Low-Complexity Switch Scheduling Algorithms: Delay Optimality in Heavy Traffic

Prakirt Raj Jhunjhunwala, Siva Theja Maguluri
prakirt@gatech.edu, siva.theja@gatech.edu
Georgia Institute of Technology

***Abstract*—Motivated by applications in data center networks, in this paper, we study the problem of scheduling in an input queued switch. While throughput maximizing algorithms in a switch are well-understood, delay analysis was developed only recently. It was recently shown that the well-known MaxWeight algorithm achieves optimal scaling of mean queue lengths in steady state in the heavy-traffic regime, and is within a factor less than 2 of a universal lower bound. However, MaxWeight is not used in practice because of its high time complexity. In this paper, we study several low complexity algorithms and show that their heavy-traffic performance is identical to that of MaxWeight. We first present a negative result that picking a random schedule does not have optimal heavy-traffic scaling of queue lengths even under uniform traffic. We then show that if one picks the best among two matchings or modifies a random matching even a little, using the so-called flip operation, it leads to MaxWeight like heavy-traffic performance under uniform traffic. We then focus on the case of non-uniform traffic and show that a large class of low time complexity algorithms have the same heavy-traffic performance as MaxWeight, as long as it is ensured that a MaxWeight matching is picked often enough. We also briefly discuss the performance of these algorithms in the large scale heavy-traffic regime when the size of the switch increases simultaneously with the load. Finally, we perform empirical study on a new algorithm to compare its performance with some existing algorithms.**

***Index Terms*—Data Centers, MaxWeight, Scheduling, State Space Collapse, Power-of-d, Bipartite Matching**

I. INTRODUCTION

Input queued crossbar switches are essential components in building networks and have been studied since the 90's [1]. There is now renewed interest in studying input queued switches because they are good approximations of data center networks built using Clos topologies [2][3].

The throughput performance of various algorithms was studied in the past. It was shown in [1][4] that the celebrated MaxWeight algorithm maximizes throughput. However, implementing a MaxWeight algorithm involves computing a maximum weight bipartite matching at every time, which has a complexity of $O(n^{2.5})$ [5], which is impractical given the size of today's data center networks. Therefore, lower complexity algorithms that also maximize throughput were studied in [6][7][8][9][10]. A low complexity algorithm with distributed implementation is presented in [11].

While maximizing throughput is a first order metric and easy to study, the objective in a real world data center is to minimize delay. Due to Little's law, studying steady-state delay is the same as studying steady-state mean queue length. However, evaluating either of these is challenging in queueing systems. Therefore, they are studied in various asymptotic regimes such as heavy-traffic. The primary focus of this paper is heavy-traffic regime, where the switch is loaded close to its capacity. In this regime, the mean queue length goes to infinity, and we study the rate at which it goes to infinity by considering the sum of the queue lengths in heavy-traffic, multiplied by a heavy-traffic parameter (ϵ) that captures the distance to the capacity region.

Heavy-traffic queue length behavior under MaxWeight was recently studied in [12][13][14] and an exact expression for the heavy-traffic scaled mean sum queue lengths was obtained. Moreover, it was shown that the queue lengths are within a factor of 2 from a universal lower bound, thus establishing that MaxWeight has an optimal scaling. Moreover, using Little's law, this result implies that the mean delay is $O(1)$ independent of the size of the switch. This result was obtained in [12] using a novel drift method. The key step is to establish a state space collapse (SSC) result, which shows that in heavy traffic, the n^2 dimensional queue length vector lives close to a $(2n-1)$ dimensional cone. The main challenge here was due to the multidimensional nature of the SSC. The goal of this paper is to study low complexity scheduling algorithms that have MaxWeight like queue length performance on heavy-traffic, i.e., within a constant factor of the universal lower bound.

A. Main Contributions

We first consider the switch under uniform traffic and study random scheduling, where a matching is picked every time uniformly at random. We show in Section III-A that, under uniform traffic, while random scheduling achieves the maximum possible throughput, its heavy-traffic behavior is much worse. In particular, we show that for random scheduling, the heavy traffic scaled mean sum queue length is $\Theta(n^2)$, as opposed to $\Theta(n)$ for MaxWeight. This is because random scheduling does not exhibit state space collapse.

Then, in Section III-C, we study the power-of-d scheduling, where d matchings are picked uniformly at random and the best among them is used. We show that under uniform traffic, power-of-d scheduling not only maximizes throughput, but also has MaxWeight like heavy-traffic behavior. Inspired from [15], we further propose an algorithm that we call random d-flip , where one matching is sampled at random, and one tries to improve it by trying to flip two queues in the matching. We show that under uniform traffic, this is enough to get maximum throughput and MaxWeight like heavy-traffic behavior.

This research was supported in part by the NSF grant CCF 1850439.

TABLE I
RESULTS PRESENTED IN THIS PAPER

Algorithm	Throughput Optimality	$\lim_{\epsilon \downarrow 0} \epsilon \sum_{ij} \bar{q}_{ij}$	$\mathbb{E}[\sum_{ij} \bar{q}_{ij}] = O(n^{1+\beta})$ for β	Amortized Complexity	Reference
MaxWeight	Yes	$O(n)$	> 4	$O(n^{2.5})$	[12]
Random	Uniform traffic	$O(n^2)$	N/A	$O(n)$	Sec III-A
Power-of-d	Uniform traffic	$O(n)$	> 6	$O(dn)$	Sec III-C, Sec VI-A
Random d-Flip	Uniform traffic	$O(n)$	> 6	$O(n+d)$	Sec III-D, Sec VI-A
Bursty MaxWeight	Yes	$O(n)$	$> 3+\max(\gamma, 1)$	$O(n^{2.5}/m)$	Sec IV-B, Sec VI-A, [7]
Pipelined MaxWeight	Yes	$O(n)$	$> 3+\max(\gamma, 1)$	$O(n^{2.5})$, parallelizable	Sec IV-C, Sec VI-A, [7]
Randomly Delayed MaxWeight	Yes	$O(n)$	Unknown	$O(\delta n^{2.5})$	Sec V
Pick and Compare (PC-d)	Yes	$O(n)$	Unknown	$O(dn)$	Sec V
LAURA	Yes	$O(n)$	Unknown	$O(n \log^2 n)$	Sec V. [8]
SERENA	Yes	$O(n)$	Unknown	$O(n)$	Sec V, [8]
d-Flip	Unknown	Unknown	Unknown	$O(d)$	Sec VII

We then consider variants of MaxWeight algorithm under general traffic in Section IV. We show that bursty MaxWeight algorithm and pipelined MaxWeight algorithm [7] have the same heavy-traffic performance as MaxWeight. In bursty MaxWeight, a maximum weight matching is computed every m time steps, and the same matching is used for m steps. In pipelined MaxWeight, a maximum weight matching is computed at every time, but it takes m time steps to complete this computation, and so the matching is used only m steps later. This is amenable to a parallelized implementation. We present a general theorem that characterizes the heavy-traffic performance of a broad class of algorithms including both these algorithms.

We then consider another general class of linear complexity algorithms proposed by Tassiulas [6] that are shown to be throughput optimal. In these algorithms, at any time, there is a small δ chance of picking a MaxWeight matching. If not, a matching is sampled according to some distribution, and it is compared with the previous matching, and the best among the two is used. This framework was used in [8] and [9] to develop several low complexity algorithms including APSARA, SERENA and LAURA. We show in Section V that this large class of algorithms also has the same heavy-traffic behavior as MaxWeight.

While all the algorithms that we study have same heavy-traffic performance as that of MaxWeight, they all are not equally good in practice. This is because while heavy-traffic analysis is finer than throughput optimality, it does not capture subtle differences in performance. In particular, any algorithm that exhibits SSC has MaxWeight like heavy-traffic performance. However, different algorithms may have slightly different quality of SSC. In order to capture this performance difference, we consider the large system heavy-traffic regime [16][17][18][19] in Section VI-A. In this regime, size of the switch increases simultaneously while the traffic approaches the capacity, and we study the performance difference of the above algorithms in this regime.

All the results are summarized in Table I. In Section VII, we use simulations to exhibit the performance of the proposed algorithm d-Flip. We finally conclude in Section VIII, along with a few pointers on future research directions. We will now start with the model, notation and other preliminaries such as a formal definition of state-space collapse and heavy traffic optimality in Section II.

