 2.1: Interference constraints for six users and three links and the corresponding interference graph

The resource allocation problem then reduces to the problem of choosing a vector $N(t) = (N_1(t), N_2(t), \dots, N_M(t))$ that is a feasible configuration vector for the cloud. Here, we say that a vector N is a feasible vector for the cloud if can be written as $N = \sum_i N(i)$ where $N(i)$ is a feasible configuration for the server i.

2.1.1 Preemptive Algorithm

We first assume that all servers can be reconfigured at the beginning of each time slot. This means that a job can be interrupted at the beginning of each time and put back in the centralized queue for that job type. In other words, we assume that complete preemption is allowed.

This problem is then identical to the problem of scheduling in an ad hoc wireless network. An ad hoc network consists of a collection of wireless nodes. A link in such a network refers to a transmitter-receiver pair of nodes. Not all the links can be simultaneously active because of interference. These constraints are represented by an interference graph. Vertices in the interference graph correspond to the links. If there is an edge between two vertices, then the corresponding links interfere and so cannot transmit at the same time. An example is shown in Figure 2.1.

Packets arrive to be transmitted over the links and are queued. Given the queue lengths at each link, a scheduling algorithm has to choose a set of links that can transmit at each given time, without violating interference constraints. In other words, at any given time, the scheduler should choose an independent set from the interference graph.

In their seminal paper, Tassiulas and Ephrimedes [15] presented the MaxWeight scheduling algorithm for this problem and showed that it is throughput optimal. Each link is associated with a weight which is a function of the queue length, usually the queue length itself, and a schedule with the maximum weight is chosen in each time slot from all possible schedules.

Since the centralized and preemptive scheduling problem is identical to the wireless network scheduling problem, MaxWeight algorithm is also throughput optimal. However, the Server-by-server MaxWeight algorithm (Algorithm 1) is equivalent to finding the maximum weight feasible vector for the cloud.

Algorithm 1 Server-by-server MaxWeight allocation for a centralized queueing system with complete preemption

At the beginning of time slot t, consider the i^{th} server. If the set of jobs on the server are not finished, move them back to the central queue. Find a VM-configuration $N^*(t)$ such that

$$N^{(i)*}(t) \in \arg\max_{N \in \mathcal{N}_i} \sum_m q_m(t) N_m.$$

At server i, we create up to $N_m^{(i)*}(t)$ type-m VMs depending on the number of jobs that are backlogged. Let $N_m^{(i)}(t)$ be the actual number of VMs that were created. Then, we set

$$q_m(t+1) = \left(q_m(t) + a_m(t) - \sum_i N_m^{(i)} \right).$$

Then, as in [15], we have the following proposition.

Proposition 2.1. *Consider the cloud system with centralized queues and assume that complete preemption of jobs is allowed. The server-by-server MaxWeight allocation presented in Algorithm 1 is throughput optimal, i.e.,*

$$\lim_{t \to \infty} E\left[\sum_m q_m(t) \right] < \infty$$

whenever $\check{\lambda} \in int\mathcal{C}$.

The proof is based on bounding the drift of a quadratic Lyapunov function.

The drift is shown to be negative outside a finite set and the Foster-Lyapunov theorem is invoked to prove positive recurrence of the Markov chain corresponding to the system as long as the arrivals are within the capacity region. We skip the proof of this proposition since it is much simpler and is in the same lines as that of Theorem 2.1.

One drawback of MaxWeight scheduling in wireless networks is that its complexity increases exponentially with the number of wireless nodes. Moreover, it needs to be implemented in a centralized policy. However, for the cloud system the server by server implementation in Algorithm 1 is has much lower complexity and can be implemented in a distributed manner. Consider the following example. If there are L servers and each server has S allowed configurations, then 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.

2.1.2 Nonpreemptive Algorithms

One of the simplifying assumptions made in the previous subsection is that jobs can be interrupted and reallocated later, possibly on different servers. In practice, a job may not be interruptible or interrupting a job can be very costly (the system needs to store a snapshot of the VM to be able to restart the VM later). In the rest of this chapter and the next, we assume that jobs are not allowed to be interrupted.

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 include these jobs.

Therefore, since one cannot choose MaxWeight schedule in every time slot, a natural alternative is to somehow choose MaxWeight schedule every few time slots and then perform some 'reasonable' scheduling between these time slots. It turns out that using MaxWeight schedule once in a while is good enough.

Before we present the algorithm, we outline the basic ideas first. We group T time slots into a super time slot, where $T > S_{\max}$. At the beginning of a super time slot, a configuration is chosen according to the MaxWeight algorithm. When jobs depart a server, the remaining resources in the server are filled again using the MaxWeight algorithm; however, we impose the constraint that only jobs that can be completed within the super slot can be served. So the algorithm myopically (without consideration of the future) uses resources, but is queue-length aware since it uses the MaxWeight algorithm. This is described more precisely in Algorithm 2

Algorithm 2 Myopic MaxWeight allocation:

We group T time slots into a super time slot. At time slot t, consider the i^{th} server. Let $N^{(i)}(t^-)$ be the set of VMs that are hosted on server i at the beginning of time slot t, i.e., these correspond to the jobs that were scheduled in the previous time slot but are still in the system. These VMs cannot be reconfigured due to our nonpreemption requirement. The central controller finds a new vector of configurations $\tilde{N}^{(i)}(t)$ to fill up the resources not used by $N^{(i)}(t^-)$, i.e.,

$$\tilde{N}^{(i)}(t) \in \arg \max_{N: N+N^{(i)}(t^-) \in \mathcal{N}_i} \sum_m q_m(t) N_m,$$

The central controller selects as many jobs as available in the queue, up to a maximum of $\tilde{N}^{(i)}_m(t)$ type-m jobs at server i, and subject to the constraint that a type-m job can only be served if its size $S_j \leq T - (t \mod T)$. Let $\bar{N}^{(i)}_m(t)$ denote the actual number of type-m jobs selected. Server i then serves the $\bar{N}_m(t)^(i)$ new jobs of type m, and the set of jobs $N^{(i)}(t^-)$ left over from the previous time slot. The queue length is updated as follows:

$$q_m(t+1) = q_m(t) + a_m(t) - \sum_i \left(N^{(i)}_m(t^-) + \bar{N}^{(i)}_m(t) \right).$$

Note that this myopic MaxWeight allocation algorithm differs from the server-by-server MaxWeight allocation in two aspects: (i) jobs are not interrupted when served and (ii) when a job departs from a server, new jobs are accepted without reconfiguring the server. The following proposition characterizes the throughput achieved by the myopic MaxWeight.

Proposition 2.2. *Consider the myopic MaxWeight algorithm in Algorithm 2. Any job load that satisfies* $\frac{T}{T-S_{\max}}\check{\lambda} \in \mathcal{C}$ *is stabilizable under the myopic MaxWeight allocation.*

The proof of this proposition again uses the Foster-Lyapunov theorem based on a quadratic Lyapunov function.However, instead of the one step drift, the drift over every super time slot is shown to be negative (outside a finite set). This then gives that a system sampled at the beginning of every super time slot is stable. Since the mean arriving workload in each time slot is bounded, we then have stability of the original system is also stable. We again skip the proof as most of the same ideas are presented in the proof of Theorem 2.1.

It is important to note that, unlike best fit, the myopic MaxWeight algorithm can be made to achieve any arbitrary fraction of the capacity region by choosing T sufficiently large.

2.2 Resource Allocation with Load Balancing

In the previous section, we considered the case where there was a single queue for jobs of the same type, being served at different servers. This requires a central authority to maintain a single queue for all servers in the system. A more distributed solution is to maintain queues at each server and route jobs as soon as they arrive. To the best of our knowledge, this problem does not fit into the scheduling/routing model in [15]. However, we show that one can still use MaxWeight-type scheduling if the servers are load-balanced using a join-the-shortest-queue (JSQ) routing rule.

So, we now assume that each server maintains M different queues for different types of jobs. It then uses the information about backlogged workload in each of these queues in making scheduling decisions. Let $\mathbf{q}$ denote the vector of these backlogged workloads where $\mathbf{q}_{mi}$ is the backlogged workload of type m jobs at server i. Routing and scheduling are performed as described in Algorithm 3.

Algorithm 3 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 backlogged workload for type m jobs i.e., the server $i^*_m(t) = \underset{i \in \{1,2,,,L\}}{\arg\min} q_{mi}(t)$. Therefore, the arrivals to q_{mi} in time slot t are given by

$$a_{mi}(t) = \begin{cases} \widehat{A}_m(t) & \text{if } i = i^*_m(t) \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

2. *Scheduling Algorithm (Myopic MaxWeight Scheduling) for each server i*: T time slots are grouped into a super time slot. A MaxWeight configuration is chosen at the beginning of a super time slot. So, for $t = nT$, configuration $\tilde{N}^{(i)}(t)$ is chosen according to

$$\tilde{N}^{(i)}(t) \in \underset{N \in \mathcal{N}_i}{\arg\max} \sum_m q_{mi}(t) N_m.$$

For all other t, at the beginning of the time slot, a new configuration is chosen as follows:

$$\tilde{N}^{(i)}(t) \in \underset{N : N + N^{(i)}(t^-) \in \mathcal{N}_i}{\arg\max} \sum_m q_{mi}(t) N_m,$$

where $N^{(i)}(t^-)$ is the configuration of jobs at server i that are still in service at the end of the previous time slot. As many jobs as available are selected for service from the queue, up to a maximum of $\tilde{N}^{(i)}_m(t)$ jobs of type m, and subject to the constraint that a new type m job is served only if it can finish its service by the end of the super time slot, i.e., only if $S_j \le T - (t \bmod T)$. Let $\overline{N}^{(i)}_m(t)$ denote the actual number of type m jobs selected at server i and define $N^{(i)}(t) = N^{(i)}(t^-) + \overline{N}^{(i)}(t)$. The queue lengths are updated as follows:

$$q_{mi}(t+1) = q_{mi}(t) + a_{mi}(t) - N^{(i)}_m(t). \tag{2.3}$$

The following theorem characterizes the throughput performance of the algorithm.

Theorem 2.1. *Any job load vector that satisfies* $\frac{T}{T-S_{\max}}\check{\lambda} \in \mathcal{C}$ *is stabilizable under the JSQ routing and myopic MaxWeight allocation as described in Algorithm 3.*

Proof. The idea behind the proof is again to bound the drift of a quadratic Lyapunov function over a super time slot. However, now the load balancing algorithm also plays a role in the drift.

Let $Y_{mi}(t)$ denote the state of the queue for type-m jobs at server i. If there are I such jobs, $Y_{mi}(t)$ is a vector of size I and $Y_{mi}^{j}(t)$ is the (backlogged) size of the j^{th} type-m job at server i. First, it is easy to see that $\mathbf{Y}(t) = \{Y_{mi}(t)\}_{m,i}$ is a Markov chain under the myopic MaxWeight scheduling. Further define $\mathcal{S} = \{y : \Pr(\mathbf{Y}(t) = y|\mathbf{Y}(0) = \mathbf{0}) \text{ for some } t\}$, then $\mathbf{Y}(t)$ is an irreducible Markov chain on state space $\mathcal{S}$ assuming $\mathbf{Y}(0) = \mathbf{0}$. This claim holds because (i) any state in $\mathcal{S}$ is reachable from $\mathbf{0}$ and (ii) since $\Pr(a_m(t) = 0) \geq \epsilon_A$ for all m and t, the Markov chain can move from $\mathbf{Y}(t)$ to $\mathbf{0}$ in finite time with a positive probability. Further $q_{mi}(t) = \sum_j Y_{m,i}^{j}(t)$, i.e., $q_{mi}(t)$ is a function of $Y_{mi}(t)$.

We will first show that the increase of $\sum\limits_m q_{mi}(t)N_m^{(i)}(t)$ is bounded within a super time slot. For any t such that $1 \leq (t \bmod T) \leq T - S_{\max}$, for each server i,

$$\begin{aligned}
&\sum_m q_{mi}(t)N_m^{(i)}(t-1)\\
=&\sum_m q_{mi}(t)N_m^{(i)}(t^-) + \sum_m q_{mi}(t)\left(N_m^{(i)}(t-1) - N_m^{(i)}(t^-)\right)\\
\overset{(a)}{\leq}&\sum_m q_{mi}(t)N_m^{(i)}(t^-) + \sum_m q_{mi}(t)\tilde{N}_m^{(i)}(t)\\
=&\sum_m \left(q_{mi}(t)N_m^{(i)}(t^-) + q_{mi}(t)\tilde{N}_m^{(i)}(t)\right)\mathbb{I}_{q_{mi}(t)\geq S_{\max}N_{\max}}\\
&+\sum_m \left(q_{mi}(t)N_m^{(i)}(t^-) + q_{mi}(t)\tilde{N}_m^{(i)}(t)\right)\mathbb{I}_{q_{mi}(t)< S_{\max}N_{\max}}\\
\overset{(b)}{\leq}&\sum_m q_{mi}(t)N_m^{(i)}(t) + MS_{\max}N_{\max}^2,
\end{aligned}$$

where the inequality (a) follows from the definition $\tilde{N}_m^{(i)}(t)$; and inequality (b)

holds because when $q_{mi}(t) \geq S_{\max} N_{\max}$, there are enough number of type-m jobs to be allocated to the servers, and when $1 \leq (t \mod T) \leq T - S_{\max}$, all backlogged jobs are eligible to be served in terms of job sizes. Now since $|q_{mi}(t) - q_{mi}(t-1)| = \left| a_{mi}(t-1) - N_m^{(i)}(t) \right| \leq a_{mi}(t-1) + N_{\max}$, we have

$$\sum_m q_{mi}(t-1) N_m^{(i)}(t-1) \leq \beta' + \sum_m q_{mi}(t) N_m^{(i)}(t) + \sum_m a_{mi}(t-1) N_{\max}, \quad (2.4)$$

where $\beta' = M N_{\max}^2 (S_{\max} + 1)$.

Let $V(t) = |\mathbf{q}(t)|^2$ be the Lyapunov function. Let $t = nT + \tau$ for $0 \leq \tau < T$. Then,

$$\begin{aligned}
& E[V(nT+\tau+1) - V(nT+\tau) | \mathbf{q}(nT) = \mathbf{q}] \\
=& E\left[\sum_i \sum_m \left(q_{mi}(t) + a_{mi}(t) - N_m^{(i)}(t) \right)^2 - q_{mi}^2(t) \middle| \mathbf{q}(nT) = \mathbf{q} \right] && (2.5) \\
=& E\left[2 \sum_i \sum_m q_{mi}(t) \left(a_{mi}(t) - N_m^{(i)}(t) \right) \right. \\
& \left. + \sum_i \sum_m \left(a_{mi}(t) - N_m^{(i)}(t) \right)^2 \middle| \mathbf{q}(nT) = \mathbf{q} \right] && (2.6) \\
\leq& K_1 + 2E\left[\sum_m \sum_i q_{mi}(t) a_{mi}(t) - \sum_i \sum_m q_{mi}(t) N_m^{(i)}(t) \middle| \mathbf{q}(nT) = \mathbf{q} \right] && (2.7) \\
=& K_1 + 2 \sum_m E[q_{mi_m^*(t)}(t) a_m(t) | \mathbf{q}(nT) = \mathbf{q}] \\
& - 2E\left[\sum_i \sum_m q_{mi}(t) N_m^{(i)}(t) \middle| \mathbf{q}(nT) = \mathbf{q} \right] && (2.8) \\
=& K_1 + 2 \sum_m \check{\lambda}_m E[q_{mi_m^*(t)}(t) | \mathbf{q}(nT) = \mathbf{q}] \\
& - 2E\left[\sum_i \sum_m q_{mi}(t) N_m^{(i)}(t) \middle| \mathbf{q}(nT) = \mathbf{q} \right] && (2.9) \\
\leq& K_1 + 2 \sum_m \check{\lambda}_m^2 \tau + 2 \sum_m \check{\lambda}_m E[q_{mi_m^*(nT)}(nT) | \mathbf{q}(nT) = \mathbf{q}] \\
& - 2E\left[\sum_i \sum_m q_{mi}(t) N_m^{(i)}(t) \middle| \mathbf{q}(nT) = \mathbf{q} \right] && (2.10) \\
=& K_1 + 2 \sum_m \check{\lambda}_m^2 \tau + 2 \sum_m \check{\lambda}_m q_{mi_m^*}
\end{aligned}$$

$$-2E\left[\sum_i\sum_m q_{mi}(t)N_m^{(i)}(t)\middle|\mathbf{q}(nT)=\mathbf{q}\right], \tag{2.11}$$

where $K_1 = MLN_{\max}^2 + \sum_m(\check{\lambda}_m^2 + \sigma_m^2 + 2\check{\lambda}_m N_{\max})$ and $i_m^* = i_m^*(nT) = \arg\min_{i\in\{1,2,,,L\}} q_{mi}$. Equation (2.8) follows from the definition of a_{mi} in the routing algorithm in (5.2). Equation (2.9) follows from the independence of the arrival process from the queue length process. Inequality (2.10) follows from the fact that $E[q_{mi_m^*(t)}(t)] \le E[q_{mi_m^*(nT)}(t)] \le E[q_{mi_m^*(nT)}(nT) + \sum_{t'=nT}^{t-1} a_m(t)] = E[q_{mi_m^*}(nT)] + \tau\check{\lambda}$.

Now, applying (2.4) repeatedly for $t \in [nT, (n+1)T - S_{\max}]$, and summing over i and using the fact that $\sum_i a_{mi}(t) = a_m(t)$, we get

$$\begin{aligned}&-\sum_i\sum_m q_{mi}(t)N_m^{(i)}(t)\\&\le L(t-nT)\beta' - \sum_i\sum_m q_{mi}(nT)N_m^{(i)}(nT) + \sum_{t'=nT}^{(n+1)T-S_{\max}-1}\sum_m a_m(t')N_{\max}.\end{aligned} \tag{2.12}$$

Since $\frac{(1+\epsilon)T}{T-S_{\max}}\check{\lambda} \in int(\mathcal{C})$, there exists $\epsilon > 0$ s.t. $\frac{(1+\epsilon)T}{T-S_{\max}}\check{\lambda} \in \mathcal{C}$, and so there exists $\{\check{\lambda}^i\}_i$ such that $\frac{(1+\epsilon)T}{T-S_{\max}}\check{\lambda}^i \in \mathrm{Conv}(\mathcal{N}_i)$ for all i and $\check{\lambda} = \sum_i \check{\lambda}^i$. According to the scheduling algorithm, for each i, we have that

$$(1+\epsilon)\frac{T}{T-S_{\max}}\sum_m q_{mi}(nT)\check{\lambda}_m^i \le \sum_m q_{mi}(nT)N_m^{(i)}(nT). \tag{2.13}$$

Thus, we get,

$$\begin{aligned}&-\sum_i\sum_m q_{mi}(t)N_m^{(i)}(t)\\&\le L(t-nT)\beta' - \sum_i\sum_m q_{mi}(nT)N_m^{(i)}(nT) + \sum_{t'=nT}^{(n+1)T-S_{\max}-1}\sum_m a_m(t')N_{\max}\end{aligned} \tag{2.14}$$

$$\le L(t-nT)\beta' - \frac{(1+\epsilon)T}{T-S_{\max}}\sum_i\sum_m q_{mi}(nT)\check{\lambda}_m^i + \sum_{t'=nT}^{(n+1)T-S_{\max}-1}\sum_m a_m(t')N_{\max}. \tag{2.15}$$

Substituting this in (2.11), for $t \in [nT, (n+1)T - S_{\max}]$, we get

$$\begin{aligned} & E[V(nT+\tau+1) - V(nT+\tau) | \mathbf{q}(nT) = \mathbf{q}] \\ \leq & K + 2\tau \sum_m (\check{\lambda}_m^2 + \check{\lambda}_m N_{\max}) + 2L(t - nT)\beta' \\ & + 2\sum_m \check{\lambda}_m q_{mi_m^*} - 2(1+\epsilon)\frac{T}{T - S_{\max}} \sum_i \sum_m q_{mi} \check{\lambda}_m^i . \end{aligned} \tag{2.16}$$

Note that $\check{\lambda}_m q_{mi_m^*} = \sum_i \check{\lambda}_m^i q_{mi_m^*} \leq \sum_i \check{\lambda}_m^i q_{mi}$. We will now use this relation and sum the drift for τ from 0 to $T-1$. Using (2.16) for $\tau \in [0, T - S_{\max}]$, and (2.11) for the remaining τ, we get

$$\begin{aligned} & E[V((n+1)T) - V(nT) | \mathbf{q}(nT) = \mathbf{q}] \\ \leq & TK_1 + 2\sum_m (\check{\lambda}_m^2 + \check{\lambda}_m N_{\max}) \sum_{\tau=0}^{T-1} \tau + 2L\beta' \sum_{\tau=0}^{T-S_{\max}-1} \tau \\ & + 2T \sum_{i,m} q_{mi}\check{\lambda}_m^i - 2\frac{(1+\epsilon)T}{T - S_{\max}} \sum_{i,m} q_{mi}\check{\lambda}_m^i (T - S_{\max}) \\ \leq & K_2 - 2\epsilon T \sum_i \sum_m q_{mi}\check{\lambda}_m^i , \end{aligned}$$

where $K_2 = TK_1 + 2\sum_m (\check{\lambda}_m^2 + \check{\lambda}_m N_{\max}) \sum_{\tau=0}^{T-1} \tau + 2L\beta' \sum_{\tau=0}^{T-S_{\max}-1} \tau$. Let $\mathcal{B} = \{\mathbf{Y} : \sum_i \sum_m q_{mi}(\mathbf{Y})\check{\lambda}_m^i \leq K_2/\epsilon T\}$. The set $\mathcal{B}$ is finite. This is because there are only a finite number of $\mathbf{q} \in \mathbb{Z}_+^M L$ vectors satisfying $\sum_i \sum_m q_{mi}\check{\lambda}_m^i$ and for each $\mathbf{q}$, there are a finite number of states $\mathbf{Y}$ such that $\mathbf{q} = \mathbf{q}(\mathbf{Y})$. Clearly the drift $E[V((n+1)T) - V(nT)|\mathbf{q}(nT) = \mathbf{q}]$ is negative outside the finite set $\mathcal{B}$. Then from the Foster-Lyapunov theorem [22, 23], we have that the sampled Markov chain $\widetilde{\mathbf{Y}}(n) \triangleq \mathbf{Y}(nT)$ is positive recurrent and so $\lim_{n\to\infty} E[\sum_i \sum_m q_{mi}(nT)] < \infty$. For any time t between nT and $(n+1)T$, we have

$$\begin{aligned} E\left[\sum_i \sum_m q_{mi}(t)\right] \leq & E\left[\sum_i \sum_m q_{mi}(nT) + \sum_i \sum_m \sum_{t'=nT}^{t-1} a_{mi}(t')\right] \\ = & E\left[\sum_i \sum_m q_{mi}(nT)\right] + T\sum_m \check{\lambda}_m . \end{aligned}$$

Therefore, we have

$$\begin{aligned}\lim_{n\to\infty} E\left[\sum_i \sum_m q_{mi}(t)\right] &\leq \lim_{n\to\infty} E\left[\sum_i \sum_m q_{mi}(nT)\right] + T\sum_m \check{\lambda}_m \\ &< \infty.\end{aligned}$$

This proves the theorem. □

Since the work load $q_{mi}(t)$ is always at least as much as the number of jobs, $Q_{mi}(t)$, from Proposition 1.1, we have that any arrival rate $\check{\lambda} \notin \mathcal{C}$ is not supportable. Thus, we have that Algorithm 3 is *almost throughput optimal*. By this, we mean that given any arrival rate $\check{\lambda} \in \mathcal{C}$, we can choose the parameter T so that the system is stable.

2.3 Simpler Load Balancing Algorithms

Though JSQ routing algorithm is (almost) throughput optimal, the job scheduler needs the workload information from all the servers. This imposes a considerable communication overhead as the arrival rates of jobs and number of servers increase. In this section, we present two alternatives which have much lower routing complexity.

2.3.1 Power-of-two-choices Routing and Myopic MaxWeight Scheduling

An alternative to JSQ routing is the power-of-two-choices algorithm [24, 25, 26], which is much simpler to implement. When a job arrives, two servers are sampled at random, and the job is routed to the server with the smaller queue for that job type. In our algorithm, in each time slot t, for each type of job m, two servers $i_1^m(t)$ and $i_2^m(t)$ are chosen uniformly at random. The job scheduler then routes all the type m job arrivals in this time slot to the server with shorter backlogged workload among these two, i.e., $i_m^*(t) =$

$\arg\min_{i\in\{i_1^m(t),i_2^m(t)\}} q_{mi}(t)$ and so

$$a_{mi}(t) = \begin{cases} a_m(t) & \text{if } i = i_m^*(t) \\ 0 & \text{otherwise.} \end{cases}$$

Otherwise, the algorithm is identical to the JSQ-Myopic MaxWeight algorithm considered earlier. The following theorem shows that the throughput performance using the power-of-two-choices algorithm is similar to that of JSQ routing algorithm when all the servers are identical.

Theorem 2.2. *When all the severs are identical, any load vector that satisfies $\frac{T}{T-S_{\max}}\check{\lambda} \in int(\mathcal{C})$ is stabilizable under the power-of-two-choices routing and myopic MaxWeight allocation algorithm.*

Proof. Again, we use $V(t) = |\mathbf{q}(t)|^2$ as the Lyapunov function. Then, from (2.7), we have

$$\begin{aligned} E[V(t+1) - V(t)|\mathbf{q}(nT) = \mathbf{q}] \le & K_1 + 2E\left[\sum_m \sum_i q_{mi}(t)a_{mi}(t)\middle| \mathbf{q}(nT) = \mathbf{q}\right] \\ & - 2E\left[\sum_i \sum_m q_{mi}(t)N_m^{(i)}(t)\middle| \mathbf{q}(nT) = \mathbf{q}\right] \end{aligned}. \tag{2.17}$$

For fixed m, let $X_m(t)$ be the random variable which denotes the two servers that were chosen by the routing algorithm at time t for jobs of type m. $X_m(t)$ is then uniformly distributed over all sets of two servers. Now, using the tower property of conditional expectation, we have

$$\begin{aligned} & E\left[\sum_i q_{mi}(t)a_{mi}(t)\middle| \mathbf{q}(nT) = \mathbf{q}\right] \\ = & E\left[E\left[\sum_i q_{mi}(t)a_{mi}(t)\middle| \mathbf{q}(nT) = \mathbf{q}, X_m(t) = \{i', j'\}\right]\middle| \mathbf{q}(nT) = \mathbf{q}\right] \\ = & E\left[E\left[\min\left(q_{mi'}(t), q_{mj'}(t)\right) a_m(t) \middle| \mathbf{q}(nT) = \mathbf{q}, X(t) = \{i', j'\}\right]\middle| \mathbf{q}(nT) = \mathbf{q}\right] \end{aligned} \tag{2.18}$$

$$\begin{aligned} \le & E\left[E\left[\frac{q_{mi'}(t) + q_{mj'}(t)}{2} a_m(t)\middle| \mathbf{q}(nT) = \mathbf{q}, X(t) = \{i', j'\}\right]\middle| \mathbf{q}(nT) = \mathbf{q}\right] \\ = & E\left[\frac{L-1}{\binom{L}{2}}\frac{1}{2}\sum_i q_{mi}(t)\check{\lambda}_m\middle| \mathbf{q}(nT) = \mathbf{q}\right] \end{aligned} \tag{2.19}$$

$$
\begin{aligned}
=&\frac{\check{\lambda}_m}{L} E\left[\sum_i q_{mi}(t) \middle| \mathbf{q}(nT) = \mathbf{q}\right] \\
\leq&\frac{\check{\lambda}_m}{L} E\left[\sum_i q_{mi}(nT) + \sum_{t'=nT}^{t-1} \sum_i a_{mi}(t') \middle| \mathbf{q}(nT) = \mathbf{q}\right] \\
\leq&\frac{\check{\lambda}_m}{L} \left(\sum_i q_{mi} + \tau\check{\lambda}_m\right), \qquad (2.20)
\end{aligned}
$$

where $\tau = t - nT$. Equation (2.18) follows from the routing algorithm and (2.19) follows from the fact that $X_m(t)$ is uniformly distributed. Inequality 2.20 follows from the fact that $E[\sum_i a_{mi}(t')] = E[a_m(t')] = \check{\lambda}$.

Since the scheduling algorithm is identical to Algorithm 3, the bound in (2.12) still holds

$$
\begin{aligned}
&-\sum_i \sum_m q_{mi}(t) N_m^{(i)}(t) \\
&\leq L(t-nT)\beta' - \sum_i \sum_m q_{mi}(nT) N_m^{(i)}(nT) + \sum_{t'=nT}^{(n+1)T-S_{\max}-1} \sum_m a_m(t') N_{\max}. \qquad (2.21)
\end{aligned}
$$

Since $\frac{(1+\epsilon)T}{T-S_{\max}}\check{\lambda} \in int(\mathcal{C})$, there exists $\epsilon > 0$ s.t. $\frac{(1+\epsilon)T}{T-S_{\max}}\check{\lambda} \in \mathcal{C}$. We have assumed that all the servers are identical, so $\mathcal{C}$ is obtained by summing L copies of $\text{Conv}(\mathcal{N})$. Thus, $\frac{(1+\epsilon)T}{T-S_{\max}}\check{\lambda} \in \mathcal{C}$, means $\frac{(1+\epsilon)T}{T-S_{\max}}\frac{\check{\lambda}}{L} \in \text{Conv}(\mathcal{N}) = \text{Conv}(\mathcal{N}_i)$ for all i. According to the scheduling algorithm, for each i, we have that

$$
(1+\epsilon)\frac{T}{T-S_{\max}} \sum_m q_{mi}(nT)\frac{\check{\lambda}_m}{L} \leq \sum_m q_{mi}(nT) N_m^{(i)}(nT).
$$

Thus, we get

$$
\begin{aligned}
&-\sum_i \sum_m q_{mi}(t) N_m^{(i)}(t) \\
&\leq L(t-nT)\beta' - \frac{(1+\epsilon)T}{T-S_{\max}} \sum_i \sum_m q_{mi}(nT)\frac{\check{\lambda}_m}{L} + \sum_{t'=nT}^{(n+1)T-S_{\max}-1} \sum_m a_m(t') N_{\max}.
\end{aligned}
$$

Now, substituting this and (2.20) in (2.17) and summing over $t \in [nT, (n+1)T-1]$, as in the proof of Theorem 2.1, we get

$$\begin{aligned}
&E[V((n+1)T) - V(nT)|\mathbf{q}(nT) = \mathbf{q}] \\
\leq &TK_1 + 2\sum_m (\check{\lambda}_m^2 + \check{\lambda}_m N_{\max}) \sum_{\tau=0}^{T-1} \tau + 2L\beta' \sum_{\tau=0}^{T-S_{\max}-1} \tau \\
&+ 2T \sum_{i,m} q_{mi} \frac{\check{\lambda}_m}{L} - 2\frac{(1+\epsilon)T}{T-S_{\max}} \sum_{i,m} q_{mi} \check{\lambda}_m^i (T - S_{\max}) \\
\leq &K_2 - 2\epsilon T \sum_i \sum_m q_{mi} \frac{\check{\lambda}_m}{L}.
\end{aligned}$$

This proof can be completed by applying the Foster-Lyapunov theorem [22, 23] and then observing that the workload does not change by much within a supertime slot, as in the proof of Theorem 2.1. □

2.3.2 Pick-and-Compare Routing and Myopic MaxWeight Scheduling

One drawback of the power-of-two-choices scheduling is that it is throughput optimal only when all servers are identical. In the case of nonidentical servers, one can use pick-and-compare routing algorithm instead of power-of-two-choices. The algorithm is motivated by the pick-and-compare algorithm for wireless scheduling and switch scheduling [27], and is as simple to implement as power-of-two-choices, and can be shown to be optimal even if the servers are not identical. We describe this next. The scheduling algorithm is identical to the previous case.

Pick-and-compare routing works as follows. In each time slot t, for each type of job m, a server $i_m(t)$ is chosen uniformly at random and compared with the server to which jobs were routed in the previous time slot. The server with the shorter backlogged workload among the two is chosen and all the type m job arrivals in this time slot are routed to that server. Let $i_m^*(t)$ be the server to which jobs will be routed in time slot t. Then, $i_m^*(t) =$

$\underset{i\in\{i_m(t),i^*_m(t-1)\}}{\arg\min} \; q_{mi}(t)$ and so

$$a_{mi}(t) = \begin{cases} a_m(t) & \text{if } i = i^*_m(t) \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 2.3. *Assume $a_m(t) \le a_{\max}$ for all t and m. Any job load vector that satisfies $\frac{T}{T-S_{\max}}\check{\lambda} \in int(\mathcal{C})$ is stabilizable under the pick-and-compare routing and myopic MaxWeight allocation algorithm.*

Proof. Consider the irreducible Markov chain $\mathcal{Y}(t) = (\mathbf{Y}(t), \{i^*_m(t)\}_m)$ and the Lyapunov function $V(t) = |\mathbf{q}(t)|^2$. Then, as in the proof of Theorem 2.2, similar to (2.17) for $t \ge nT$, we have

$$\begin{aligned} &E[V(t+1) - V(t)|\mathbf{q}(nT) = \mathbf{q}, i^*_m(nT) = i'] \\ \le & K_1 + 2E\left[\sum_m \sum_i q_{mi}(t)a_{mi}(t) \middle| \mathbf{q}(nT) = \mathbf{q}, i^*_m(nT) = i'\right] \\ & - 2E\left[\sum_i \sum_m q_{mi}(t)N^{(i)}_m(t) \middle| \mathbf{q}(nT) = \mathbf{q}, i^*_m(nT) = i'\right]. \end{aligned} \tag{2.22}$$

Since $\frac{T}{T-S_{\max}}\check{\lambda} \in int(\mathcal{C})$, there exists and $\epsilon > 0$ such that, $(1+\epsilon)\frac{T}{T-S_{\max}}\check{\lambda} \in \mathcal{C}$ and so there exists $\{\check{\lambda}^i\}_i$ such that $(1+\epsilon)\frac{T}{T-S_{\max}}\check{\lambda}^i \in \text{Conv}(\mathcal{N}_i)$ for all i and $\check{\lambda} = \sum_i \check{\lambda}^i$. This $\{\check{\lambda}^i\}_i$ can be chosen so that there is a κ so that $\check{\lambda}_m \le \kappa\check{\lambda}^i_m$. This is possible because if $\check{\lambda}_m > 0$ and $\check{\lambda}_m$ is not on the boundary of $\mathcal{C}$, one can always find $\{\check{\lambda}^i_m\}_i$ so that $\check{\lambda}^i_m > 0$.

Since the scheduling part of the algorithm is identical to Algorithm 3, (2.15) still holds for $t \in [nT, (n+1)T - S_{\max}]$. Thus, we have

$$\begin{aligned} &-\sum_i \sum_m q_{mi}(t)N^{(i)}_m(t) \\ &\le L(t-nT)\beta'' - \frac{(1+\epsilon)T}{T-S_{\max}} \sum_i \sum_m q_{mi}(nT)\check{\lambda}^i_m, \end{aligned} \tag{2.23}$$

where $\beta'' = MN_{\max}(a_{\max} + N_{\max}) + MS_{\max}N^2_{max}$.