II. MODEL AND PRELIMINARIES

In this section, we present the model and introduce the required notation. Moreover, we present several known results from the previous literature. In any time slot t, $q_{ij}(t)$ (also called queue length) denotes the number of packets that need to be transferred from the input i to the output j, $\mathbf{q}(t)$ is a $n \times n$ queue length matrix with elements $q_{ij}(t)$. Throughout this paper, the letters in bold denotes vectors in $\mathbb{R}^{n \times n}$. Also, for any process $x(t)$ that converges in distribution, $\bar{x}$ denotes the limiting random variable to which $x(t)$ converges.

A. Arrival and Service Process

At any time t, $a_{ij}(t)$ ($\mathbf{a}(t)$ in matrix form) denotes the number of packets that arrive at the input port i to be delivered to output port j. The term matrix and vector are used interchangeably throughout the paper. The mean arrival rate vector is denoted by $\mathbb{E}[\mathbf{a}(t)] = \boldsymbol{\lambda}$ and variance $\text{Var}(\mathbf{a}(t)) = \boldsymbol{\sigma}^2$.

Assumption 1. *For the arrival process:*

(i) *For any given pair (i, j), $a_{ij}(t)$ are independent and identically distributed with respect to t.*

(ii) *The arrival process is also independent across input-output pair, i.e., for all i, j, i' and j' such that $(i, j) \neq (i', j')$, $a_{ij}(t)$ is independent of $a_{i'j'}(t)$.*

(iii) *There exists $a_{\max}$ such that $\forall i, j, t$, $a_{ij}(t) \leq a_{\max} < \infty$.*

(iv) *There is non-zero probability of no arrivals, i.e., $\mathbb{P}(\mathbf{a}(t) = \mathbf{0}) > 0$, where $\mathbf{0}$ is a $n \times n$ vector of all zeros.*

The assumptions mentioned in Assumption 1 are quite general for a switch system. Due to the structure of the switch system, in each time slot, each input can be matched with at most one output and vice-versa. The switch system can also be thought of as a complete bipartite graph with $2n$ nodes and n^2 edges. And the weight of each edge (i, j) is $q_{ij}(t)$. A schedule is then a matching on the corresponding graph, which is represented by a $n \times n$ matrix with entries either 0 or 1. We use $\mathbf{s}(t)$ to denote the schedule in time slot t. The element $s_{ij}(t) = 1$ if and only if the input i is connected with the output j at time t. In this paper, without loss of generality, we consider a schedule to be a perfect matching between input and output nodes, i.e., no more connections between input and

output nodes can be made. It follows that the set of possible schedules $\mathcal{X}$ is just the set of all $n \times n$ permutation matrices.

The weight of the schedule is the sum of the queue lengths that are being served in the given time slot. A scheduling algorithm or policy picks the schedule $\mathbf{s}(t)$ in every time slot. *MaxWeight* is a scheduling algorithm that always picks the schedule with the highest weight. If the algorithm picks schedules only from $\mathcal{X}$, it might happen that $s_{ij}(t)$ is 1 but there are no packets available to be transferred from input i to output j. In such a case, we say that the service is wasted. As a result, the queue length evolve according to the following equation,

$$\begin{aligned} q_{ij}(t+1) &= [q_{ij}(t) + a_{ij}(t) - s_{ij}(t)]^+ \\ &= q_{ij}(t) + a_{ij}(t) - s_{ij}(t) + u_{ij}(t), \end{aligned}$$

where $[x^+] = \max(0, x)$ and $u_{ij}(t)$ denotes the unused service on link (i, j). By writing this into matrix form, we get

$$\mathbf{q}(t+1) = \mathbf{q}(t) + \mathbf{a}(t) - \mathbf{s}(t) + \mathbf{u}(t)$$

It can be observed that if $q_{ij}(t+1) > 0$ then $u_{ij}(t) = 0$. This gives us the condition that, $q_{ij}(t+1)u_{ij}(t) = 0$ for all (i, j) which implies that $\langle \mathbf{q}(t+1), \mathbf{u}(t) \rangle = 0$. Let $\sigma(\mathcal{H}_t)$ be the σ-algebra generated by $\mathcal{H}_t$, where $\mathcal{H}_t$ denotes the history till time t, i.e.,

$$\mathcal{H}_t = \{\mathbf{q}(0), \mathbf{s}(0), \mathbf{q}(1), \ldots, \mathbf{s}(t-1), \mathbf{q}(t)\}. \tag{1}$$

Similarly, we define $\sigma(\tilde{\mathcal{H}}_t)$ to be the σ-algebra generated by $\tilde{\mathcal{H}}_t$, where

$$\tilde{\mathcal{H}}_t = \{\mathbf{q}(0), \mathbf{s}(0), \mathbf{q}(1), \ldots, \mathbf{s}(t-1), \mathbf{q}(t), \mathbf{s}(t)\}. \tag{2}$$

For an arbitrary scheduling algorithm, it not necessary that $\mathbf{q}(t)$ forms a Markov chain. For example, in Section V, we look at the algorithm named as randomly delayed MaxWeight, where the system uses the MaxWeight schedule with probability δ, and with probability $(1-\delta)$, it uses the schedule used in previous time slot. In such a case, the system needs to remember the schedule used in previous time slot and so using $\mathbf{q}(t)$ as the state of Markov chain is not enough. The correct definition for the state of the Markov chain in this case would be $(\mathbf{q}(t), \mathbf{s}(t))$. For the switch system considered in this paper, we assume that there is a process $X(t)$ such that $X(t)$ forms a Markov chain and we define two conditions on $X(t)$ as given below.

A.1. The Markov chain $X(t)$ is $\sigma(\tilde{\mathcal{H}}_t)$-measurable and it is also irreducible and aperiodic.

A.2. There exists a function $g(\cdot)$ such that $\mathbf{q}(t) = g(X(t))$. Further, let $\mathcal{A} \subset \mathbb{Z}^{n \times n}$ and suppose $g^{-1}(\mathcal{A}) = \{X : g(X) \in \mathcal{A}\}$. Then, if $|\mathcal{A}| < \infty$ then $|g^{-1}(\mathcal{A})| < \infty$, where $|\cdot|$ denotes the size of the set.

For all the algorithms considered in this paper, we prove that condition A.1 and A.2 are satisfied. The condition A.1 is required to use the Lyapunov's drift argument to establish the positive recurrence of the Markov chain $X(t)$. Note that irreducibility is not a major condition as otherwise, we can just consider the communicating class of $X(0)$ to be the state space. The first part of condition A.2 essentially says that $\mathbf{q}(t)$ is a deterministic function of the state $X(t)$, which implies that the state of the Markov chain holds full information about the queue length. Second part of condition A.2 is more technical. Overall condition A.2 is not very limiting because it is satisfied by most of the algorithms studied in the previous literature.

In this paper, we say that the switch system is *stable* if the corresponding Markov chain $X(t)$ is positive recurrent. The capacity region $\mathcal{C}$ of the switch is the set of mean arrival rate vector $\boldsymbol{\lambda}$ for which there exists some scheduling policy under which the switch system is stable. As given in [4], the capacity region for a switch, denoted by $\mathcal{C}$ is

$$\mathcal{C} = \Big\{\boldsymbol{\lambda} \in \mathbb{R}_+^{n \times n} : \sum_{i=1}^{n} \lambda_{ij} < 1, \sum_{j=1}^{n} \lambda_{ij} < 1 \ \forall i, j\Big\}.$$

An algorithm for which the the queue length vector $\mathbf{q}(t)$ is stable for all $\boldsymbol{\lambda} \in \mathcal{C}$ is called throughput optimal. In [20], it was proved that MaxWeight is throughput optimal.

The set $\mathcal{F}$ denotes the set of doubly stochastic matrices. The set $\mathcal{F}$ forms a facet [21, Chapter 3] of the closure of the capacity region $\mathcal{C}$. Throughout this paper, we use $\boldsymbol{\lambda}$ to denote a matrix in $\mathcal{C}$ and $\boldsymbol{\nu}$ to denote a matrix in $\mathcal{F}$.

A switch system is in heavy traffic regime if the mean arrival rate matrix is very close to the boundary of the capacity region. Note that for any $\boldsymbol{\lambda} \in \mathcal{C}$, there exists $\boldsymbol{\nu} \in \mathcal{F}$ and $\epsilon_{ij} \in [0, 1]$ such that $\lambda_{ij} = (1 - \epsilon_{ij})\nu_{ij}$. In order to make the theoretical analysis simpler, we take $\epsilon_{ij} = \epsilon$ for all (i, j). Otherwise we can pick an ϵ such that $\epsilon_{ij} \geq \epsilon \ \forall (i, j)$ and many of our upper bound results would still be valid. This is also called *Completely Saturated Case* in [12].

Assumption 2. *The mean arrival rate vector is $\boldsymbol{\lambda} = (1-\epsilon)\boldsymbol{\nu}$, for some $\boldsymbol{\nu} \in \mathcal{F}$ and $\epsilon \in (0, 1)$, such that*

$$\nu_{\min} \triangleq \min_{ij} \nu_{ij} > 0.$$

Also, there exists $\tilde{\boldsymbol{\sigma}}^2$ such that the variance $\boldsymbol{\sigma}^2 \to \tilde{\boldsymbol{\sigma}}^2$ as $\epsilon \downarrow 0$.

The parameter ϵ in Assumption 2 is a measure of how far $\boldsymbol{\lambda} \in \mathcal{C}$ is from the boundary $\mathcal{F}$. In this paper, we refer ϵ as the heavy traffic parameter. The switch system is in heavy traffic regime if ϵ is very close to 0.

From here onwards, we will assume that the arrival satisfies Assumption 1 and 2. Throughout the paper, ϵ denotes the distance of $\boldsymbol{\lambda}$ from its corresponding $\boldsymbol{\nu}$ as given in Assumption 2. Also, note that even though the parameters of the arrival process depends on ϵ, we do not attach ϵ to their symbols just to keep the notations simple.

An arrival process is said to be under *uniform traffic* if the mean arrival rate for every input-output pair is same, i.e. $\lambda_{ij} = \lambda_{i'j'}$ for all i, j, i' and j'. Also, even though the mean arrival rates are same, the variance might differ across the input-output node pairs. It is easy to observe that for an arrival process that is in the capacity region and under uniform traffic, the mean arrival rate lies in $\mathcal{C}^* \subset \mathcal{C}$ given by

$$\mathcal{C}^* = \Big\{\boldsymbol{\lambda} \in \mathbb{R}_+^{n \times n} : \lambda_{ij} < \frac{1}{n}, \ \forall i, j\Big\}.$$

Let $\mathbf{1}$ be an $n \times n$ matrix of all ones. If the uniform traffic arrival process satisfies Assumption 2, then we can take $\boldsymbol{\lambda} = \frac{1-\epsilon}{n}\mathbf{1}$. Furthermore, if the arrival process is *uniform Bernoulli*

traffic, i.e., the arrivals $a_{ij}(t)$ are Bernoulli random variables, then $\|\boldsymbol{\sigma}\|^2 = (1-\epsilon)(n-1+\epsilon)$, which gives $\lim_{\epsilon\downarrow 0}\|\boldsymbol{\sigma}\|^2 = \|\tilde{\boldsymbol{\sigma}}\|^2 = n-1$.

B. Geometry

Let $\mathbf{e}^i$ be an $n\times n$ matrix with i^{th} row being all ones and zeros everywhere else and $\tilde{\mathbf{e}}^j$ is a $n\times n$ matrix with j^{th} column begin all ones and zeros everywhere else. Consider the subspace $\mathcal{S}\subset\mathbb{R}^{n\times n}$ defined as,

$$\mathcal{S}=\Big\{\mathbf{x}:\mathbf{x}=\sum_i w_i\mathbf{e}^i+\sum_j \tilde{w}_j\tilde{\mathbf{e}}^j \text{ s.t. } w_i,\tilde{w}_j\in\mathbb{R}\ \forall i,j\Big\}.$$

We define the cone $\mathcal{K}$ to be the intersection of $\mathcal{S}$ with the positive orthant, i.e., $\mathcal{K}=\mathcal{S}\cap\mathbb{R}_+^{n\times n}$. The dimension of cone $\mathcal{K}$ is $2n-1$ as it is spanned by $2n-1$ independent vectors out of 2n vectors $\{\mathbf{e}^i\}$ and $\{\tilde{\mathbf{e}}^j\}$. For two matrices $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^{n\times n}$, $\langle\mathbf{x},\mathbf{y}\rangle$ denotes the Frobenius inner product and $\|\mathbf{x}\|=\sqrt{\langle\mathbf{x},\mathbf{x}\rangle}$.

For any vector $\mathbf{x}$, $\mathbf{x}_\parallel$ denotes the projection to the space $\mathcal{S}$ with $\mathbf{x}_\perp=\mathbf{x}-\mathbf{x}_\parallel$. Similarly, $\mathbf{x}_{\parallel\mathcal{K}}$ denotes the projection to the cone $\mathcal{K}$ with $\mathbf{x}_{\perp\mathcal{K}}=\mathbf{x}-\mathbf{x}_{\parallel\mathcal{K}}$. Some important properties regarding the set $\mathcal{S}$ and $\mathcal{K}$ are provided in Appendix A (in the supplement file).

C. State-space collapse

The main workhorse in heavy-traffic analysis is state-space collapse, viz., the phenomenon that a queueing system in heavy-traffic behaves like a system with a smaller number of queues. It was shown in [12] that in the switch system operating under MaxWeight scheduling algorithm, the state $\mathbf{q}(t)$ (of dimension n^2) collapses to the cone $\mathcal{K}$ (of dimension $2n-1$). This was established by showing that in steady state, $\mathbf{q}_{\perp\mathcal{K}}$ is significantly smaller than $\mathbf{q}_{\parallel\mathcal{K}}$. The following definition presents this notion of state space collapse more formally.

Definition 1. *A scheduling algorithm is said to achieve State-Space Collapse (SSC) if the switch system is stable, the corresponding Markov chain $X(t)$ satisfies condition A.1 and A.2 and there exists $\epsilon_0>0$ such that for $0<\epsilon\le\epsilon_0$, the steady state queue length vector satisfies*

$$\mathbb{E}\Big[\,\|\bar{\mathbf{q}}_{\perp\mathcal{K}}\|^r\,\Big]\le C_r\quad\forall r\in\{1,2,\dots\},\tag{3}$$

where C_r is a constant, independent of ϵ.

Theorem 2. *Consider a switch system which achieves state-space collapse according to Definition 1, then the heavy traffic scaled queue length satisfies*

$$\lim_{\epsilon\downarrow 0}\epsilon\mathbb{E}\Big[\sum_{ij}\bar{q}_{ij}\Big]=\Big(1-\frac{1}{2n}\Big)\|\tilde{\boldsymbol{\sigma}}\|^2.\tag{4}$$

The proof of the result in Eq. (4) for MaxWeight was given in [12]. However, the proof in [12, Theorem 1] implies that Eq. (4) holds for any scheduling algorithm that satisfies SSC as given by Definition 1. Now, we formally define the term *MaxWeight-like*.

Definition 3. *For a switch scheduling algorithm, its heavy traffic performance is said to be MaxWeight-like if the algorithm satisfies Eq.* (4).

According to Theorem 2, to show that an algorithm is *MaxWeight-like* it is enough to prove that the algorithm achieves the SSC according to Definition 1. Although MaxWeight satisfies Eq. (4), there might exist algorithms that perform better than MaxWeight in heavy traffic. As mentioned in [12], we only know that for any scheduling algorithm,

$$\epsilon\mathbb{E}\Big[\sum_{ij}\bar{q}_{ij}\Big]\ge\frac{1}{2}\|\boldsymbol{\sigma}\|^2-\frac{\epsilon(1-\epsilon)}{2}\xrightarrow{\epsilon\downarrow 0}\frac{1}{2}\|\tilde{\boldsymbol{\sigma}}\|^2.\tag{5}$$

This means that the heavy traffic scaled mean sum queue length for MaxWeight is within a factor of 2 of the optimal. In [22], the authors presented an algorithm which performs better than MaxWeight, although they did not provide the heavy traffic limit for it.

D. Lyapunov Drift

We use Lyapunov drift arguments to obtain the heavy-traffic results in this paper. To that end, in this subsection, we present some Lyapunov functions, their drift and some known results on Lyapunov drift.