We also need a bound on the increase in $-\sum_i \sum_m q_{mi}(t)N^{(i)}_m(t)$ over multiple super time slots. So, for any n', we have

$$\begin{aligned}
&\sum_i \sum_m q_{mi}(nT) N_m^{(i)}(nT) \\
\leq &\sum_i \sum_m q_{mi}((n+n')T) N_m^{(i)}(nT) + n'TLMN_{\max}^2 \\
\leq &\sum_i \sum_m q_{mi}((n+n')T) N_m^{(i)}((n+n')T) + n'TL\beta'',
\end{aligned}$$

where the second inequality follows from the fact that we use maxweight scheduling every T slots and from the definition of β''. Now, again, using (2.13), and (2.23), for any t such that $1 \leq (t \bmod T) \leq T - S_{\max}$, we have

$$\begin{aligned}
&-\sum_i \sum_m q_{mi}(t) N_m^{(i)}(t) \\
&\leq L(t-nT)\beta'' - \frac{(1+\epsilon)T}{T-S_{\max}} \sum_i \sum_m q_{mi}(nT) \check{\lambda}_m^i. \qquad (2.24)
\end{aligned}$$

Fix m. Let $i_{\min}^m = \underset{i \in \{1,2,,,\mathrm{L}\}}{\arg\min}\, q_{mi}(nT)$. Note that $|q_{mi}(t) - q_{mi}(t-1)| = \left|a_{mi}(t) - N_m^{(i)}(t)\right| \leq a_{\max} + N_{\max}$. Therefore, once there is a $t_0 \geq nT$ such that $i_m^*(t_0)$ satisfies

$$q_{mi_m^*(t_0)}(t_0) \leq q_{mi_{min}^m}(t_0), \qquad (2.25)$$

then, for all $t \geq t_0$, we have $q_{mi_m^*(t)}(t) \leq q_{mi_{min}^m}(nT) + (t-nT)\left(a_{\max} + N_{\max}\right)$. Probability that (2.25) does not happen is at most $\left(1 - \frac{1}{L}\right)^{(t_0 - nT)}$. Choose t_0 so that this probability is less than $p = \epsilon/4\kappa$. Then, $(1+\kappa p) = 1 + \epsilon/4$. Choose k so that $kT > (t_0 - nT)$ and $((n+k)T - t_0) + \kappa(t_0 - nT) \leq kT(1+\epsilon/4)$.

Then

$$\begin{aligned}
&\sum_{t=nT}^{(n+k)T-1} E\left[\sum_i q_{mi}(t) a_{mi}(t) \middle| \mathbf{q}(nT) = \mathbf{q}, i_m^*(nT) = i'\right] \\
=&\sum_{t=nT}^{t_0} E\left[\sum_i q_{mi}(t) a_{mi}(t) \middle| \mathbf{q}(nT) = \mathbf{q}, i_m^*(nT) = i'\right] \\
&+ \sum_{t=t_0}^{(n+k)T-1} E\left[\sum_i q_{mi}(t) a_{mi}(t) \middle| \mathbf{q}(nT) = \mathbf{q}, i_m^*(nT) = i'\right] \qquad (2.26) \\
\leq&\check{\lambda}_m (t_0 - nT) \sum_i q_{mi} + \sum_{t=nT}^{t_0} (t-nT)\left(a_{\max} + N_{\max}\right) L a_{\max}
\end{aligned}$$

$$
\begin{aligned}
&+ \sum_{t=t_0}^{(n+k)T-1} (1-p)\check{\lambda}_m \left(q_{mi^m_{min}} + (t-nT)\left(a_{\max}+N_{\max}\right)\right) \\
&+ p\check{\lambda}_m \left((n+k)T - t_0\right) \sum_i q_{mi} \\
&+ p \sum_{t=t_0}^{(n+k)T-1} (t-nT)\left(a_{\max}+N_{\max}\right) La_{\max} \qquad (2.27)
\end{aligned}
$$

$$
\begin{aligned}
\leq &(1-p)\left((n+k)T - t_0\right) \sum_i q_{mi^m_{min}} \check{\lambda}^i_m + \sum_{\tau=0}^{kT} \tau \left(a_{\max}+N_{\max}\right) La_{\max} \\
&+ (1-p)\check{\lambda}_m (t_0 - nT) \sum_i q_{mi} + p\check{\lambda}_m kT \sum_i q_{mi} \qquad (2.28) \\
\leq &K_3 + (1-p)\left((n+k)T - t_0\right) \sum_i q_{mi}\check{\lambda}^i_m \\
&+ (1-p)\kappa(t_0 - nT) \sum_i q_{mi}\check{\lambda}^i_m + \kappa pkT \sum_i q_{mi}\check{\lambda}^i_m \qquad (2.29) \\
\leq &K_3 + (1-p)kT(1+\epsilon/4) \sum_i q_{mi}\check{\lambda}^i_m + (1+\epsilon/4)\kappa pkT \sum_i q_{mi}\check{\lambda}^i_m \qquad (2.30) \\
\leq &K_3 + kT(1+\epsilon/4)^2 \sum_i q_{mi}\check{\lambda}^i_m \qquad (2.31) \\
\leq &K_3 + kT(1+3\epsilon/4) \sum_i q_{mi}\check{\lambda}^i_m \qquad (2.32)
\end{aligned}
$$

where $K_3 = \sum_{\tau=0}^{kT} \tau \left(a_{\max}+N_{\max}\right) La_{\max}$. Equations (2.30) and (2.31) follow from our choice of k and p respectively.

Now, substituting (2.32) and (2.24) in (2.22) and summing over $t \in [nT, (n+1)T-1]$, we get

$$
\begin{aligned}
&E[V((n+k)T) - V(nT) | \mathbf{q}(nT) = \mathbf{q}, i^*_m(nT) = i'] \\
\leq &K_4 + 2kT(1+3\epsilon/4) \sum_m \sum_i q_{mi}\check{\lambda}^i_m \\
&- \sum_{t=nT}^{(n+k)T-1} 2E\left[\sum_i \sum_m q_{mi}(t) N^{(i)}_m(t) \middle| \mathbf{q}(nT) = \mathbf{q}, i^*_m(t) = i' \right] \\
\leq &K_4 + 2kT(1+3\epsilon/4) \sum_m \sum_i q_{mi}\check{\lambda}^i_m \\
&- 2(1+\epsilon)\frac{T}{T - S_{\max}} \sum_m \sum_i q_{mi}\check{\lambda}^i_m k(T - S_{\max})
\end{aligned}
$$

$$\leq K_4 + -\frac{1}{2} kT\epsilon \sum_m \sum_i q_{mi} \check{\lambda}_m^i$$

where $K_4 = kTK_1 + MK_3 + 2L\beta'' \sum_{\tau=0}^{kT-S_{\max}-1} \tau$. The result follows from the Foster-Lyapunov theorem [22, 23]. $\square$

Note that the assumption that the arrivals in each time slot are bounded by $a_{\max}$ can easily be relaxed, similar to the proof of theorems 2.1 and 2.2.

2.4 Discussion and Simulations

The parameter T in our algorithms can be arbitrarily large, but needs to be finite for the proofs to be valid. A natural question is: What happens if we let T go to infinity? When $T = \infty$, our scheduling algorithm reduces to a myopic MaxWeight scheduling in each time slot. Under no additional assumptions, such an algorithm is not throughput optimal. For instance, the same example given to show that Best-Fit policy is not throughput optimal in Chapter 1 would also show that such an algorithm is also not throughput optimal. To completely characterize the optimality of such a myopic MaxWeight scheduling algorithm is an open question. However, we will address a slightly related question in the next chapter.

In this section, we use simulations to compare the performance for different values of T. We will also compare the centralized myopic MaxWeight scheduling algorithm, and the joint routing and scheduling algorithm, based on the power-of-two-choices and MaxWeight scheduling. We consider a cloud computing cluster with 100 identical servers, and each server has the hardware configuration specified in Example 1.1. We assume jobs being served in this cloud belong to one of the three types specified in Table 1.1. So VM configurations $(2, 0, 0)$, $(1, 0, 1)$, and $(0, 1, 1)$ are the three maximal VM configurations for each server. It is easy to verify that the load vector $\lambda^{(1)} = (1, \frac{1}{3}, \frac{2}{3})$ is on the boundary of the capacity region of a server.

To model the large variability in jobs sizes, we assume job sizes are distributed as follows: when a new job is generated, with probability 0.7, the size is an integer that is uniformly distributed in the interval $[1, 50]$, 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

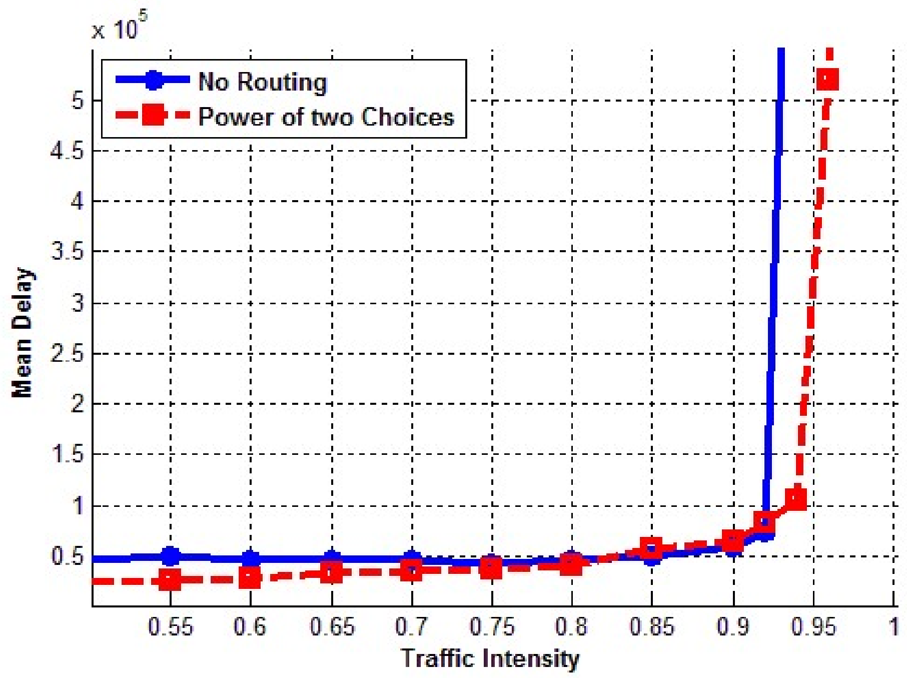


Figure 2.2: Comparison of the mean delays in the cloud computing cluster in the case with a common queue and in the case with power-of-two-choices routing when frame size is 4000

and 500. Therefore, the average job size is 130.5 and the maximum job size is 500. We call this distribution A.

We further assume the number of type-i jobs arriving at each time slot follows a binomial distribution with parameter $(\rho\frac{\lambda_i^{(1)}}{130.5}, 100)$. We varied the traffic intensity parameter ρ from 0.5 to 1 in our simulations. Here traffic intensity is the factor by which the load vector has to be divided so that it lies on the boundary of the capacity region. Each simulation was run for $500,000$ time slots. First we study the difference between power-of-two-choice routing and JSQ routing by comparing the mean delays of the two algorithms at various traffic intensities for different choices of frame sizes. Our simulation results indicate that the delay performance of the two algorithms was not very different. For concision, we only provide a representative sample of our simulations here for the case where the frame size is 4000 in Figure 2.2.

Next, we show the performance of our algorithms for various values of the frame size T in Figure 2.3. Again, we have only shown a representative sample for the power-of-two-choices routing (with myopic MaxWeight scheduling). From Theorems 2.1 and 2.2, we know that any load less than $\frac{T-S_{\max}}{T}$ is

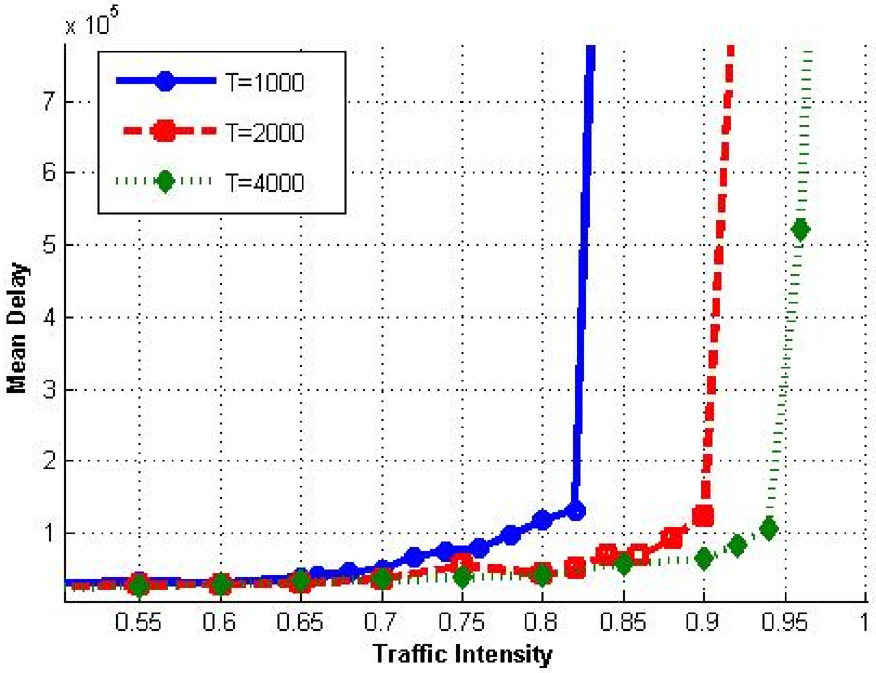


Figure 2.3: Comparison of power-of-two-choices routing algorithm for various frame lengths T

supportable. The simulations indicate that the system is stable even for the loads greater than this value. This is to be expected since our proofs of Theorems 2.1 and 2.2 essentially ignore the jobs that are scheduled in the last $S_{\max}$ time slots of a frame. However, the fact that the stability region is larger for larger values of T is confirmed by the simulations.

It is even more interesting to observe the delay performance of our algorithms as T increases. Figure 2.3 indicates that the delay performance does not degrade as T increases and the throughput increases with T. So the use of queue-length information seems to be the key ingredient of the algorithm while the optimal implementation of the MaxWeight algorithm seems to be secondary.

2.5 Conclusions

In this chapter, we have studied a few different resource allocation algorithms for the stochastic model of IaaS cloud computing that was presented in the Introduction. We assume that job sizes are known and are bounded and that preemption is not allowed. We made a connection with scheduling in ad hoc wireless networks and proposed a frame based myopic MaxWeight algorithm. We presented three different routing algorithms, viz., join the shortest queue, power-of-two-choices and pick-and-compare. We have shown that all these algorithms can be made nearly throughput-optimal by choosing sufficiently long frame durations. Simulations indicate that long frame durations are not only good from a throughput perspective but also seem to provide good delay performance.

CHAPTER 3

UNKNOWN JOB SIZES

The algorithms presented in Chapter 2 assume that the job sizes are bounded and are known when the job arrives into the system. This assumption is not realistic in some settings. Many times, the duration of a job is not known until the job finishes. Moreover the bounded job sizes assumption excludes simple job size distributions like geometric distribution because it is not bounded. In this chapter, we will present algorithms when job sizes are not known.

The scheduling algorithm presented in this chapter is inspired by the one studied in [19] for input-queued switched. Since a MaxWeight schedule cannot be chosen in every time slot without disturbing the jobs in service, a MaxWeight schedule is chosen only at every refresh time. A time slot is called a *refresh time* if no jobs 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 [19] 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 [19] that the refresh time for a system with infinitely backlogged 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 original system be called system 1 and the system with infinitely backlogged 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 [19] 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 infinitely backlogged 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 [28] as well as an example similar to the counter example to Best-Fit policy in Chapter 1 show that the policy presented in [19] 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.

In this chapter we present 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 the algorithm in Chapter 2. The scheduling part of our algorithm is based on [19], but includes an additional routing component. Further, our proof of throughput-optimality is different from the one in [19] due to the earlier mentioned reasons.

If the job sizes are known, one can then use backlogged workload as the weight in the algorithm presented here. In that case, this algorithm does not waste any resources unlike the algorithms in Chapter 2 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 chapter works even when the job sizes are not bounded, for instance, when the job sizes are geometrically distributed.

In terms of proof technique, in this chapter, 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 [29]) in the case of geometric job sizes. For general job sizes, we use a log-weight functions. Log-weight functions are known to have good performance properties [30] and are also amenable to low-complexity implementations using randomized algorithms [31, 32].

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 [33] 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 [33].

We will first present the refresh times based scheduling algorithm and argue that the refresh times are bounded. We illustrate the use of this result by first proving throughput optimality in the simple case when the job sizes are geometrically distributed. In the following section, we present the proof for the case of general job size distributions. The results in this chapter have been presented in [34] and [35].

3.1 Algorithm

There are two main challenges in this setting. The first is that, since the job sizes are unknown, the total backlogged workload cannot be used to calculate the weight of different schedules. We address this by using number of jobs as a proxy for the backlogged workload. However, it is not clear if such an algorithm is still optimal. It turns out that one can use a modified MaxWeight algorithm with a different weight function. It was shown in [29] that a wide class of weight functions can be used in MaxWeight scheduling for wireless channels. When the total backlogged workload is not known, a logarithmic function of the queue length can be used for throughput optimality.

The second challenge is that since the job sizes are not bounded, one cannot use the idea of super time slots, because one cannot make sure that a MaxWeight schedule can be chosen at the beginning of every super time slot. To address this challenge, recall that the key intuition from the results in the previous section is that, a MaxWeight schedule should be chosen 'often enough.' However, we since we cannot choose a MaxWeight schedules at some fixed time slots, we choose them whenever we can. To precisely state such an algorithm, we need the notion of 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 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. Based on these ideas, we present Algorithm 4.

To state the optimality result for this algorithm in general, we need the following assumption on the job sizes. Let $\mathcal{S}$ be the support of the random variable S, the job size, i.e., $\mathcal{S} = \{n \in \mathbb{N} : P(S = n) > 0\}$. The job size distribution is assumed to satisfy the following assumption.

Assumption 3.1. 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 (3.1) means that when the job sizes are not bounded, they have geometric tails. For example, truncated heavy-tailed distributions with

Algorithm 4 JSQ Routing and 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 $i^*_m(t) = \underset{i \in \{1,2,,,L\}}{\arg\min} \mathbf{Q}_{mi}(t)$. Therefore, the arrivals to $\mathbf{Q}_{mi}$ in time slot t are given by

$$\mathbf{A}_{mi}(t) = \begin{cases} A_m(t) & \text{if } i = i^*_m(t) \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

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.,

$$\widetilde{N}^{(i)}(t) \in \underset{N \in \mathcal{N}_i}{\arg\max} \sum_m g(\mathbf{Q}_{mi}(t)) N_m. \tag{3.2}$$

   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, $N^{(i)}(t) = \widetilde{N}^{(i)}_m(t)$.

arbitrarily high variance would be allowed but purely heavy-tailed distributions would not be allowed under our model.

3.1.1 Refresh Times

One of the key ideas in the proof is to show that the refresh times occur often enough. We will now present a lemma to this effect. Let us denote the n^{th} refresh time by t_n. Let $z_n = t_{n+1} - t_n$ be the duration (in slots) between the n^{th} and $(n+1)^{\text{th}}$ refresh times. Then, the duration between the refresh times is bounded as follows.

Lemma 3.1. *There exists constants $\mathcal{K}_1 > 0$ and $\mathcal{K}_2 > 0$ such that $\mathbb{E}[z_n] < \mathcal{K}_1$ and $\mathbb{E}[z_n^2] < \mathcal{K}_2$.*

This lemma is proved by first obtaining a lower bound on the probability that the next time slot is a refresh time. This bound is then used to construct a coupled Bernoulli process that gives the required bound. We present the complete proof now.

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 at least C. Therefore, 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 at least $\overline{C} \triangleq C^{LMN_{\max}} > 0$. 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.

□

3.2 Throughput Optimality - Geometric Job Sizes

Before presenting the general result, in this section, we will characterize the throughput performance of Algorithm 4 in the special case when the job sizes are geometrically distributed. The proof of this result is simpler than the general proof and illustrates some of the ideas. We will consider a more general case in the next section. Note that when the job sizes are geometric, Assumption 3.1 is clearly satisfied and so Lemma 3.1 is applicable.

In the case of geometric job sizes, a wide class of functions $g(y)$ can be used to obtain a stable MaxWeight policy [29]. 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 3.1. *Assume that the job arrivals satisfy $A_m(t) \leq A_{\max}$ for all m and t. When the job sizes are geometrically distributed with mean job size vector $\overline{S}$, any job load vector that satisfies $(\lambda, \overline{S}) \in int(\widehat{\mathcal{C}})$ is supportable under the JSQ routing and MaxWeight allocation as described in Algorithm 4 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 4.

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 the 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 $0 \le \tau < z_n$.

$$\begin{aligned}
&(V(t+1) - V(t)) \\
=&\sum_i \sum_m \overline{S}_m \left(\mathbf{Q}_{mi}(t) + \mathbf{A}_{mi}(t) - \mathbf{D}_{mi}(t)\right)^2 - \overline{S}_m \mathbf{Q}_{mi}^2(t) \\
=&2\sum_i \sum_m \overline{S}_m \mathbf{Q}_{mi}(t) \left(\mathbf{A}_{mi}(t) - \mathbf{D}_{mi}(t)\right) \\
&+\sum_i \sum_m \overline{S}_m \left(\mathbf{A}_{mi}(t) - \mathbf{D}_{mi}(t)\right)^2 \\
\le&2\sum_m \sum_i \overline{S}_m \mathbf{Q}_{mi}(t) \left(\mathbf{A}_{mi}(t) - \mathbf{D}_{mi}(t)\right) + K_5,
\end{aligned}$$

where $K_5 = L(A_{\max} + N_{\max})^2 (\sum_m \overline{S}_m)$.

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 $E_{\mathbf{q}}\left[(.)\right]$.

$$\begin{aligned}
&E[V(\widetilde{\mathbf{Q}}(n+1)) - V(\widetilde{\mathbf{Q}}(n))|\widetilde{\mathbf{Q}}(n) = \mathbf{q}] \\
=&E[V(t_{n+1}) - V(t_n)|\mathbf{Q}(t_n) = \mathbf{q}] \\
=&E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1} V(t_n+\tau+1) - V(t_n+\tau)\right] \\
\le&E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1} 2\sum_{m,i} \left(\overline{S}_m \mathbf{Q}_{mi}(t_n+\tau)\mathbf{A}_{mi}(t_n+\tau)\right.\right. \\
&\left.\left. - \overline{S}_m \mathbf{Q}_{mi}(t_n+\tau)\mathbf{D}_{mi}(t_n+\tau)\right) + K_5\right]. \qquad (3.3)
\end{aligned}$$

The last term above is bounded by $K_5\mathcal{K}_1$ from Lemma 3.1. We will now bound the first term in (3.3). From the definition of $\mathbf{A}_{mi}$ in the routing algorithm in (3.1), we have

$$\begin{aligned}
&2E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_m\sum_i \overline{S}_m \mathbf{Q}_{mi}(t_n+\tau)\mathbf{A}_{mi}(t_n+\tau)\right]\\
=&2E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_m \overline{S}_m \mathbf{Q}_{mi_m^*(t_n+\tau)}(t_n+\tau)A_m(t_n+\tau)\right]\\
\leq&2E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_m \overline{S}_m\Big(\mathbf{Q}_{mi_m^*(t_n)}(t_n)A_m(t_n+\tau)+\tau A_{\max}^2\Big)\right]\\
\leq&2\sum_m \overline{S}_m \mathbf{q}_{mi_m^*}E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}A_m(t_n+\tau)\right]+\sum_m A_{\max}^2\overline{S}_m E_{\mathbf{q}}\left[z_n^2\right]\\
\leq&A_{\max}^2\mathcal{K}_2\sum_m \overline{S}_m + 2E_{\mathbf{q}}\left[z_n\right]\sum_m \overline{S}_m \mathbf{q}_{mi_m^*}\lambda_m, \qquad (3.4)
\end{aligned}$$

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[1] [36, Chap 6, Cor 3.1 and Sec 4(a)] and Lemma 3.1, we have (3.4).

Now we will bound the second term in (3.3). 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 the memoryless property of

[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 $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$, $E[\sum_{n=1}^N X_n] = E[X_n]E[N]$.

geometric distribution, $\mathcal{I}^i_{j,m}(t)$ are i.i.d. Bernoulli with mean $1/\overline{S}_m$.

If the j^{th} job was scheduled, $\mathcal{I}^i_{j,m}(t)$ corresponds to an actual departure. If not (i.e., if there was unused service), there is no actual departure corresponding to this. Let $\widehat{\mathbf{D}}_{mi}(t) = \sum_{j=1}^{\widetilde{N}^{(i)}_m(t_n)} \mathcal{I}^i_{j,m}(t)$ 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

$$\begin{aligned}
\mathbf{Q}_{mi}(t)\mathbf{D}_{mi}(t) &= \left(\mathbf{Q}_{mi}(t)\mathbf{D}_{mi}(t)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t)\geq N_{\max}} \\
&\quad + \left(\mathbf{Q}_{mi}(t)\mathbf{D}_{mi}(t)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t)< N_{\max}} \\
&\geq \left(\mathbf{Q}_{mi}(t)\widehat{\mathbf{D}}_{mi}(t)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t)\geq N_{\max}} \\
&\quad + \left(\mathbf{Q}_{mi}(t)\left(\widehat{\mathbf{D}}_{mi}(t) - N_{\max}\right)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t)< N_{\max}} \\
&\geq \mathbf{Q}_{mi}(t)\widehat{\mathbf{D}}_{mi}(t) - N^2_{\max}. \qquad (3.5)
\end{aligned}$$

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

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

Using this with (3.5), we can bound the second term in (3.3) as follows:

$$\begin{aligned}
&2E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_{i,m}\overline{S}_m\mathbf{Q}_{mi}(t_n+\tau)\mathbf{D}_{mi}(t_n+\tau)\right] \\
\geq& 2E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_{i,m}\overline{S}_m\mathbf{Q}_{mi}(t_n+\tau)\widehat{\mathbf{D}}_{mi}(t_n+\tau)\right] \\
&- LN^2_{\max}\sum_m \overline{S}_m 2E_{\mathbf{q}}\left[z_n\right] \\
\geq& 2E_{\mathbf{q}}\left[\sum_{i,m}\overline{S}_m\mathbf{Q}_{mi}(t_n)\sum_{\tau=0}^{z_n-1}\widehat{\mathbf{D}}_{mi}(t_n+\tau)\right] - K_6 \\
=& 2E_{\mathbf{q}}\left[z_n\right]\sum_{i,m}\mathbf{q}_{mi}\widetilde{N}^{(i)}_m - K_6 \qquad (3.6)
\end{aligned}$$

where $K_6 = LN^2_{\max}\sum_m \overline{S}_m(2\mathcal{K}_1 + \mathcal{K}_2)$. Let $\widehat{\mathcal{F}}^{(n)}_\tau$ denote the filtration generated by $\{\mathbf{Y}(t_n+\tau), \widehat{D}(t_n+\tau)\}$. Then, $\mathcal{F}^{(n)}_\tau \subseteq \widehat{\mathcal{F}}^{(n)}_\tau$. 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 we get (3.6).

Since $(\lambda, \overline{S}) \in int(\widehat{\mathcal{C}})$, there exists $\epsilon > 0$ such that $((1+\epsilon)\lambda, \overline{S}) \in \widehat{\mathcal{C}}$, there exists $\{(1+\epsilon)\lambda^i\}_i$ such that $\lambda^i \circ \overline{S} \in \mathrm{Conv}(\mathcal{N}_i)$ for all i and $\lambda = \sum\limits_i \lambda^i$. According to the scheduling algorithm (3.2), for each server i, we have that

$$\sum_m \mathbf{Q}_{mi}(t_n)(1+\epsilon)\lambda_m^i \overline{S}_m \le \sum_m \mathbf{Q}_{mi}(t_n)\widetilde{N}_m^{(i)}(t_n). \tag{3.7}$$

Then, from (3.4), (3.3) and (3.6), we get

$$\begin{aligned}
&E[V(\widetilde{\mathbf{X}}(n+1)) - V(\widetilde{\mathbf{X}}(n))|\widetilde{\mathbf{Q}}(n) = \mathbf{q}, \widetilde{\mathbf{Y}}(n)] \\
\le& K_7 + 2E_{\mathbf{q}}[z_n]\sum_m \overline{S}_m \overline{\mathbf{q}}_{mi_m^*}\lambda_m - 2E_{\mathbf{q}}[z_n]\sum_{i,m}\mathbf{q}_{mi}\widetilde{N}_m^{(i)} \\
\overset{(a)}{\le}& K_7 - 2\epsilon E_{\mathbf{q}}[z_n]\sum_i\sum_m \mathbf{q}_{mi}\lambda_m^i\overline{S}_m \\
\overset{(ba)}{\le}& K_7 - 2\epsilon\sum_i\sum_m \mathbf{q}_{mi}\lambda_m^i\overline{S}_m,
\end{aligned}$$

where $K_7 = A_{\max}^2\mathcal{K}_2\sum_m \overline{S}_m + K_6$. Inequality (a) follows from $\lambda = \sum\limits_i \lambda^i$ and (3.7). Inequality (b) follows from $z_n \ge 1$.

Then, from the Foster-Lyapunov theorem [22, 23], 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 $\lim_{n\to\infty}\sum_m\sum_i E[\mathbf{Q}_{mi}(t)] \le \mathcal{K}_3$.

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

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

As $t \to \infty$, we get

$$\begin{aligned}
&\limsup_{t\to\infty}\sum_m\sum_i E[\mathbf{Q}_{mi}(t)] \\
\le& \limsup_{n\to\infty}\sum_m\sum_i E[(\mathbf{Q}_{mi}(t_n) + z_n(A_{\max} + N_{\max}))]
\end{aligned}$$

$$\leq \mathcal{K}_3 + \mathcal{K}_1 LM(A_{\max} + N_{\max}).$$

□

3.3 Throughput Optimality - General Job Size Distribution

In this section, we will consider a general job size distribution that satisfies Assumption 3.1. We will show that Algorithm 4 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. We first present some preliminaries.

The process $\mathbf{X}(t) = (\mathbf{Q}(t), \mathbf{Y}(t))$ is a Markov chain, where $\mathbf{Y}(t)$ is defined in the section 3.2. 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

$$\overline{\mathbf{D}}_{mi}(t) = \begin{cases} W_m(l) & \text{with prob } \widehat{p}_{ml} \\ W_m(l) - W_m(l+1) & \text{with prob } 1 - \widehat{p}_{ml}. \end{cases} \tag{3.8}$$

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 3.1, the expected remaining workload is upper bounded by that of a geometric distribution. We will now show this formally.

From Assumption 3.1 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,

$$\begin{aligned} W_m(l) = E[S_m - l | S_m > l] \qquad &= \sum_{k=0}^{\infty} P(S_m > l+k | S_m > l) \\ &\leq \sum_{k=0}^{\infty} (1-C)^k \ \leq 1/C. \end{aligned} \tag{3.9}$$

Then from (3.8), 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

$$\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t) \leq A_{\max}\overline{S}_{\max} + \frac{N_{\max}}{C}. \tag{3.10}$$

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

$$\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t) \geq -\frac{N_{\max}}{C}. \tag{3.11}$$

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 3.2. *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 3.3. *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. Recall $\widehat{p}_{ml} = P(S_m = l+1|S_m > l)$. We have

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

Thus, we have

$$W_m(l) - W_m(l+1) = 1 - W_m(l+1)\,(\widehat{p}_{ml})\,. \tag{3.12}$$

Then, from (3.8),

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

from (3.12).

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 job 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

$$G(q) = \int_0^q g(y)dy = \int_0^q \log(1 + y)dy = (1+q)\log(1+q) - q.$$

We will show that the drift of $V(\overline{\mathbf{Q}})$ between two refresh times is negative. To do this, we will need the following generalized form the well-known Wald's identity.

Lemma 3.4 (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 $E[S_N] = E[T_N]$.

Then, to use the 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 job sizes are not bounded, this set may not be finite and so the Foster-Lyapunov theorem is not applicable. We will instead use the following result by Hajek [33, Thm 2.3, Lemma 2.1], which can be thought of as a generalization of the Foster-Lyapunov theorem for non-Markovian random processes.

Lemma 3.5. *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, $E[Z_{n+1} - Z_n|\mathcal{F}_n] \leq -\epsilon_0$ 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 lemma 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 $U_\alpha(\overline{\mathbf{Q}}) = (\sum_{i,m}(\overline{\mathbf{Q}}_{mi})^{(1+\alpha)})^{\frac{1}{1+\alpha}}$ [37]. 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.

$$U(\overline{\mathbf{Q}}) = G^{-1}(V(\overline{\mathbf{Q}})) = G^{-1}(\sum_{i,m} G(\overline{\mathbf{Q}}_{mi})). \tag{3.13}$$

This is related to the Lambert 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 3.6. *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 [37]. The proof is presented in Appendix A.

We will need the following Lemma to verify the conditions C1 and C2 in Lemma 3.5.

Lemma 3.7. *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 now present the main theorem, which establishes the throughput optimality of Algorithm 4 when $g(q)=\log(1+q)$.

Theorem 3.1. *Assume that the job arrivals satisfy* $A_m(t)\leq A_{\max}$ *for all* m *and* t *and that the job size distribution satisfies Assumption 3.1. Then, any job load vector that satisfies* $(\lambda,\overline{S})\in int(\widehat{\mathcal{C}})$ *is supportable under JSQ routing and MaxWeight allocation as described in Algorithm 4 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 4.

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 Lemma 3.5. We will study the drift of $U(n)$ in terms of drift of $V(n)$ using Lemma 3.6. First consider the following one step drift of $V(t)$.

$$\begin{aligned}
&(V(t+1)-V(t))\\
=&\sum_{m,i}\left(G\left(\overline{\mathbf{Q}}_{mi}(t+1)\right)-G\left(\overline{\mathbf{Q}}_{mi}(t)\right)\right)\\
\leq&\sum_{m,i}\left(\overline{\mathbf{Q}}_{mi}(t+1)-\overline{\mathbf{Q}}_{mi}(t)\right)g(\overline{\mathbf{Q}}_{mi}(t+1)) \qquad (3.14)\\
=&\sum_{m,i}\left(\overline{\mathbf{Q}}_{mi}(t+1)-\overline{\mathbf{Q}}_{mi}(t)\right)\left(g(\overline{\mathbf{Q}}_{mi}(t+1))-g(\overline{\mathbf{Q}}_{mi}(t))\right)
\end{aligned}$$

$$+\sum_{m,i}\left(\overline{\mathbf{A}}_{mi}(t)-\overline{\mathbf{D}}_{mi}(t)\right)g(\overline{\mathbf{Q}}_{mi}(t)), \tag{3.15}$$

where (3.14) follows from the convexity of $G(.)$. To bound the first term in (3.15), 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

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

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

So the first term in (3.15) can be bounded as

$$\begin{aligned}&\sum_{m,i}\left(\overline{\mathbf{Q}}_{mi}(t+1)-\overline{\mathbf{Q}}_{mi}(t)\right)\left(g(\overline{\mathbf{Q}}_{mi}(t+1))-g(\overline{\mathbf{Q}}_{mi}(t))\right)\\&\leq\sum_{m,i}\left|\overline{\mathbf{Q}}_{mi}(t+1)-\overline{\mathbf{Q}}_{mi}(t)\right|\left|g(\overline{\mathbf{Q}}_{mi}(t+1))-g(\overline{\mathbf{Q}}_{mi}(t))\right|\\&\leq\sum_{m,i}\left|\left(\overline{\mathbf{Q}}_{mi}(t+1)-\overline{\mathbf{Q}}_{mi}(t)\right)\right|^2\qquad\leq K_8,\end{aligned}$$

where $K_8=LM(A_{\max}\overline{S}_{\max}+\frac{N_{\max}}{C})^2$. The last inequality follows from (3.10) and (3.11). Thus, we have

$$V(t+1)-V(t)\leq K_8+\sum_{m,i}(\overline{\mathbf{A}}_{mi}(t)-\overline{\mathbf{D}}_{mi}(t))g(\overline{\mathbf{Q}}_{mi}(t)). \tag{3.16}$$

Similarly, it can be shown that

$$V(t)-V(t+1)\leq K_8+\sum_{m,i}\left(\overline{\mathbf{D}}_{mi}(t)-\overline{\mathbf{A}}_{mi}(t)\right)g(\overline{\mathbf{Q}}_{mi}(t+1)). \tag{3.17}$$

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 (3.16) in the drift of the sampled system, we get

$$E[V(\widetilde{\mathbf{X}}(n+1))-V(\widetilde{\mathbf{X}}(n))|\widetilde{\mathbf{Q}}(n)=\mathbf{q},\widetilde{\mathbf{Y}}(n)]$$

$$
\begin{aligned}
&= E[V(t_{n+1}) - V(t_n)|\mathbf{Q}(t_n) = \mathbf{q}, \mathbf{Y}(t_n)] \\
&= E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1} V(t_n+\tau+1) - V(t_n+\tau)\right] \\
&\leq E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\left(\sum_{m,i}\left(g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{A}}_{mi}(t_n+\tau)\right.\right.\right. \\
&\qquad \left.\left.\left. -g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\right)\right) + K_8\right]. \qquad (3.18)
\end{aligned}
$$

The last term above is bounded by $K_8\mathcal{K}_1$ from Lemma 3.1. We will now bound the first term in (3.18).

$$
\begin{aligned}
&E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_m\sum_i g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{A}}_{mi}(t_n+\tau)\right] \\
&= E_{\mathbf{q}}[\sum_{\tau=0}^{z_n-1}\sum_m g(\overline{\mathbf{Q}}_{mi^*_m(t_n+\tau)}(t_n+\tau))A_m(t_n+\tau)\overline{S}_m] \\
&\overset{(a)}{\leq} E_{\mathbf{q}}[\sum_{\tau=0}^{z_n-1}\sum_m g(\overline{\mathbf{Q}}_{mi^*_m}(t_n)+\tau A_{\max}\overline{S}_m+\tau N_{\max}/C)A_m(t_n+\tau)\overline{S}_m] \\
&\overset{(b)}{\leq} E_{\mathbf{q}}[\sum_{\tau=0}^{z_n-1}\sum_m g(\overline{\mathbf{Q}}_{mi^*_m}(t_n))A_m(t_n+\tau)\overline{S}_m+\tau A_{\max}^2\overline{S}_m^2+\tau A_{\max}\overline{S}_m N_{\max}/C] \\
&\leq \sum_m \overline{S}_m g(\overline{\mathbf{q}}_{mi^*_m})E[\sum_{\tau=0}^{z_n-1}A_m(t_n+\tau)] + E[z_n^2]\sum_m A_{\max}^2\overline{S}_m^2 + A_{\max}\overline{S}_m N_{\max}/C \\
&\overset{(c)}{\leq} E[z_n]\sum_m \overline{S}_m g(\overline{\mathbf{q}}_{mi^*_m})\lambda_m + A_{\max}^2\mathcal{K}_2\sum_m\overline{S}_m^2 + \frac{A_{\max}N_{\max}}{C}\mathcal{K}_2\sum_m\overline{S}_m \\
&\leq K_9 + E[z_n]\sum_m \overline{S}_m g(\overline{\mathbf{q}}_{m\hat{i}_m})\lambda_m, \qquad (3.19)
\end{aligned}
$$

where $i^*_m(t) = \underset{i\in\{1,2,,,L\}}{\arg\min}\,\mathbf{Q}_{mi}(t)$, $i^*_m = i^*_m(t_n)$, $\hat{i}_m(t) = \underset{i\in\{1,2,,,L\}}{\arg\min}\,\overline{\mathbf{Q}}_{mi}(t)$, $\hat{i}_m = \hat{i}_m(t_n)$ and $K_9 = A_{\max}^2\mathcal{K}_2\sum_m\overline{S}_m^2 + \frac{A_{\max}N_{\max}}{C}\mathcal{K}_2\sum_m\overline{S}_m + \mathcal{K}_1\sum_m\overline{S}_m\log(\widetilde{C})\lambda_m$. The first equality follows from the definition of $\mathbf{A}_{mi}$ in the routing algorithm in (3.1). Since $\overline{\mathbf{Q}}_{mi^*_m(t_n+\tau)}(t_n+\tau) \leq \overline{\mathbf{Q}}_{mi^*_m(t_n)}(t_n+\tau) \leq \overline{\mathbf{Q}}_{mi^*_m}(t_n) + \tau\overline{S}_m A_{\max} + \tau N_{\max}/C$ because the load at each queue cannot increase by more than $A_{\max}\overline{S}_m + N_{\max}/C$ 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 and Lemma 3.1. 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 3.2 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 (3.19).