Let $X(t)$ be an irreducible and aperiodic Markov chain over a countable state space $\mathcal{A}$. Suppose $Z:\mathcal{A}\to\mathbb{R}_+$ is a non-negative Lyapunov function. The drift of Z at X is the change in the value of $Z(\cdot)$ after one step transition. Mathematically,

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

where $\mathcal{I}(\cdot)$ is the indicator function. We define three different conditions on the drift:

C.1. There exists $\eta>0$ and $\kappa<\infty$ such that $\forall t>0$ and $\forall X\in\mathcal{A}$ with $Z(X)\ge\kappa$,

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

C.2. There exists $D<\infty$ such that $\forall X\in\mathcal{A}$,

$$\mathbb{P}\big(|\Delta Z(X)|\le D\big)=1.$$

C.3. There exists a non-negative random variable M such that $|\Delta Z(X)|$ is stochastically dominated by M for all $t\ge 0$, i.e., for any $c>0$,

$$\mathbb{P}\big(|\Delta Z(X)|>c\big|X(t)=X\big)\le\mathbb{P}(M>c)\quad\forall t\ge 0,$$

and $\mathbb{E}[e^{\theta M}]<\infty$ for some $\theta>0$.

It is easy to observe that the condition C.2 is stronger than condition C.3. We define C.2 and C.3 differently because we can state a stronger result if the condition C.2 holds. Some important results related to the drift analysis of switch system is given in Appendix B (in the supplement file).

III. Class 1: Modifications of Random Scheduling

In this section, we study random scheduling and some modifications of it. For a switch system, random scheduling is not throughput optimal. The capacity region of random scheduling is known to be $\mathcal{C}^*$. Throughout this section, we assume that the arrival process is under uniform traffic. We

show that heavy traffic behaviour of random scheduling is not MaxWeight-like, but there are some variants of random scheduling which have MaxWeight-like heavy traffic behaviour.

A. Random Scheduling

Random scheduling, as the name suggests, is a scheduling policy for which the schedule $\mathbf{s}(t)$ is chosen uniformly at random from the set of permutation matrices $\mathcal{X}$. The time complexity of generating a random schedule is $O(n)$ by using Fisher–Yates shuffle [23, Example 12].

Proposition 4. *Consider a switch system under uniform traffic. For random scheduling, the process $\mathbf{q}(t)$ forms a positive recurrent Markov chain and,*

$$\lim_{\epsilon \downarrow 0} \epsilon \mathbb{E}\Big[\sum_{ij} \bar{q}_{ij}\Big] = \frac{n}{2}\left\|\tilde{\boldsymbol{\sigma}}\right\|^2 + \frac{n(n-1)}{2}. \tag{6}$$

Moreover, if the arrival process is uniform Bernoulli traffic,

$$\lim_{\epsilon \downarrow 0} \epsilon \mathbb{E}\Big[\sum_{ij} \bar{q}_{ij}\Big] = n(n-1). \tag{7}$$

From Theorem 2, we know that any scheduling algorithm that satisfies SSC has optimal queue length scaling of $O(\|\tilde{\boldsymbol{\sigma}}\|^2)$ in heavy traffic. While from Proposition 4, the heavy traffic scaled mean sum queue length for random scheduling is $O(n\|\tilde{\boldsymbol{\sigma}}\|^2)$. This shows that random scheduling does not have optimal queue length scaling.

Proposition 4 under uniform Bernoulli traffic was presented in [7, Theorem 2], and was proved by noting that under random scheduling, each of the n^2 queues of the switch can be analyzed separately by assuming that the service to each of the queue is a Bernoulli random variable with success probability $1/n$. The proof for general traffic can be shown similarly, and we present the details in Appendix C (in the supplement file) for completeness.

B. State space collapse

In this section, we will present the heavy-traffic results for a class of scheduling algorithms. Later on we provide some examples that lie in this class like power-of-d, and random d-flip scheduling algorithms that are modification of random scheduling.

Definition 5. *A scheduling algorithm lies in class $\Pi_1(\boldsymbol{\nu})$ if the corresponding Markov chain $X(t)$ satisfies condition A.1 and A.2 and there exists a constant $W_1 > 0$ such that in any time slot $t \geq 0$, the expected weight satisfies*

$$\mathbb{E}[\langle \mathbf{q}(t), \mathbf{s}(t)\rangle | X(t) = X] \geq \langle \mathbf{q}, \boldsymbol{\nu}\rangle + W_1 \left\|\mathbf{q}_{\perp\mathcal{K}}\right\|, \tag{8}$$

where W_1 is independent of ϵ and $\mathbf{q} = g(X)$.

MaxWeight lies in class $\Pi_1(\boldsymbol{\nu})$ for any $\boldsymbol{\nu}$ for which $\nu_{\min} > 0$, in which case $W_1 = \nu_{\min}$ [12]. We later on show that, power-of-d and random d-flip scheduling lies in $\Pi_1\left(\frac{1}{n}\mathbf{1}\right)$ with $W_1 = \frac{1}{2n^3}$. Also, it is easy to observe that random scheduling does not lie in class $\Pi_1\left(\frac{1}{n}\mathbf{1}\right)$. Next, we claim that any scheduling algorithm that lies in class $\Pi_1(\boldsymbol{\nu})$ satisfies SSC if the mean arrival rate is $\boldsymbol{\lambda} = (1-\epsilon)\boldsymbol{\nu}$.

Theorem 6. *Suppose the mean arrival rate is of the form $\boldsymbol{\lambda} = (1-\epsilon)\boldsymbol{\nu}$ and the scheduling algorithm lies in the class $\Pi_1(\boldsymbol{\nu})$. Then, the scheduling algorithm achieves SSC and so its heavy traffic behaviour is MaxWeight-like.*

The proof of Theorem 6 follows by showing that if we pick the Lyapunov function to be $\|\mathbf{q}_{\perp\mathcal{K}}\|$, then this Lyapunov function satisfy the conditions C.1 and C.2. After that we can use existing results to show that all the moments of $\|\mathbf{q}_{\perp\mathcal{K}}\|$ are bounded by a constant. This implies that scheduling algorithm achieves SSC according to Definition 1 and so it is heavy traffic behaviour is MaxWeight-like. The details of the proof are provided in Appendix D (in the supplement file). Next, we provide some examples of the algorithms that lie in this class.

C. Power-of-d scheduling

The *power-of-d* scheduling is a variant of random scheduling in which the system samples $d \geq 2$ schedules uniformly at random with replacement from the set of permutation matrices $\mathcal{X}$ and chooses the one with the largest weight. In time slot t, let $\{\mathbf{s}_1(t), \ldots, \mathbf{s}_d(t)\}$ denotes the schedules sampled by the power-of-d algorithm. The schedule chosen by power-of-d is

$$\mathbf{s}(t) = \arg\max\{\langle \mathbf{q}(t), \mathbf{s}_1(t)\rangle, \ldots, \langle \mathbf{q}(t), \mathbf{s}_d(t)\rangle\}.$$

We assume that schedules are sampled with replacement just for simplicity. The results does not change qualitatively even if the schedules are sampled without replacement. Generating a random schedule has a time-complexity of $O(n)$. And as power-of-d generated d random schedule times, the time complexity of power-of-d is $O(dn)$.

It is known that power-of-d scheduling is not throughput optimal [8]. However, it is stable under all the arrival rates in $\mathcal{C}^*$, and so we can study its heavy traffic behavior under uniform traffic.

D. Random d-Flip scheduling

Random d-flip scheduling algorithm is another variant of random scheduling. For any given schedule $\mathbf{s}_1$ and a queue length matrix $\mathbf{q}$, a *flip step* constitutes of following three steps,

- Sample two indices (i,j) and (k,l) uniformly at random such that $s_{1,ij} = s_{1,kl} = 1$, where $s_{1,ij}$ is the $(i,j)^{th}$ element of the schedule $\mathbf{s}_1$.
- Create a different schedule $\mathbf{s}_2$ such that $s_{2,ij} = s_{2,kl} = 0$ and $s_{2,il} = s_{2,kj} = 1$.
- Select the schedule with the larger weight, i.e.,

$$\mathbf{s} = \arg\max_{\mathbf{s}_1, \mathbf{s}_2}\{\langle \mathbf{q}, \mathbf{s}_1\rangle, \langle \mathbf{q}, \mathbf{s}_2\rangle\}.$$

Note that to compare the weight of the matching $\mathbf{s}_1$ and $\mathbf{s}_2$ in the flip step, the system does not need to calculate the weight of the schedule. It suffices to compare the value of $q_{ij} + q_{kl}$ and $q_{il} + q_{kj}$. Thus the flip step has a complexity of only $O(1)$.

In each time slot t, random d-flip samples a schedule $\mathbf{s}(t)$ uniformly at random from the set $\mathcal{X}$ and then uses the flip step on $\mathbf{s}(t)$, d times consecutively. As the complexity of generating a random schedule is $O(n)$ and complexity of flip step is $O(1)$, the complexity of random d-flip is $O(n+d)$.

The flip step considered in this paper is random flipping and it is not necessary that flip step improves the schedule generated by random sampling, but there are more ways to implement the flip step. In [15], authors provide another method of implementing the flip step, which strictly improves the weight of the schedule but the complexity of each flip step is $O(n)$. The algorithm APSARA in [8] is also based on flip step mentioned above.

Lemma 7. *In any time slot t, the schedule chosen by power-of-d or by random d-flip satisfies,*

$$\mathbb{E}\big[\langle \mathbf{q}(t), \mathbf{s}(t)\rangle \big| \mathbf{q}(t) = \mathbf{q}\big] \geq \frac{1}{n}\langle \mathbf{q}, \mathbf{1}\rangle + \frac{1}{2n^3}\left\|\mathbf{q}_{\perp\mathcal{K}}\right\|. \quad (9)$$

Note that for random scheduling, the expected weight is $\frac{1}{n}\langle \mathbf{q}, \mathbf{1}\rangle$. Power-of-$d$ generates more schedules to improve the weight. We show that in expectation, this improvement is at least $\frac{1}{2n^3}\left\|\mathbf{q}_{\perp\mathcal{K}}\right\|$. The detailed proof of Lemma 7 for power-of-d is provided in Appendix E (in the supplement file).

Similarly, the algorithm random d-flip first samples a random schedule and then implements the flip steps that strictly improves the expected weight. We prove that the expected improvement by the first flip step is at least $\frac{1}{2n^3}\left\|\mathbf{q}_{\perp\mathcal{K}}\right\|$. The details of the proof of Lemma 7 for random d-flip is provided in Appendix F (in the supplement file).

Proposition 8. *Under uniform traffic, power-of-d scheduling and random d-flip achieve SSC and so their heavy traffic behaviour is MaxWeight-like.*

Proof. For power-of-d and random d-flip, the process $\mathbf{q}(t)$ forms a Markov chain and satisfy condition A.1 and A.2. The aperiodicity of Markov chain $\mathbf{q}(t)$ in this case follows from part (iv) of Assumption 1, as the state $\mathbf{q} = \mathbf{0}$ has a self loop. And irreducibility follows by taking the state space to be the set of states reachable from $\mathbf{0}$ as given in [24, Exercise 4.2]. Also, it is evident that the chain $\mathbf{q}(t)$ satisfies condition A.2.

From Lemma 7, we know that power-of-d and random d-flip lies in the class $\Pi(\frac{1}{n}\mathbf{1})$ as given in Definition 5. Then, Proposition 8 follows directly from Theorem 6. □

So far, we considered uniform traffic since power-of-d and random d-flip scheduling algorithms are not throughput optimal. The switch is unstable under general non-uniform traffic under these algorithms. One way to overcome this limitation is by using a load-balanced switch [25]. A load balanced switch is a two-stage architecture consisting of two switches in tandem. The first stage aims to equalize the arrival rate across the inputs of the switch at the second stage, so that the second stage is operating under uniform traffic. The on-line complexity of operating the first switch is just $O(1)$ so it does not affect the overall performance. More details regarding the load balanced setup can be found in [24] and [25].

IV. Class 2: Approximate MaxWeight

In this section, we will present another class of scheduling policies that achieves SSC and so are heavy traffic optimal. In [7], the authors present two efficient approximations of the MaxWeight named bursty MaxWeight and pipelined MaxWeight. Next, we define a class of algorithm that contains these two algorithms, and provide heavy result for that class.

A. State space collapse

Now we prove the heavy traffic result for a class of algorithms which includes bursty and pipelined MaxWeight.

Definition 9. *A scheduling algorithm lies in class Π_2 if the corresponding Markov chain $X(t)$ satisfies condition A.1 and A.2 and there exists a constant $W_2 \geq 0$ such that in any time slot $t \geq 0$, the expected weight satisfies*

$$\mathbb{E}\big[\langle \mathbf{q}(t), \mathbf{s}(t)\rangle \big| X(t) = X\big] \geq \max_{\mathbf{s}} \langle \mathbf{q}, \mathbf{s}\rangle - W_2, \quad (10)$$

where W_2 is independent of ϵ and $\mathbf{q} = g(X)$.

The class Π_2 presented in Definition 9 is based on the class of algorithms presented in [7]. MaxWeight lies in class Π_2 with $W_2 = 0$. In [7], it was proved that any scheduling algorithm in the class Π_2 is throughput optimal. Next, we look at the SSC and heavy traffic optimality of scheduling algorithms in class Π_2.

Theorem 10. *Any scheduling algorithm that lies in the class Π_2 achieves SSC and so its heavy traffic behaviour is MaxWeight-like.*

Theorem 10 shows that bursty MaxWeight and pipelined MaxWeight satisfies SSC and thus their heavy traffic behaviour is MaxWeight-like. The proof of Theorem 10 follows on similar lines as the proof of Theorem 6. From the definition of the algorithms in class Π_2, the weight of schedule for any scheduling algorithm in Π_2 is at most a constant difference away from MaxWeight. If the queue lengths are very large (like in heavy traffic), the weight of the MaxWeight schedule is much larger compared to difference W_2 and so the performance of the scheduling algorithm is quite close to that of MaxWeight. Thus, the heavy traffic performance of algorithms in Π_2 is similar to MaxWeight scheduling. The proof of Theorem 10 is provided in Appendix G (in the supplement file).

B. Bursty MaxWeight

This scheduling algorithm evaluates the MaxWeight schedule after every m time-slots and then uses the same schedule consecutively for next m time slots. For a $n \times n$ switch, the time-complexity of computing the MaxWeight schedule is $O(n^{2.5})$. Thus, the amortized time-complexity of bursty MaxWeight is $O(n^{2.5}/m)$. Note that if m is chosen to be $\Theta(n^{2.5})$, this leads to a constant amortized complexity.

C. Pipelined MaxWeight

This takes m time slots to compute the MaxWeight schedule, so the MaxWeight schedule corresponding to $\mathbf{q}(t)$ is used in time slot $t + m$. While pipelined MaxWeight still has a high complexity of $O(n^{2.5})$, it is amenable to a parallelized implementation which makes it useful in practice.

Proposition 11. *Bursty and Pipelined MaxWeight achieve SSC and so their heavy traffic behaviour is MaxWeight-like.*

Proof. The proof of condition A.1 and A.2 for both algorithms are given in Appendix H (in the supplement file). For both algorithms, as shown in [7],

$$\langle \mathbf{q}(t), \mathbf{s}(t)\rangle \geq \max_{\mathbf{s}} \langle \mathbf{q}(t), \mathbf{s}\rangle - 2mna_{\max}.$$

Thus, bursty and pipelined MaxWeight lies in the class Π_2 as given in Definition 9 (Section IV-A) and then by using Theorem 10, both algorithms satisfy SSC and so their heavy traffic behaviour is MaxWeight-like. □

Both bursty MaxWeight and pipelined MaxWeight depend on the parameter m. Even though the result in Proposition 11 holds for any value of m, it does not mean that the heavy traffic performance of bursty MaxWeight or pipelined MaxWeight is not affected by the value of m. The larger the value of m, the further away these algorithms are from MaxWeight. Later, in Section VI-A, we provide an intuition of the effect of m on the heavy traffic behavior of the switch.

V. Class 3: Randomized Algorithms with Memory

In this section, we look at the third class of algorithms that satisfies SSC. The description of the class is as follows.