Now we will bound the second term in (3.18). 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:

$$\widehat{\mathcal{D}}^{(j)}_{mi}(t) = \begin{cases} \overline{\mathcal{D}}^{(j)}_{mi}(t) & \text{if } j^{\text{th}} \text{ job in } \widetilde{N}^{(i)}_m(t_n) \text{ was scheduled} \\ 1 & \text{if the } j^{\text{th}} \text{ job was unused.} \end{cases} \tag{3.20}$$

$$\widehat{\mathbf{D}}_{mi}(t) = \sum_{j=1}^{\widetilde{N}^{(i)}_m(t_n)} \widehat{\mathcal{D}}^{(j)}_{mi}(t). \tag{3.21}$$

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

$$\begin{aligned}
& g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau) \\
&= \left(g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t_n+\tau)<N_{\max}} \\
&\quad + \left(g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\right)\mathbb{I}_{\mathbf{Q}_{mi}(t_n+\tau)\geq N_{\max}} \\
&\overset{(a)}{\geq} \Big(g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\Big(\widehat{\mathbf{D}}_{mi}(t_n+\tau) - N_{\max}\Big)\Big)\mathbb{I}_{\mathbf{Q}_{mi}(t_n+\tau)<N_{\max}} \\
&\quad + \Big(g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\widehat{\mathbf{D}}_{mi}(t_n+\tau)\Big)\mathbb{I}_{\mathbf{Q}_{mi}(t_n+\tau)\geq N_{\max}} \\
&\overset{(b)}{\geq} g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\widehat{\mathbf{D}}_{mi}(t_n+\tau) - N_{\max}\, g(\widetilde{C}N_{\max}).
\end{aligned} \tag{3.22}$$

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 3.2 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 (3.10) and (3.11). Then, using $\widehat{\mathbf{D}}_{mi}(t_n+\tau) \leq N_{\max}/C$ from (3.11) in (3.22), we get

$$\begin{aligned}
& g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau) \\
\geq & g(\overline{\mathbf{Q}}_{mi}(t_n))\widehat{\mathbf{D}}_{mi}(t_n+\tau) - K_{10}\tau - N_{\max}g(\widetilde{C}N_{\max}),
\end{aligned}$$

where $K_{10} = (N_{\max}/C)((A_{\max}\overline{S}_{\max} + N_{\max}/C)$. Then, using Lemma 3.1, the second term in (3.18) can be bounded as follows:

$$\begin{aligned}
& E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_{i,m} g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\right] \\
\geq & E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\sum_{i,m} g(\overline{\mathbf{Q}}_{mi}(t_n))\widehat{\mathbf{D}}_{mi}(t_n+\tau)\right] - K_{11} \\
= & \sum_{i,m} g(\overline{\mathbf{q}}_{mi}) E_{\mathbf{q}}\left[\sum_{\tau=0}^{z_n-1}\widehat{\mathbf{D}}_{mi}(t_n+\tau)\right] - K_{11}, \qquad (3.23)
\end{aligned}$$

where $K_{11} = LM(K_{10}\mathcal{K}_2 + N_{\max}g(N_{\max})\mathcal{K}_1)$. We will now use the generalized Wald's Identity (Lemma 3.4), 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 3.3.

From definition of $\widehat{\mathbf{D}}_{mi}(t_n+\tau)$, from (3.8) and (3.9), $|\widehat{\mathbf{D}}_{mi}(t_n+\tau)| \leq N_{\max}/C$. 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 3.1.

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

Proof. Consider the departures due to each job, $\widehat{\mathcal{D}}_{mi}^{(j)}(t)$ as defined in (3.20). 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 3.3, 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 3.3. We will now prove this more formally.

The event $\{z_n \geq \tau\}$ is a union of many (but finite) disjoint events $\{E_\alpha : \alpha = 1 \dots \mathcal{A}\}$. Each of these events E_α is of the form $\{(\mathbf{q}(t_n), \mathbf{A}(t_n), \mathbf{D}(t_n)), (\mathbf{q}(t_n+1), \mathbf{A}(t_n+1), \mathbf{D}(t_n+1)), \dots (\mathbf{q}(t_n+\tau-1), \mathbf{A}(t_n+\tau-1), \mathbf{D}(t_n+\tau-1))\}$. 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

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

The last inequality follows from Lemma 3.3 and definition of $\widehat{\mathcal{D}}_{mi}^{(j)}(t)$ in terms of $\overline{\mathcal{D}}_{mi}^{(j)}(t)$ (3.20). 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 3.3 and (3.20), we have $E_{\mathbf{q}}\left[\widehat{\mathcal{D}}_{mi}^{(j)}(t_n+\tau)\right] = 1$. Summing over j, from (3.21), 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 the generalized Wald's identity (Lemma 3.4) in (3.23), we have

$$\begin{aligned}E_{\mathbf{q}}&\left[\sum_{\tau=0}^{z_n-1}\sum_{i,m}g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))\overline{\mathbf{D}}_{mi}(t_n+\tau)\right]\\&\geq\sum_{i,m}g(\overline{\mathbf{q}}_{mi})E_{\mathbf{q}}\left[z_n\right]\widetilde{N}_m^{(i)}(t_n)-K_{11}. \quad (3.24)\end{aligned}$$

The key idea is to note that the expected departures of workload for each scheduled job is 1 from Lemma (3.3). We use this, along with the generalized Wald's theorem, to bound the departures similar to the case of geometric job sizes.

Since $(\lambda,\overline{S})\in\widehat{\mathcal{C}}$, there exists $\{\lambda^i\}_i$ such that $\lambda=\sum_i\lambda^i$ and $\lambda^i\circ\overline{S}\in$ *int*(Conv($\mathcal{N}_i$)) for all i . Then, there exists an $\epsilon>0$ such that $(\lambda^i+\epsilon)\circ\overline{S}\in$ Conv($\mathcal{N}_i$) for all i. From Lemma 3.2, 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 [38] [39]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 4 since $\widetilde{N}_m^{(i)}(t_n)$ is chosen according to MaxWeight policy. The last inequality again follows from Lemma 3.2. Substituting this in (3.24) and from (3.19) and (3.18), we get

$$E[V(\widetilde{\mathbf{X}}(n+1))-V(\widetilde{\mathbf{X}}(n))|\widetilde{\mathbf{Q}}(n)=\mathbf{q},\widetilde{\mathbf{Y}}(n)]$$

$$\begin{aligned}
&\le K_{12}+E_{\mathbf{q}}[z_n]\sum_m\left(g(\overline{\mathbf{q}}_{m\hat{i}_m})\lambda_m\overline{S}_m-\sum_i g(\overline{\mathbf{q}}_{mi})(\lambda_m^i+\epsilon)\overline{S}_m\right)\\
&\overset{(a)}{\le} K_{12}-\epsilon\overline{S}_{\min}E_{\mathbf{q}}[z_n]\sum_i\sum_m g(\overline{\mathbf{q}}_{mi})\\
&\le K_{13}-\epsilon\overline{S}_{\min}\log(1+G^{-1}(V(\overline{\mathbf{q}}))),
\end{aligned}$$

where $K_{12} = K_8\mathcal{K}_1 + K_9 + K_{11} + \log(\widetilde{C})\sum_m(\lambda_m + L\epsilon)\overline{S}_m$ and $K_{13} = K_{12} + \epsilon\overline{S}_{\min}\mathcal{K}_1$. Inequality (a) follows from $\lambda = \sum_i \lambda^i$ and $\overline{\mathbf{q}}_{m\hat{i}_m} \le \overline{\mathbf{q}}_{mi}$. The last inequality follows from Lemma 3.7 and since $z_n \ge 1$.

If the job 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 3.2, Foster-Lyapunov theorem can be used to show that the sampled Markov chain $\widetilde{\mathbf{X}}(n)$ is positive recurrent. We need the bounded job 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 job sizes are not bounded in general, we will use Lemma 3.5 to show stability of Algorithm 4 for the random process $U(n)$. From Lemma 3.6, we have

$$\begin{aligned}
&E[U(\widetilde{\mathbf{X}}(n+1)) - U(\widetilde{\mathbf{X}}(n))|\widetilde{\mathbf{X}}(n) = \mathbf{x} = (\mathbf{q},\mathbf{y})]\\
&\le E\left[\left.\frac{V(\widetilde{\mathbf{X}}(n+1)) - V(\widetilde{\mathbf{X}}(n))}{\log(1+U(\widetilde{\mathbf{X}}(n))}\right|\widetilde{\mathbf{X}}(n) = \mathbf{x} = (\mathbf{q},\mathbf{y})\right]\\
&\le \frac{K_{13}}{\log(1+U(\overline{\mathbf{q}}))} - \epsilon\overline{S}_{\min}\mathcal{K}_1 \quad \le -\frac{\epsilon\overline{S}_{\min}\mathcal{K}_1}{2}
\end{aligned}$$

whenever $U(\overline{\mathbf{q}}) > e^{(2K_{13}/\epsilon\overline{S}_{\min}\mathcal{K}_1)}$. Thus, $U(n)$ satisfies condition C1 of Lemma 3.5 for the filtration generated by the $\{\widetilde{\mathbf{X}}(n)\}$. From Lemma 3.6, Lemma 3.7 and (3.16), we have

$$\begin{aligned}
&(U(t_n+\tau+1) - U(t_n+\tau))\\
&\le \frac{[V(t_n+\tau+1) - V(t_n+\tau)]}{\log(1+G^{-1}(V(\overline{\mathbf{Q}}(t_n+\tau))))}\\
&\le \frac{K_8 + A_{\max}\sum_{m,i}\overline{S}_m g(\overline{\mathbf{Q}}_{mi}(t_n+\tau))}{\log(1+G^{-1}(V(\overline{\mathbf{Q}}(t_n+\tau))))}
\end{aligned}$$

$$
\begin{aligned}
&\leq \frac{K_8}{\log(1 + G^{-1}(V(\overline{\mathbf{Q}}(t_n+\tau))))} + \frac{A_{\max}\overline{S}_{\max}}{LM} \\
&\overset{(a)}{\leq} K_{14} \quad \text{if } U(t_n+\tau) > 0,
\end{aligned}
$$

where $K_{14} = \frac{K_8}{\log(2)} + \frac{A_{\max}\overline{S}_{\max}}{LM}$. 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 (3.16), we have $(U(t_n+\tau+1) - U(t_n+\tau)) \leq K_{15} \triangleq G^{-1}(K_4)$. Thus, we have

$$
(U(t_n+\tau+1) - U(t_n+\tau)) \leq K_{16},
$$

where $K_{16} = \max\{K_{14}, K_{15}\}$. Similarly, from (3.17) it can be shown that

$$
(U(t_n+\tau) - U(t_n+\tau+1)) \leq K_{18},
$$

where $K_{18} = \max\{K_{17}, K_{15}\}$ and $K_{17} = \frac{K_8}{\log(2)} + \frac{N_{\max}}{LM}$. Setting $K_{19} = \max\{K_{16}, K_{18}\}$, we have

$$
\begin{aligned}
(|U(t_n+\tau) - U(t_n+\tau+1)|) &\leq K_{19} \\
(|U(t_n+\tau) - U(t_n+\tau+1)|| \, \mathbf{X}(t_n)) &\leq K_{19} \\
\left(|U(\widetilde{\mathbf{X}}(n+1)) - U(\widetilde{\mathbf{X}}(n))| \middle| \widetilde{\mathbf{X}}(n)\right) &\leq K_{19} z_n, \quad \leq K_{19}\overline{z}_n
\end{aligned}
$$

where $\overline{z}_n$ is the coupled random variable constructed in the proof of Lemma 3.1. Since $\overline{z}_n$ is a geometric random variable by construction, it satisfies condition C1 in Lemma 3.5. 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

$$
G\left(\frac{\sum_{m,i} \overline{\mathbf{Q}}_{mi}(t_n)}{LM}\right) \leq \frac{\sum_{m,i} G\left(\overline{\mathbf{Q}}_{mi}(t_n)\right)}{LM} \leq V(\overline{\mathbf{Q}}(t_n)). \tag{3.25}
$$

Then, from Lemma 3.2 and (b), we get

$$
\sum_{m,i} \mathbf{Q}_{mi}(t_n) \leq \sum_{m,i} \overline{\mathbf{Q}}_{mi}(t_n) \leq LMU(\overline{\mathbf{Q}}(t_n)) \leq \frac{LM}{\theta^*} e^{\theta^* U(\widetilde{\mathbf{X}}(n))}.
$$

Thus, we have $\lim_{n\to\infty}\sum_m\sum_i E[\mathbf{Q}_{mi}(t_n)] \le \frac{LM}{\theta^*}\mathcal{K}_4$.

For any $t > 0$, if t_{n+1} is the next refresh time after t, from (3.11) we have

$$\begin{aligned}
\mathbf{Q}_{mi}(t) \le \overline{\mathbf{Q}}_{mi}(t) &\le \overline{\mathbf{Q}}_{mi}(t_{n+1}) + z_n\frac{N_{\max}}{C} \\
&\le \widetilde{C}\mathbf{Q}_{mi}(t_{n+1}) + z_n\frac{N_{\max}}{C} \\
\sum_m\sum_i E[\mathbf{Q}_{mi}(t)] &\le \sum_m\sum_i E[\widetilde{C}\left(\mathbf{Q}_{mi}(t_{n+1}) + z_nN_{\max}/C\right)].
\end{aligned}$$

As $t \to \infty$, we get

$$\begin{aligned}
\limsup_{t\to\infty}\sum_{m,i}E[\mathbf{Q}_{mi}(t)] &\le \limsup_{n\to\infty}\sum_{m,i}E\left[\widetilde{C}\mathbf{Q}_{mi}(t_n)+z_n\frac{N_{\max}}{C}\right] \\
&\le \frac{LM}{\theta^*}\widetilde{C}\mathcal{K}_4 + \frac{\mathcal{K}_1LMN_{\max}}{C}.
\end{aligned}$$

□

A centralized queuing architecture was considered in Chapter 2. 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 4 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 Theorem 3.1 and so we skip it.

3.4 Local Refresh Times

According to Algorithm 4, each server performs MaxWeight scheduling only at refresh times. At other times, it uses the same schedule as before. Since a refresh time happens only when none of the servers are serving any jobs, refresh times could be pretty infrequent in practice. Moreover, refresh times become rarer as the number of servers increases. This may lead to large queue lengths and delays in practice.

Another disadvantage with the use of (*global*) refresh times is that there needs to be some form of coordination between the servers to know if a time slot is a refresh time or not. So, we propose the use of *local refresh times* instead. 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 5.* Routing is done according to the Join the shortest Queue algorithm as before. For scheduling, each server chooses a MaxWeight schedule only at local refresh times. Between the local refresh times, a server maintains the same configuration. It is not clear if this is throughput optimal or not. Each server may have multiple local refresh times between two (global) refresh times and the schedule changes at these refresh times. So the proof approach from the previous section is not applicable here. Investigating throughput optimality of this algorithm is an open problem.

We propose Algorithm 6 with a simpler routing algorithm which is more tractable analytically. In traditional load balancing problem without any scheduling (i.e., when the jobs and servers are one dimensional), random routing is known to be throughput optimal when all the servers are identical. In practice, many data centers have identical servers. In such a case, the following proposition presents throughput optimality of Algorithm 6.

Algorithm 6 Random Routing and MaxWeight Scheduling at Local Refresh times

---

1. *Routing Algorithm (JSQ Routing)*: Each job that arrives into the system is routed to one of the servers uniformly at random.

2. *Scheduling Algorithm (MaxWeight Scheduling) for each server i*: Let $\widetilde{N}_m^{(i)}(t)$ denote a configuration chosen in each time slot. If the time slot is a *local* refresh time, $\widetilde{N}_m^{(i)}(t)$ is chosen according to the MaxWeight policy, i.e.,

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

   If it is not a refresh time, $\widetilde{N}_m^{(i)}(t) = \widetilde{N}_m^{(i)}(t-1)$.

Proposition 3.2. *Assume that all the servers are identical and the job size distribution satisfies Assumption 3.1. Then, any job load vector that satisfies $(\lambda, \overline{S}) \in int(\widehat{\mathcal{C}})$ is supportable under random routing and MaxWeight scheduling at local refresh times as described in Algorithm 6 with $g(q) = \log(1+q)$.*

We skip the proof here because it is very similar to that of Theorem 3.1. Since routing is random, each server is independent of other servers in the system. So, one can show that each server is stable under the job load vector $(\lambda/L, \overline{S})$ using the Lyapunov function in (3.13). This then implies that the whole system is stable.

In the next section, we study the performance of these algorithms by simulations.

3.5 Simulations

In this section, we use simulations to compare the performance of the Algorithms presented so far. We use the same simulation set up as in Chapter 2 with 100 identical servers, three types of jobs and three maximal VM configurations for each server viz., $(2, 0, 0)$, $(1, 0, 1)$, and $(0, 1, 1)$. 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 the same bounded distribution that was considered in Chapter 2, which models the high variability in jobs sizes. 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.

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 $(\rho\frac{\lambda_i}{130.5}, 100)$.

All the plots in this section compare the mean delay of the jobs under various algorithms. The parameter ρ is varied to simulate different traffic intensities. Each simulation was run for one million time slots.

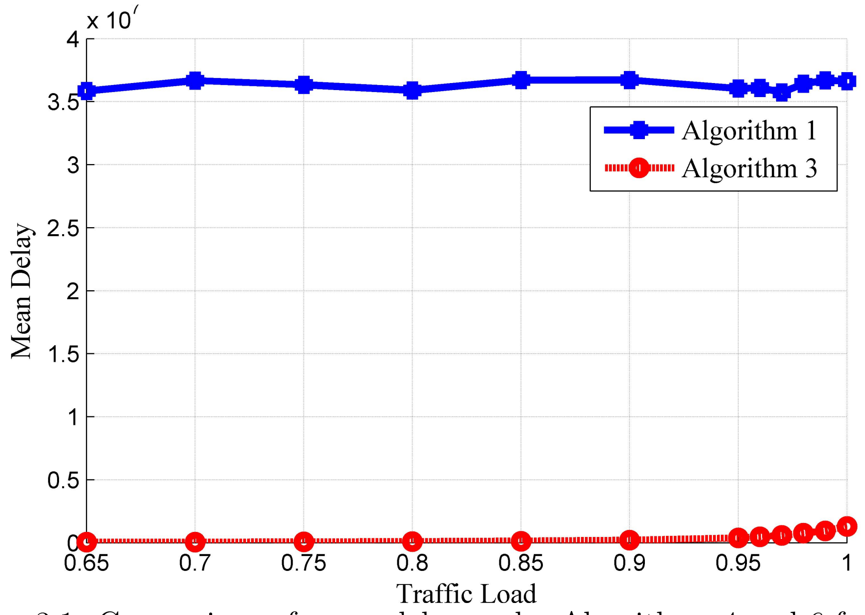


Figure 3.1: Comparison of mean delay under Algorithms 4 and 6 for load vector $\lambda^{(1)}$ and job size distribution A

3.5.1 Local vs. Global Refresh Times

In this subsection, we compare the performance of Algorithms 4 and 6 which are proven to be throughput optimal. Figure 3.1 shows the mean delay of the jobs under the job size distribution A and load vector $\lambda^{(1)}$.

Algorithm 4 has poor performance because of the amount of time between two refresh times. However, using Algorithm 6 with local refresh times gives much better performance (in the case when servers are identical). Even though both algorithms are throughput optimal, Algorithm 6 has better performance in practice.

3.5.2 Heuristics

In this section, we study the performance of some heuristic algorithms. We have seen in the previous subsection that the idea of using local refresh times is good. Since JSQ routing provides better load balancing than random routing, a natural algorithm to study is one that does JSQ routing and updates schedules at local refresh times. This leads us to Algorithm 5. Since we don't know if Algorithm 5 is throughput optimal, we study its performance using simulations.

We also consider another heuristic algorithm motivated by Algorithm 4

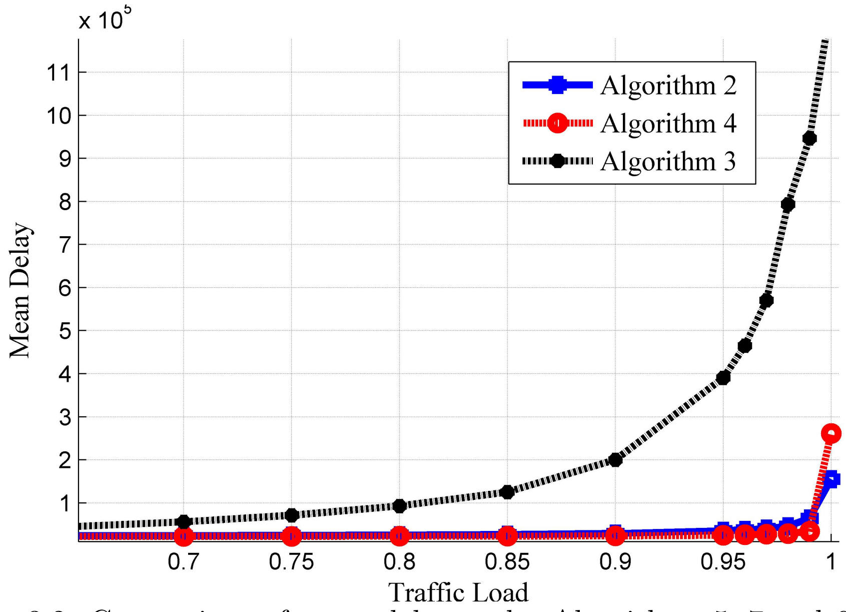


Figure 3.2: Comparison of mean delay under Algorithms 5, 7 and 6 for load vector $\lambda^{(1)}$ and job size distribution A

and Algorithm 3 in Chapter 2 as follows. Routing is done according to the join the shortest queue algorithm. At refresh times, a MaxWeight schedule is chosen at each server. At all other times, each server tries to choose a MaxWeight schedule myopically. 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 without disturbing the jobs in service. This algorithm tries to emulate a MaxWeight schedule in every time slot by greedily adding new jobs with higher priority to long queues. We call this *Algorithm 7.*

This algorithm has the advantage that, at refresh times, an exact MaxWeight schedule is chosen automatically, so the servers need not keep track of the refresh times. It is not clear if this algorithm is throughput optimal. The proof in section 3.3 is not applicable here because one cannot use Wald's identity to bound the drift. This algorithm is an extension of Algorithm 3 Chapter 2 when the super time slots are taken to be infinite. However as stated in Chapter 2, Algorithm 3 is almost throughput optimal only when the super time slot is finite. Investigating throughput optimality of 7 under appropriate assumptions is an open problem.

Figures 3.2, 3.3 and 3.4 compare the mean delay of the jobs under Algorithms 5, 7 and 6 with the three job size distributions using the load vector $\lambda^{(1)}$. Figure 3.5 shows the case when the load vector $\lambda^{(2)}$ is used.

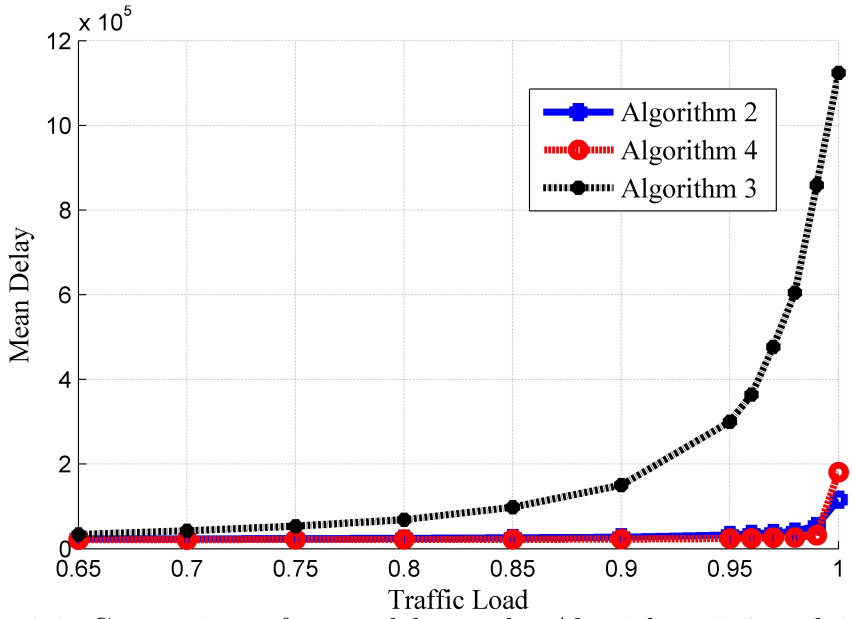


Figure 3.3: Comparison of mean delay under Algorithms 5, 7 and 6 for load vector $\lambda^{(1)}$ and job size distribution B

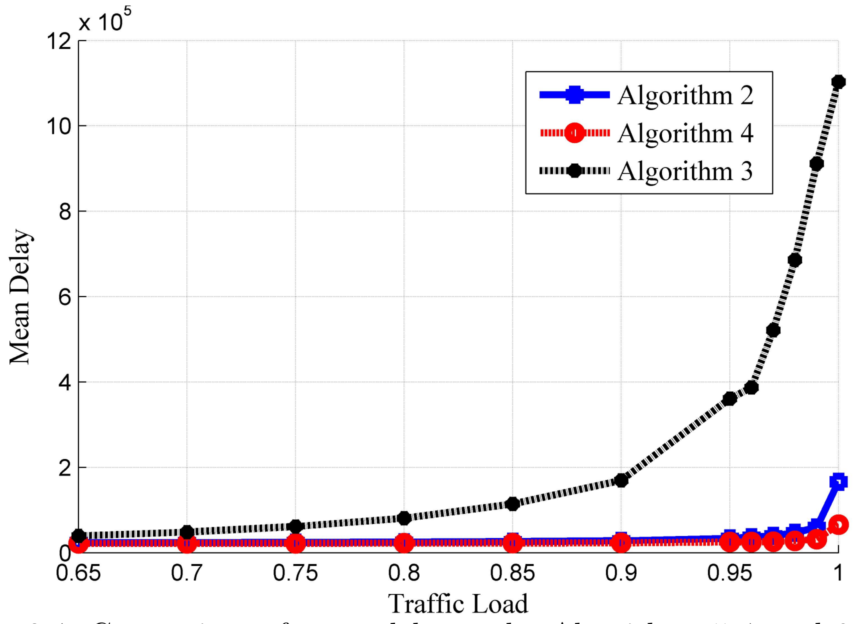


Figure 3.4: Comparison of mean delay under Algorithms 5, 7 and 6 for load vector $\lambda^{(1)}$ and job size distribution C

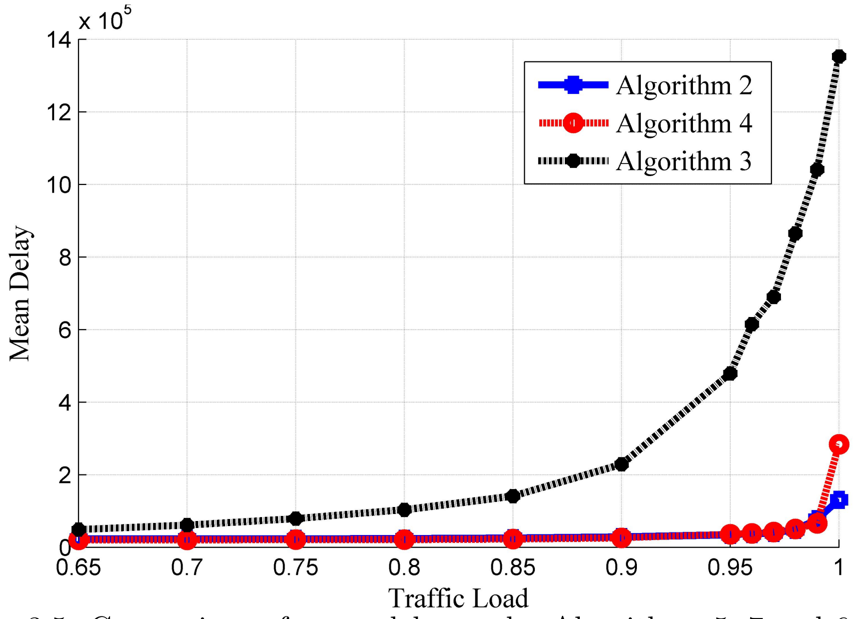


Figure 3.5: Comparison of mean delay under Algorithms 5, 7 and 6 for load vector $\lambda^{(2)}$ and job size distribution A

The simulations indicate that both Algorithms 5 and 7 have better delay performance than Algorithm 6 for all job size distributions and both the load vectors. The performance improvement is more significant at higher traffic intensities. In the cases studied, simulations suggest that Algorithms 5 and 7 are also throughput optimal. Since we do not know if this always true, it is an open question for future research to characterize the throughput region of Algorithms 5 and 7.

In sections 3.2, it was noted that a wide class of weight functions can be used for MaxWeight schedule in the case of geometric job sizes. However, the proof in section 3.3 required a $\log(1+q)$ weight function for general job size distributions. So, we now study the delay performance under linear and log weight functions. Figure 3.6 shows the delay of Algorithms 5 and 7 under the weight functions, q and $\log(1+q)$. Job size distribution A was used with the load vector $\lambda^{(1)}$. It can be seen that there is no considerable difference in performance between the two weight functions. It is an open question whether Algorithms 4 and 6 are throughput optimal under more weight functions.

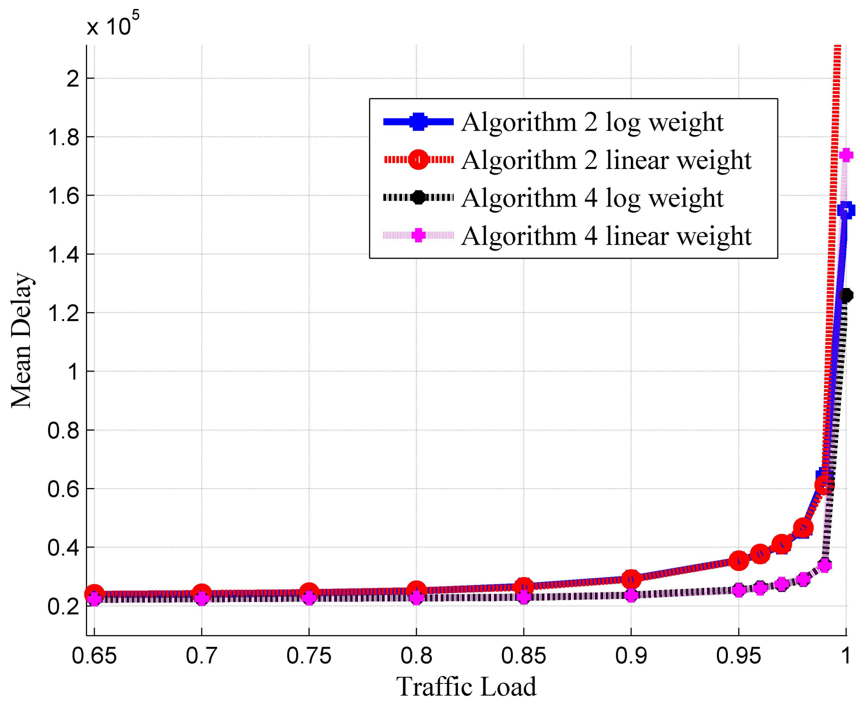


Figure 3.6: Comparison of mean delay under Algorithms 5 and 7 for load vector $\lambda^{(1)}$ and job size distribution A with log and linear weights

3.6 Conclusion

In this chapter, we studied various algorithms for the problem of routing and nonpreemptive scheduling jobs with variable, unknown and unbounded sizes in a cloud computing data center. The key idea in these algorithms is to choose a MaxWeight schedule at either local or global refresh times. We have presented two algorithms that are throughput optimal. The key idea in the proof is to show that the refresh times occur often enough and then use this to show that the drift of a Lyapunov function is negative. We then presented two heuristic algorithms and studied their performance using simulations.

CHAPTER 4

LIMITED PREEMPTION

In the last two chapters, we have assumed that preemption is absolutely not allowed. For the algorithm studied in chapter 2, we were only able to characterize an inner bound of the capacity region. Due to the wasted time with in a super time slot, these algorithms did not achieve 100% of the capacity region. In Chapter 3, when global refresh times were used even though we were able to prove throughput optimality, poor performance of the algorithm makes it impractical to use. When local refresh times were used in conjunction with random routing, we were able to prove throughput optimality, performance was not satisfactory. However, using JSQ routing gave much better performance.

In this chapter, we assume that jobs are allowed to be preempted every T time slots. This is a reasonable assumption because in practice, when a job lasts for a long time, it is preempted to accommodate other potentially shorter jobs. Every T time slots are grouped into a super time slot. We assume that at the beginning of every super time slot, one is allowed to interrupt/reshuffle jobs. Therefore, a MaxWeight algorithm can be chosen at the beginning of every super time slot. Between super time slots, we will use myopic MaxWeight scheduling. We will primarily focus on JSQ routing in this chapter. The results can be extended to power-of-two choices routing algorithm as well.

In this chapter, we consider the case when job sizes are unknown, similar to the results in Chapter 3. In the next chapter, we will assume that job sizes are known, similar to the case studied in Chapter 2 and study delay optimality.

4.1 Unknown Job Sizes

As in Chapter 3, in this section we will assume that the job sizes are not known either at arrival or during the service. Since reshuffling of jobs is allowed every super time slot, we do not need the notion of refresh times any more. The details of the algorithm are presented in Algorithm 8.

Algorithm 8 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 $i^*_m(t) = \underset{i \in \{1,2,,,L\}}{\arg\min}\, \mathbf{Q}_{mi}(t)$. Therefore, the arrivals to $\mathbf{Q}_{mi}$ in time slot t are given by

$$\mathbf{A}_{mi}(t) = \begin{cases} A_m(t) & \text{if } i = i^*_m(t) \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

2. *Scheduling Algorithm (Myopic MaxWeight Scheduling) for each server i*: T time slots are grouped into a super time slot. A MaxWeight configuration is chosen at the beginning of a super time slot. So, for $t = nT$, configuration $\widetilde{N}^{(i)}(t)$ is chosen according to

$$\widetilde{N}^{(i)}(t) \in \underset{N \in \mathcal{N}_i}{\arg\max} \sum_m g(\mathbf{Q}_{mi}(t)) N_m. \tag{4.2}$$

For all other t, at the beginning of the time slot, a new configuration is chosen as follows:

$$\widetilde{N}^{(i)}(t) \in \underset{N : N + N^{(i)}(t^-) \in \mathcal{N}_i}{\arg\max} \sum_m g(\mathbf{Q}_{mi}(t)) N_m,$$

where $N^{(i)}(t^-)$ is the configuration of jobs at server i that are still in service at the end of the previous time slot. 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, $N^{(i)}(t) = \widetilde{N}^{(i)}(t)$.

To prove throughput optimality, Assumption 3.1 in Chapter 3 can now be relaxed. Before we state the new assumption, we need the following definition.

Definition 4.1. For a random variable S, $W(s) = E[S - s|S > s]$ is called its *Mean Residual Life* function or Mean excess function.

If S is a random variable denoting the job size distribution, then the mean residual life is the expected remaining amount of service of that job given that it has been served for s time slots.

Assumption 4.1. The mean residual function of the job size distribution of type m jobs is upper bounded by a constant $1/C_m$.

Example 4.1. A geometric random variable has a constant mean residual function and so satisfies Assumption 4.1. Any distribution with finite support also satisfies Assumption 4.1.

Example 4.2. Zeta distribution, a heavy-tailed distribution and a discrete counterpart of Pareto distribution, is given by $P(S = k) = k^{-a}/\zeta(a)$ where $\zeta(a)$ is the Reimann-Zeta function and $a > 1$. Then, the mean residual function is

$$\begin{aligned} W(s) &= E[S - s|S > s] \\ &= \frac{\sum_{s'=s+1}^{\infty} P(S \geq s')}{P(S \geq s+1)} \\ &= \frac{\sum_{s'=s+1}^{\infty} \sum_{s''=s'}^{\infty} (s'')^{-a}}{\sum_{s'=s+1}^{\infty} (s')^{-a}}. \end{aligned}$$

Note that $\sum_{s''=s'}^{\infty} (s'')^{-a} > \int_{s'}^{\infty} x^{-a} dx = \frac{(s')^{1-a}}{a-1}$ since $a > 1$. So, we have

$$\begin{aligned} W(s) &> \frac{1}{a-1} \frac{\sum_{s'=s+1}^{\infty} (s')^{1-a}}{\sum_{s'=s+1}^{\infty} (s')^{-a}} \\ &\geq \frac{s+1}{a-1} \frac{\sum_{s'=s+1}^{\infty} (s')^{-a}}{\sum_{s'=s+1}^{\infty} (s')^{-a}} \\ &= \frac{s+1}{a-1}. \end{aligned}$$

Therefore, $W(s)$ cannot be upper bounded by a constant and so Zeta distribution does not satisfy Assumption 4.1.

We will now show that Algorithm 8 is throughput optimal with appropriately chosen $g(.)$ when the job size distributions satisfy Assumption 4.1.

The process $\mathbf{X}(t) = (\mathbf{Q}(t), \mathbf{Y}(t))$ is a Markov chain, where $\mathbf{Y}(t)$ is defined in the section 3.2 of Chapter 3. Let $W_m(l)$ be the mean residual life 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

$$\overline{\mathbf{D}}_{mi}(t) = \begin{cases} W_m(l) & \text{with prob } \widehat{p}_{ml} \\ W_m(l) - W_m(l+1) & \text{with prob } 1 - \widehat{p}_{ml}. \end{cases} \tag{4.3}$$

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.

Then from (4.3), the increase in backlog of workload due to 'departure' for each scheduled job can increase by at most $W_m(l+1)$, which is upper bounded by $1/C_m$ due to Assumption 4.1. There are at most $N_{\max}$ jobs of each type that are scheduled. The arrival in backlog queue is at most $A_{\max}\overline{S}_{\max}$. Then, defining $C = \min_m\{C_m\}$ we have

$$\begin{aligned} \overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t) &\le A_{\max}\overline{S}_{\max} + \frac{N_{\max}}{C_m} \\ &\le A_{\max}\overline{S}_{\max} + \frac{N_{\max}}{C}. \end{aligned} \tag{4.4}$$

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

$$\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t) \geq -\frac{N_{\max}}{C_m} \geq -\frac{N_{\max}}{C}. \tag{4.5}$$

Theorem 4.1. *Assume that the job arrivals satisfy $A_m(t) \leq A_{\max}$ for all m and t and that the job size distribution satisfies Assumption 4.1. Then, any job load vector that satisfies $(\lambda, \overline{S}) \in int(\widehat{\mathcal{C}})$ is supportable under JSQ routing and myopic myopic MaxWeight allocation as described in Algorithm 8 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 with one entry for each job that was partially served. Some of the partially served jobs are currently scheduled and some were interrupted at the beginning of a super time slot. Therefore, the size of $\mathbf{Y}_{mi}(t)$ is at most $\mathbf{Q}_{mi}(t)$ and $\mathbf{Y}^j_{mi}(t)$ is the amount of time the j^{th} partially served type-m job that is in service at server i has been served.

It is easy to see that $\mathbf{X}(t) = (\mathbf{Q}(t), \mathbf{Y}(t))$ is a Markov chain under Algorithm 8.

Obtain a new process, $\widetilde{\mathbf{X}}(n)$, by sampling the Markov chain $\mathbf{X}(t)$ every T time slots, i.e., $\widetilde{\mathbf{X}}(n) = \mathbf{X}(nT)$. Note that $\widetilde{\mathbf{X}}(n)$ is also a Markov chain.

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.

With slight abuse of notation, we will use $V(t)$ for $V(\overline{\mathbf{Q}}(t))$. Similarly, $V(n)$, $U(t)$ and $U(n)$. We will establish this result by showing that the Lyapunov function $U(n)$ satisfies both the conditions of Lemma 3.5. We will study the drift of $U(n)$ in terms of drift of $V(n)$ using Lemma 3.6. First consider the following one step drift of $V(t)$:

$$\begin{aligned}
&(V(t+1) - V(t)) \\
=&\sum_{m,i} \left(G\left(\overline{\mathbf{Q}}_{mi}(t+1)\right) - G\left(\overline{\mathbf{Q}}_{mi}(t)\right)\right) \\
\leq&\sum_{m,i} \left(\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t)\right) g(\overline{\mathbf{Q}}_{mi}(t+1)) \qquad (4.6)\\
=&\sum_{m,i} \left(\overline{\mathbf{Q}}_{mi}(t+1) - \overline{\mathbf{Q}}_{mi}(t)\right) \left(g(\overline{\mathbf{Q}}_{mi}(t+1)) - g(\overline{\mathbf{Q}}_{mi}(t))\right)
\end{aligned}$$

$$+\sum_{m,i}\left(\overline{\mathbf{A}}_{mi}(t)-\overline{\mathbf{D}}_{mi}(t)\right)g(\overline{\mathbf{Q}}_{mi}(t)), \tag{4.7}$$

where (4.6) follows from the convexity of $G(.)$. To bound the first term in (4.7), 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

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

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

So the first term in (4.7) can be bounded as

$$\begin{aligned}
&\sum_{m,i}\left(\overline{\mathbf{Q}}_{mi}(t+1)-\overline{\mathbf{Q}}_{mi}(t)\right)\left(g(\overline{\mathbf{Q}}_{mi}(t+1))-g(\overline{\mathbf{Q}}_{mi}(t))\right)\\
\leq&\sum_{m,i}\left|\overline{\mathbf{Q}}_{mi}(t+1)-\overline{\mathbf{Q}}_{mi}(t)\right|\left|g(\overline{\mathbf{Q}}_{mi}(t+1))-g(\overline{\mathbf{Q}}_{mi}(t))\right|\\
\leq&\sum_{m,i}\left|\left(\overline{\mathbf{Q}}_{mi}(t+1)-\overline{\mathbf{Q}}_{mi}(t)\right)\right|^2\qquad\leq K_8,
\end{aligned}$$

where $K_8=LM(A_{\max}\overline{S}_{\max}+\frac{N_{\max}}{C})^2$ as defined in Chapter 3. The last inequality follows from (4.4) and (4.5). Thus, we have

$$V(t+1)-V(t)\leq K_8+\sum_{m,i}(\overline{\mathbf{A}}_{mi}(t)-\overline{\mathbf{D}}_{mi}(t))g(\overline{\mathbf{Q}}_{mi}(t)). \tag{4.8}$$

Similarly, it can be shown that

$$V(t)-V(t+1)\leq K_8+\sum_{m,i}\left(\overline{\mathbf{D}}_{mi}(t)-\overline{\mathbf{A}}_{mi}(t)\right)g(\overline{\mathbf{Q}}_{mi}(t+1)). \tag{4.9}$$

Again, we use $E_{\mathbf{q}}[(.)]$ to denote $E[(.)|\mathbf{Q}(nT)=\mathbf{q},\mathbf{Y}(nT)]$. Now using (3.16) in the drift of the sampled system, we get

$$\begin{aligned}
&E[V(\widetilde{\mathbf{X}}(n+1))-V(\widetilde{\mathbf{X}}(n))|\widetilde{\mathbf{Q}}(n)=\mathbf{q},\widetilde{\mathbf{Y}}(n)]\\
=&E[V((n+1)T)-V(nT)|\mathbf{Q}(nT)=\mathbf{q},\mathbf{Y}(nT)]
\end{aligned}$$

$$
\begin{aligned}
=&E_{\mathbf{q}}\left[\sum_{\tau=0}^{T-1} V(nT+\tau+1)-V(nT+\tau)\right]\\
\leq&E_{\mathbf{q}}\left[\sum_{\tau=0}^{T-1}\left(\sum_{m,i}\big(g(\overline{\mathbf{Q}}_{mi}(nT+\tau))\overline{\mathbf{A}}_{mi}(nT+\tau)\right.\right.\\
&\left.\left.-g(\overline{\mathbf{Q}}_{mi}(nT+\tau))\overline{\mathbf{D}}_{mi}(nT+\tau)\big)\right)+K_8\right]. \qquad (4.10)
\end{aligned}
$$

The last term above is bounded by K_8T. We will now bound the first term in (4.10).

$$
\begin{aligned}
&E_{\mathbf{q}}\left[\sum_{\tau=0}^{T-1}\sum_m\sum_i g(\overline{\mathbf{Q}}_{mi}(nT+\tau))\overline{\mathbf{A}}_{mi}(nT+\tau)\right]\\
=&E_{\mathbf{q}}[\sum_{\tau=0}^{T-1}\sum_m g(\overline{\mathbf{Q}}_{mi^*_m(nT+\tau)}(nT+\tau))A_m(nT+\tau)\overline{S}_m]\\
\overset{(a)}{\leq}&E_{\mathbf{q}}[\sum_{\tau=0}^{T-1}\sum_m g(\overline{\mathbf{Q}}_{mi^*_m}(nT)+\tau A_{\max}\overline{S}_m+\tau N_{\max}/C)A_m(nT+\tau)\overline{S}_m]\\
\overset{(b)}{\leq}&E_{\mathbf{q}}[\sum_{\tau=0}^{T-1}\sum_m g(\overline{\mathbf{Q}}_{mi^*_m}(nT))A_m(nT+\tau)\overline{S}_m+\tau A_{\max}^2\overline{S}_m^2+\tau\frac{N_{\max}}{C}A_{\max}\overline{S}_m]\\
=&\sum_m \overline{S}_m g(\overline{\mathbf{q}}_{mi^*_m})E[\sum_{\tau=0}^{T-1}A_m(nT+\tau)]+K_{20}\\
\leq&K_{21}+T\sum_m \overline{S}_m g(\overline{\mathbf{q}}_{m\hat{i}_m})\lambda_m, \qquad (4.11)
\end{aligned}
$$

where $i^*_m(t)=\underset{i\in\{1,2,,,L\}}{\arg\min}\,\mathbf{Q}_{mi}(t)$, $i^*_m=i^*_m(nT)$, $\hat{i}_m(t)=\underset{i\in\{1,2,,,L\}}{\arg\min}\,\overline{\mathbf{Q}}_{mi}(t)$, $\hat{i}_m=\hat{i}_m(nT)$ and $K_{20}=\sum_m(A_{\max}^2\overline{S}_m^2+N_{\max}A_{\max}/C)T(T-1)/2$ $K_{21}=K_{20}+T\sum_m\overline{S}_m\log(\widetilde{C})\lambda_m$. The first equality follows from the definition of $\mathbf{A}_{mi}$ in the routing algorithm in (4.1). Since $\overline{\mathbf{Q}}_{mi^*_m(nT+\tau)}(nT+\tau)\leq\overline{\mathbf{Q}}_{mi^*_m(nT)}(nT+\tau)\leq\overline{\mathbf{Q}}_{mi^*_m}(nT)+\overline{S}_mA_{\max}\tau+\tau N_{\max}/C$ because the load at each queue cannot increase by more than $A_{\max}\overline{S}_m+N_{\max}/C$ in each time slot from (4.4), we get (a). Inequality (b) follows from concavity of $g(.)$ and $g'(.)\leq 1$. Note that Lemma 3.2 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 (4.11).

Now consider the second term in (4.10).

$$
E_{\mathbf{q}}\left[\sum_{\tau=0}^{T-1} g(\overline{\mathbf{Q}}_{mi}(nT+\tau))\overline{\mathbf{D}}_{mi}(nT+\tau)\right]
$$

$$
\begin{aligned}
=&\sum_{\tau=0}^{T-1} E_{\mathbf{q}}\left[g(\overline{\mathbf{Q}}_{mi}(nT+\tau))\overline{\mathbf{D}}_{mi}(nT+\tau)\right] \\
=&\sum_{\tau=0}^{T-1} E_{\mathbf{q}}\left[E\left[\sum_{i,m} g(\overline{\mathbf{Q}}_{mi}(nT+\tau))\overline{\mathbf{D}}_{mi}(nT+\tau)\right.\right. \\
&\qquad\left.\left.\middle|\mathbf{Y}(nT+\tau),\mathbf{Q}(nT+\tau),N_m^{(i)}(nT+\tau)\right]\right] \\
=&E_{\mathbf{q}}\left[\sum_{\tau=0}^{T-1}\sum_{i,m} g(\overline{\mathbf{Q}}_{mi}(nT+\tau))N_m^{(i)}(nT+\tau)\right], \qquad (4.12)
\end{aligned}
$$

where the last equality follows from Lemma 3.3 since the expected departure is 1 for any job that is scheduled.

To bound this term, we will first show that the increase of $\sum_m g(\overline{\mathbf{Q}}_{mi}(t))N_m^{(i)}(t)$ is bounded in a super time slot. For any t such that $nT \le t < (n+1)T$, for each server i,

$$
\begin{aligned}
&\sum_m g(\overline{\mathbf{Q}}_{mi}(t))N_m^{(i)}(t-1) \\
=&\sum_m g(\overline{\mathbf{Q}}_{mi}(t))N_m^{(i)}(t^-) \\
&+\sum_m g(\overline{\mathbf{Q}}_{mi}(t))\left(N_m^{(i)}(t-1)-N_m^{(i)}(t^-)\right) \\
\overset{(a)}{\le}&\sum_m g(\overline{\mathbf{Q}}_{mi}(t))N_m^{(i)}(t^-)+\sum_m g(\overline{\mathbf{Q}}_{mi}(t))\widetilde{N}_m^{(i)}(t) \\
=&\sum_m \Big(g(\overline{\mathbf{Q}}_{mi}(t))N_m^{(i)}(t^-)+g(\overline{\mathbf{Q}}_{mi}(t))\widetilde{N}_m^{(i)}(t)\Big)\mathbb{I}_{\mathbf{Q}_{mi}(t)\ge N_{\max}} \\
&+\sum_m \Big(g(\overline{\mathbf{Q}}_{mi}(t))N_m^{(i)}(t^-)+g(\overline{\mathbf{Q}}_{mi}(t))\widetilde{N}_m^{(i)}(t)\Big)\mathbb{I}_{\mathbf{Q}_{mi}(t)< N_{\max}} \\
\overset{(b)}{\le}&\sum_m g(\overline{\mathbf{Q}}_{mi})(t)N_m^{(i)}(t)+MN_{\max}g(\widetilde{C}N_{\max}),
\end{aligned}
$$

where inequality (a) follows from the definition $\widetilde{N}_m^{(i)}(t)$; and inequality (b) holds for the following reason. When $\mathbf{Q}_{mi}(t) \ge N_{\max}$, there are enough type-$m$ jobs to be allocated to the servers, and so $\overline{N}_m^{(i)}(t) = \widetilde{N}_m^{(i)}(t)$ and $N_m^{(i)}(t) = N_m^{(i)}(t^-) + \widetilde{N}_m^{(i)}(t)$. For the other case, we just use the fact that $\overline{\mathbf{Q}}_{mi}(t) \le \mathbf{Q}_{mi}(t)\widetilde{C}$ from Lemma 3.2 and $N_m^{(i)}(t) \le N_{\max}$

From (4.5), since $\overline{\mathbf{Q}}_{mi}(t-1) - \overline{\mathbf{Q}}_{mi}(t) \le N_{\max}/C$, again from concavity of

$g(.)$ and $g'(.) \leq 1$, we have

$$\sum_m g(\overline{\mathbf{Q}}_{mi}(t-1))N_m^{(i)}(t-1) \leq \beta' + \sum_m g(\overline{\mathbf{Q}}_{mi}(t))N_m^{(i)}(t),$$

where $\beta' = MN_{\max}g(\widetilde{C}N_{\max}) + MN_{\max}^2$. Using this recursively for $t = nT+\tau$ such that $\tau < T$, we get

$$-\sum_m g(\overline{\mathbf{Q}}_{mi}(nT+\tau))N_m^{(i)}(nT+\tau) \leq \tau\beta' - \sum_m g(\overline{\mathbf{Q}}_{mi}(nT))N_m^{(i)}(nT).$$

Since $(\lambda, \overline{S}) \in \widehat{\mathcal{C}}$, 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 $\epsilon > 0$ such that $(\lambda^i + \epsilon) \circ \overline{S} \in \text{Conv}(\mathcal{N}_i)$ for all i. From Lemma 3.2, we have $g(\overline{\mathbf{Q}}_{mi}(nT)) \leq g(\widetilde{C}\mathbf{Q}_{mi}(nT)) \leq \log(\widetilde{C}(1+\mathbf{Q}_{mi}(nT))) \leq g(\mathbf{Q}_{mi}(nT)) + \log(\widetilde{C})$. The last inequality which is an immediate consequence of the log function has also been exploited in [38], [39] for a different problem. For each server i, we have

$$\begin{aligned}
&\sum_m g(\overline{\mathbf{Q}}_{mi}(nT))(\lambda_m^i + \epsilon)\overline{S}_m - \log(\widetilde{C})\sum_m(\lambda_m^i + \epsilon)\overline{S}_m \\
\leq &\sum_m g(\mathbf{Q}_{mi}(nT))(\lambda_m^i + \epsilon)\overline{S}_m \\
= &\sum_m \left(g(\mathbf{Q}_{mi}(nT))(\lambda_m^i + \epsilon)\overline{S}_m\right)\mathbb{I}_{\mathbf{Q}_{mi}(nT)\geq N_{\max}} \\
&+ \sum_m \left(g(\mathbf{Q}_{mi}(nT))(\lambda_m^i + \epsilon)\overline{S}_m\right)\mathbb{I}_{\mathbf{Q}_{mi}(nT)< N_{\max}} \\
\leq &\sum_m g(\mathbf{Q}_{mi}(nT))N_m^{(i)}(nT) + g(N_{\max})\sum_m(\lambda_m^i + \epsilon)\overline{S}_m \\
\leq &\sum_m g(\overline{\mathbf{Q}}_{mi}(nT))N_m^{(i)}(nT) + g(N_{\max})\sum_m(\lambda_m^i + \epsilon)\overline{S}_m.
\end{aligned}$$

Since the schedule chosen is MaxWeight schedule whenever $\mathbf{Q}_{mi}(nT) \geq N_{\max}$, we have the first inequality. The last inequality again follows from Lemma 3.2. Summing over i, we get

$$\begin{aligned}
&-\sum_i\sum_m g(\overline{\mathbf{Q}}_{mi}(nT+\tau))N_m^{(i)}(nT+\tau) \\
&\leq L\tau\beta' - \sum_i\sum_m g(\overline{\mathbf{Q}}_{mi}(nT))N_m^{(i)}(nT) \\
&\leq L\tau\beta' - \sum_i\sum_m g(\overline{\mathbf{Q}}_{mi}(nT))(\lambda_m^i + \epsilon)\overline{S}_m + K_{22},
\end{aligned}$$

where $K_{22} = (g(N_{\max})+\log(\widetilde{C}))\sum_m(\lambda_m+L\epsilon)\overline{S}_m$. Substituting this in (4.12), summing over τ and using Lemma 3.1, we get

$$\begin{aligned}
&-E_{\mathbf{q}}\left[\sum_{\tau=0}^{T-1}\sum_{i,m} g(\overline{\mathbf{Q}}_{mi}(nT+\tau))\overline{\mathbf{D}}_{mi}(nT+\tau)\right]\\
\leq &K_{23} - T\sum_i\sum_m g(\overline{\mathbf{q}}_{mi}(nT))(\lambda_m^i+\epsilon)\overline{S}_m,
\end{aligned}$$

where $K_{23} = K_{22}T + L\beta' T(T-1)/2$.

Substituting this and (4.11) in (3.18), we get

$$\begin{aligned}
&E[V(\widetilde{\mathbf{X}}(n+1)) - V(\widetilde{\mathbf{X}}(n))|\widetilde{\mathbf{Q}}(n)=\mathbf{q},\widetilde{\mathbf{Y}}(n)]\\
\leq &K_{24}+T\sum_m\left(g(\overline{\mathbf{q}}_{m\hat{i}_m})\lambda_m\overline{S}_m - \sum_i g(\overline{\mathbf{q}}_{mi})(\lambda_m^i+\epsilon)\overline{S}_m\right)\\
\overset{(a)}{\leq} &K_{24} - \epsilon\overline{S}_{\min}T\sum_i\sum_m g(\overline{\mathbf{q}}_{mi})\\
\leq &K_{25} - \epsilon\overline{S}_{\min}T\log(1+G^{-1}(V(\overline{\mathbf{q}}))),
\end{aligned}$$

where $K_{24} = K_8 + K_{21} + K_{23} + \log(\widetilde{C})\sum_m(\lambda_m+L\epsilon)\overline{S}_m$ and $K_{25} = K_{24} + \epsilon\overline{S}_{\min}T$. 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 3.7.

If the job 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 3.2, the Foster-Lyapunov theorem can be used to show that the sampled Markov Chain $\widetilde{\mathbf{X}}(n)$ is positive recurrent. We need the bounded job size assumption here because, without it the set $\mathcal{B}$ could 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 job sizes are not bounded in general, we will use Lemma 3.5 to show stability of Algorithm 3 for the random process $U(n)$. From Lemma 3.6, we have

$$\begin{aligned}
&E[U(\widetilde{\mathbf{X}}(n+1)) - U(\widetilde{\mathbf{X}}(n))|\widetilde{\mathbf{X}}(n)=\mathbf{x}=(\mathbf{q},\mathbf{y})]\\
\leq &E\left[\left.\frac{V(\widetilde{\mathbf{X}}(n+1)) - V(\widetilde{\mathbf{X}}(n))}{\log(1+U(\widetilde{\mathbf{X}}(n))}\right|\widetilde{\mathbf{X}}(n)=\mathbf{x}=(\mathbf{q},\mathbf{y})\right]
\end{aligned}$$

$$\leq \frac{K_{25}}{\log(1+U(\overline{\mathbf{q}}))} - \epsilon \overline{S}_{\min} T \quad \leq -\frac{\epsilon \overline{S}_{\min} T}{2}$$

whenever $U(\overline{\mathbf{q}}) > e^{(2K_{25}/\epsilon \overline{S}_{\min} T)}$. Thus, $U(n)$ satisfies condition C1 of Lemma 3.5 for the filtration generated by the $\{\widetilde{\mathbf{X}}(n)\}$. We will now verify condition C2. From Lemma 3.6, Lemma 3.7 and (4.8), we have

$$\begin{aligned}
&(U(nT+\tau+1) - U(nT+\tau)) \\
\leq& \frac{[V(nT+\tau+1) - V(nT+\tau)]}{\log(1+G^{-1}(V(\overline{\mathbf{Q}}(nT+\tau))))} \\
\leq& \frac{K_8 + A_{\max} \sum_{m,i} \overline{S}_m g(\overline{\mathbf{Q}}_{mi}(nT+\tau))}{\log(1+G^{-1}(V(\overline{\mathbf{Q}}(nT+\tau))))} \\
\leq& \frac{K_8}{\log(1+G^{-1}(V(\overline{\mathbf{Q}}(nT+\tau))))} + \frac{A_{\max}\overline{S}_{\max}}{LM} \\
\overset{(a)}{\leq}& K_{14} \quad \text{if } U(nT+\tau) > 0,
\end{aligned}$$

where $K_{14} = \frac{K_8}{\log(2)} + \frac{A_{\max}\overline{S}_{\max}}{LM}$. 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 $V(nT+\tau) > G(1)$. This gives the inequality (a). If $U(nT+\tau) = 0$, from (3.16), we have $(U(nT+\tau+1) - U(nT+\tau)) \leq K_{15} = G^{-1}(K_4)$. Thus, we have

$$(U(nT+\tau+1) - U(nT+\tau)) \leq K_{16},$$

where $K_{16} = \max\{K_{14}, K_{15}\}$. Similarly, from (3.17) it can be shown that

$$(U(nT+\tau) - U(nT+\tau+1)) \leq K_{18},$$

where $K_{18} = \max\{K_{17}, K_{15}\}$ and $K_{17} = \frac{K_8}{\log(2)} + \frac{N_{\max}}{LM}$. Constants $K_{14} - K_{18}$ were defined in Chapter 3. Setting $K_{19} = \max\{K_{16}, K_{18}\}$, we have

$$\begin{aligned}
(|U(nT+\tau) - U(nT+\tau+1)|) &\leq K_{19} \\
(|U(nT+\tau) - U(nT+\tau+1)| | \mathbf{X}(nT)) &\leq K_{19} \\
\left(|U(\widetilde{\mathbf{X}}(n+1)) - U(\widetilde{\mathbf{X}}(n))| \middle| \widetilde{\mathbf{X}}(n)\right) &\leq K_{19}T.
\end{aligned}$$

Thus, $U(n)$ satisfies condition C2 in Lemma 3.5 and so 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

$$G\left(\frac{\sum_{m,i}\overline{\mathbf{Q}}_{mi}(nT)}{LM}\right) \le \frac{\sum_{m,i} G\left(\overline{\mathbf{Q}}_{mi}(nT)\right)}{LM} \le V(\overline{\mathbf{Q}}(nT)). \tag{4.13}$$

Then, from Lemma 3.2 and (b), we get

$$\sum_{m,i}\mathbf{Q}_{mi}(nT) \le \sum_{m,i}\overline{\mathbf{Q}}_{mi}(nT) \le LMU(\overline{\mathbf{Q}}(nT)) \le \frac{LM}{\theta^*}e^{\theta^* U(\widetilde{\mathbf{X}}(n))}.$$

Thus, we have $\lim_{n\to\infty}\sum_m\sum_i E[\mathbf{Q}_{mi}(nT)] \le \frac{LM}{\theta^*}\mathcal{K}_4$.

For any $(n-1)T \le t \le nT$, from (4.5) and Lemma 3.2, we have

$$\mathbf{Q}_{mi}(t) \le \overline{\mathbf{Q}}_{mi}(t) \le \overline{\mathbf{Q}}_{mi}(nT) + T\frac{N_{\max}}{C} \le \widetilde{C}\mathbf{Q}_{mi}(nT) + T\frac{N_{\max}}{C}$$
$$\sum_m\sum_i E[\mathbf{Q}_{mi}(t)] \le \sum_m\sum_i E\left[\left(\widetilde{C}\mathbf{Q}_{mi}(nT) + TN_{\max}/C\right)\right].$$

As $t\to\infty$, we get

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

□

4.2 Conclusion

To ameliorate the poor performance of the completely nonpreemptive algorithms studied so far, in this chapter we have studied the cloud resource allocation problem when jobs are allowed to be preempted once in a while. We assumed that the job sizes are unknown and unbounded, and have presented a throughput optimal algorithm.

CHAPTER 5

DELAY OPTIMALITY

The results in previous chapters study only throughput optimality of resource allocation algorithms. Throughput optimality makes sure that the best usage of available resources is made. However, in practice, delay is more important.

In this chapter, we will study delay optimality in an asymptotic regime called heavy traffic regime. However, at this point, we do not know if any of the algorithms presented so far are delay optimal. In particular, we do not have delay optimality when job sizes are unknown. So, we now reconsider the case when job sizes are known as in Chapter 2 and as in the previous chapter, we assume that preemption is allowed every T time slots.

In this chapter, we present a myopic MaxWeight algorithm with JSQ routing, described in Algorithm 9, and we prove its throughput optimality and delay optimality in heavy traffic regime.

Studying mean delay is equivalent to studying mean queue length of backlogged workload because they are related by Little's law.

Characterizing the exact delay or queue length in general is difficult. So, various asymptotic limits have been studied in the literature. One popular approach is to study the system in the heavy-traffic regime, i.e., when the exogenous arrival rate is close to the boundary of the capacity region. We say that an algorithm is heavy-traffic optimal if it minimizes $\lim_{\epsilon\to 0}\epsilon\mathbb{E}\left[f(\mathbf{q})\right]$ where ϵ is the distance of the arrival rate vector from the boundary of the capacity region, $\mathbf{q}$ is the vector of queue lengths of backlogged workload and $f(.)$ is a function which we will clearly define later.

In the heavy-traffic regime, for some systems, the multi-dimensional state of the system reduces to a single dimension, called state-space collapse. In [40, 41], a method was outlined to use the state-space collapse for studying the diffusion limits of several queuing systems. This procedure has been successfully applied to a variety of multiqueue models served by multiple servers [42, 43, 44, 45].

Stolyar [46], generalized this notion of state-space collapse and resource pooling to a generalized switch model. This was used to establish the heavy traffic optimality of the MaxWeight algorithm.

Most of these results are based on considering a scaled version of queue lengths and time, which converges to a regulated Brownian motion, and then show sample-path optimality in the scaled time over a finite time interval. This then allows a natural conjecture about steady state distribution. In [37], the authors present an alternate method to prove heavy traffic optimality that is not only simpler, but shows heavy traffic optimality in unscaled time. In addition, this method directly obtains heavy-traffic optimality in steady state. The method consists of the following three steps.

1. *Lower bound:* First a lower bound is obtained on the weighted sum of expected queue lengths of backlogged workload by comparing with a single-server queue. A lower bound for the single-server queue, similar to the Kingman bound [47], then gives a lower bound to the original system. This lower bound is a universal lower bound satisfied by any joint routing and scheduling algorithm.

2. *State-space collapse*: The second step is to show that the state of the system collapses to a single dimension. Here, it is not a complete state-space collapse, as in the Brownian limit approach, but an approximate one. In particular, this step is to show that the queue length along a certain direction increases as the exogenous arrival rate gets closer to the boundary of the capacity region but the queue length in any perpendicular direction is bounded.

3. *Upper bound*: The state-space collapse is then used to obtain an upper bound on the weighted queue length. This is obtained by using a natural Lyapunov function suggested by the resource pooling. Heavy-traffic optimality can be obtained if the upper bound coincides with the lower bound.

Under the special case of $T = 1$, i.e., when jobs are allowed to be preempted every time slot we have shown heavy traffic optimality in [48] and [49]. Here, we generalize this result to arbitrary finite T. In the next section, we present the resource allocation algorithm that is based on myopic

MaxWeight scheduling and JSQ routing. In section 5.2, we show its throughput optimality. In section 5.3, we apply the above three-step procedure and prove heavy-traffic optimality when all the servers are identical. The lower bound in this case is identical to the case of the MaxWeight scheduling problem in ad hoc wireless networks in [37]. The state-space collapse and upper bound do not directly follow from the corresponding results for the MaxWeight algorithm in [37] due to the additional routing step here. In section 5.4 we present heavy traffic optimality when power-of-two-choices routing is used instead of JSQ for the indentical server case.

Note on Notation: The set of real numbers, the set of non-negative real numbers, and the set of positive real numbers are denoted by $\mathbb{R}$, $\mathbb{R}_+$ and $\mathbb{R}_{++}$ respectively. We use a slightly different notation in this chapter. We denote vectors in $\mathbb{R}^M$ or $\mathbb{R}^L$ by x, in normal font. We use bold font $\mathbf{x}$ only to denote vectors in $\mathbb{R}^{ML}$. Dot product in the vector spaces $\mathbb{R}^M$ or $\mathbb{R}^L$ is denoted by $\langle x, y\rangle$ and the dot product in $\mathbb{R}^{ML}$ is denoted by $\langle \mathbf{x}, \mathbf{y}\rangle$.

5.1 Algorithm with Limited Preemption - Known Job Sizes

In this chapter, we use the same notation as in Chapter 2, for instance, q for the backlogged workload, etc. Recall that $a_m(t)$ is the total workload of type m that arrives in time slot t, $\mathbb{E}[a_m(t)] = \check{\lambda}_m$ $var[a_m(t)] = \sigma_m^2$, $\check{\lambda} = (\check{\lambda}_1, ...\check{\lambda}_M)$ and $\sigma = (\sigma_1, ...\sigma_M)$. We denote $\sigma^2 = (\sigma_1^2, ...\sigma_M^2)$. We assume that the job sizes are upper bounded by $D_{\max}$.

Consider server i. In this chapter, we say that server i is in configuration $s = (s_1, s_2, ..., s_M) \in (\mathbb{Z}_+)^M$ if the server is serving s_1 jobs of type 1, s_2 jobs of type 2, etc. We have earlier used N to denote this. Let s_{max} be the maximum number of jobs of any type that can be scheduled on any server. Recall that $\mathcal{N}_i$ is the set of feasible configurations on server i. Let $\mathcal{C}_i^*$ be the convex hull of the maximal configurations of server i. Let $\mathcal{C}_i = \{s \in (\mathbb{R}_+)^M : s \leq s^* \text{ for some } s^* \in \mathcal{C}_i^*\}$. Here $s \leq s^*$ means $s_m \leq s_m^* \forall m \in \{1, 2, ..., M\}$. $\mathcal{C}_i$ can be thought of as the capacity region for server i. $\mathcal{C}_i$ is a convex polytope in the nonnegative quadrant of $\mathbb{R}^M$.

Then the capacity region $\mathcal{C}$ would be $\mathcal{C} = \sum_{i=1}^{L} \mathcal{C}_i = \{s \in (\mathbb{R}_+)^M : \exists s^i \in$

$\mathcal{C}_i \; \forall \; i$ s.t. $s \leq \sum_{i=1}^{L} s^i\}$. Here s^i just denotes an element in $\mathcal{C}_i$ and not i^{th} power of s. Here $\sum$ denotes the Minkowski sum of sets. Therefore, $\mathcal{C}$ is again a convex polytope in the nonnegative quadrant of $\mathbb{R}^M$, and it can be described by a set of hyperplanes as follows:

$$\mathcal{C} = \{s \geq 0 : \left\langle c^{(k)}, s \right\rangle \leq b^{(k)}, k = 1, ...K\},$$

where K is the number of hyperplanes that completely defines $\mathcal{C}$, and $(c^{(k)}, b^{(k)})$ completely defines the k^{th} hyperplane $\mathcal{H}^{(k)}$, $\left\langle c^{(k)}, s \right\rangle = b^{(k)}$. Since $\mathcal{C}$ is in the first quadrant, we have

$$||c^{(k)}|| = 1 \quad , c^{(k)} \geq 0, \quad b^{(k)} \geq 0 \quad for \; k = 1, 2, ...K.$$

Similar to $\mathcal{C}$, define $\mathcal{S} = \sum_{i=1}^{L} \mathcal{S}_i$. WLOG, we assume that the $\mathcal{C}$ is full-dimensional, i.e., it is J-dimensional.