Definition 12. *A scheduling algorithm lies in class Π_3 if the corresponding Markov chain $X(t)$ satisfy condition A.1 and A.2 and*

(i) *There exists a $\delta > 0$ such that for every time $t \geq 0$ the chosen schedule $\mathbf{s}(t)$ satisfies*

$$\mathbb{P}\big(\langle \mathbf{q}(t), \mathbf{s}(t)\rangle = \max_{\mathbf{s}} \langle \mathbf{q}(t), \mathbf{s}\rangle \big| \mathcal{H}_t\big) \geq \delta, \qquad (11)$$

where $\mathcal{H}_t$ is given by Eq. (1)*.*

(ii) *For every time $t \geq 1$, the chosen schedule $\mathbf{s}(t)$ satisfies*

$$\langle \mathbf{q}(t), \mathbf{s}(t)\rangle \geq \langle \mathbf{q}(t), \mathbf{s}(t-1)\rangle. \qquad (12)$$

(iii) *There exists a deterministic function $f(\cdot)$, such that $(\mathbf{q}(t), \mathbf{s}(t)) = f(X(t))$.*

It is easy to observe that MaxWeight scheduling lies in Π_3 with $\delta = 1$. The definition of class Π_3 in this paper is based on the class of algorithms presented in [6]. The algorithms presented in [6] uses a two-step procedure to choose the schedule $\mathbf{s}(t)$.

- *Sampling step:* The system samples a schedule $\tilde{\mathbf{s}}(t)$ which satisfies Eq. (11).
- *Comparison step:* The sampled schedule $\tilde{\mathbf{s}}(t)$ is compared with $\mathbf{s}(t-1)$, i.e.,

$$\mathbf{s}(t) = \underset{\tilde{\mathbf{s}}(t), \mathbf{s}(t-1)}{\arg\max} \big\{ \langle \mathbf{q}(t), \tilde{\mathbf{s}}(t)\rangle, \langle \mathbf{q}(t), \mathbf{s}(t-1)\rangle \big\}.$$

Any scheduling algorithm that uses the above mentioned steps satisfies Eq. (11) and Eq. (12). Note that the complexity of the comparison step is $O(n)$, so the comparison step does not affect the complexity of the algorithm with worse than linear time complexity. The comparison step is very useful because it plays a key role in making the scheduling algorithm throughput optimal. Some of the algorithms based on the procedure given in [6] are as follows,

- *Randomly Delayed MaxWeight:* This is randomized version of bursty MaxWeight. The system chooses to implement MaxWeight with probability δ or uses the previous schedule with probability $1-\delta$. The amortized complexity of this algorithm is $O(\delta n^{2.5})$.
- *Pick and Compare (PC-d):* This algorithm is an extension of power-of-d. In this algorithms, the system generates the random schedule $\tilde{\mathbf{s}}(t)$ during the sampling step using power-of-d and then uses the comparison step. In this case, δ can be taken to be $d/n!$ and the complexity of pick and compare or PC-d is $O(dn)$.
- *LAURA and SERENA:* In [9] and [8], the authors presented several low-complexity algorithms, like LAURA (complexity $O(n \log^2 n)$) and SERENA (complexity $O(n)$), that also lie in the class Π_3 under the assumption that the arrival process is Bernoulli.

Part (iii) of Definition 12 is an extension of condition A.2, as here we also need the information about schedule to be a part of the state of the Markov chain. This condition is satisfied by the example algorithms presented above. More complicated algorithms such as non-Markovian scheduling policies may not satisfy this assumption. The corresponding Markov chain for algorithms mentioned above and the class of algorithms in [6] is given by $X(t) = (\mathbf{q}(t), \mathbf{s}(t))$, and it can be observed that $X(t)$ satisfy condition A.1 and A.2, so they also satisfy Part (iii) of Definition 12. Thus, the class of algorithms in [6] also lies in class Π_3. By the arguments presented in [6], for any scheduling algorithm that lies in class Π_3, the queue length process $\mathbf{q}(t)$ is stable, so we skip the proof of stability here.

Theorem 13. *Suppose the scheduling algorithm lies in the class Π_3. Then, the process $\mathbf{q}(t)$ is stable. Also, the scheduling algorithm achieves SSC and its heavy traffic behaviour is MaxWeight-like.*

Theorem 13 shows that the class of scheduling algorithms presented in [6] achieves SSC and have the same heavy traffic scaled queue length as MaxWeight. Therefore, the same result holds for the algorithms presented in [9].

Let $\{T_k\}_{k\geq 0}$ be the sequence time instants at which the chosen schedule matches with the MaxWeight schedule, i.e.,

$$\langle \mathbf{q}(T_k), \mathbf{s}(T_k)\rangle = \max_{\mathbf{s}} \langle \mathbf{q}(T_k), \mathbf{s}\rangle.$$

As $(\mathbf{q}(t), \mathbf{s}(t)) = f(X(t))$, it follows that $\{T_k\}_{k\geq 0}$ form a sequence of stopping times for the Markov chain $X(t)$. We define another process $\{Y_k\}_{k\geq 0}$ such that $Y_k = X(T_k)$. As $\{T_k\}_{k\geq 0}$ are stopping times, by strong Markov property, Y_k forms a Markov chain. Also, for $Y_k = Y$, take $(\mathbf{q}, \mathbf{s}) = f(Y)$, and by the construction of $\{Y_k\}_{k\geq 0}$, $\mathbf{s}$ is the MaxWeight schedule corresponding to $\mathbf{q}$.

Lemma 14. *For any $k \geq 1$, let $\tau_k = T_{k+1} - T_k$. Then, for any Y, the random variable $\{\tau_k | Y_k = Y\}$ is stochastically dominated by a random variable M which is Geometrically distributed with mean $1/\delta$.*

The proof of Lemma 14 follows from part (i) of Definition 12. The idea is that in any time slot, there is at least δ probability that the MaxWeight schedule is picked. So, we can bound the probability $\mathbb{P}(\tau_k > c | Y_k = Y)$ with $\mathbb{P}(M > c)$. The proof of Lemma 14 is provided in Appendix I (in the supplement file). Next, we provide the proof sketch for Theorem 13.

The idea behind the proof of Theorem 13 is that the scheduling algorithms lying in Π_3 chooses the MaxWeight schedule frequently. In fact, Lemma 14 shows that the time difference between choosing the two MaxWeight schedules is

stochastically dominated by a geometrically distributed random variable.

Once the MaxWeight schedule is chosen, the weight of the schedule chosen by the scheduling algorithm in subsequent time slots is not much worse than the weight of the MaxWeight schedule in those time slot. This happens because the queue lengths cannot deviate too much in a single time slot (as arrivals are bounded) and the algorithm tries to improve upon the schedule used in previous time slot.

Mathematically, we use the Lyapunov drift argument on the Markov chain $\{Y_k\}_{k\geq 0}$. We show that for $\{Y_k\}_{k\geq 0}$, if we choose the Lyapunov function to be $\|\mathbf{q}_{\perp\mathcal{K}}\|$, then this Lyapunov function satisfies the condition C.1 and C.3. The detailed proof of Theorem 13 is provided in Appendix J (in the supplement file).

VI. Comparison of algorithms

So far, we studied the performance of three classes of algorithms, and saw that they all have MaxWeight like heavy-traffic performance. In this section, we present a comparison and contrast them and Table I presents a summary. The three classes are clearly not disjoint. For instance, as mentioned before, MaxWeight lies in all three classes. It is not hard to construct other algorithms that lie in all three classes. One can combine multiple scheduling algorithms to generate a new algorithm. The algorithm PC-d (Section V) was created by implementing an extra comparison step after doing power-of-d. So PC-d lies in both $\Pi_1(\frac{1}{n}\mathbf{1})$ and Π_3. A similar modification can be done with random d-flip without changing the complexity of the algorithm. One can also create a scheduling algorithm that generates two schedules, one by bursty MaxWeight and other by PC-d and then chooses the one with larger weight. Such an algorithm will lie in all three classes.

While we proved that they all have MaxWeight-like heavy-traffic mean delay performance, their performance under other metrics can be different. First consider throughput optimality. As mentioned before, while the algorithms in class Π_2 and class Π_3 are throughput optimal, the algorithms in $\Pi_1(\boldsymbol{\nu})$ are not. Power-of-d and random d-flip are known to be stable only in a subset of the capacity region $\mathcal{C}^*$. In particular, they support maximum possible load only when the traffic is uniform. There are known examples [8, Theorem 1] showing how they are unstable under non-uniform load for certain arrival rate vectors within the capacity region. In the next subsection, we present the large scale heavy traffic regime, which is yet another asymptotic performance view, that enables us to distinguish between the performance of the algorithms presented so far.

A. Large scale heavy traffic regime

Stochastic networks such as an input queued switch are in general hard to analyze and so, are usually studied in various asymptotic regimes with heavy-traffic being a prominent one that is the main focus of this paper. In the heavy-traffic regime, we fix the size of the switch n, and load it to its maximum capacity, i.e., we let the heavy-traffic parameter $\epsilon \to 0$. Another popular regime is the large scale limit, where the load is fixed (ϵ is fixed) and the size of the system, n is sent to infinity [26]. Different regimes present different view points of the system, and obtaining results in various regimes presents a more holistic view. For example, several algorithms that have the same performance in one regime may have different performance in another regime.

In this section, we consider a whole spectrum of asymptotic regimes between the large scale regime and the heavy-traffic regime, where the size of the switch simultaneously grows to infinity as the arrival rate approaches the boundary of the capacity region. These are called the *large scale heavy traffic regime*. These regimes are of special interest today, since the size of today's data center networks is huge. For a given n, let the $\epsilon(n)$ denote the heavy traffic parameter of the system such that $\epsilon(n)$ is $\Omega(n^{-\beta})$ for some $\beta > 0$. As $\beta \to \infty$, one can heuristically think of this as the heavy-traffic regime studied in the previous sections. To see this, note that we have that n is $O(\epsilon^{-1/\beta})$. So, when $\beta \to \infty$, n does not scale as $\epsilon \to 0$. Thus, the heavy-traffic regime where n is constant and $\epsilon \to 0$ can be thought of as a special case of large scale heavy traffic regime when $\beta \to \infty$. Similarly, one can think of the special case of $\beta \to 0$ as corresponding to the large scale regime. Also, for simplicity, we assume that the arrival process is uniform Bernoulli traffic in this section. This means that $\boldsymbol{\lambda} = \frac{(1-\epsilon(n))}{n}\mathbf{1}$ and $\|\boldsymbol{\sigma}\|^2 = (1-\epsilon(n))(n-1+\epsilon(n)) = n-1+o(n)$. In this case, from the universal lower bound given in Eq. (5), we know that if $\epsilon(n) = \Omega(n^{-\beta})$ then, $\mathbb{E}\big[\sum_{ij}\bar{q}_{ij}\big]$ is $\Omega(n^{1+\beta})$ for any value of $\beta > 0$. Therefore, a natural question is if there is an algorithm under which, we can also obtain an upper bound that is $O(n^{1+\beta})$ for all $\beta > 0$. At this point, while this is still an open question [19], it is known [12, Corollary 1] that under MaxWeight algorithm, $\mathbb{E}\big[\sum_{ij}\bar{q}_{ij}\big]$ is $\Theta(n^{1+\beta})$ for $\beta > 4$. More precisely, for $\beta > 4$, under MaxWeight algorithm,

$$\lim_{n\to\infty} \frac{\epsilon(n)}{n}\mathbb{E}\Big[\sum_{ij}\bar{q}_{ij}\Big] = 1. \tag{13}$$

Theorem 15. *Consider a switch system under uniform Bernoulli traffic such that $\lambda = \frac{(1-\epsilon(n))}{n}\mathbf{1}$, and $\epsilon(n)$ is $\Omega(n^{-\beta})$. Then, we have Eq.* (13)

(i) *under algorithms in class $\Pi_1(\frac{1}{n}\mathbf{1})$ for $\beta > 3+\alpha_1$, where $\alpha_1 > 0$ and $W_1 = O(n^{-\alpha_1})$. In particular, power-of-d and random d-flip algorithms satisfy Eq.* (13) *for $\beta > 6$.*

(ii) *under algorithms in class Π_2 for $\beta > 2 + \max\{\alpha_2, 2\}$, where $\alpha_2 > 0$ and $W_2 = O(n^{\alpha_2})$. In particular, bursty MaxWeight and pipelined MaxWeight algorithms satisfy Eq.* (13) *for $\beta > 3 + \max\{\gamma, 1\}$, where m is $O(n^{\gamma})$.*

The proof of the theorem is presented in Appendix K (in the supplement file). The main tool to prove results of the form Eq. (13) in general, and the above theorem in particular is the following result that exploits a finer handle on the state space collapse, which follows from [12, Corollary 1]. The details are also provided in Appendix L (in the supplement file).

Lemma 16. *Consider a switch system under uniform Bernoulli traffic such that $\lambda = \frac{(1-\epsilon(n))}{n}\mathbf{1}$, and $\epsilon(n)$ is $\Omega(n^{-\beta})$. Suppose the scheduling algorithm satisfy state space collapse with,*

$$\mathbb{E}\Big[\,\|\bar{\mathbf{q}}_{\perp\mathcal{K}}\|^r\,\Big] \leq C_r \quad \forall r \in \{1, 2, \dots\},$$

such that C_r is $O(n^{\alpha r})$ for all $r \in \{1, 2, \dots\}$, then Eq. (13) *holds for all $\beta > \alpha + 1$.*

Note that for classical heavy-traffic results, we just need existence of SSC as in Definition 1 that only cares about the existence of parameter C_r, and not about its dependence on the system size n. In contrast, here the quality of SSC, i.e., the exact dependence of C_r on the system size n plays a key role. The quality of such SSC is in turn influenced by the drift with which the algorithm pushes towards the cone. Among the algorithms studied so far, MaxWeight has the strongest drift towards the cone, and so we have from prior work [12] that Eq. (13) is valid for $\beta > 4$. The class of algorithms studied in this paper have weaker drift towards the cone. While this distinction was not evident in the performance in the classical heavy-traffic regime, it becomes clearer in Theorem 15.

More precisely, for Class 1 algorithms, the quality of drift towards the cone is determined by the constant W_1 in Eq. (8). For MaxWeight, which also lies in Class 1, under uniform traffic, $W_1 = \nu_{\min} = 1/n$. For power-of-d and and random d-flip, from Lemma 7, we have that $W_1 = \frac{1}{2n^3}$, which is worse by a factor of $O(1/n^2)$. This leads to $\beta > 6$ in Theorem 15 for these algorithms, more than $\beta > 4$ for MaxWeight.

For Class 2 algorithms, the quality of drift towards the cone is determined by the constant W_2 in Eq. (10). For bursty MaxWeight and pipelined MaxWeight in this class, according to Definition 9 we have that $W_2 = 2nma_{\max}$ which is same as $O(n^{1+\gamma})$ when m is $O(n^{\gamma})$. On the other hand, for MaxWeight (which also lies in Class 2), $W_2 = 0$. It turns out that distinction leads to $\beta > 3 + \max\{1, \gamma\}$ for pipelined and bursty MaxWeight algorithms. The details of the proof of Theorem 15 is provided in Appendix K (in the supplement file).

At this point, it is not clear if a result similar to Theorem 15 can be proved for Class 3 algorithms and that is an open question for further investigation. The key challenge is that the parameter C_r in Lemma 16 depends on $1/\delta$, and can be as large as $n!$. Another open question is investigating these algorithms for the values of β not covered in Theorem 15. It is known [19] to be a challenging open problem to even study the MaxWeight algorithm when $\beta \leq 4$.

VII. d-FLIP: EMPIRICAL RESULTS

In this section, we present simulation results on an $O(d)$ algorithm named as d-flip. The algorithm d-flip differs from the random d-flip in the sense that d-flip do not generate a random schedule, it just uses the flip step d times on the schedule used in previous time slot, i.e. it generates $\mathbf{s}(t)$ by applying d flip steps on $\mathbf{s}(t-1)$. We only present empirical results related to d-Flip, as we cannot claim that d-flip lies in any of the three class mentioned in this paper. The complexity of d-flip is $O(d)$ as it uses just d flip step, each of complexity $O(1)$. For simplicity, we will use the term *Q-length* to denote $\mathbb{E}\big[\sum_{ij} \bar{q}_{ij}\big]$. Also, the term *Load* denotes the value $(1-\epsilon)$, where ϵ is the heavy traffic parameter.