Lemma 5.1. Given the k^{th} hyperplane $\mathcal{H}^{(k)}$ of the capacity region $\mathcal{C}$ (i.e., $\left\langle c^{(k)}, \check{\lambda} \right\rangle = b^{(k)}$), for each server i, there is a $b_i^{(k)}$ *such that* $\left\langle c^{(k)}, \check{\lambda} \right\rangle = b_i^{(k)}$ is the boundary of the capacity region $\mathcal{C}_i$, and $b^{(k)} = \sum_{i=1}^{L} b_i^{(k)}$ *. Moreover, for* every set $\left\{ \check{\lambda}_i^{(k)} \in \mathcal{C}_i \right\}_i$ such that $\check{\lambda}^{(k)} = \sum_{i=1}^{L} \check{\lambda}_i^{(k)}$ and $\check{\lambda}^{(k)} \in \mathcal{C}$ lies on the k^{th} hyperplane $\mathcal{H}^{(k)}$, we have that $\left\langle c^{(k)}, \check{\lambda}_i^{(k)} \right\rangle = b_i^{(k)}$.

Proof. Define $b_i^{(k)} = \max_{s \in \mathcal{C}_i} \left\langle c^{(k)}, s \right\rangle$. Then, since
$\mathcal{C} = \sum_{i=1}^{L} \mathcal{C}_i$, we have that $b^{(k)} = \sum_{i=1}^{L} b_i^{(k)}$.

Again, by the definition of $\mathcal{C}$, for every $\check{\lambda} \in \mathcal{C}$, there are $\check{\lambda}_i^{(k)} \in \mathcal{C}_i$ for each i such that $\check{\lambda}^{(k)} = \sum_{i=1}^{L} \check{\lambda}_i^{(k)}$. However, these may not be unique. We will prove that for every such $\left\{ \check{\lambda}_i^{(k)} \right\}_i$, for each i, $\left\langle c^{(k)}, \check{\lambda}_i^{(k)} \right\rangle = b_i^{(k)}$. Suppose, for some server i_1, $\left\langle c^{(k)}, \check{\lambda}_{i_1}^{(k)} \right\rangle < b_{i_1}^{(k)}$. Then since $\left\langle c^{(k)}, \sum_{i=1}^{L} \check{\lambda}_i^{(k)} \right\rangle = \sum_{i=1}^{L} b_i^{(k)}$, there exists i_2 such that
$\left\langle c^{(k)}, \check{\lambda}_{i_2}^{(k)} \right\rangle > b_{i_2}^{(k)}$ which is a contradiction. Thus, we have the lemma. $\square$

The JSQ routing and myopic MaxWeight scheduling is described in Algorithm 9.

Algorithm 9 JSQ Routing and myopic MaxWeight Scheduling

1. *Routing Algorithm:* All the type m arrivals in a time slot are routed to the server with the smallest backlogged workload for type m jobs, i.e., the server $i^*_m = \underset{i\in\{1,2,\ldots L\}}{\arg\min}\, q_{mi}$. Ties are broken uniformly at random.

2. *Scheduling Algorithm (Myopic MaxWeight Scheduling) for each server i:*

   T time slots are grouped into a super time slot. A MaxWeight configuration is chosen at the beginning of a super time slot. So, for $t = nT$, configuration $\overline{s^i} \in \mathcal{C}^*_i$ is chosen according to

   $$\overline{s^i}(t) \in \underset{s\in\mathcal{C}^*_i}{\arg\max} \sum_{m=1}^{M} s_m q_{mi}.$$

   It then schedules up to a maximum of $\overline{s^i_m}(t)$ jobs of type m (in a preemptive manner). Note that even if the queue length is greater than the allocated service, all of it may not be utilized, e.g., when the backlogged size is from a single job, since different chunks of the same job cannot be scheduled simultaneously. Denote the actual number of jobs chosen by $s^i_m(t)$. Note that if $q_{mi} \geq D_{max}s_{max}$, then $\overline{s^i_m} = s^i_m$. For all other t, at the beginning of the time slot, a new configuration is chosen as follows:

   $$\overline{s^i}(t) \in \underset{s:s+s^i(t^-)\in\mathcal{C}^*_i}{\arg\max} \sum_{m=1}^{M} s_m q_{mi},$$

   where $s^i(t^-)$is the configuration of jobs at server i that are still in service at the end of the previous time slot.

   However, $\overline{s^i_m}(t)$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 $\widetilde{s^i_m}(t)$ denotes the actual number of type m jobs selected at server m, then the configuration at time t is $s^i(t) = \widetilde{s^i}(t)$. Otherwise, i.e., if there are enough number of jobs at server i, $s^i(t) = \overline{s^i}(t)$.

Let $Y_{mi}(t)$ denote the state of the queue for type-m jobs at server i. If there are J such jobs, $Y_{mi}(t)$ is a vector of size J and $Y_{mi}^j(t)$ is the (backlogged) size of the j^{th} type-m job at server i. It is easy to see that $\mathbf{Y}(t) = \{Y_{mi}(t)\}_{mi}$ is a Markov chain under the JSQ routing and MaxWeight scheduling. Let $q_{mi}(t)$ denote the queue length of backlogged workload of type-m jobs at server i. Then the vector of backlogged workloads, $\mathbf{q}(t) = \{q_{mi}(t)\}_{mi}$, is a function $\mathbf{q}(\mathbf{Y})$ of the state $Y_{mi}(t)$ given by $q_{mi}(t) = \sum_j Y_{mi}^j(t)$.

However, note that $\mathbf{Y}(t)$ is not a time-homogeneous Markov process. Sampling the process $\mathbf{Y}(t)$ every T time slots, we obtain the process $\widetilde{\mathbf{Y}}(n) = \mathbf{Y}(nT)$, which is a time-homogeneous Markov process. Moreover, note that the process $\widehat{\mathbf{Y}}(n) = (\mathbf{Y}(nT), \mathbf{Y}(nT+1) \dots \mathbf{Y}(nT+T-1))$ is also a time-homogeneous Markov process.

The queue lengths of backlogged workload evolve according to the following equation:

$$\begin{aligned} q_{mi}(t+1) &= q_{mi}(t) + a_{mi}(t) - s_m^i(t) \\ &= q_{mi}(t) + a_{mi}(t) - \overline{s_m^i}(t) + \overline{u}_{mi}(t), \end{aligned} \tag{5.1}$$

where $\overline{u}_{mi}(t)$ is the unused service, given by $\overline{u}_{mi}(t) = \overline{s_m^i}(t) - s_m^i(t)$, $\overline{s_m^i}(t)$ is the MaxWeight or myopic MacWeight schedule as defined in Algorithm 9 and $s_m^i(t)$ is the actual schedule chosen by the scheduling algorithm and the arrivals are

$$a_{mi}(t) = \begin{cases} a_m(t) & \text{if } i = i_m^*(t) \\ 0 & \text{otherwise} \end{cases}. \tag{5.2}$$

Here, i_m^* is the server chosen by the routing algorithm for type m jobs. Note that

$$\overline{u}_{mi}(t) = 0 \text{ when } q_{mi}(t) + a_{mi}(t) \geq D_{max} s_{max}. \tag{5.3}$$

Also, denote $s = (s_m)_m$ where

$$s_m = \sum_{i=1}^{L} s_m^i. \tag{5.4}$$

Denote $\mathbf{a} = (a_{mi})_{mi}$, $\mathbf{s} = (s_m^i)_{mi}$ and $\overline{\mathbf{u}} = (\overline{u}_{mi})_{mi}$. Also denote $\mathbf{1}$ to be the vector with 1 in all components.

5.2 Throughput Optimality

In this section, we will first show that Algorithm 9 is throughput optimal. We show not only stability in the sense that queue lengths are finite, but we show a stronger result, viz., positive recurrence.

Theorem 5.1. *Assume $a_m(t) \leq a_{\max}$ for all t and m and assume that the job load vector satisfies $\check{\lambda} \in int(\mathcal{C})$. Then, the JSQ routing and myopic MaxWeight scheduling as described in Algorithm 9 stabilizes the system. Moreover, the sampled Markov processes $\widetilde{\mathbf{Y}}(n)$ as well as the Markov process $\widehat{\mathbf{Y}}(n)$ are positive recurrent. Consequently, both the processes have a steady state distribution.*

Proof. The proof of positive recurrence of $\widetilde{\mathbf{Y}}(n)$ and stability of the system is very similar to the proof of Theorem 2.1 in Chapter 2 and so we skip the details here. Recall that the proof uses a quadratic Lyapunov function, $V(\mathbf{Y}) = \|\mathbf{q}(\mathbf{Y})\|^2 = \sum\limits_{i=1}^{L}\sum\limits_{m=1}^{M} q_{mi}^2$ and shows that its drift over T time slots is negative outside the finite set $\mathcal{B} = \{\mathbf{Y} : \sum_i \sum_m q_{mi}(\mathbf{Y})\check{\lambda}_m^i \leq K_2/\epsilon T\}$. In other words, we have shown the following:

$$\begin{aligned}
&E[V(\widetilde{\mathbf{Y}}(n+1)) - V(\widetilde{\mathbf{Y}}(n))|\widetilde{\mathbf{Y}}(n) = \widetilde{\mathbf{Y}}]\\
=&E[V(\mathbf{Y}((n+1)T)) - V(\mathbf{Y}(nT))|\mathbf{Y}(nT) = \widetilde{\mathbf{Y}}]\\
\leq& - K_2 \qquad \text{whenever } \widetilde{\mathbf{Y}} \notin \mathcal{B}.
\end{aligned}$$

Then the Foster-Lyapunov theorem was used to show positive recurrence of $\widetilde{\mathbf{Y}}(n)$.

Now for the Markov process $\widehat{\mathbf{Y}}(n)$, consider the Lyapunov function $\widehat{V}(\widehat{\mathbf{Y}}) = \widehat{V}(\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_{T-1}) \triangleq V(\mathbf{Y}_1) = \|\mathbf{q}(\mathbf{Y}_1)\|^2$. Then the one-step drift of this Lyapunov function is

$$\begin{aligned}
&E\left[\widehat{V}(\widehat{\mathbf{Y}}(n+1)) - V(\widehat{\mathbf{Y}}(n))\middle|\, \widehat{\mathbf{Y}}(n) = \widehat{\mathbf{Y}} = (\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_{T-1})\right]\\
=&E\left[V(\mathbf{Y}((n+1)T)) - V(\mathbf{Y}(nT))\right.\\
&\qquad\left.|(\mathbf{Y}(nT), \mathbf{Y}(nT+1), \ldots, \mathbf{Y}(nT+T-1)) = (\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_{T-1})\right]\\
=&E\left[E[V(\mathbf{Y}((n+1)T)) - V(\mathbf{Y}(nT))|\mathbf{Y}(nT) = \mathbf{Y}_0]\right.\\
&\qquad\left.|(\mathbf{Y}(nT), \mathbf{Y}(nT+1), \ldots, \mathbf{Y}(nT+T-1)) = (\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_{T-1})\right]\\
\leq& - K_2 \qquad \text{whenever } \mathbf{Y}_0 \notin \mathcal{B}.
\end{aligned}$$

Now define the set $\mathcal{B}'$ as follows:

$$\mathcal{B}' = \left\{ (\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_{T-1}) : \begin{array}{l} \mathbf{Y}_0 \in \mathcal{B}. \text{ for } \tau \in \{1, 2, \ldots, T-1\} and, \\ \mathbf{Y}_\tau \text{ is such that it is a feasible state for} \\ \text{the Markov chain } \mathbf{Y}(t) \text{ at time } \tau \text{ assuming} \\ \text{it has started in state } \mathbf{Y}_0 \text{ at time } 0 \end{array} \right\}.$$

Since we assume that the total number of job arrivals, job sizes and total number of departures in the queue is bounded in each time slot and since the set $\mathcal{B}$ is finite, we have that set $\mathcal{B}'$ is also finite. Then, using the Foster-Lyapunov theorem, we have that the Markov process $\widehat{\mathbf{Y}}(n)$ is also positive recurrent. Note that this is the only instance in this chapter where we assume that $a_m(t) \leq a_{\max}$. This assumption can easily be relaxed. □

Note that the steady-state distributions of the processes $\widetilde{\mathbf{Y}}(n)$ and $\widehat{\mathbf{Y}}(n)$ are related. Let $(\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{T-1})$ be the queue lengths of backlogged workloads corresponding to the state of the Markov process $\widehat{\mathbf{Y}}(n)$. Let $(\pi_0(\mathbf{q}), \pi_1(\mathbf{q}), \ldots, \pi_{T-1}(\mathbf{q}))$ denote their marginal probability distributions in steady state. Then, the probability distribution of the queue lengths of the sampled process $\widetilde{\mathbf{Y}}(n)$ is given by $\pi_0(\mathbf{q})$.

Now that we have shown that a steady-state exists, in the next section, we will show that the queue lengths of backlogged workload are optimal in the steady state in the heavy-traffic regime.

5.3 Heavy Traffic Optimality

Recall that the capacity region is bounded by K hyperplanes, each hyperplane $\mathcal{H}^{(k)}$ described by its normal vector $c^{(k)}$ and the value $b^{(k)}$. Then, for any $\check{\lambda} \in interior(\mathcal{C})$, we can define the distance of $\check{\lambda}$ to $\mathcal{H}^{(k)}$ and the closest point, respectively, as

$$\begin{aligned} \epsilon^{(k)} &= \min_{s \in \mathcal{H}^{(k)}} ||\check{\lambda} - s|| \\ \check{\lambda}^{(k)} &= \check{\lambda} + \epsilon^{(k)} c^{(k)}, \end{aligned} \tag{5.5}$$

where $\epsilon^{(k)} > 0$ for each k since $\check{\lambda} \in interior(\mathcal{C})$. We let $\epsilon \triangleq \left(\epsilon^{(k)}\right)_{k=1}^{K}$ denote the vector of distances to all hyperplanes. Note that $\check{\lambda}^{(k)}$ may be outside the capacity region $\mathcal{C}$ for some hyperplanes. So define

$$\mathcal{K}_{\check{\lambda}} \triangleq \left\{k \in \{1, 2, ...K\} : \check{\lambda}^{(k)} \in \mathcal{C}\right\}.$$

The set $\mathcal{K}_{\check{\lambda}}$ identifies the set of *dominant hyperplanes* whose closest point to $\check{\lambda}$ is on the boundary of the capacity region $\mathcal{C}$ and hence is a feasible average rate for service. Note that for any $\check{\lambda} \in \mathit{interior}(\mathcal{C})$, the set $\mathcal{K}_{\check{\lambda}}$ is non-empty, and hence is well-defined. We further define

$$\mathcal{K}^o_{\check{\lambda}} \triangleq \left\{k \in \mathcal{K}_{\check{\lambda}} : \check{\lambda}^{(k)} \in \mathit{Relint}(\mathcal{F}^{(k)})\right\},$$

where $\mathcal{F}^{(k)}$ denotes the face on which $\check{\lambda}^{(k)}$ lies and *Relint* means relative interior. Thus, $\mathcal{K}^o_{\check{\lambda}}$ is the subset of faces in $\mathcal{K}_{\check{\lambda}}$ for which the projection of $\check{\lambda}$ is not shared by more than one hyperplane.

For $\epsilon \triangleq \left(\epsilon^{(k)}\right)_{k=1}^{K} > 0$, let $\check{\lambda}^{(\epsilon)}$ be the arrival rate in the interior of the capacity region so that its distance from the hyperplane $\mathcal{H}^{(k)}$ is $\epsilon^{(k)}$. Let $\check{\lambda}^{(k)}$ be the closest point to $\check{\lambda}^{(\epsilon)}$ on $\mathcal{H}^{(k)}$. Thus, we have

$$\check{\lambda}^{(k)} = \check{\lambda}^{(\epsilon)} + \epsilon^{(k)} c^{(k)}. \tag{5.6}$$

Let $\mathbf{q}^{(\epsilon)}(t)$ be the backlogged workload queue length process when the arrival rate is $\check{\lambda}^{(\epsilon)}$.

Define $\mathbf{c}^{(k)} \in \mathbb{R}^{ML}_{+}$, indexed by m, i as $\mathbf{c}^{(k)}_{mi} = \frac{c^{(k)}_m}{\sqrt{L}}$. We expect that the state space collapse occurs along the direction $\mathbf{c}^{(k)}$. This is intuitive. For a fixed m, JSQ routing tries to equalize the queue lengths of backlogged workload across servers. For a fixed server i, we expect that the state space collapse occurs along $c^{(k)}$ when approaching the hyperplane $\mathcal{H}^{(k)}$, as shown in [37]. Thus, for JSQ routing and MaxWeight, we expect that the state space collapse occurs along $\mathbf{c}^{(k)}$ in $\mathbb{R}^{ML}$.

For each $k \in \mathcal{K}^o_{\check{\lambda}^{(\epsilon)}}$, define the projection and perpendicular component of $\mathbf{q}^{(\epsilon)}$ to the vector $\mathbf{c}^{(k)}$ as follows:

$$\mathbf{q}^{(\epsilon,k)}_{||} \triangleq \left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)} \right\rangle \mathbf{c}^{(k)}$$
$$\mathbf{q}^{(\epsilon,k)}_{\perp} \triangleq \mathbf{q}^{(\epsilon)} - \mathbf{q}^{(\epsilon,k)}_{||}.$$

In this section, we will prove the following theorem.

Theorem 5.2. *Consider the cloud computing system described in section 5.1 with the assumption that the job arrivals satisfy $a_m(t) \leq a_{\max}$ for all m and t. Assume all the servers are identical and that JSQ routing and myopic MaxWeight scheduling as described in Algorithm 9 are used. Let the exogenous arrival rate be $\check{\lambda}^{(\epsilon)} \in Interior(\mathcal{C})$ and the standard deviation of the arrival vector be $\sigma^{(\epsilon)} \in \mathbb{R}^M_{++}$ where the parameter $\epsilon = \left(\epsilon^{(k)}\right)_{k=1}^K$ is such that $\epsilon^{(k)}$ is the distance of $\check{\lambda}^{(\epsilon)}$ from the k^{th} hyperplane $\mathcal{H}^{(k)}$ as defined in (5.5). Then for each $k \in \mathcal{K}^o_{\check{\lambda}^{(\epsilon)}}$, the steady state queue length of backlogged workload satisfies*

$$\epsilon^{(k)}\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle\right] \leq \frac{\zeta^{(\epsilon,k)}}{2} + B_3^{(\epsilon,k)},$$

where $\zeta^{(\epsilon,k)} = \frac{1}{\sqrt{L}}\left\langle \left(c^{(k)}\right)^2, \left(\sigma^{(\epsilon)}\right)^2\right\rangle + \frac{\left(\epsilon^{(k)}\right)^2}{\sqrt{L}}$, $B_3^{(\epsilon,k)}$ is $o(\frac{1}{\epsilon^{(k)}})$.

In the heavy traffic limit as $\epsilon^{(k)} \downarrow 0$, this bound is tight, i.e.,

$$\lim_{\epsilon^{(k)}\downarrow 0} \epsilon^{(k)}\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle\right] = \frac{\zeta^{(k)}}{2},$$

where $\zeta^{(k)} = \frac{1}{\sqrt{L}}\left\langle \left(c^{(k)}\right)^2, (\sigma)^2\right\rangle$.

Note that since the process $\mathbf{Y}(t)$ is not time-homogeneous, the steady state referred to in the theorem is that of the process $\widehat{\mathbf{Y}}(n)$. The theorem gives a bound on $\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle\right]$, where the expectation is taken according to the distribution $\pi_{t \mod T}(.)$ and the bound is valid for all $\tau \in \{0, 1, 2, \ldots, T-1\}$. We do no need the assumption $a_m(t) \leq a_{\max}$ in the proof of heavy-traffic optimality. This assumption was needed for throughput optimality in Theorem 5.1 (and so for the existence of steady-state) and can easily be relaxed.

We will prove this theorem by following the three-step procedure described before, by first obtaining a lower bound, then showing state space collapse and finally using the state space collapse result to obtain an upper bound.

5.3.1 Lower Bound

We will obtain a lower bound on $\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}\right\rangle\right] = \mathbb{E}\left[\sum_{m=1}^{M} \frac{c_m^{(k)}}{\sqrt{L}}\left(\sum_{i=1}^{L} q_{mi}\right)\right]$ in steady state using the lower bound on the queue length of a single server queue. For the cloud system, given the capacity region and the face $\mathcal{F}^{(k)}$, we will construct a single server queue with appropriate arrivals and service rates to obtain a lower bound.

Consider a single server queuing system, $\phi^{(\epsilon)}(t)$, with arrival process $\frac{1}{\sqrt{L}}\left\langle c^{(k)}, a^{(\epsilon)}(t)\right\rangle$ and service process given by $\frac{b^{(k)}}{\sqrt{L}}$ at each time slot. Then $\phi^{(\epsilon)}(t)$ is stochastically smaller than $\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle$. Thus, for every t, we have

$$\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle\right] \geq \mathbb{E}\left[\phi^{(\epsilon)}(t)\right].$$

The process $\phi^{(\epsilon)}(t)$ is a time-homogeneous Markov process and it is positive recurrent since the arrival rate $\frac{1}{\sqrt{L}}\left\langle c^{(k)}, \check{\lambda}^{(\epsilon)}(t)\right\rangle$ is smaller than the departure rate $\frac{b^{(k)}}{\sqrt{L}}$. Therefore, a steady state distribution exists for $\phi^{(\epsilon)}(t)$. Using ϕ^2 as Lyapunov function and noting that the drift of it should be zero in steady state, we get that in steady state [37],

$$\epsilon^{(k)}\mathbb{E}\left[\phi^{(\epsilon)}(t)\right] \geq \frac{\zeta^{(\epsilon,k)}}{2} - B_1^{(\epsilon,k)}, \tag{5.7}$$

where $\left(c^{(k)}\right)^2 = \left(\left(c_m^{(k)}\right)^2\right)_{m=1}^{M}$, $B_1^{(\epsilon,k)} = \frac{b^{(k)}\epsilon^{(k)}}{2}$ and $\zeta^{(\epsilon,k)} = \frac{1}{\sqrt{L}}\left\langle \left(c^{(k)}\right)^2, \left(\sigma^{(\epsilon)}\right)^2\right\rangle + \frac{\left(\epsilon^{(k)}\right)^2}{\sqrt{L}}$.

Thus, in steady state, in the heavy traffic limit as $\epsilon^{(k)} \downarrow 0$, we have that

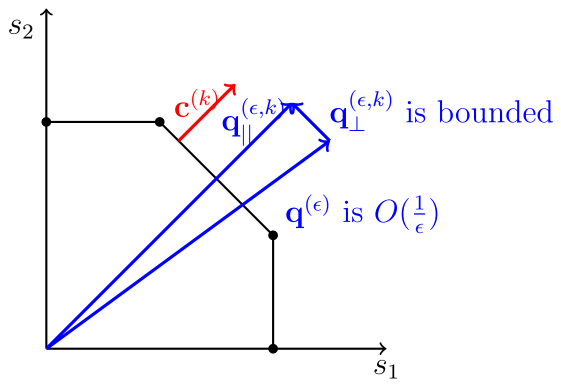
$$\lim_{\epsilon^{(k)}\downarrow 0} \epsilon^{(k)}\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle\right] \geq \frac{\zeta^{(k)}}{2}, \tag{5.8}$$

where $\zeta^{(k)} = \frac{1}{\sqrt{L}}\left\langle \left(c^{(k)}\right)^2, (\sigma)^2\right\rangle$.

Note that this lower bound is a universal lower bound that is valid for any joint routing and scheduling algorithm.

5.3.2 State Space Collapse

In this subsection, we will show that there is a state space collapse along the direction $\mathbf{c}^{(k)}$. We know that as the arrival rate approaches the boundary of the capacity region, i.e., $\epsilon^{(k)} \to 0$, the steady state mean queue length of backlogged workload, $\mathbb{E}[||\mathbf{q}||] \to \infty$. We will show that as $\epsilon^{(k)} \to 0$, queue length of backlogged workload projected along any direction perpendicular



s_2
s_1
$\mathbf{c}^{(k)}$
$\mathbf{q}_{||}^{(\epsilon,k)}$
$\mathbf{q}_{\perp}^{(\epsilon,k)}$ is bounded
$\mathbf{q}^{(\epsilon)}$ is $O(\frac{1}{\epsilon})$

Figure 5.1: Illustration of the capacity region, the vector $\mathbf{c}^{(k)}$ and state space collapse. As the arrival rate approaches the boundary of the capacity region, the queue length vector $\mathbf{q}$ is increasing as $O(\frac{1}{\epsilon})$. But the component perpendicular to $\mathbf{c}$, i.e. $\mathbf{q}_{\perp}$ is bounded.

to $\mathbf{c}^{(k)}$ is bounded. This is illustrated in Figure 5.1. So the constant does not contribute to the first order term in $\frac{1}{\epsilon^{(k)}}$, in which we are interested. Therefore, it is sufficient to study a bound on the queue length of backlogged workload along $\mathbf{c}^{(k)}$. This is called state-space collapse.

Define the following Lyapunov functions:

$$V(\mathbf{q}) \triangleq \|\mathbf{q}\|^2 = \sum_{i=1}^{L}\sum_{m=1}^{M} q_{mi}^2,\ W_{\perp}^{(k)}(\mathbf{q}) \triangleq \left\|\mathbf{q}_{\perp}^{(k)}\right\|,\ W_{||}^{(k)}(\mathbf{q}) \triangleq \left\|\mathbf{q}_{||}^{(k)}\right\|$$

$$V_{||}^{(k)}(\mathbf{q}) \triangleq \left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)} \right\rangle^2 = \left\|\mathbf{q}_{||}^{(k)}\right\|^2 = \frac{1}{L}\left(\sum_{i=1}^{L}\sum_{m=1}^{M} q_{mi}c_m\right)^2.$$

With a slight abuse of notation, we will use $V(\mathbf{Y})$ for $V(\mathbf{q}(\mathbf{Y}))$ and similarly with the other Lyapunov functions. Define the one step drift of the above Lyapunov functions for the original system $\mathbf{Y}(t)$ and for the sampled system $\widetilde{\mathbf{Y}}(n)$ respectively as follows:

$$\Delta V(\mathbf{Y}) \triangleq [V(\mathbf{q}(\mathbf{Y}(t+1))) - V(\mathbf{q}(\mathbf{Y}(t)))]\,\mathcal{I}(\mathbf{Y}(t) = \mathbf{Y})$$
$$\Delta \widetilde{V}(\widetilde{\mathbf{Y}}) \triangleq [V(\mathbf{q}(\mathbf{Y}((n+1)T))) - V(\mathbf{q}(\mathbf{Y}(nT)))]\,\mathcal{I}(\mathbf{Y}(nT) = \widetilde{\mathbf{Y}}).$$

Similarly define $\Delta W_{\perp}^{(k)}(\mathbf{Y})$, $\Delta V_{||}^{(k)}(\mathbf{Y})$, $\Delta \widetilde{W}_{\perp}^{(k)}(\widetilde{\mathbf{Y}})$ and $\Delta \widetilde{V}_{||}^{(k)}(\widetilde{\mathbf{Y}})$. We will show that the state space collapse happens along the direction of $\mathbf{c}^{(k)}$ for the sampled system $\widetilde{\mathbf{Y}}(n)$. We will need Lemma 3.5 by Hajek [33], which gives a bound on $\left\|\widetilde{\mathbf{q}}_{\perp}^{(k)}\right\|$ in steady-state if the drift of $\widetilde{W}_{\perp}^{(k)}(\widetilde{\mathbf{Y}})$ is negative. Here we use the following special case of Lemma 3.5, as presented in [37].

Lemma 5.2. *For an irreducible and aperiodic Markov chain $\{X[t]\}_{t\geq 0}$ over a countable state space $\mathcal{X}$, suppose $Z : \mathcal{X} \to \mathbb{R}_+$ is a nonnegative-valued Lyapunov function. We define the drift of Z at X as*

$$\Delta Z(X) \triangleq [Z(X[t+1]) - Z(X[t])]\, \mathcal{I}(X[t] = X),$$

where $\mathcal{I}(.)$ is the indicator function. Thus, $\Delta Z(X)$ is a random variable that measures the amount of change in the value of Z in one step, starting from state X. This drift is assumed to satisfy the following conditions:

1. *There exists an $\eta > 0$, and a $\kappa < \infty$ such that for all $X \in \mathcal{X}$ with $Z(X) \geq \kappa$,*

$$\mathbb{E}[\Delta Z(X) | X[t] = X] \leq -\eta.$$

2. *There exists a $D < \infty$ such that for all $X \in \mathcal{X}$,*

$$\mathbb{P}\left(|\Delta Z(X)| \leq D\right) = 1.$$

Then, there exists a $\theta^\star > 0$ and a $C^\star < \infty$ such that

$$\limsup_{t\to\infty} \mathbb{E}\left[e^{\theta^\star Z(X[t])}\right] \leq C^\star.$$

If we further assume that the Markov chain $\{X[t]\}_t$ is positive recurrent, then $Z(X[t])$ converges in distribution to a random variable $\bar{Z}$ for which

$$\mathbb{E}\left[e^{\theta^\star \bar{Z}}\right] \leq C^\star,$$

which directly implies that all moments of $\bar{Z}$ exist and are finite.

We will use this result for the Markov process $\widetilde{\mathbf{Y}}(n)$ and the Lyapunov function $\widetilde{W}_{\perp}^{(k)}(\widetilde{\mathbf{Y}})$. We will use the following lemma (similar to Lemma 7 in [37]) to bound the drift $\Delta\widetilde{W}_{\perp}^{(k)}(\widetilde{\mathbf{Y}})$ in terms of the drifts $\Delta\widetilde{V}(\widetilde{\mathbf{q}})$ and $\Delta\widetilde{V}_{||}^{(k)}(\widetilde{\mathbf{Y}})$. The proof follows from concavity of square-root function and using Pythagoras theorem. See [37] for details.

Lemma 5.3. *Let* $\widetilde{\mathbf{q}} \triangleq \mathbf{q}(\widetilde{\mathbf{Y}})$. *The drift of* $\widetilde{W}_{\perp}^{(k)}(\widetilde{\mathbf{Y}})$ *can be bounded as follows:*

$$\Delta\widetilde{W}_{\perp}^{(k)}(\widetilde{\mathbf{Y}}) \leq \frac{1}{2\left\|\widetilde{\mathbf{q}}_{\perp}^{(k)}\right\|}\left(\Delta\widetilde{V}(\widetilde{\mathbf{Y}}) - \Delta\widetilde{V}_{||}^{(k)}(\widetilde{\mathbf{Y}})\right) \quad \forall\ \widetilde{\mathbf{q}} \in \mathbb{R}_{+}^{ML}. \tag{5.9}$$

Note that

$$\begin{aligned}
&\mathbb{E}\left[\Delta\widetilde{V}(\widetilde{\mathbf{Y}})\middle|\,\mathbf{Y}(nT) = \widetilde{\mathbf{Y}}\right]\\
=&\mathbb{E}\left[V(\mathbf{Y}((n+1)T)) - V(\mathbf{Y}(nT))|\,\mathbf{Y}(nT) = \widetilde{\mathbf{Y}}\right]\\
=&\sum_{t=nT}^{nT+T-1}\mathbb{E}\left[V(\mathbf{Y}(t+1)) - V(\mathbf{Y}(t))|\,\mathbf{Y}(nT) = \widetilde{\mathbf{Y}}\right]\\
=&\sum_{t=nT}^{nT+T-1}\mathbb{E}\left[\mathbb{E}\left[V(\mathbf{Y}(t+1)) - V(\mathbf{Y}(t))|\,\mathbf{Y}(t) = \mathbf{Y}, \mathbf{Y}(nT) = \widetilde{\mathbf{Y}}\right]\middle|\,\mathbf{Y}(nT) = \widetilde{\mathbf{Y}}\right]\\
\overset{(a)}{=}&\sum_{t=nT}^{nT+T-1}\mathbb{E}\left[\mathbb{E}\left[\Delta V(\mathbf{Y})|\,\mathbf{Y}(t) = \mathbf{Y}\right]|\,\mathbf{Y}(nT) = \widetilde{\mathbf{Y}}\right],
\end{aligned}$$

where (a) follows from the fact that $\mathbf{Y}(t)$ is Markov. Similarly, one can bound the conditional drift of $\Delta\widetilde{V}_{||}(\widetilde{\mathbf{Y}})$. Then using these equalities in (5.9), we get

$$\begin{aligned}
&\mathbb{E}\left[\Delta\widetilde{W}_{\perp}^{(k)}(\widetilde{\mathbf{Y}})\middle|\,\mathbf{Y}(nT) = \widetilde{\mathbf{Y}}\right]\\
\leq&\frac{1}{2\left\|\widetilde{\mathbf{q}}_{\perp}^{(k)}\right\|}\left(\sum_{t=nT}^{nT+T-1}\mathbb{E}\left[\mathbb{E}\left[\Delta V(\mathbf{Y})|\,\mathbf{Y}(t) = \mathbf{Y}\right]\right.\right.\\
&\left.\left.-\mathbb{E}\left[\Delta V_{||}^{(k)}(\mathbf{Y})\middle|\,\mathbf{Y}(t) = \mathbf{Y}\right]\middle|\,\mathbf{Y}(nT) = \widetilde{\mathbf{Y}}\right]\right).
\end{aligned} \tag{5.10}$$

In the following, we will use $\mathbf{q}^{(\epsilon)}$ for $\mathbf{q}(\mathbf{Y}^{(\epsilon)})$ and $\widetilde{\mathbf{q}}^{(\epsilon)}$ for $\mathbf{q}(\widetilde{\mathbf{Y}}^{(\epsilon)})$. We will start by bounding the one step drift of $V_{||}^{(k)}$.

$$\begin{aligned}
&\mathbb{E}\left[\triangle\ V_{||}^{(k)}(\mathbf{Y}^{(\epsilon)})\middle|\,\mathbf{Y}^{(\epsilon)}(t) = \mathbf{Y}^{(\epsilon)}\right]\\
=&\mathbb{E}\left[V_{||}^{(k)}(\mathbf{q}^{(\epsilon)}(t+1)) - V_{||}^{(k)}(\mathbf{q}^{(\epsilon)}(t))\middle|\,\mathbf{Y}^{(\epsilon)}(t) = \mathbf{Y}^{(\epsilon)}\right]\\
=&\mathbb{E}\left[\left\langle\mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t+1)\right\rangle^2 - \left\langle\mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle^2\middle|\,\mathbf{Y}(t) = \mathbf{Y}^{(\epsilon)}\right]\\
=&\mathbb{E}\left[\left\langle\mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t) + \mathbf{a}^{(\epsilon)}(t) - \mathbf{s}^{(\epsilon)}(t) + \overline{\mathbf{u}}^{(\epsilon)}(t)\right\rangle^2 - \left\langle\mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle^2\middle|\,\mathbf{Y}(t) = \mathbf{Y}^{(\epsilon)}\right]\\
=&\mathbb{E}\left[\left\langle\mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t) + \mathbf{a}^{(\epsilon)}(t) - \mathbf{s}^{(\epsilon)}(t)\right\rangle^2 + \left\langle\mathbf{c}^{(k)}, \overline{\mathbf{u}}^{(\epsilon)}(t)\right\rangle^2 - \left\langle\mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle^2\right.
\end{aligned}$$

$$
\begin{aligned}
&+2\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)+\mathbf{a}^{(\epsilon)}(t)-\mathbf{s}^{(\epsilon)}(t)\right\rangle \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}^{(\epsilon)}(t)\right\rangle \Big| \mathbf{Y}(t)=\mathbf{Y}^{(\epsilon)}\Big] \\
\geq &\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{a}^{(\epsilon)}(t)-\mathbf{s}^{(\epsilon)}(t)\right\rangle^2 - 2\left\langle \mathbf{c}^{(k)}, \mathbf{s}^{(\epsilon)}(t)\right\rangle \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}^{(\epsilon)}(t)\right\rangle\right. \\
&\left.+2\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle \left\langle \mathbf{c}^{(k)}, \mathbf{a}^{(\epsilon)}(t)-\mathbf{s}^{(\epsilon)}(t)\right\rangle \Big| \mathbf{Y}(t)=\mathbf{Y}^{(\epsilon)}\right] \\
\geq &2\left\langle \mathbf{c}^{(k)}, \mathbf{q}(\mathbf{Y}^{(\epsilon)})\right\rangle \left(\left\langle \mathbf{c}^{(k)}, \mathbb{E}\left[\mathbf{a}^{(\epsilon)}(t)\big| \mathbf{Y}(t)=\mathbf{Y}^{(\epsilon)}\right] - \mathbb{E}\left[\mathbf{s}^{(\epsilon)}(t)\big| \mathbf{Y}(t)=\mathbf{Y}^{(\epsilon)}\right]\right\rangle\right) \\
&-2\left\langle \mathbf{c}^{(k)}, s_{max}\mathbf{1}\right\rangle^2 \\
= &\frac{2||\mathbf{q}_{||}^{(\epsilon,k)}||}{\sqrt{L}}\sum_{m=1}^{M} c_m \left(\sum_{i=1}^{L}\mathbb{E}\left[a_{mi}^{(\epsilon)}(t)|\mathbf{Y}(t)=\mathbf{Y}^{(\epsilon)}\right] - \sum_{i=1}^{L}\mathbb{E}\left[s_m^{i(\epsilon)}(t)|\mathbf{Y}(t)=\mathbf{Y}^{(\epsilon)}\right]\right) \\
&- K_{26} \\
= &\frac{2||\mathbf{q}_{||}^{(\epsilon,k)}||}{\sqrt{L}}\sum_{m=1}^{M} c_m \left(\check{\lambda}_m^{(\epsilon)} - \sum_{i=1}^{L}\mathbb{E}\left[s_m^{i(\epsilon)}(t)|\mathbf{Y}(t)=\mathbf{Y}^{(\epsilon)}\right]\right) - K_{26} && (5.11) \\
= &\frac{2||\mathbf{q}_{||}^{(\epsilon,k)}||}{\sqrt{L}}\sum_{m=1}^{M} c_m \left(\check{\lambda}_m^{(k)} - \epsilon^{(k)} c_m^{(k)} - \sum_{i=1}^{L}\mathbb{E}\left[s_m^{i(\epsilon)}(t)|\mathbf{Y}(t)=\mathbf{Y}^{(\epsilon)}\right]\right) - K_{26} && (5.12) \\
= &\frac{2||\mathbf{q}_{||}^{(\epsilon,k)}||}{\sqrt{L}}\sum_{m=1}^{M} c_m \left(\sum_{i=1}^{L}\check{\lambda}_m^{m(k)} - \sum_{i=1}^{L}\mathbb{E}\left[s_m^{m(\epsilon)}(t)|\mathbf{Y}(t)=\mathbf{Y}^{(\epsilon)}\right]\right) - K_{26} \\
&- \frac{2\epsilon^{(k)}}{\sqrt{L}}||\mathbf{q}_{||}^{(\epsilon,k)}|| && (5.13) \\
= &\frac{2||\mathbf{q}_{||}^{(\epsilon,k)}||}{\sqrt{L}}\sum_{i=1}^{L}\sum_{m=1}^{M} c_m \left(\check{\lambda}_m^{i(k)} - \mathbb{E}\left[s_m^{i(\epsilon)}(t)|\mathbf{Y}(t)=\mathbf{Y}^{(\epsilon)}\right]\right) - K_{26} - \frac{2\epsilon^{(k)}}{\sqrt{L}}||\mathbf{q}_{||}^{(\epsilon,k)}|| \\
\geq &- K_{26} - \frac{2\epsilon^{(k)}}{\sqrt{L}}||\mathbf{q}_{||}^{(\epsilon,k)}||, && (5.14)
\end{aligned}
$$

where $K_{26} = 2JMs_{max}^2$ and recall $\mathbf{q}_{||}^{(\epsilon,k)} \triangleq \mathbf{q}(\mathbf{Y}^{(\epsilon)})_{||}^{(k)}$. Equation (5.11) follows from the fact that the sum of arrival rates at each server is same as the external arrival rate. Equation (5.12) follows from (5.6). From the definition of $\mathcal{C}$, we have that there exists $\check{\lambda}^{i(k)} \in \mathcal{C}_i$ such that $\check{\lambda}^{(k)} = \sum_{i=1}^{L}\check{\lambda}^{i(k)}$. This gives (5.13). From Lemma 5.1, we have that for each i, there exists $b_i^{(k)}$ such that $\sum_{m=1}^{M} c_m\check{\lambda}_m^{i(k)} = b_i^{(k)}$ and $\left\langle c^{(k)}, s^{i(\epsilon)}\right\rangle \leq b_i^{(k)}$ for every $s^{i(\epsilon)}(t) \in \mathcal{C}_i$. Therefore, we have, for each i,

$$
\sum_{m=1}^{M} c_m \left(\check{\lambda}_m^{i(k)} - \mathbb{E}\left[s_m^{i(\epsilon)}(t)|\mathbf{Y}(t)=\mathbf{Y}^{(\epsilon)}\right]\right) \geq 0
$$

and so (5.14) is true.

Now, we will bound the one step drift of $V(.)$. By expanding the drift of $V(\mathbf{q}^{(\epsilon)})$ and using (5.3), it can be easily seen that

$$\mathbb{E}\left[\triangle\, V(\mathbf{Y}^{(\epsilon)})|\mathbf{Y}^{(\epsilon)}(t)=\mathbf{Y}^{(\epsilon)}\right] \leq K_{27}+\mathbb{E}_{\mathbf{Y}^{(\epsilon)}}\left[\sum_{i=1}^{L}\sum_{m=1}^{M}2q_{mi}^{(\epsilon)}\left(a_{mi}(t)-s_m^i(t)\right)\right], \tag{5.15}$$

where $K_{27}=M\left(\sum_m\left(\check{\lambda}_m^2+\sigma_m^2\right)+2Ms_{max}^2(1+D_{max})\right)$ and $\mathbb{E}_{\mathbf{Y}^{(\epsilon)}}[.]$ is shorthand for $\mathbb{E}[.|\mathbf{Y}^{(\epsilon)}(t)=\mathbf{Y}^{(\epsilon)}]$.

By definition of $a_{mi}(t)$, (5.2) we have

$$\begin{aligned}\mathbb{E}_{\mathbf{Y}^{(\epsilon)}}\left[\sum_{i=1}^{L}\sum_{m=1}^{M}2q_{mi}^{(\epsilon)}a_{mi}(t)\right] &= \mathbb{E}_{\mathbf{Y}^{(\epsilon)}}\left[\sum_{m=1}^{M}2q_{mi_m^*}^{(\epsilon)}a_m(t)\right]\\ &=\sum_{m=1}^{M}2q_{mi_m^*}^{(\epsilon)}\check{\lambda}_m^{(\epsilon)}.\end{aligned}$$

Then we have

$$\begin{aligned}&\mathbb{E}\left[\triangle\, V(\mathbf{Y}^{(\epsilon)})|\mathbf{Y}^{(\epsilon)}(t)=\mathbf{Y}^{(\epsilon)}\right]\\ \leq& K_{27}+2\sum_{m=1}^{M}\check{\lambda}_m^{(\epsilon)}q_{mi_m^*}^{(\epsilon)}-2\sum_{i=1}^{L}\mathbb{E}_{\mathbf{q}^{(\epsilon)}}\left[\sum_{m=1}^{M}q_{mi}^{(\epsilon)}s_m^i(t)\right]\\ =&K_{27}+2\sum_{m=1}^{M}\check{\lambda}_m^{(\epsilon)}q_{mi_m^*}^{(\epsilon)}-2\sum_{m=1}^{M}\check{\lambda}_m^{(\epsilon)}\sum_{i=1}^{L}\frac{q_{mi}^{(\epsilon)}}{L} \qquad (5.16)\\ &+2\sum_{m=1}^{M}\check{\lambda}_m^{(\epsilon)}\sum_{i=1}^{L}\frac{q_{mi}^{(\epsilon)}}{L}-2\sum_{i=1}^{L}\mathbb{E}_{\mathbf{q}^{(\epsilon)}}\left[\sum_{m=1}^{M}q_{mi}^{(\epsilon)}s_m^i(t)\right]. \qquad (5.17)\end{aligned}$$

We will bound the terms in (5.17).

$$\begin{aligned}&2\sum_{m=1}^{M}\check{\lambda}_m^{(\epsilon)}\sum_{i=1}^{L}\frac{q_{mi}^{(\epsilon)}}{L}-2\sum_{i=1}^{L}\mathbb{E}_{\mathbf{Y}^{(\epsilon)}}\left[\sum_{m=1}^{M}q_{mi}^{(\epsilon)}s_m^i(t)\right]\\ =&\sum_{m=1}^{M}2\left(\check{\lambda}_m^{(k)}-\epsilon^{(k)}c_m^{(k)}\right)\sum_{i=1}^{L}\frac{q_{mi}^{(\epsilon)}}{L}-2\sum_{i=1}^{L}\mathbb{E}_{\mathbf{Y}^{(\epsilon)}}\left[\sum_{m=1}^{M}q_{mi}^{(\epsilon)}s_m^i(t)\right]\\ =&-\frac{2\epsilon^{(k)}}{\sqrt{L}}||\mathbf{q}_{||}^{(\epsilon,k)}||+2\sum_{i=1}^{L}\mathbb{E}_{\mathbf{Y}^{(\epsilon)}}\left[\sum_{m=1}^{M}q_{mi}^{(\epsilon)}\left(\frac{\check{\lambda}_m^{(k)}}{L}-s_m^i(t)\right)\right]. \qquad (5.18)\end{aligned}$$

Substituting all the bounds, i.e., (5.14), (5.17), and (5.18) in (5.10), and using $\widetilde{\mathbf{q}}^{(\epsilon)}$ for $\mathbf{q}(\widetilde{\mathbf{Y}}^{(\epsilon)})$, we get

$$\mathbb{E}\left[\Delta\widetilde{W}_{\perp}^{(k)}(\widetilde{\mathbf{Y}}^{(\epsilon)})\middle|\mathbf{Y}^{(\epsilon)}(nT)=\widetilde{\mathbf{Y}}^{(\epsilon)}\right]$$

$$\leq\frac{1}{2\left\|\widetilde{\mathbf{q}}_{\perp}^{(\epsilon,k)}\right\|}\left(\sum_{t=nT}^{nT+T-1}\mathbb{E}\left[K_{28}+2\sum_{m=1}^{M}\check{\lambda}_m^{(\epsilon)}q_{mi_m^*(t)}^{(\epsilon)}(t)-2\sum_{m=1}^{M}\check{\lambda}_m^{(\epsilon)}\sum_{i=1}^{L}\frac{q_{mi}^{(\epsilon)}(t)}{L}\right.\right. \tag{5.19}$$

$$\left.\left.+2\sum_{i=1}^{L}\mathbb{E}_{\mathbf{Y}^{(\epsilon)}}\left[\sum_{m=1}^{M}q_{mi}^{(\epsilon)}\left(\frac{\check{\lambda}_m^{(k)}}{L}-s_m^i(t)\right)\right]\middle|\mathbf{Y}^{(\epsilon)}(nT)=\widetilde{\mathbf{Y}}^{(\epsilon)}\right]\right), \tag{5.20}$$

where $K_{28}=K_{26}+K_{27}$ and $i_m^*(t)$ denotes the server to which type m jobs were routed at time t. We will first bound the terms in (5.19). Note that from the queue evolution equation (5.1), we have that

$$|q_{mi}^{(\epsilon)}(t+1)-q_{mi}^{(\epsilon)}(t)|\leq a_{mi}(t)+s_{\max}$$

$$|q_{mi}^{(\epsilon)}(t)-q_{mi}^{(\epsilon)}(nT)|\leq\sum_{t=nT}^{nT+T-1}a_{mi}(t)+Ts_{\max}\text{ for }t\in\{nT+1,\ldots,nT+T-1\} \tag{5.21}$$

$$\begin{aligned}q_{mi_m^*(t)}^{(\epsilon)}(t)&\overset{(a)}{\leq}q_{mi_m^*(nT)}^{(\epsilon)}(t)\\&\leq q_{mi_m^*(nT)}^{(\epsilon)}(nT)+\sum_{t=nT}^{nT+T-1}a_{mi_m^*(nT)}(t)+Ts_{\max}\\\frac{1}{L}\sum_{i=1}^{L}q_{mi}^{(\epsilon)}(nT)&\leq\frac{1}{L}\sum_{i=1}^{L}\left(q_{mi}^{(\epsilon)}(t)+\sum_{t=nT}^{nT+T-1}a_{mi}(t)+Ts_{\max}\right),\end{aligned}$$

where (a) follows from the fact that the server $i_m^*(t)$ has the smallest workload at time t for type-m jobs. Using $\mathbb{E}_{\widetilde{\mathbf{Y}}^{(\epsilon)}}[.]$ for $\mathbb{E}[.|\mathbf{Y}^{(\epsilon)}(nT)=\widetilde{\mathbf{Y}}^{(\epsilon)}]$, we bound the terms in (5.19). We will *assume* that the arrival rate $\check{\lambda}^{(\epsilon)}$ is such that there exists a $\delta>0$ such that $\check{\lambda}_j^{(\epsilon)}>\delta$ for all m. This assumption is reasonable because we are interested in the limit when the arrival rate is on the boundary of the capacity region.

$$\sum_{t=nT}^{nT+T-1}2\sum_{m=1}^{M}\check{\lambda}_m^{(\epsilon)}\mathbb{E}_{\widetilde{\mathbf{Y}}^{(\epsilon)}}\left[q_{mi_m^*(t)}^{(\epsilon)}(t)-\sum_{i=1}^{L}\frac{q_{mi}^{(\epsilon)}(t)}{L}\right]$$

$$\leq \sum_{t=nT}^{nT+T-1} 2\sum_{m=1}^{M} \check{\lambda}_m^{(\epsilon)} \left(\mathbb{E}_{\widetilde{\mathbf{Y}}^{(\epsilon)}} \left[q_{mi_m^*(nT)}^{(\epsilon)}(nT) - \sum_{i=1}^{L} \frac{q_{mi}^{(\epsilon)}(nT)}{L} \right] + 2T\check{\lambda}_m^{(\epsilon)} + 2Ts_{\max} \right)$$

$$\leq K_{29} + \sum_{t=nT}^{nT+T-1} 2\sum_{m=1}^{M} \check{\lambda}_m^{(\epsilon)} \left(\widetilde{q}_{mi_m^*}^{(\epsilon)} - \sum_{i=1}^{L} \frac{\widetilde{q}_{mi}^{(\epsilon)}}{L} \right)$$

$$= K_{29} - 2T\sum_{m=1}^{M} \check{\lambda}_m^{(\epsilon)} \left(\sum_{i=1}^{L} \left(\frac{\widetilde{q}_{mi}^{(\epsilon)}}{L} - \frac{\widetilde{q}_{mi_m^*}^{(\epsilon)}}{L} \right) \right)$$

$$= K_{29} - \frac{2T}{L}\sum_{m=1}^{M} \check{\lambda}_m^{(\epsilon)} \left(\sum_{i=1}^{L} \left| \widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi_m^*}^{(\epsilon)} \right| \right)$$

$$\leq K_{29} - \frac{2T}{L}\sum_{m=1}^{M} \check{\lambda}_m^{(\epsilon)} \left(\sqrt{\sum_{i=1}^{L} \left(\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi_m^*}^{(\epsilon)} \right)^2} \right) \tag{5.22}$$

$$\leq K_{29} - \frac{2T}{L}\sum_{m=1}^{M} \check{\lambda}_m^{(\epsilon)} \left(\sqrt{\sum_{i=1}^{L} \left(\widetilde{q}_{mi}^{(\epsilon)} - \frac{1}{M}\sum_{i'=1}^{L} \widetilde{q}_{mi'}^{(\epsilon)} \right)^2} \right) \tag{5.23}$$

$$\leq K_{29} - \frac{2T\delta}{L}\sum_{m=1}^{M} \left(\sqrt{\sum_{i=1}^{L} \left(\widetilde{q}_{mi}^{(\epsilon)} - \frac{1}{M}\sum_{i'=1}^{L} \widetilde{q}_{mi'}^{(\epsilon)} \right)^2} \right) \tag{5.24}$$

$$= K_{29} - \frac{2T\delta}{L}\sum_{m=1}^{M} \sqrt{\sum_{i=1}^{L} \left(\widetilde{q}_{mi}^{(\epsilon)} \right)^2 - \frac{1}{M} \left(\sum_{i'=1}^{L} \widetilde{q}_{mi'}^{(\epsilon)} \right)^2}$$

$$\leq K_{29} - \frac{2T\delta}{L} \sqrt{\sum_{m=1}^{M}\sum_{i=1}^{L} \left(\widetilde{q}_{mi}^{(\epsilon)} \right)^2 - \frac{1}{M}\sum_{m=1}^{M} \left(\sum_{i'=1}^{L} \widetilde{q}_{mi'}^{(\epsilon)} \right)^2}, \tag{5.25}$$

where $K_{29} = 4T^2 \sum_{m=1}^{M} (\check{\lambda}_m^{(\epsilon)})^2 + 4T^2 s_{\max} \sum_{m=1}^{M} \check{\lambda}_m^{(\epsilon)}$. Equation (5.22) follows from the fact that the ℓ_1 norm of a vector is no more than its ℓ_2 norm for a vector in $\mathbb{R}^L$. The minimum mean square constant estimator of a vector is its empirical mean. In other words, for a vector x in $\mathbb{R}^L$, the convex function $\sqrt{\sum_i (x_i - y)^2}$ is minimized for $y = \frac{1}{L}\sum_i x_i$. This gives (5.23). Equation (5.24) follows from the assumption that $\check{\lambda}_m^{(\epsilon)} > \delta$. Equation (5.25) follows from the observation that $(\sum_m \sqrt{x_m})^2 \geq \sum_m x_m$.

We will now bound the terms in (5.20). To do this, we will first show that the sum $\sum_{m=1}^{J} q_{mi}^{(\epsilon)}(t) s_m^i(t)$ does not change by much with in T time-slots.

For any $t \in \{nT+1, \dots, (n+1)T\}$, we have

$$\begin{aligned}
&\sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t) s_m^i(t-1) \\
=&\sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t) s_m^i(t^-) + \sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t)(s_m^i(t-1) - s_m^i(t^-)) \\
\overset{(a)}{\leq}&\sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t) s_m^i(t^-) + \sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t)\overline{s_m^i}(t) \\
=&\left(\sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t) s_m^i(t^-) + \sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t)\overline{s_m^i}(t)\right)\mathcal{I}_{q_{mi}^{(\epsilon)}(t)\geq D_{\max}s_{\max}} \\
&\quad + \left(\sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t) s_m^i(t^-) + \sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t)\overline{s_m^i}(t)\right)\mathcal{I}_{q_{mi}^{(\epsilon)}(t)< D_{\max}s_{\max}} \\
\overset{(b)}{\leq}&\sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t) s_m^i(t) + MD_{\max}s_{\max}^2,
\end{aligned}$$

where Inequality (a) follows from the scheduling algorithm and definition of $\overline{s_m^i}(t)$. Inequality (b) holds because when $q_{mi}^{(\epsilon)}(t) \geq D_{\max}s_{\max}$, there are enough number of jobs so that there is no unused service. From (5.1), we get

$$\begin{aligned}
\sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t-1) s_m^i(t-1) \overset{(c)}{\leq}& Ms_{\max}^2(D_{\max}+1) + \sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t) s_m^i(t) \\
-\sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t) s_m^i(t) \leq& K_{30} - \sum_{m=1}^{M} q_{mi}^{(\epsilon)}(nT) s_m^i(nT)
\end{aligned}$$

where $K_{30} = T\left(Ms_{\max}^2(D_{\max}+1)\right)$. Repeatedly applying Inequality (c), we get the last relation. We now use this relation along with (5.21) to bound the terms in (5.20).

$$\begin{aligned}
&2\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\sum_{i=1}^{L} \mathbb{E}_{\mathbf{Y}^{(\epsilon)}}\left[\sum_{m=1}^{M} q_{mi}^{(\epsilon)}\left(\frac{\check{\lambda}_m^{(k)}}{L} - s_m^i(t)\right)\right]\middle| \mathbf{Y}^{(\epsilon)}(nT) = \widetilde{\mathbf{Y}}^{(\epsilon)}\right] \\
=&2\sum_{t=nT}^{nT+T-1}\sum_{i=1}^{L} \mathbb{E}_{\widetilde{\mathbf{Y}}^{(\epsilon)}}\left[\sum_{m=1}^{M} q_{mi}^{(\epsilon)}(t)\left(\frac{\check{\lambda}_m^{(k)}}{L} - s_m^i(t)\right)\right] \\
\leq&2\sum_{t=nT}^{nT+T-1}\sum_{i=1}^{L} \mathbb{E}_{\widetilde{\mathbf{Y}}^{(\epsilon)}}\left[\sum_{m=1}^{M} q_{mi}^{(\epsilon)}(nT)\left(\frac{\check{\lambda}_m^{(k)}}{L} - s_m^i(nT)\right)\right] + 2T^2\sum_{m=1}^{M}\check{\lambda}_m^{(k)}\check{\lambda}_m^{(\epsilon)}
\end{aligned}$$

$$
\begin{aligned}
&+2TLK_{30}\\
=&2T\sum_{i=1}^{L}\mathbb{E}_{\widetilde{\mathbf{Y}}^{(\epsilon)}}\left[\sum_{m=1}^{M}q_{mi}^{(\epsilon)}(nT)\left(\frac{\check{\lambda}_m^{(k)}}{L}-s_m^i(nT)\right)\right]+2T^2\sum_{m=1}^{M}\check{\lambda}_m^{(k)}\check{\lambda}_m^{(\epsilon)}+2TLK_{30}\\
\overset{(a)}{=}&K_{31}+2T\sum_{i=1}^{L}\left[\min_{r^i\in\mathcal{C}_i}\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}\left(\frac{\check{\lambda}_m^{(k)}}{L}-r_m^i\right)\right],
\end{aligned}
\tag{5.26}
$$

where $K_{31} = 2T^2\sum_{m=1}^{M}\check{\lambda}_m^{(k)}\check{\lambda}_m^{(\epsilon)} + 2TLK_{30} + +2TMLD_{max}s_{max}^2$. Equation (a) is true because of MaxWeight scheduling. Note that in Algorithm 9, the actual service allocated to jobs of type m at server i is same as that of the MaxWeight schedule as long as the corresponding queue length of backlogged workload is greater than $D_{max}s_{max}$. This gives the additional $2MLD_{max}s_{max}^2$ term.

Assuming *all the servers are identical*, we have that for each i, $\mathcal{C}_i = \{\check{\lambda}/L : \check{\lambda}\in\mathcal{C}\}$, so $\mathcal{C}_i$ is a scaled version of $\mathcal{C}$. Thus, $\check{\lambda}^i = \check{\lambda}/L$. Since $k\in\mathcal{K}^o_{\check{\lambda}^{(\epsilon)}}$, we also have that $k\in\mathcal{K}^o_{\check{\lambda}^{i(\epsilon)}}$ for the capacity region $\mathcal{C}_i$. Thus, there exists $\delta^{(k)}>0$ so that

$$
\mathcal{B}^{(k)}_{\delta^{(k)}} \triangleq \mathcal{H}^{(k)}\cap\{r\in\mathbb{R}^M_+ : ||r-\check{\lambda}^{(k)}/L||\le\delta^{(k)}\}
$$

lies strictly within the face of $\mathcal{C}_i$ that corresponds to $\mathcal{F}^{(k)}$. (Note that this is the only instance in the proof of Theorem 5.2 in which we use the assumption that all the servers are identical.) Call this face $\mathcal{F}_i^{(k)}$. Thus we have

$$
\begin{aligned}
&\sum_{i=1}^{L}\left[\min_{r^i\in\mathcal{C}_i}\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}\left(\frac{\check{\lambda}_m^{(k)}}{L}-r_m^i\right)\right]\\
\le&\sum_{i=1}^{L}\left[\min_{r^i\in\mathcal{B}^{(k)}_{\delta^{(k)}}}\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}\left(\frac{\check{\lambda}_m^{(k)}}{L}-r_m^i\right)\right] &(5.27)\\
=&\sum_{i=1}^{L}\left[\min_{r^i\in\mathcal{B}^{(k)}_{\delta^{(k)}}}\sum_{m=1}^{M}\left(\widetilde{q}_{mi}^{(\epsilon)}-\left(\sum_{m'=1}^{M}\widetilde{q}_{m'i}^{(\epsilon)}c_{m'}\right)c_m\right)\left(\frac{\check{\lambda}_m^{(k)}}{L}-r_m^i\right)\right] &(5.28)\\
=&-\delta^{(k)}\sum_{i=1}^{L}\sqrt{\sum_{m=1}^{M}\left(\widetilde{q}_{mi}^{(\epsilon)}-\left(\sum_{m'=1}^{M}\widetilde{q}_{m'i}^{(\epsilon)}c_{m'}\right)c_m\right)^2} &(5.29)\\
=&-\delta^{(k)}\sum_{i=1}^{L}\sqrt{\sum_{m=1}^{M}\left(\widetilde{q}_{mi}^{(\epsilon)}\right)^2-\left(\sum_{m'=1}^{M}\widetilde{q}_{m'i}^{(\epsilon)}c_{m'}\right)^2} &(5.30)
\end{aligned}
$$

$$\leq -\,\delta^{(k)}\sqrt{\sum_{i=1}^{L}\sum_{m=1}^{M}\left(\widetilde{q}_{mi}^{(\epsilon)}\right)^2-\sum_{i=1}^{L}\left(\sum_{m'=1}^{M}\widetilde{q}_{m'i}^{(\epsilon)}c_{m'}\right)^2}. \tag{5.31}$$

Equation (5.28) is true because c is a vector perpendicular to the face $\mathcal{F}_i^{(k)}$ of $\mathcal{C}_i$ whereas both $\check{\lambda}^{(k)}/L$ and r^i lie on the face $\mathcal{F}_i^{(k)}$. So, $\left(\sum_{m'=1}^{M}\widetilde{q}_{m'i}^{(\epsilon)}c_{m'}\right)\sum_{m=1}^{M}c_m\left(\frac{\check{\lambda}_m^{(k)}}{L}-r_m^i\right)=0$. The vector $\widetilde{q}_i^{(\epsilon)}\triangleq\left(\widetilde{q}_{mi}^{(\epsilon)}\right)_m$ is in $\mathbb{R}^M$. Its component along $c\in\mathbb{R}^M$ is $\widetilde{q}_{i||}^{(\epsilon)}=\left(\left\|\widetilde{q}_{i||}^{(\epsilon)}\right\|c_m\right)_m$ where $\left\|\widetilde{q}_{i||}^{(\epsilon)}\right\|=\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}c_m$. Then, the component perpendicular to c is $\widetilde{q}_{i\perp}^{(\epsilon)}=\left(\widetilde{q}_{mi}^{(\epsilon)}-\left(\sum_{m'=1}^{M}\widetilde{q}_{m'i}^{(\epsilon)}c_{m'}\right)c_m\right)_m$ and the term in (5.28) is $\sum_m(\widetilde{q}_{i\perp}^{(\epsilon)})_m\left(\frac{\check{\lambda}_m^{(k)}}{L}-r_m^i\right)$. This term is an inner product in $\mathbb{R}^M$ which is minimized when r^i is chosen to be on the boundary of $\mathcal{B}_{\delta^{(k)}}^{(k)}$ so that $\left(\frac{\check{\lambda}_m^{(k)}}{L}-r_m^i\right)_m$ points in the opposite direction to $\widetilde{q}_{i\perp}^{(\epsilon)}$ and the minimum value is $-\delta^{(k)}\|\widetilde{q}_{i\perp}^{(\epsilon)}\|$. This gives (5.29). Equation (5.30) can be obtained either by expanding or by using Pythagorean theorem, viz., $\|\widetilde{q}_{i\perp}^{(\epsilon)}\|^2=\|\widetilde{q}_{i}^{(\epsilon)}\|^2-\|\widetilde{q}_{i||}^{(\epsilon)}\|^2$. Similar to (5.25), since $(\sum_i\sqrt{x_i})^2\geq\sum_i x_i$, we get (5.31).

Now substituting (5.25) and (5.31) in (5.20), we get

$$\mathbb{E}\left[\Delta\widetilde{W}_{\perp}^{(k)}(\widetilde{\mathbf{Y}}^{(\epsilon)})\middle|\,\mathbf{Y}^{(\epsilon)}(nT)=\widetilde{\mathbf{Y}}^{(\epsilon)}\right]$$

$$\leq\frac{1}{2\left\|\widetilde{\mathbf{q}}_{\perp}^{(\epsilon,k)}\right\|}\left(K_{28}T+K_{29}-\frac{2T\delta}{L}\sqrt{\sum_{m=1}^{M}\sum_{i=1}^{L}\left(\widetilde{q}_{mi}^{(\epsilon)}\right)^2-\frac{1}{L}\sum_{m=1}^{M}\left(\sum_{i'=1}^{L}\widetilde{q}_{mi'}^{(\epsilon)}\right)^2}\right.$$

$$\left.+K_{31}-2T\delta^{(k)}\sqrt{\sum_{i=1}^{L}\sum_{m=1}^{M}\left(\widetilde{q}_{mi}^{(\epsilon)}\right)^2-\sum_{i=1}^{L}\left(\sum_{m'=1}^{M}\widetilde{q}_{m'i}^{(\epsilon)}c_{m'}\right)^2}\right) \tag{5.32}$$

$$\overset{(a)}{\leq}\frac{K_{32}}{2\left\|\widetilde{\mathbf{q}}_{\perp}^{(\epsilon,k)}\right\|}-\frac{T\delta'}{\left\|\widetilde{\mathbf{q}}_{\perp}^{(\epsilon,k)}\right\|}\left[\sum_{m=1}^{M}\sum_{i=1}^{L}\left(\widetilde{q}_{mi}^{(\epsilon)}\right)^2-\frac{1}{L}\sum_{m=1}^{M}\left(\sum_{i=1}^{L}\widetilde{q}_{mi}^{(\epsilon)}\right)^2\right.$$

$$\left.+\sum_{i=1}^{L}\sum_{m=1}^{M}\left(\widetilde{q}_{mi}^{(\epsilon)}\right)^2-\sum_{i=1}^{L}\left(\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}c_m\right)^2\right]^{\frac{1}{2}}$$

$$\overset{(b)}{\leq}\frac{K_{32}}{2\left\|\widetilde{\mathbf{q}}_{\perp}^{(\epsilon,k)}\right\|}-\frac{T\delta'}{\left\|\widetilde{\mathbf{q}}_{\perp}^{(\epsilon,k)}\right\|}\sqrt{\sum_{m=1}^{M}\sum_{i=1}^{L}\left(\widetilde{q}_{mi}^{(\epsilon)}\right)^2-\left(\sum_{i=1}^{L}\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}\frac{c_m}{\sqrt{L}}\right)^2}$$

$$
\begin{aligned}
=&\frac{K_{32}}{2\left\|\widetilde{\mathbf{q}}_{\perp}^{(\epsilon,k)}\right\|}-\frac{T\delta'}{\left\|\widetilde{\mathbf{q}}_{\perp}^{(\epsilon,k)}\right\|}\sqrt{\|\widetilde{\mathbf{q}}^{(\epsilon)}\|^2-\left\|\widetilde{\mathbf{q}}_{\|}^{(\epsilon,k)}\right\|^2}\\
=&\frac{K_{32}}{2\left\|\widetilde{\mathbf{q}}_{\perp}^{(\epsilon,k)}\right\|}-T\delta'\\
\leq&\frac{-T\delta'}{2}\text{ whenever }\left(W_{\perp}^{(k)}(\mathbf{q}(\widetilde{\mathbf{Y}}^{(\epsilon)}))=\left\|\widetilde{\mathbf{q}}_{\perp}^{(\epsilon,k)}\right\|\geq\frac{K_{32}}{T\delta'}\right),
\end{aligned}
\tag{5.33}
$$

where $K_{32}=K_{28}T+K_{29}+K_{31}$ and $\delta'=\min\{\frac{\delta}{M},\delta^{(k)}\}$. Inequality (a) follows from the fact that $(\sqrt{x}+\sqrt{y})^2\geq x+y$. Inequality (b) follows from the following claim, which is proved in C.

Claim 5.1. *For any $\widetilde{\mathbf{q}}\in\mathbb{R}^{ML}$,*

$$
-\frac{1}{L}\sum_{m=1}^{M}\left(\sum_{i=1}^{L}\widetilde{q}_{mi}^{(\epsilon)}\right)^2+\sum_{i=1}^{L}\sum_{m=1}^{M}\left(\widetilde{q}_{mi}^{(\epsilon)}\right)^2-\sum_{i=1}^{L}\left(\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}c_m\right)^2
\geq-\frac{1}{L}\left(\sum_{i=1}^{L}\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}c_m\right)^2.
$$

In addition to (5.33), since the departures in each time slot are bounded and the arrivals are finite there is a $D<\infty$ such that $\mathbb{P}\left(|\Delta Z(X)|\leq D\right)$ almost surely. Now, applying Lemma 5.2, we have the following proposition.

Proposition 5.1. *Assume all the servers are identical and the arrival rate $\check{\lambda}^{(\epsilon)}\in int(\mathcal{C})$ is such that there exists a $\check{\lambda}_m^{(\epsilon)}>\delta$ for all m for some $\delta>0$. Consider the sampled system $\widetilde{\mathbf{Y}}^{(\epsilon)}(n)$ under JSQ routing and myopic MaxWeight scheduling according to Algorithm 9. For every $k\in\mathcal{K}^o_{\check{\lambda}^{(\epsilon)}}$, there exists a set of finite constants $\{N_r^{(k)}\}_{r=1,2,...}$ such that in steady state, $\mathbb{E}\left[\left\|\mathbf{q}_{\perp}^{(k)}(\widetilde{\mathbf{Y}}^{(\epsilon)}(n))\right\|^r\right]\leq N_r^{(k)}$ for all $\epsilon>0$ and for each $r=1,2,....$*

As in [46, 37], note that $k\in\mathcal{K}^o_{\check{\lambda}^{(\epsilon)}}$ is an important assumption here. This is called the 'Complete Resource Pooling' assumption and was used in [46, 50, 51]. If $k\in\mathcal{K}\setminus\mathcal{K}^o_{\check{\lambda}^{(\epsilon)}}$, i.e., if the arrival rate approaches a corner point of the capacity region as $\epsilon^{(k)}\to 0$, then there is no constant $\delta^{(k)}$ so that $\mathcal{B}_{\delta^{(k)}}^{(k)}$ lies in the face $\mathcal{F}^{(k)}$. In other words, the $\delta^{(k)}$ depends on $\epsilon^{(k)}$ and so the bound obtained by Lemma 5.2 also depends on $\epsilon^{(k)}$.

Remark: As stated in Theorem 5.2, our results hold only for the case of identical servers, which is the most practical scenario. However, we have written the proofs more generally whenever we can so that it is clear where we need the identical server assumption. In particular, in this subsection, up to Equation (5.3.2), we do not need this assumption, but we have used the assumption after that, in analyzing the drift of $V(\mathbf{q})$. The upper bound in the next section is valid more generally if one can establish state-space collapse for the non-identical server case. However, at this time, this is an open problem.

5.3.3 Upper Bound

In this section, we will obtain an upper bound on the steady state weighted queue length of backlogged workload, $\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle\right]$ and show that in the asymptotic limit as $\epsilon^{(k)} \downarrow 0$, this coincides with the lower bound. For ease of exposition, we will omit the superscript $^{(\epsilon)}$ in this section. In order to obtain a matching upper bound, we consider the drift of the same Lyapunov function that was used in the lower bound, viz., $V_{||}^{(k)}(.)$. As a result, we have the following lemma.

Lemma 5.4. *In steady state,*

$$\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) - \mathbf{a}(t)\right\rangle\right] \tag{5.34}$$

$$= \sum_{t=nT}^{nT+T-1} \frac{\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) - \mathbf{a}(t)\right\rangle^2\right]}{2} + \sum_{t=nT}^{nT+T-1} \frac{\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle^2\right]}{2} \tag{5.35}$$

$$+ \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t) + \mathbf{a}(t) - \overline{\mathbf{s}}(t)\right\rangle \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle\right]. \tag{5.36}$$

Proof. First we expand the drift of the Lyapunov function $V_{||}^{(k)}$ for the sampled system $\widetilde{\mathbf{Y}}(n)$, i.e., $\Delta\widetilde{V}_{||}^{(k)}(\widetilde{\mathbf{Y}})$.

$$\begin{aligned}
&\Delta\widetilde{V}^{(k)}(\mathbf{Y}(nT)) \\
&= \left[V_{||}^{(k)}(\mathbf{q}(\mathbf{Y}((n+1)T))) - V_{||}^{(k)}(\mathbf{q}(\mathbf{Y}(nT)))\right] \\
&= \sum_{t=nT}^{nT+T-1} \left[V_{||}^{(k)}(\mathbf{q}(t+1) - V_{||}^{(k)}(\mathbf{q}(t))\right]
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{t=nT}^{nT+T-1} \left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t+1)\right\rangle^2 - \left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle^2\right] \\
&\overset{(a)}{=} \sum_{t=nT}^{nT+T-1} \left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)+\mathbf{a}(t)-\overline{\mathbf{s}}(t)+\overline{\mathbf{u}}(t)\right\rangle^2 - \left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle^2\right] \\
&= \sum_{t=nT}^{nT+T-1} \left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)+\mathbf{a}(t)-\overline{\mathbf{s}}(t)\right\rangle^2 + 2\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)+\mathbf{a}(t)-\overline{\mathbf{s}}(t)\right\rangle \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle \right. \\
&\qquad\qquad \left. + \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle^2 - \left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle^2\right] \\
&= \sum_{t=nT}^{nT+T-1} \left[\left\langle \mathbf{c}^{(k)}, \mathbf{a}(t)-\overline{\mathbf{s}}(t)\right\rangle^2 + 2\left\langle \mathbf{c}^{(k)}, \mathbf{a}(t)-\overline{\mathbf{s}}(t)\right\rangle \left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle \right. \\
&\qquad\qquad \left. + 2\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)+\mathbf{a}(t)-\overline{\mathbf{s}}(t)\right\rangle \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle + \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle^2\right].
\end{aligned}
$$

Equation (a) follows from (5.1). Noting that the expected drift of $\Delta V_{||}^{(k)}$ is zero in steady state, we have the lemma.

□

Note that the expectation in the lemma is according to the steady state distribution of the process $\widehat{\mathbf{Y}}(n)$, i.e., at time t, the queue length distribution is $\pi_\tau(\mathbf{q})$ where $\tau = t \mod T$.

We will obtain an upper bound on $\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle\right]$ by bounding each of the above terms. Before that, we need the following definitions and results.

Let $\pi_\tau^{(k)}$ be the steady-state probability that the MaxWeight schedule chosen is from the face $\mathcal{F}^{(k)}$ at a time t such that $t \mod T = \tau$, i.e.,

$$\pi_\tau^{(k)} = \mathbb{P}\left(\langle c, \overline{s}(t)\rangle = b^{(k)}\right) \text{ whenever } t \mod t = \tau,$$

where $\overline{s}_m = \sum_{i=1}^{L} \overline{s_m^i}$ as defined in (5.4). Let $\pi^{(k)}$ denote $\frac{1}{T}\sum_{\tau=0}^{T} \pi_\tau^{(k)}$. Also, define

$$\gamma^{(k)} = \min\left\{b^{(k)} - \langle c, r\rangle : r \in \mathcal{S} \setminus \mathcal{F}^{(k)}\right\}.$$

Then we have the following Claim.

Claim 5.2. *For any $\epsilon^{(k)} \in \left(0, \gamma^{(k)}\right)$, Then, note that*

$$\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t)\right\rangle\right)^2\right] \leq \frac{T\epsilon^{(k)}}{L\gamma^{(k)}}\left(\left(b^{(k)}\right)^2 + \langle c, s_{max}1\rangle^2\right). \quad (5.37)$$

Proof. We will first show that

$$\left(1-\pi^{(k)}\right) \le \frac{\epsilon^{(k)}}{\gamma^{(k)}}.$$

From the stability of the system, we have that in steady state,

$$\begin{aligned}
\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left\langle c^{(k)}, \overline{s(q(t))}\right\rangle\right] &\ge T\left\langle c^{(k)}, \check{\lambda}^{\epsilon}\right\rangle \\
&= T(b^{(k)}-\epsilon^{(k)})
\end{aligned}$$

$$\begin{aligned}
&\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left\langle c^{(k)}, \overline{s(q(t))}\right\rangle \mathcal{I}\left(\langle c, \overline{s}(t)\rangle = b^{(k)}\right)\right] \\
&\quad + \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left\langle c^{(k)}, \overline{s(q(t))}\right\rangle \mathcal{I}\left(\langle c, \overline{s}(t)\rangle \ne b^{(k)}\right)\right] \ge T(b^{(k)}-\epsilon^{(k)}) \\
&\quad \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left\langle c^{(k)}, \overline{s(q(t))}\right\rangle \mathcal{I}\left(\langle c, \overline{s}(t)\rangle \ne b^{(k)}\right)\right] \ge T(b^{(k)}-\epsilon^{(k)}) - Tb^{(k)}\pi^{(k)}.
\end{aligned}$$

Also note that

$$\begin{aligned}
&\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left\langle c^{(k)}, \overline{s(q(t))}\right\rangle \mathcal{I}\left(\langle c, \overline{s}(t)\rangle \ne b^{(k)}\right)\right] \\
&\le \sum_{t=nT}^{nT+T-1} (b^{(k)}-\gamma^{(k)})(1-\pi^{(k)}_{t \bmod T}) \\
&\le T(b^{(k)}-\gamma^{(k)})(1-\pi^{(k)}).
\end{aligned}$$

Combining the lower and upper bounds, we get $\left(1-\pi^{(k)}\right) \le \frac{\epsilon^{(k)}}{\gamma^{(k)}}$.

Then, note that

$$\begin{aligned}
&\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t)\right\rangle\right)^2\right] \\
=&\frac{1}{L}\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(b^{(k)} - \langle c, \overline{s}(t)\rangle\right)^2\right] \\
=&\frac{1}{L}\sum_{t=nT}^{nT+T-1} \left(1-\pi^{(k)}_{t \bmod T}\right) \mathbb{E}\left[\left(b^{(k)} - \langle c, \overline{s}(t)\rangle\right)^2 \mid \left(\langle c, \overline{s}(t)\rangle \ne b^{(k)}\right)\right] \\
\le&\frac{T}{L}(1-\pi^{(k)})\left(\left(b^{(k)}\right)^2 + \langle c, s_{max}1\rangle^2\right)
\end{aligned}$$

$$\leq \frac{T\epsilon^{(k)}}{L\gamma^{(k)}} \left(\left(b^{(k)}\right)^2 + \langle c, s_{max} 1 \rangle^2 \right).$$

$\square$

Define $\widetilde{\mathcal{C}_i} \subseteq \mathbb{R}_+^{JM}$ as $\widetilde{\mathcal{C}_i} = \mathcal{C}_1 \times ... \times \mathcal{C}_M$. Then, $\widetilde{\mathcal{C}_i}$ is a convex polygon.

Claim 5.3. *Let $q^i \in \mathbb{R}_+^M$ for each $i \in \{1, 2, ...L\}$. Denote $\mathbf{q} = (q^i)_{i=1}^L \in \mathbb{R}_+^{ML}$. If, for each i, $(s^i)^*$ is a solution of $\max\limits_{s \in \mathcal{C}_i} \langle q^i, s \rangle$ then $\mathbf{s}^* = ((s^i)^*)_i$ is a solution of $\max\limits_{\mathbf{s} \in \widetilde{\mathcal{C}_i}} \langle \mathbf{q}, \mathbf{s} \rangle$.*

Proof. Since $\mathbf{s}^* \in \widetilde{\mathcal{C}_i}$, $\langle \mathbf{q}, \mathbf{s}^* \rangle \leq \max\limits_{\mathbf{s} \in \widetilde{\mathcal{C}_i}} \langle \mathbf{q}, \mathbf{s} \rangle$. Note that $\max\limits_{\mathbf{s} \in \widetilde{\mathcal{C}_i}} \langle \mathbf{q}, \mathbf{s} \rangle = \sum\limits_{i=1}^L \max\limits_{s^i \in \mathcal{C}_i} \langle q^i, s^i \rangle$. Therefore, if $\langle \mathbf{q}, \mathbf{s}^* \rangle < \max\limits_{\mathbf{s} \in \widetilde{\mathcal{C}_i}} \langle \mathbf{q}, \mathbf{s} \rangle$, we have $\sum\limits_{i=1}^L \left\langle q^i, (s^i)^* \right\rangle < \sum\limits_{i=1}^L \max\limits_{s^i \in \mathcal{C}_i} \langle q^i, s^i \rangle$. Then there exists an $i \leq L$ such that $\left\langle q^i, (s^i)^* \right\rangle < \max\limits_{s^i \in \mathcal{C}_i} \langle q^i, s^i \rangle$, which is a contradiction. $\square$

Therefore, choosing a MaxWeight schedule at each server is the same as choosing a MaxWeight schedule from the convex polygon, $\widetilde{\mathcal{C}_i}$. Since there are a finite number of feasible schedules, given $\mathbf{c}^{(k)} \in \mathbb{R}_+^{ML}$ such that $||\mathbf{c}^{(k)}|| = 1$, there exists an angle $\theta^{(k)} \in (0, \frac{\pi}{2}]$ such that, for all $\mathbf{q} \in \left\{ \mathbf{q} \in \mathbb{R}_+^{ML} : ||\mathbf{q}_{||}^{(k)}|| \geq ||\mathbf{q}|| \cos\left(\theta^{(k)}\right) \right\}$ (i.e., for all $\mathbf{q} \in \mathbb{R}_+^{ML}$ such that $\theta_{\mathbf{q}\mathbf{q}_{||}^{(k)}} \leq \theta^{(k)}$ where $\theta_{\mathbf{ab}}$ represents the angle between vectors $\mathbf{a}$ and $\mathbf{b}$), we have

$$\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \right\rangle \mathcal{I}\left(\mathbf{q}(t) = \mathbf{q}\right) = b^{(k)} / \sqrt{L} \mathcal{I}\left(\mathbf{q}(t) = \mathbf{q}\right).$$

We can bound the unused service at each time t in steady-state as follows.

$$\begin{aligned}
\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t) \right\rangle\right] &\leq \mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) - \mathbf{a}(t) \right\rangle\right] \\
&= \frac{1}{\sqrt{L}} \left(\mathbb{E}\left[\left\langle c^{(k)}, \overline{s}(t) \right\rangle\right] - \left\langle c^{(k)}, \check{\lambda}^{\epsilon} \right\rangle \right) \\
&= \frac{1}{\sqrt{L}} \left(\mathbb{E}\left[\left\langle c^{(k)}, \overline{s}(t) \right\rangle\right] - \left(b^{(k)} - \epsilon^{(k)} \right) \right) \\
&\leq \frac{\epsilon^{(k)}}{\sqrt{L}},
\end{aligned} \tag{5.38}$$

where the last inequality follows from the fact that the MaxWeight schedule lies inside the capacity region and so $\mathbb{E}\left[\left\langle c^{(k)}, \overline{s}(t) \right\rangle\right] \leq b^{(k)}$.

We will also need the following bound. Since the change in workload between T time-slots is bounded as in (5.21), we get

$$\begin{aligned}
||\mathbf{q}_{||}^{(k)}(t) - \mathbf{q}_{||}^{(k)}(nT)|| =& \left|\left\langle \mathbf{q}^{(k)}(t) - \mathbf{q}^{(k)}(nT), \mathbf{c}^{(k)}\right\rangle\right| \\
\leq& ||\mathbf{q}^{(k)}(t) - \mathbf{q}^{(k)}(nT)|| \\
\leq& \left\| \sum_{\breve{t}=nT}^{nT+T-1} \mathbf{a}(\breve{t}) \right\| + \sqrt{MLT} s_{\max}. \qquad (5.39)
\end{aligned}$$

Now, we will bound each of the terms in (5.36). Let us first consider the first term in (5.35). Again, using the fact that the arrival rate is $\check{\lambda}^{\epsilon}$, we have

$$\begin{aligned}
&\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \bar{\mathbf{s}}(t) - \mathbf{a}(t)\right\rangle^2\right] \\
\overset{(a)}{=}& \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(\left\langle \mathbf{c}^{(k)}, \mathbf{a}(t)\right\rangle - \frac{b^{(k)}}{\sqrt{L}}\right)^2\right] + \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \bar{\mathbf{s}}(t)\right\rangle\right)^2\right] \\
&\qquad - 2\frac{\epsilon^{(k)}}{\sqrt{L}} \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \bar{\mathbf{s}}(t)\right\rangle\right)\right] \\
\overset{(b)}{\leq}& \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(\frac{1}{\sqrt{L}}\left\langle c^{(k)}, a(t) - \check{\lambda}^{\epsilon}\right\rangle + \frac{\left\langle c^{(k)}, \check{\lambda}^{\epsilon}\right\rangle - b^{(k)}}{\sqrt{L}}\right)^2\right] \\
&\qquad + \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \bar{\mathbf{s}}(t)\right\rangle\right)^2\right] \\
\overset{(c)}{\leq}& \frac{1}{L} \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(\left\langle c^{(k)}, a(t) - \check{\lambda}^{\epsilon}\right\rangle\right)^2\right] + 2\frac{\epsilon^{(k)}}{\sqrt{L}} \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left\langle c^{(k)}, a(t) - \check{\lambda}^{\epsilon}\right\rangle\right] \\
&+ T\frac{\left(\epsilon^{(k)}\right)^2}{L} + \frac{T}{L}\frac{\epsilon^{(k)}}{\gamma^{(k)}}\left(\left(b^{(k)}\right)^2 + \langle c, s_{max}1\rangle^2\right) \\
\leq& \frac{T}{L}\left\langle \left(c^{(k)}\right)^2, \sigma^2\right\rangle + \frac{T\left(\epsilon^{(k)}\right)^2}{L} + \frac{T}{L}\frac{\epsilon^{(k)}}{\gamma^{(k)}}\left(\left(b^{(k)}\right)^2 + \langle c, s_{max}1\rangle^2\right) \qquad (5.40) \\
=& \frac{T}{\sqrt{L}}\left(\zeta^{(\epsilon,k)} + \frac{1}{\sqrt{L}}\frac{\epsilon^{(k)}}{\gamma^{(k)}}\left(\left(b^{(k)}\right)^2 + \langle c, s_{max}1\rangle^2\right)\right), \qquad (5.41)
\end{aligned}$$

where $\zeta^{(\epsilon,k)}$ was earlier defined as $\zeta^{(\epsilon,k)} = \frac{\left(\epsilon^{(k)}\right)^2}{\sqrt{L}} + \frac{1}{\sqrt{L}}\left\langle \left(c^{(k)}\right)^2, \left(\sigma^{(\epsilon)}\right)^2\right\rangle$. The last term in (a) is dropped to get (b) since it is negative. Inequality (c) follows from (5.37) in Claim 5.2. The first term in (5.40) is obtained by noting that $\mathbb{E}[a(t)] = \check{\lambda}^{\epsilon}$ and so $\mathbb{E}\left[\left(\left\langle c^{(k)}, a(t) - \check{\lambda}^{\epsilon}\right\rangle\right)^2\right] = var\left(\left\langle c^{(k)}, a(t) - \check{\lambda}^{\epsilon}\right\rangle\right) =$

$\left\langle c^{(k)}, var(a(t) - \check{\lambda}^{\epsilon})\right\rangle$. Consider the second term in (5.35).

$$\begin{aligned}\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle^2\right] &\leq \sum_{t=nT}^{nT+T-1} \left\langle \mathbf{c}^{(k)}, \mathbf{1}s_{max}\right\rangle \mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle\right] \\ &\leq \frac{T\epsilon^{(k)}}{\sqrt{L}} \left\langle \mathbf{c}^{(k)}, \mathbf{1}s_{max}\right\rangle, \qquad (5.42)\end{aligned}$$

where the last inequality follows from (5.38).

Now, we consider the term in (5.36). We need some definitions so that we can only consider the non-zero components of c. Let $\mathcal{L}_{++}^{(k)} = \left\{m \in \{1, 2, ...M\} : c_m^{(k)} > 0\right\}$. Define $\check{\mathbf{c}}^{(k)} = \left(c_{mi}^{(k)}\right)_{m\in\mathcal{L}_{++}^{(k)}}$,$\check{\mathbf{q}} = (q_{mi})_{m\in\mathcal{L}_{++}^{(k)}}$ and $\check{\mathbf{u}} = (\overline{u}_{mi})_{m\in\mathcal{L}_{++}^{(k)}}$. Also define, the projections, $\check{\mathbf{q}}_{||}^{(k)} = \left\langle \check{\mathbf{c}}^{(k)}, \check{\mathbf{q}}\right\rangle \check{\mathbf{c}}^{(k)}$ and $\check{\mathbf{q}}_{\perp}^{(k)} = \check{\mathbf{q}} - \check{\mathbf{q}}_{||}^{(k)}$. Similarly, define $\check{\mathbf{u}}_{||}^{(k)}$ and $\check{\mathbf{u}}_{\perp}^{(k)}$. Then in steady-state, for all time t we have

$$\begin{aligned}&\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t) + \mathbf{a}(t) - \overline{\mathbf{s}}(t)\right\rangle \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle\right] \\ =&\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t+1)\right\rangle \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle\right] - \mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle^2\right] \\ \leq&\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t+1)\right\rangle \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle\right] \\ =&\mathbb{E}\left[\left\langle \check{\mathbf{c}}^{(k)}, \check{\mathbf{q}}(t+1)\right\rangle \left\langle \check{\mathbf{c}}^{(k)}, \check{\mathbf{u}}(t)\right\rangle\right] \\ =&\mathbb{E}\left[||\check{\mathbf{q}}_{||}^{(k)}(t+1)||||\check{\mathbf{u}}_{||}^{(k)}||\right] \\ =&\mathbb{E}\left[\left\langle \check{\mathbf{q}}_{||}^{(k)}(t+1), \check{\mathbf{u}}_{||}^{(k)}(t)\right\rangle\right] \\ =&\mathbb{E}\left[\left\langle \check{\mathbf{q}}_{||}^{(k)}(t+1), \check{\mathbf{u}}(t)\right\rangle\right] \\ =&\mathbb{E}\left[\left\langle \check{\mathbf{q}}(t+1), \check{\mathbf{u}}(t)\right\rangle\right] + \mathbb{E}\left[\left\langle -\check{\mathbf{q}}_{\perp}^{(k)}(t+1), \check{\mathbf{u}}(t)\right\rangle\right] \\ \leq&\mathbb{E}\left[\left\langle D_{max}s_{max}\mathbf{1}, \check{\mathbf{u}}(t)\right\rangle\right] + \sqrt{\mathbb{E}\left[||\check{\mathbf{q}}_{\perp}^{(k)}(t+1)||^2\right]\mathbb{E}\left[||\check{\mathbf{u}}(t)||^2\right]} \qquad (5.43) \\ \leq&D_{max}s_{max}\mathbb{E}\left[\left\langle \mathbf{1}, \check{\mathbf{u}}(t)\right\rangle\right] + \sqrt{N_2^{(k)}\mathbb{E}\left[\left\langle \check{\mathbf{u}}(t), \check{\mathbf{u}}(t)\right\rangle\right]} \qquad (5.44) \\ \leq&D_{max}s_{max}\mathbb{E}\left[\left\langle \mathbf{1}, \check{\mathbf{u}}(t)\right\rangle\right] + \sqrt{N_2^{(k)}s_{max}\mathbb{E}\left[\left\langle \mathbf{1}, \check{\mathbf{u}}(t)\right\rangle\right]},\end{aligned}$$

where (5.43) follows from (5.3) and from Cauchy-Schwarz inequality. Equation (5.44) follows from state-space collapse (Proposition 5.1), since $\mathbb{E}\left[||\check{\mathbf{q}}_{\perp}^{(k)}||^2\right] \leq \mathbb{E}\left[||\overline{\mathbf{q}}_{\perp}^{(k)}||^2\right] \leq N_2^{(k)}$.

Note that

$$\begin{aligned}\mathbb{E}\left[\langle \mathbf{1}, \breve{\mathbf{u}}(t)\rangle\right] &\leq \frac{1}{c_{min}^{(k)}}\mathbb{E}\left[\left\langle \breve{\mathbf{c}}^{(k)}, \breve{\mathbf{u}}(t)\right\rangle\right]\\ &= \frac{1}{c_{min}^{(k)}}\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle\right]\\ &\leq \frac{\epsilon^{(k)}}{\sqrt{L}c_{min}^{(k)}},\end{aligned}$$

where $c_{min}^{(k)} \triangleq \min_{m\in\mathcal{L}_{++}^{(k)}} c_m^{(k)} > 0$ and the last inequality follows from (5.38). Thus, we have

$$\begin{aligned}&\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)+\overline{\mathbf{s}}(t)-\mathbf{a}(t)\right\rangle \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle\right]\\ \leq &TD_{max}s_{max}\frac{\epsilon^{(k)}}{\sqrt{L}c_{min}^{(k)}} + T\sqrt{N_2^{(k)}s_{max}\frac{\epsilon^{(k)}}{\sqrt{L}c_{min}^{(k)}}}. \qquad (5.45)\end{aligned}$$

We will now consider the left hand side term in (5.34). Given that the arrival rate is $\check{\lambda}^{\epsilon}$ for every t in steady-state, we have

$$\begin{aligned}&\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t)-\mathbf{a}(t)\right\rangle\right]\\ =&\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle\right]\left(\frac{b^{(k)}}{\sqrt{L}} - \frac{1}{\sqrt{L}}\left\langle c^{(k)}, \check{\lambda}\right\rangle\right) - \mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle\left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t)\right\rangle\right)\right]\\ =&\frac{\epsilon^{(k)}}{\sqrt{L}}\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle\right] - \mathbb{E}\left[||\mathbf{q}_{||}^{(k)}(t)||\left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t)\right\rangle\right)\right]. \qquad (5.46)\end{aligned}$$

Now, we will bound the last term in this equation, summed over T time slots in steady-state using the following lemma.

Lemma 5.5. *In steady-state, we have*

$$\begin{aligned}&\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[||\mathbf{q}_{||}^{(k)}(t)||\left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t)\right\rangle\right)\right]\\ \leq &T\sqrt{K_{33}+\cot^2\left(\theta^{(k)}\right)K_{34}}\sqrt{\frac{\epsilon^{(k)}}{L\gamma^{(k)}}\left(\left(b^{(k)}\right)^2 + \langle c, s_{max}1\rangle^2\right)}, \qquad (5.47)\end{aligned}$$

where $K_{33} = MLT^2s_{\max}^2 + 2\sqrt{ML}T^2s_{\max}\sqrt{\|\check{\lambda}^{(\epsilon)}\| + ||\sigma||^2} + T^2\|\check{\lambda}^{(\epsilon)}\|^2 + T^2||\sigma||^2$

and $K_{34} = N_2^{(k)} + 4N_1^{(k)} K_{33} \left(T\sqrt{\|\check{\lambda}^{(\epsilon)}\|^2 + \|\sigma\|^2} + \sqrt{ML} T s_{\max} \right) + 4K_{33}$.

Proof. Using the definition of $\theta^{(k)}$, we will consider three cases.
Case(i): $\theta_{\mathbf{q}(nT)\mathbf{c}^{(k)}} > \theta^{(k)}$

In this case, we have $||\mathbf{q}_{||}^{(k)}(nT)|| = ||\mathbf{q}(nT)|| \cos\left(\theta_{\mathbf{q}(nT)\mathbf{c}^{(k)}}\right) \leq ||\mathbf{q}_{\perp}^{(k)}(nT)|| \cot\left(\theta_{\mathbf{q}(nT)\mathbf{c}^{(k)}}\right) \leq ||\mathbf{q}_{\perp}^{(k)}(nT)|| \cot\left(\theta^{(k)}\right)$. Intuitively this means that, in steady state, when $\theta_{\mathbf{q}(nT)\mathbf{c}^{(k)}} > \theta^{(k)}$, $\mathbf{q}(nT)$ must be small. Otherwise, it would contradict the state-space collapse result that $\mathbf{q}_{\perp}^{(k)}(nT)$ is small. Here is the precise argument.

$$\begin{aligned}
&\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[||\mathbf{q}_{||}^{(k)}(t)|| \left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \right\rangle \right) \right] \\
\leq& \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(||\mathbf{q}_{||}^{(k)}(nT)|| + \left\| \sum_{\breve{t}=nT}^{nT+T-1} \mathbf{a}(\breve{t}) \right\| + \sqrt{ML} T s_{\max} \right) \left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \right\rangle \right) \right] \\
\leq& \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[||\mathbf{q}_{\perp}^{(k)}(nT)|| \cot\left(\theta^{(k)}\right) \left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \right\rangle \right) \right] \\
&\quad + \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(\left\| \sum_{\breve{t}=nT}^{nT+T-1} \mathbf{a}(\breve{t}) \right\| + \sqrt{ML} T s_{\max} \right) \left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \right\rangle \right) \right] \\
\leq& \sum_{t=nT}^{nT+T-1} \sqrt{ \mathbb{E}\left[\left(\frac{b^{(k)}}{\sqrt{L}} - \langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \rangle \right)^2 \right] } \\
&\times \sqrt{ \cot^2\left(\theta^{(k)}\right) \mathbb{E}\left[||\mathbf{q}_{\perp}^{(k)}(nT)||^2 \right] + \mathbb{E}\left[\left(\left\| \sum_{\breve{t}=nT}^{nT+T-1} \mathbf{a}(\breve{t}) \right\| + \sqrt{ML} T s_{\max} \right)^2 \right] }
\end{aligned} \tag{5.48}$$

$$\leq \sqrt{K_{33} + \cot^2\left(\theta^{(k)}\right) N_2^{(k)}} \sqrt{ T \sum_{t=nT}^{nT+T-1} \mathbb{E}\left[\left(\frac{b^{(k)}}{\sqrt{L}} - \langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \rangle \right)^2 \right] } \tag{5.49}$$

$$\leq T \sqrt{K_{33} + \cot^2\left(\theta^{(k)}\right) N_2^{(k)}} \sqrt{ \frac{\epsilon^{(k)}}{M\gamma^{(k)}} \left(\left(b^{(k)}\right)^2 + \langle c, s_{max} 1 \rangle^2 \right) }.$$

Equation (5.48) follows from Cauchy-Schwarz inequality. In (5.49), we get $N_2^{(k)}$ from state-space collapse, Proposition 5.1, the summation was moved inside the square root (along with an additional T) term due to Jensen's inequality, and finally the bound K_{33} uses the fact that $||\mathbf{a}(t)|| \leq ||a(t)||$. The last inequality follows from (5.37).

Case(ii): $\theta_{\mathbf{q}(t)\mathbf{c}^{(k)}} \leq \theta^{(k)}$ for all $t \in \{nT, nT+1, \ldots, nT+T-1\}$

Since a MaxWeight schedule is chosen at $t = nT$, by definition of $\theta^{(k)}$, we have that $\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(nT) \right\rangle = \frac{b^{(k)}}{\sqrt{L}}$. One can then inductively argue as follows for other times t. Suppose that at time t $\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \right\rangle = \frac{b^{(k)}}{\sqrt{L}}$. Then at time $t+1$, $\overline{\mathbf{s}}(t)$ is still feasible and it maximizes $\langle \mathbf{q}(t+1), \overline{\mathbf{s}} \rangle$ since $\theta_{\mathbf{q}(t+1)\mathbf{c}^{(k)}} \leq \theta^{(k)}$. Therefore, myopic MaxWeight chooses a schedule such that $\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t+1) \right\rangle = \frac{b^{(k)}}{\sqrt{L}}$. Therefore, in this case,

$$\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[||\mathbf{q}_{||}^{(k)}(t)|| \left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \right\rangle \right) \right] = 0.$$

Case(iii): $\theta_{\mathbf{q}(nT)\mathbf{c}^{(k)}} \leq \theta^{(k)}$ but $\theta_{\mathbf{q}(t)\mathbf{c}^{(k)}} > \theta^{(k)}$for some $t \in \{nT+1, \ldots, nT+T-1\}$

Let t_0 denote the first time $\theta_{\mathbf{q}(t)\mathbf{c}^{(k)}}$ exceeds $\theta^{(k)}$. Then, similar to Case (ii), up to time t_0, a schedule is chosen so that $\left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \right\rangle = \frac{b^{(k)}}{\sqrt{L}}$. We can now bound the remaining terms similar to Case (i). As in Case(i), we have that $||\mathbf{q}_{||}^{(k)}(t_0)|| \leq ||\mathbf{q}_{\perp}^{(k)}(t_0)|| \cot\left(\theta^{(k)}\right)$. Also, note that from (5.21), we can show that the bound in (5.39) is valid for $||\mathbf{q}_{||}^{(k)}(t) - \mathbf{q}_{||}^{(k)}(t_0)||$ too. Then, using the same argument as in Case (i), we get

$$\begin{aligned}
&\sum_{t=nT}^{nT+T-1} \mathbb{E}\left[||\mathbf{q}_{||}^{(k)}(t)|| \left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \right\rangle \right) \right] \\
=&\sum_{t=t_0}^{nT+T-1} \mathbb{E}\left[||\mathbf{q}_{||}^{(k)}(t)|| \left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \right\rangle \right) \right] \\
\leq&\sum_{t=t_0}^{nT+T-1} \mathbb{E}\left[\left(||\mathbf{q}_{||}^{(k)}(t_0)|| + \left\| \sum_{t=nT}^{nT+T-1} \mathbf{a}(t) \right\| + \sqrt{MLT} s_{\max} \right) \left(\frac{b^{(k)}}{\sqrt{L}} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t) \right\rangle \right) \right] \\
\leq& T\sqrt{K_{33} + \cot^2\left(\theta^{(k)}\right) \mathbb{E}\left[||\mathbf{q}_{\perp}^{(k)}(t_0)||^2 \right]} \sqrt{\frac{\epsilon^{(k)}}{L\gamma^{(k)}} \left(\left(b^{(k)}\right)^2 + \langle c, s_{max}1 \rangle^2 \right)}.
\end{aligned} \tag{5.50}$$

We will now bound $\sqrt{\mathbb{E}\left[||\mathbf{q}_{\perp}^{(k)}(t_0)||^2 \right]}$ using state-space collapse.

First, from (5.39), we get

$$\begin{aligned}
||\mathbf{q}_{\perp}^{(k)}(t)-\mathbf{q}_{\perp}^{(k)}(nT)|| =& || \left(\mathbf{q}(t)-\mathbf{q}(nT)\right) - \left(\mathbf{q}_{||}^{(k)}(t)-\mathbf{q}_{||}^{(k)}(nT)\right) || \\
\leq& ||\mathbf{q}(t)-\mathbf{q}(nT)|| + ||\mathbf{q}_{||}^{(k)}(t)-\mathbf{q}_{||}^{(k)}(nT)|| \\
\leq& 2\left\|\sum_{\breve{t}=nT}^{nT+T-1}\mathbf{a}(\breve{t})\right\| + 2\sqrt{ML}Ts_{\max} \\
\mathbb{E}\left[||\mathbf{q}_{\perp}^{(k)}(t_0)||^2\right] \leq& \mathbb{E}\left[\left(||\mathbf{q}_{\perp}^{(k)}(nT)|| + 2\left\|\sum_{\breve{t}=nT}^{nT+T-1}\mathbf{a}(\breve{t})\right\| + 2\sqrt{ML}Ts_{\max}||\right)^2\right] \\
\leq& N_2^{(k)} + 4N_1^{(k)}\left(T\sqrt{\|\check{\lambda}^{(\epsilon)}\|^2+\|\sigma\|^2}+\sqrt{ML}Ts_{\max}\right) + 4K_{33}.
\end{aligned}$$

The last inequality follows from state-space collapse, Proposition 5.1, $\mathbb{E}\left[||\mathbf{q}_{\perp}^{(k)}(nT)||^2\right] \leq N_2^{(k)}$ and $\mathbb{E}\left[||\mathbf{q}_{\perp}^{(k)}(nT)||\right] \leq N_1^{(k)}$. Combining the three cases, we have the lemma. $\square$

Now, substituting (5.46), (5.47), (5.41), (5.42) and (5.45) in (5.36), we get

$$\epsilon^{(k)}\sum_{t=nT}^{nT+T-1}\mathbb{E}\left[\left\langle \mathbf{c}^{(k)},\mathbf{q}(t)\right\rangle\right] \leq \frac{T\zeta^{(\epsilon,k)}}{2} + TB_2^{(\epsilon,k)},$$

where

$$\begin{aligned}
B_2^{(\epsilon,k)} =& \frac{1}{2\sqrt{L}}\frac{\epsilon^{(k)}}{\gamma^{(k)}}\left(\left(b^{(k)}\right)^2 + \langle c, s_{max}1\rangle^2\right) + \frac{D_{max}s_{max}\epsilon^{(k)}}{c_{min}^{(k)}} + \frac{\epsilon^{(k)}}{2}\left\langle \mathbf{c}^{(k)}, \mathbf{1}s_{max}\right\rangle \\
&+ \sqrt{K_{33}+\cot^2\left(\theta^{(k)}\right)K_{34}}\sqrt{\frac{\epsilon^{(k)}}{\gamma^{(k)}}\left(\left(b^{(k)}\right)^2 + \langle c, s_{max}1\rangle^2\right)} \\
&+ \sqrt{\sqrt{L}N_2^{(k)}s_{max}\frac{\epsilon^{(k)}}{c_{min}^{(k)}}}.
\end{aligned}$$

We will now use (5.39) to get a bound on $\mathbb{E}\left[\left\langle \mathbf{c}^{(k)},\mathbf{q}(t)\right\rangle\right]$ in steady-state for any time t. Let $nT \leq t \leq nT+T-1$. Then,

$$\begin{aligned}
\epsilon^{(k)}\mathbb{E}\left[\left\langle \mathbf{c}^{(k)},\mathbf{q}(t)\right\rangle\right] \leq& \epsilon^{(k)}\mathbb{E}\left[\left\langle \mathbf{c}^{(k)},\mathbf{q}(nT)\right\rangle\right] + \epsilon^{(k)}\mathbb{E}\left[||\mathbf{q}_{||}^{(k)}(t)-\mathbf{q}_{||}^{(k)}(nT)||\right] \\
\leq& \frac{\epsilon^{(k)}}{T}\sum_{\breve{t}}^{nT+T-1}\left(\mathbb{E}\left[\left\langle \mathbf{c}^{(k)},\mathbf{q}(\breve{t})\right\rangle\right] + \mathbb{E}\left[||\mathbf{q}_{||}^{(k)}(nT)-\mathbf{q}_{||}^{(k)}(\breve{t})||\right]\right)
\end{aligned}$$

$$
\begin{aligned}
&\quad + \epsilon^{(k)}\mathbb{E}\left[||\mathbf{q}_{||}^{(k)}(t) - \mathbf{q}_{||}^{(k)}(nT)||\right] \\
&\le \frac{\zeta^{(\epsilon,k)}}{2} + B_2^{(\epsilon,k)} + 2\epsilon^{(k)}\mathbb{E}\left[\left\|\sum_{\breve{t}=nT}^{nT+T-1}\mathbf{a}(\breve{t})\right\|\right] + 2\epsilon^{(k)}\sqrt{ML}Ts_{\max} \\
&\le \frac{\zeta^{(\epsilon,k)}}{2} + B_3^{(\epsilon,k)},
\end{aligned}
$$

where $B_3^{(\epsilon,k)} = B_2^{(\epsilon,k)} + 2\epsilon^{(k)}T\sqrt{\|\check{\lambda}^{(\epsilon)}\| + ||\sigma||^2} + 2\epsilon^{(k)}\sqrt{ML}Ts_{\max}$

Note that in the heavy traffic limit as $\epsilon^{(k)} \downarrow 0$, $B_3^{(\epsilon,k)} \to 0$. Thus, in the heavy traffic limit as $\epsilon^{(k)} \downarrow 0$, for any time t, we have

$$
\lim_{\epsilon^{(k)}\downarrow 0} \epsilon^{(k)}\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle\right] \le \frac{\zeta^{(k)}}{2}, \tag{5.51}
$$

where $\zeta^{(k)}$ was defined as $\zeta^{(k)} = \frac{1}{\sqrt{L}}\left\langle \left(c^{(k)}\right)^2, (\sigma)^2\right\rangle$. Thus, (5.8) and (5.51) establish the first moment heavy-traffic optimality of JSQ routing and MaxWeight scheduling policy. The proof of Theorem 5.2 is now complete.

5.3.4 Some Extensions

We have so far, obtained heavy traffic optimality of $\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle$, which depends on the vector $\mathbf{c}^{(k)}$. In other words, we have optimality of a particular linear combination of queue lengths and we do not have a choice on this linear combination. In this subsection, we will extend the heavy traffic optimality result to $||\mathbf{q}(t)||^2$. We will do that by first obtaining lower and upper bounds on $\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle^n$ for any $n \ge 1$.

Proposition 5.2. *Consider the cloud computing system described in section 5.1. Under the same conditions as in Theorem 5.2, for any $n \ge 1$, we have*

$$
n!\left(\frac{\zeta_1^{(\epsilon,k)}}{2}\right)^n - B_{1,n}^{(\epsilon,k)}, \le (\epsilon^{(k)})^n\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle^n\right] \le n!\left(\frac{\zeta_1^{(\epsilon,k)}}{2}\right)^n + B_{2,n}^{(\epsilon,k)},
$$

where $\zeta_1^{(\epsilon,k)} = \frac{1}{\sqrt{L}}\left\langle \left(c^{(k)}\right)^2, \left(\sigma^{(\epsilon)}\right)^2\right\rangle$ and $B_{1,n}^{(\epsilon,k)}$, $B_{2,n}^{(\epsilon,k)}$ are $o(\frac{1}{\epsilon^{(k)}})$. The lower bound is a universal lower bound applicable to all resource allocation algorithms. The upper bound is attained by Algorithm 9.

In the heavy traffic limit as $\epsilon^{(k)} \downarrow 0$, this bound is tight, i.e.,

$$\lim_{\epsilon^{(k)}\downarrow 0} (\epsilon^{(k)})^n \mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}(t)\right\rangle^n\right] = n!\left(\frac{\zeta^{(k)}}{2}\right)^n,$$

where $\zeta^{(k)} = \frac{1}{\sqrt{L}}\left\langle \left(c^{(k)}\right)^2, (\sigma)^2\right\rangle$.

The proof is based on the three-step procedure as in the previous subsections. The first step is to prove the lower bound in Proposition 5.2. It follows directly from Lemma 10 in [37], which is a bound on the nth power of single server queue, in the lines of 5.7. State space collapse as stated in Proposition 5.1 is applicable here. Upper bound is obtained by first setting the drift of the following Lyapunov function to zero in steady state,

$$V_{n||}^{(k)}(\mathbf{q}) \triangleq \left\langle \mathbf{c}^{(k)}, \mathbf{q}^{(\epsilon)}\right\rangle^n = \left\|\mathbf{q}_{||}^{(k)}\right\|^n = \frac{1}{L}\left(\sum_{i=1}^{L}\sum_{m=1}^{M} q_{mi}c_m\right)^n,$$

similar to Lemma 5.4 (see Appendix D of [37]), we get

$$\sum_{t=nT}^{nT+T-1} (n+1)\left(\epsilon^{(k)}\right)^n \mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle^n\right] \tag{5.52}$$

$$= \sum_{t=nT}^{nT+T-1}\sum_{j=0}^{n-1}\binom{n+1}{j}(\epsilon^{(k)})^{n-1}\mathbb{E}\left[\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)\right\rangle^j\left(\left\langle \mathbf{c}^{(k)}, \mathbf{a}(t)\right\rangle - b^{(k)}\right)^{(n-j+1)}\right] \tag{5.53}$$

$$+ \sum_{t=nT}^{nT+T-1}\sum_{j=0}^{n}\binom{n+1}{j}(\epsilon^{(k)})^{n-1} \times \mathbb{E}\left[\left(\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)+\mathbf{a}(t)\right\rangle - b^{(k)}\right)^j\left(b^{(k)} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{s}}(t)\right\rangle\right)^{(n-j+1)}\right] \tag{5.54}$$

$$+ \sum_{t=nT}^{nT+T-1}\sum_{j=0}^{n}\binom{n+1}{j}(\epsilon^{(k)})^{n-1} \times \mathbb{E}\left[\left(\left\langle \mathbf{c}^{(k)}, \mathbf{q}(t)+\mathbf{a}(t)-\overline{\mathbf{s}}(t)\right\rangle - b^{(k)}\right)^j\left(b^{(k)} - \left\langle \mathbf{c}^{(k)}, \overline{\mathbf{u}}(t)\right\rangle\right)^{(n-j+1)}\right] \tag{5.55}$$

Each of these terms are then bounded, similar to the previous subsection and appendices C and D of [37]. We skip the details here. Unlike the proof of Theorem 5.2, we need the assumption that $a_m(t) \le a_{\max}$ in the proof of heavy traffic optimality in Proposition 5.2. We can now obtain heavy traffic optimality of $||\mathbf{q}(t)||^2$ using Proposition 5.2 for $n = 2$.

Theorem 5.3. *Consider the cloud computing system described in section 5.1. Under the same conditions as in Theorem 5.2,*

$$\frac{\left(\zeta_1^{(\epsilon,k)}\right)^2}{2} - B_{1,2}^{(\epsilon,k)}, \leq (\epsilon^{(k)})^2 \mathbb{E}\left[\left\|\mathbf{q}^{(\epsilon)}(t)\right\|^2\right] \leq \frac{\left(\zeta_1^{(\epsilon,k)}\right)^2}{2} + B_{3,2}^{(\epsilon,k)},$$

where $\zeta_1^{(\epsilon,k)} = \frac{1}{\sqrt{L}} \left\langle \left(c^{(k)}\right)^2, \left(\sigma^{(\epsilon)}\right)^2 \right\rangle$ *and* $B_{1,2}^{(\epsilon,k)}$, $B_{2,2}^{(\epsilon,k)}$ *are* $o(\frac{1}{\epsilon^{(k)}})$.

In the heavy traffic limit as $\epsilon^{(k)} \downarrow 0$*, this bound is tight, i.e.,*

$$\lim_{\epsilon^{(k)} \downarrow 0} (\epsilon^{(k)})^2 \mathbb{E}\left[\left\|\mathbf{q}^{(\epsilon)}(t)\right\|^2\right] = \frac{\left(\zeta^{(k)}\right)^2}{2},$$

The lower bound is a universal lower bound applicable to all resource allocation algorithms. The upper bound is attained by Algorithm 9. Thus, Algorithm 9 is second moment heavy traffic optimal

Proof. From the Pythagorean theorem, we have

$$\left\|\mathbf{q}_{||}^{(\epsilon)}(t)\right\|^2 \leq \left\|\mathbf{q}^{(\epsilon)}(t)\right\|^2 = \left\|\mathbf{q}_{||}^{(\epsilon)}(t)\right\|^2 + \left\|\mathbf{q}_{\perp}^{(\epsilon)}(t)\right\|^2$$

The result then follows from Proposition 5.2 and state space collapse in Proposition 5.1 that $\left\|\mathbf{q}_{\perp}^{(\epsilon)}(t)\right\|^2 \leq N_2^{(k)}$.

□

In the following section, we will study the power-of-two choices routing algorithm that is easier to implement than JSQ.

5.4 Power-of-Two-Choices Routing and MaxWeight Scheduling

Power-of-two-choices routing algorithm, studied in Chapter 2, is much simpler to use than JSQ routing algorithm. In this section, we will consider the power-of-two-choices routing algorithm with the MaxWeight scheduling algorithm for the cloud resource allocation problem. Recall that in the power-of-two-choices routing algorithm, in each time slot t, for each type of job i, two servers $i_{1m}(t)$ and $i_{2m}(t)$ are chosen uniformly at random.

All the type i job arrivals in this time slot are then routed to the server with the shorter queue length of backlogged workload among these two, i.e., $i_m^*(t) = \underset{i \in \{i_{1m}(t), i_{2m}(t)\}}{\arg\min} \; q_{mi}(t)$.

Then, we have that the cloud computing system is heavy traffic optimal. In other words, we have the following result.

Theorem 5.4. *Theorems 5.1 and 5.2 hold when power-of-two-choices routing is used instead of JSQ routing in Algorithm 9.*

Proof. Proof of Theorem 5.1 when power-of-two choices algorithm is used, is very similar to proof of Theorem 2.2 in Chapter 2 and so we skip it. Proof of this theorem is similar to that of Theorem 5.2 and so here we present only the differences that arise in the proof due to power-of-two choices routing.

We will use the same Lyapunov functions defined in the proof of Proposition 5.2. We will first bound the one-step drift of the Lyapunov function $V(.)$. Note that the bound (5.15) on the drift of $V(.)$ is valid. Recall that we use $\mathbf{q}^{(\epsilon)}$ for $\mathbf{q}(\mathbf{Y}^{(\epsilon)})$.

$$\mathbb{E}\left[\triangle\, V(\mathbf{Y}^{(\epsilon)}) | \mathbf{Y}^{(\epsilon)}(t) {=} \mathbf{Y}^{(\epsilon)}\right] \leq K_{27} + \mathbb{E}_{\mathbf{Y}^{(\epsilon)}}\left[\sum_{i=1}^{L}\sum_{m=1}^{M}\left(2q_{mi}^{(\epsilon)}\left(a_{mi}(t) - s_m^i(t)\right)\right)\right]. \tag{5.56}$$

The arrival term here can be bounded as follows. Let $X_j = (i_{1m}, i_{2m})$ denote the two servers randomly chosen by power-of-two-choices algorithms for routing of type m jobs.

$$\begin{aligned}
&\mathbb{E}_{\mathbf{Y}^{(\epsilon)}}\left[\sum_{m=1}^{M}\sum_{i=1}^{L}\left(2q_{mi}^{(\epsilon)}\left(a_{mi}(t)\right)\right)\right]\\
&= \mathbb{E}\left[\mathbb{E}\left[\sum_{m=1}^{M}\sum_{i=1}^{L}\left(2q_{mi}^{(\epsilon)}\left(a_{mi}(t)\right)\right)\middle| \mathbf{Y}(t){=}\mathbf{Y}^{(\epsilon)}, X_j{=}(i_{1m}, i_{2m})\right]\middle| \mathbf{Y}(t){=}\mathbf{Y}^{(\epsilon)}\right]\\
&\overset{(a)}{=} \mathbb{E}\left[\sum_{m=1}^{M}\sum_{(i_{1m},i_{2m}):i_{1m}<i_{2m}} \frac{1}{{}^L C_2} 2a_m(t)\min\{q_{mi_{1m}}^{(\epsilon)}, q_{mi_{2m}}^{(\epsilon)}\} | \mathbf{Y}(t) = \mathbf{Y}^{(\epsilon)}\right]\\
&= \sum_{m=1}^{M}\check{\lambda}_m \frac{1}{{}^L C_2}\sum_{(i_{1m},i_{2m}):i_{1m}<i_{2m}} 2\min\{q_{mi_{1m}}^{(\epsilon)}, q_{mi_{2m}}^{(\epsilon)}\}
\end{aligned}$$

$$\overset{(b)}{\leq}\sum_{m=1}^{M}\breve{\lambda}_m\frac{1}{{}^{L}C_2}\left(\left(\sum_{(i_{1m},i_{2m}):i_{1m}<i_{2m}} q^{(\epsilon)}_{mi_{1m}}+q^{(\epsilon)}_{mi_{2m}}\right)-(q^{(\epsilon)}_{m\max}-q^{(\epsilon)}_{m\min})\right)$$
$$=\sum_{m=1}^{M}\breve{\lambda}_m\left(\sum_{i=1}^{L}\frac{2q^{(\epsilon)}_{mi}}{L}-\frac{1}{{}^{L}C_2}(q^{(\epsilon)}_{m\max}-q^{(\epsilon)}_{m\min})\right),$$

where $q^{(\epsilon)}_{m\max}=\max_m q^{(\epsilon)}_{mi}$ and $q^{(\epsilon)}_{m\min}=\min_m q^{(\epsilon)}_{mi}$. Equation (a) follows from the definition of power-of-two-choices routing and (b) follows from the fact that $2\min\{q^{(\epsilon)}_{mi_{1m}},q^{(\epsilon)}_{mi_{2m}}\}\leq q^{(\epsilon)}_{mi_{1m}}+q^{(\epsilon)}_{mi_{2m}}$ for all i_{1m} and i_{2m} and $2\min\{q^{(\epsilon)}_{m\max},q^{(\epsilon)}_{m\min}\}\leq(q^{(\epsilon)}_{m\max}+q^{(\epsilon)}_{m\min})-(q^{(\epsilon)}_{m\max}-q^{(\epsilon)}_{m\min})$. Then, from (5.56), we have

$$\begin{aligned}\mathbb{E}\left[\triangle\, V(\mathbf{Y}^{(\epsilon)})|\mathbf{Y}^{(\epsilon)}=\mathbf{Y}^{(\epsilon)}\right]\leq&K_{27}-\sum_{m=1}^{M}\breve{\lambda}_m\frac{1}{{}^{L}C_2}(q^{(\epsilon)}_{m\max}-q^{(\epsilon)}_{m\min})\\&+2\sum_{m=1}^{M}\breve{\lambda}_m\sum_{i=1}^{L}\frac{q^{(\epsilon)}_{mi}}{L}-2\mathbb{E}_{\mathbf{q}^{(\epsilon)}}\left[\sum_{i=1}^{L}\sum_{m=1}^{M}2q^{(\epsilon)}_{mi}s^i_m(t)\right].\end{aligned}\tag{5.57}$$

The terms in (5.57) are identical to the ones in (5.17) and so can be bounded by (5.18). Moreover, the bound on one step drift of the Lyapunov function $V_{||}(.)$ in (5.14) as well as the bound on the drift $\Delta\widetilde{W}^{(k)}_{\perp}(\widetilde{\mathbf{Y}}^{(\epsilon)})$ in (5.10) are valid here since it does not depend on the routing policy. Substituting (5.14), (5.57) and (5.18) in (5.10), and using $\widetilde{\mathbf{q}}^{(\epsilon)}$ for $\mathbf{q}(\widetilde{\mathbf{Y}}^{(\epsilon)})$, we get

$$\mathbb{E}\left[\Delta\widetilde{W}^{(k)}_{\perp}(\widetilde{\mathbf{Y}}^{(\epsilon)})\middle|\mathbf{Y}^{(\epsilon)}(nT)=\widetilde{\mathbf{Y}}^{(\epsilon)}\right]$$
$$\leq\frac{1}{2\left\|\widetilde{\mathbf{q}}^{(\epsilon,k)}_{\perp}\right\|}\left(\sum_{t=nT}^{nT+T-1}\mathbb{E}\left[K_{28}-\sum_{m=1}^{M}\breve{\lambda}_m\frac{1}{{}^{L}C_2}(q^{(\epsilon)}_{m\max(t)}(t)-q^{(\epsilon)}_{m\min(t)})(t)\right.\right.\tag{5.58}$$
$$\left.\left.+2\sum_{i=1}^{L}\mathbb{E}_{\mathbf{Y}^{(\epsilon)}}\left[\sum_{m=1}^{M}q^{(\epsilon)}_{mi}\left(\frac{\breve{\lambda}^{(k)}_m}{L}-s^i_m(t)\right)\right]\middle|\mathbf{Y}^{(\epsilon)}(nT)=\widetilde{\mathbf{Y}}^{(\epsilon)}\right]\right).\tag{5.59}$$

In the notation $q^{(\epsilon)}_{m\max(t)}(t')$, $\max(t)$ denotes the server with highest workload at time t, even though we are interested in the workload at time t'. We will first bound the terms in (5.19). Using (5.21), we have

$$q^{(\epsilon)}_{m\min(t)}(t)\overset{(a)}{\leq}q^{(\epsilon)}_{m\min(nT)}(t)$$

$$
\begin{aligned}
&\leq q^{(\epsilon)}_{m\,\min(nT)}(nT) + \sum_{t=nT}^{nT+T-1} a_{m\,\min(nT)}(t) + Ts_{\max} \\
-q^{(\epsilon)}_{m\,\max(t)}(t) &\overset{(b)}{\leq} -q^{(\epsilon)}_{m\,\max(nT)}(t) \\
&\leq -q^{(\epsilon)}_{m\,\max(nT)}(nT) + \sum_{t=nT}^{nT+T-1} a_{m\,\max(nT)}(t) + Ts_{\max},
\end{aligned}
$$

where (a) follows from the fact that the server $\min(t)$ has the smallest workload at time t for type-m jobs. Using $\mathbb{E}_{\widetilde{\mathbf{Y}}^{(\epsilon)}}[.]$ for $\mathbb{E}[.|\mathbf{Y}^{(\epsilon)}(nT) = \widetilde{\mathbf{Y}}^{(\epsilon)}]$, we bound the terms in (5.58). As in Section 5.3.2, we will *assume* that the arrival rate $\check{\lambda}^{(\epsilon)}$ is such that there exists a $\delta > 0$ such that $\check{\lambda}^{(\epsilon)}_m > \delta$ for all m.

$$
\begin{aligned}
&-\sum_{t=nT}^{nT+T-1}\sum_{m=1}^{M}\check{\lambda}^{(\epsilon)}_m \frac{1}{{}^L C_2}\mathbb{E}_{\widetilde{\mathbf{Y}}^{(\epsilon)}}\left[q^{(\epsilon)}_{m\,\max(t)}(t) - q^{(\epsilon)}_{m\,\min(t)}(t)\right] \\
\leq &\sum_{t=nT}^{nT+T-1}\sum_{m=1}^{M}\check{\lambda}^{(\epsilon)}_m \frac{1}{{}^L C_2}\left(-\mathbb{E}_{\widetilde{\mathbf{Y}}^{(\epsilon)}}\left[q^{(\epsilon)}_{m\,\max(nT)}(nT) - q^{(\epsilon)}_{m\,\min(nT)}(nT)\right] + 2T\check{\lambda}^{(\epsilon)}_m + 2Ts_{\max}\right) \\
\leq &K_{29} - \sum_{t=nT}^{nT+T-1}\sum_{m=1}^{M}\check{\lambda}^{(\epsilon)}_m \frac{1}{{}^L C_2}(\widetilde{q}^{(\epsilon)}_{m\,\max} - \widetilde{q}^{(\epsilon)}_{m\,\min}) \\
= &K_{29} - \sum_{t=nT}^{nT+T-1}\sum_{m=1}^{M}\check{\lambda}^{(\epsilon)}_m \frac{1}{{}^L C_2\sqrt{L}}\sqrt{L\left(\widetilde{q}^{(\epsilon)}_{m\,\max} - \widetilde{q}^{(\epsilon)}_{m\,\min}\right)^2} \\
\leq &K_{29} - \sum_{t=nT}^{nT+T-1}\sum_{m=1}^{M}\check{\lambda}^{(\epsilon)}_m \frac{1}{{}^L C_2\sqrt{L}}\sqrt{\sum_{i=1}^{L}\left(\widetilde{q}^{(\epsilon)}_{mi} - \frac{\sum_{i'}\widetilde{q}^{(\epsilon)}_{mi'}}{L}\right)^2} \\
\leq &K_{29} - \sum_{t=nT}^{nT+T-1}\frac{2}{L}\sum_{m=1}^{M}\delta''\sqrt{\sum_{i=1}^{L}\left(\widetilde{q}^{(\epsilon)}_{mi} - \frac{\sum_{i'}\widetilde{q}^{(\epsilon)}_{mi'}}{L}\right)^2},
\end{aligned}
$$

where K_{29} was already defined as $K_{29} = 4T^2\sum_{m=1}^{M}(\check{\lambda}^{(\epsilon)}_m)^2 + 4T^2 s_{\max}\sum_{m=1}^{M}\check{\lambda}^{(\epsilon)}_m$ and $\delta'' = \frac{\delta}{(M-1)\sqrt{L}}$. This term is same as the term in (5.24) with δ'' instead of δ. This term can then be bound by the term in (5.25) with δ'' instead of δ. Noting that the terms in (5.59) are identical to the ones in (5.20), we can bound them using (5.26) and (5.31) as in section 5.3.2, and we get (5.32) with δ'' instead of δ. Note that the remainder of the proof of state-space collapse in section 5.3.2 does not use the routing policy and is valid when δ is replaced with δ''. Moreover, the proofs of lower bound in section 5.3.1 and upper bound in section 5.3.3 are also valid here. Thus, the proof of Theorem

5.4 is complete.

□

Similarly, we have the following result of heavy traffic optimality of $\mathbb{E}\left[\left\|\mathbf{q}^{(\epsilon)}(t)\right\|^2\right]$ under power-of-two choices routing, which we state without proof.

Proposition 5.3. *Proposition 5.2 and Theorem 5.3 hold when poer-of-two-choices routing algorithm is used instead of JSQ routing in Algorithm 9.*

5.5 Conclusions

In this chapter, we studied the resource allocation problem assuming that the job sizes are known and preemption is allowed once every few time slots. We presented two algorithms based on myopic MaxWeight scheduling and join the shortest queue or power-of-two choices routing algorithms. We have shown that these algorithms are not only throughput optimal, but also delay optimal in the heavy traffic regime when all the servers are assumed to be identical.

CHAPTER 6

CONCLUSION

In this thesis, we have studied the problem of optimally allocating limited server resources to virtual machines in a IaaS cloud computing data center in an optimal manner. The resource allocation algorithm has two components, viz., a load balancing or routing algorithm that assigns virtual machines to the servers and a scheduling algorithm for each server. We have shown that the widely used Best-Fit algorithm is not throughput optimal. We have presented a stochastic framework to study the resource allocation problem. We made a connection with scheduling in ad hoc wireless networks and presented different versions of MaxWeight scheduling algorithm. In conjunction with the MaxWeight scheduling, we have presented three different routing algorithms, viz., join the shortest queue, power-of-two choices and pick-and compare.

We have considered various assumptions on job sizes and preemption of jobs. We have studied the case when job sizes are known and bounded, and when they are unknown and unbounded. We have also studied nonpreemptive algorithms as well as algorithms with limited preemption. Under all these cases, we have shown that the algorithms that we have presented are throughput optimal. In the case of limited preemption and known job sizes, we have also shown that our algorithm is delay optimal in the heavy traffic limit. We have compared the performance of various algorithms through simulations.

6.1 Open Problems and Future Directions

There are several open problems and future directions that we mentioned throughout the thesis. Here we list some of them.

- The parameter T corresponding to the duration of a super time slot plays an important role in the proof of Algorithm 3. Even though any finite T stabilizes the system, a small T sacrifices a large part of the capacity region. Studying this algorithm when $T = \infty$ is an open problem. In Algorithm 4, we have used a fixed schedule between refresh times. But using a myopic MaxWeight schedule between the refresh times is more natural and we have noticed better performance in simulations. We named this Algorithm 7 and proving its throughput optimality under appropriate assumptions is an open problem. Algorithm 7 and Algorithm 3 with $T = \infty$ are similar. The only difference is in the knowledge of job sizes.

- When preemption is not allowed, the delay performance of throughput optimal algorithms that we have presented is poor. We have presented heuristic algorithms in Chapter 3 with better performance but we do not have any analytic guarantees on their throughput or delay performances. Therefore, presenting an algorithm that is provably throughput optimal with a tight bound on its delay performance is an problem for future investigation.

- When job sizes are unknown, our throughput optimality results make certain assumptions on job sizes. In particular, these assumptions do not allow for heavy-tailed job size distributions, which are common in data centers. Therefore, another important open problem is to present a throughput optimal algorithm when job sizes are unknown and are allowed to be heavy tailed.

- The result on delay optimality makes several assumptions. In particular, all the servers need to be identical and the job sizes need to be known and bounded. Relaxing any or both of these two assumptions is a direction for future research

- Our delay optimality result is valid only when approaching a corner point of the capacity region. Most of the literature on heavy traffic optimality in switches and ad hoc wireless networks and other generalized switch models faces this limitation. Addressing this issue even in the simplest possible setting of an input-queued switch has been a longstanding open problem [52].

APPENDIX A

PROOF OF LEMMA 3.6

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,

$$v = G(u)$$
$$\frac{dv}{du} = G'(u) = g(u) = g(G^{-1}(v))$$
$$\frac{du}{dv} = \frac{1}{g(G^{-1}(v))}.$$

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 3.7

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

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

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

$$
\begin{aligned}
e^{\left(\frac{1}{LM}\sum_{i,m}\log(1+\overline{\mathbf{Q}}_{mi})\right)} &\leq 1+G^{-1}\left(V(\overline{\mathbf{Q}}))\right)\\
\frac{1}{LM}\sum_{i,m}\log(1+\overline{\mathbf{Q}}_{mi}) &\leq \log\left((1+G^{-1}\left(V(\overline{\mathbf{Q}}))\right)\right). \qquad \text{(B.1)}
\end{aligned}
$$

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.

APPENDIX C

PROOF OF CLAIM 5.1

For any two job types m and m' as well as servers i and i', we have

$$0 \leq \left[\left((\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi'}^{(\epsilon)})c_{m'}\right) - \left((\widetilde{q}_{m'i}^{(\epsilon)} - \widetilde{q}_{m'i'}^{(\epsilon)})c_m\right)\right]^2$$

$$\sum_{m<m'} 2(\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi'}^{(\epsilon)})c_{m'}(\widetilde{q}_{m'i}^{(\epsilon)} - \widetilde{q}_{m'i'}^{(\epsilon)})c_m \leq \sum_{m<m'} \left((\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi'}^{(\epsilon)})c_{m'}\right)^2 + \left((\widetilde{q}_{m'i}^{(\epsilon)} - \widetilde{q}_{m'i'}^{(\epsilon)})c_m\right)^2 \tag{C.1}$$

$$\sum_{m=1}^{M}\sum_{m'=1}^{M}(\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi'}^{(\epsilon)})c_m(\widetilde{q}_{m'i}^{(\epsilon)} - \widetilde{q}_{m'i'}^{(\epsilon)})c_{m'} \leq \sum_{m=1}^{M}\sum_{m'=1}^{M}(\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi'}^{(\epsilon)})^2(c_{m'})^2 \tag{C.2}$$

$$\left(\sum_{m=1}^{M}(\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi'}^{(\epsilon)})c_m\right)^2 \leq \left[\sum_{m=1}^{M}(\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi'}^{(\epsilon)})^2\right]\sum_{m'=1}^{M}(c_{m'})^2.$$

The left-hand sides of (C.1) and (C.2) are equal for the following reason. The two sums in the LHS of (C.2) can be split into three cases, viz., $m = m'$, $m < m'$ and $m > m'$. The term corresponding to $m = m'$ is zero. The other two cases correspond to the same term which gives the factor 2 in (C.1). Considering the three cases, it can be shown that the right-hand sides of (C.1) and (C.2) are equal. Noting that $\sum_{m'=1}^{M}(c_{m'})^2 = 1$ and summing over i, i' such that $i < i'$, we get

$$\sum_{i<i'}\left[\left(\sum_{m=1}^{M}(\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi'}^{(\epsilon)})c_m\right)^2\right] \leq \sum_{i<i'}\left[\sum_{m=1}^{M}\left(\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi'}^{(\epsilon)}\right)^2\right] \tag{C.3}$$

$$\sum_{i=1}^{L}\sum_{i'=1}^{L}\left[\left(\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}c_m\right)\left(\sum_{m=1}^{M}(\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi'}^{(\epsilon)})c_m\right)\right] \leq \sum_{i=1}^{L}\sum_{i'=1}^{L}\left[\sum_{m=1}^{M}\left(\widetilde{q}_{mi}^{(\epsilon)}\right)\left(\widetilde{q}_{mi}^{(\epsilon)} - \widetilde{q}_{mi'}^{(\epsilon)}\right)\right]. \tag{C.4}$$

The left-hand side of (C.4) is obtained using the same method as in (C.2) as follows. The two sums in the LHS of (C.4) can be split into three cases, viz.,

$i = i'$, $i < i'$ and $i > i'$. The term corresponding to $i = i'$ is zero. The other two cases, can be combined to get

$$\sum_{i<i'}\left[\left(\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}c_m\right)\left(\sum_{m=1}^{M}(\widetilde{q}_{mi}^{(\epsilon)}-\widetilde{q}_{mi'}^{(\epsilon)})c_m\right)+\left(\sum_{m=1}^{M}\widetilde{q}_{mi'}^{(\epsilon)}c_m\right)\left(\sum_{m=1}^{M}(\widetilde{q}_{mi'}^{(\epsilon)}-\widetilde{q}_{mi}^{(\epsilon)})c_m\right)\right],$$

which is same as the term in the left hand side of (C.3). Similarly, the right hand side term can be obtained. Expanding the products in (C.4), we get

$$\sum_{i=1}^{L}\sum_{i'=1}^{L}\left(\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}c_m\right)^2-\sum_{i=1}^{L}\sum_{i'=1}^{L}\left(\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}c_m\right)\left(\sum_{m=1}^{M}\widetilde{q}_{mi'}^{(\epsilon)}c_m\right)$$
$$\leq\sum_{m=1}^{M}\left[\sum_{i=1}^{L}\sum_{i'=1}^{L}\left(\widetilde{q}_{mi}^{(\epsilon)}\right)^2-\sum_{i=1}^{L}\sum_{i'=1}^{L}\widetilde{q}_{mi}^{(\epsilon)}\widetilde{q}_{mi'}^{(\epsilon)}\right]$$

$$L\sum_{i=1}^{L}\left(\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}c_m\right)^2-\left(\sum_{i=1}^{L}\sum_{m=1}^{M}\widetilde{q}_{mi}^{(\epsilon)}c_m\right)^2$$
$$\leq L\sum_{m=1}^{M}\sum_{i=1}^{L}\left(\widetilde{q}_{mi}^{(\epsilon)}\right)^2-\sum_{m=1}^{M}\left(\sum_{i=1}^{L}\widetilde{q}_{mi}^{(\epsilon)}\right)^2.$$

The claim is now proved.

REFERENCES

[1] EC2, http://aws.amazon.com/ec2/.

[2] AppEngine, http://code.google.com/appengine/.

[3] Azure, http://www.microsoft.com/windowsazure/.

[4] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE'08*, 2008, pp. 1–10.

[5] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et al., "Above the clouds: A Berkeley view of cloud computing," 2009, Tech. Rep. UCB/eeCs-2009-28, EECS department, U.C. Berkeley.

[6] D. A. Menasce and P. Ngo, "Understanding cloud computing: Experimentation and capacity planning," in *Proc. 2009 Computer Measurement Group Conf.*, 2009.

[7] 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.

[8] 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.

[9] K. Tsakalozos, H. Kllapi, E. Sitaridi, M. Roussopoulos, D. Paparas, and A. Delis, "Flexible use of cloud resources through profit maximization and price discrimination," in *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, 2011, pp. 75–86.

[10] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *Proc. IEEE Infocom.*, 2011, pp. 1098–1106.

[11] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. IEEE Infocom.*, 2011, pp. 71–75.

[12] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "Kingfisher: Cost-aware elasticity in the cloud," in *Proc. IEEE Infocom.*, 2011, pp. 206–210.

[13] 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.

[14] 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.

[15] 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.

[16] T. Bonald and D. Cuda, "Rate-optimal scheduling schemes for asynchronous input-queued packet switches," in *ACM Sigmetrics MAMA Workshop*, 2012.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," Technical Report, http://hdl.handle.net/2142/28577.

[22] S. Asmussen, *Applied Probability and Queues.* New York: Springer-Verlag, 2003.

[23] S. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability.* Cambridge University Press, 2009.

[24] M. Mitzenmacher, “The power of two choices in randomized load balancing,” Ph.D. dissertation, University of California at Berkeley, 1996.

[25] Y. T. He and D. G. Down, “Limited choice and locality considerations for load balancing,” *Performance Evaluation*, vol. 65, no. 9, pp. 670 – 687, 2008.

[26] H. Chen and H. Q. Ye, “Asymptotic optimality of balanced routing,” 2010, http://myweb.polyu.edu.hk/~lgtyehq/papers/ChenYe11OR.pdf.

[27] L. Tassiulas, “Linear complexity algorithms for maximum throughput in radionetworks and input queued switches,” in *Proc. IEEE Infocom.*, 1998.

[28] 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.

[29] 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.

[30] V. Venkataramanan and X. Lin, “On the queue-overflow probability of wireless systems: A new approach combining large deviations with Lyapunov functions,” *IEEE Trans. Inform. Theory*, 2013.

[31] D. Shah and J. Shin, “Randomized scheduling algorithm for queueing networks,” *The Annals of Applied Probability*, vol. 22, no. 1, pp. 128–171, 2012.

[32] J. Ghaderi and R. Srikant, “Flow-level stability of multihop wireless networks using only MAC-layer information,” in *WiOpt*, 2012.

[33] B. Hajek, “Hitting-time and occupation-time bounds implied by drift analysis with applications,” *Advances in Applied Probability*, pp. 502–525, 1982.

[34] S. T. Maguluri and R. Srikant, “Scheduling jobs with unknown duration in clouds,” in *INFOCOM, 2013 Proceedings IEEE*, 2013, pp. 1887–1895.

[35] S. T. Maguluri and R. Srikant, “Scheduling jobs with unknown duration in clouds,” to appear in IEEE/ACM Transactions on Networking.

[36] S. Karlin and H. M. Taylor, *A First Course in Stochastic Processes.* Academic Press, 1975.

[37] A. Eryilmaz and R. Srikant, “Asymptotically tight steady-state queue length bounds implied by drift conditions,” *Queueing Systems*, pp. 1–49, 2012.

[38] J. Ghaderi, T. Ji, and R. Srikant, "Connection-level scheduling in wireless networks using only MAC-layer information," in *INFOCOM*, 2012, pp. 2696–2700.

[39] T. Ji and R. Srikant, "Scheduling in wireless networks with connection arrivals and departures," in *Information Theory and Applications Workshop*, 2011.

[40] M. Bramson, "State space collapse with application to heavy-traffic limits for multiclass queueing networks," *Queueing Systems Theory and Applications*, pp. 89 – 148, 1998.

[41] R. J. Williams, "Diffusion approximations for open multiclass queueing networks: Sufficient conditions involving state space collapse," *Queueing Systems Theory and Applications*, pp. 27 – 88, 1998.

[42] M. I. Reiman, "Some diffusion approximations with state space collapse," in *Proceedings of International Seminar on Modelling and Performance Evaluation Methodology, Lecture Notes in Control and Information Sciences.* Berlin: Springer, 1983, pp. 209–240.

[43] J. M. Harrison, "Heavy traffic analysis of a system with parallel servers: Asymptotic optimality of discrete review policies," *Ann. App. Probab.*, pp. 822–848, 1998.

[44] J. M. Harrison and M. J. Lopez, "Heavy traffic resource pooling in parallel-server systems," *Queueing Systems*, pp. 339–368, 1999.

[45] S. L. Bell and R. J. Williams, "Dynamic scheduling of a parallel server system in heavy traffic with complete resource pooling: asymptotic optimality of a threshold policy," *Electronic J. of Probability*, pp. 1044–1115, 2005.

[46] A. Stolyar, "MaxWeight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic," *Adv. Appl. Prob.*, vol. 14, no. 1, 2004.

[47] J. F. C. Kingman, "Some inequalities for the queue GI/G/1," *Biometrika*, pp. 315–324, 1962.

[48] 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.

[49] S. T. Maguluri, R. Srikant, and L. Ying, "Heavy traffic optimal resource allocation algorithms for cloud computing clusters," *Performance Evaluation*, vol. 81, pp. 20–39, 2014.

[50] A. Mandelbaum and A. L. Stolyar, “Scheduling flexible servers with convex delay costs: heavy-traffic optimality of the generalized $c\mu$-rule,” *Operations Research*, vol. 52, no. 6, pp. 836–855, 2004.

[51] H. Q. Ye and D. D. Yao, “Utility-maximizing resource control: Diffusion limit and asymptotic optimality for a two-bottleneck model,” *Operations Research*, vol. 58, 2010.

[52] D. Shah, J. Tsitsiklis, and Y. Zhong, “Optimal scaling of average queue sizes in an input-queued switch: an open problem,” *Queueing Systems*, vol. 68, no. 3-4, pp. 375–384, 2011.