Fig. 1 shows the effect for increasing the value of d in power-of-d, random d-flip and d-flip. The plot shows that increasing the d does not have a huge effect on power-of-d, but it has significant effect on random d-flip and d-flip. For small values of d, power-of-d perform better, while for large values of d, random d-flip and d-flip perform better. Intuitively, the reason behind this is that for smaller values of d, power-of-d has a higher probability of choosing a schedule with large weight as compared to random d-flip or d-flip, and this changes as the value of d increases. For example, consider a schedule which can be converted to the MaxWeight schedule by a flip step. In this case, random d-flip or d-flip have $2/n(n-1)$ probability of choosing a flip step that gives MaxWeight schedule, while for power-of-d, the probability of sampling a MaxWeight schedule is $1/n!$.

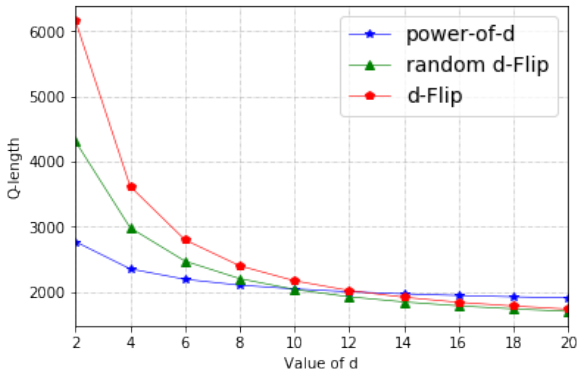

Fig. 1. *Q-length* vs *Value of d* plot for power-of-d, random d-flip and d-flip for a 16×16 switch under uniform Bernoulli traffic with $Load = 0.90$.

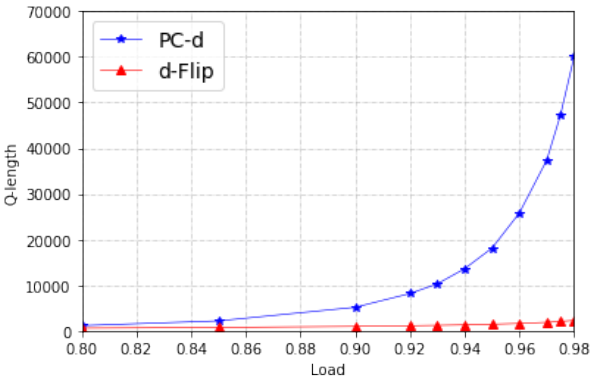

Fig. 2. *Q-length* vs *Load* plot for PC-d and d-flip with $d = 8$ for a 16×16 switch under non-uniform Bernoulli traffic.

In Fig. 2, we again show the comparison of PC-d and d-flip. In this plot, the arrival process is non-uniform. We know that PC-d lies in the class Π_3, so it is throughput optimal and also heavy traffic optimal. Even though d-flip does not lie in any of the classes mentioned in this paper, we can see that d-flip heavily outperforms PC-d.

VIII. FUTURE WORK

In this section, we present a few future directions and open problems. One open problem is characterizing the exact stability region of power-of-d or random d-flip. In this paper, we only looked at these algorithms under uniform traffic, or when the mean arrival rate lies in $\mathcal{C}^*$. But the stability region of

these algorithms is larger than $\mathcal{C}^*$. Once the capacity region is understood, one can then study these algorithms under nonuniform traffic as long as the load is within their capacity region.

While this paper studies three different classes of low complexity algorithms, there are a few more algorithms that do not fall in any of the classes, and so are not analytically understood. The d-flip is one such algorithm, which is seen to perform well in simulations presented in Section VII. Another example is iSLIP [10], which commonly used in data centers, but the heavy traffic result for iSLIP is not known.

Another future direction is the large scale analysis of algorithms in Class Π_3. Such an analysis will help us further differentiate between the algorithms in Class Π_3. Since simulations from Section VII indicate that some algorithms such as PC-d, LAURA and SERENA from the class Π_3 perform well, one expects that for these algorithms, a large scale heavy traffic regime result might be true.

REFERENCES

[1] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260–1267, 1999.

[2] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pFabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 435–446, 2013.

[3] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: a centralized" zero-queue" datacenter network," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 307–318.

[4] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, Dec 1992.

[5] R. Duan and H.-H. Su, "A scaling algorithm for maximum weight matching in bipartite graphs," in *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2012, pp. 1413–1424.

[6] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proceedings. IEEE INFOCOM '98, the Conference on Computer Communications.*, vol. 2, March 1998, pp. 533–539 vol.2.

[7] D. Shah and M. Kopikare, "Delay bounds for approximate maximum weight matching algorithms for input queued switches," in *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, June 2002, pp. 1024–1031 vol.2.

[8] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 4, pp. 546–559, 2003.

[9] D. Shah, P. Giaccone, and B. Prabhakar, "Efficient randomized algorithms for input-queued switch scheduling," *IEEE Micro*, vol. 22, no. 1, pp. 10–18, 2002.

[10] N. McKeown, "The islip scheduling algorithm for input-queued switches," *IEEE/ACM transactions on networking*, vol. 7, no. 2, pp. 188–201, 1999.

[11] L. Gong, J. Xu, L. Liu, and S. T. Maguluri, "QPS-r: A cost-effective crossbar scheduling algorithm and its stability and delay analysis," *arXiv preprint arXiv:1905.05392*, 2019.

[12] S. T. Maguluri and R. Srikant, "Heavy traffic queue length behavior in a switch under the maxweight algorithm," *Stochastic Systems*, vol. 6, no. 1, pp. 211–250, 2016. [Online]. Available: https://doi.org/10.1287/15-SSY193

[13] S. T. Maguluri, S. K. Burle, and R. Srikant, "Optimal heavy-traffic queue length scaling in an incompletely saturated switch," in *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, 2016, pp. 13–24.

[14] D. Hurtado-Lange and S. T. Maguluri, "Heavy-traffic analysis of queueing systems with no complete resource pooling," 2019.

[15] M. L. Balinski and R. E. Gomory, "A primal method for the assignment and transportation problems," *Management Science*, vol. 10, no. 3, pp. 578–593, 1964.

[16] D. Shah, J. N. 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.

[17] D. Shah, N. S. Walton, and Y. Zhong, "Optimal queue-size scaling in switched networks," *The Annals of Applied Probability*, vol. 24, no. 6, pp. 2207–2245, 2014.

[18] D. Shah, J. N. Tsitsiklis, and Y. Zhong, "On queue-size scaling for input-queued switches," *Stochastic Systems*, vol. 6, no. 1, pp. 1–25, 2016.

[19] J. Xu and Y. Zhong, "Improved queue-size scaling for input-queued switches via graph factorization," in *Abstracts of the 2019 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*, 2019, pp. 67–68.

[20] A. L. Stolyar *et al.*, "Maxweight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic," *The Annals of Applied Probability*, vol. 14, no. 1, pp. 1–53, 2004.

[21] G. M. Ziegler, *Lectures on polytopes*. Springer Science & Business Media, 2012, vol. 152.

[22] Y. Lu, S. Maguluri, M. Squillante, T. Suk, and X. Wu, "An optimal scheduling policy for the 2 x 2 input-queued switch with symmetric arrival rates," *SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 3, p. 217–223, Mar. 2018. [Online]. Available: https://doi.org/10.1145/3199524.3199563

[23] R. A. Fisher and F. Yates, *Statistical tables: For biological, agricultural and medical research*. Oliver and Boyd, 1938.

[24] R. Srikant and L. Ying, *Communication Networks: An Optimization, Control and Stochastic Networks Perspective*. Cambridge University Press, 2014.

[25] C.-S. Chang, D.-S. Lee, and Y.-S. Jou, "Load balanced birkhoff-von neumann switches, part i: One-stage buffering," *Comput. Commun.*, vol. 25, no. 6, pp. 611–622, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1016/S0140-3664(01)00427-3

[26] M. J. Neely, E. Modiano, and Y.-S. Cheng, "Logarithmic delay for $n \times n$ packet switches under the crossbar constraint," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 657–668, 2007.

Prakirt Raj Jhunjhunwala is a Ph.D. student with major in Operations Research and minor in Mathematics at ISyE, Georgia Institute of Technology. He obtained his B.Tech in Electrical Engineering from IIT Bombay. His research interests are Data Center Networks, Queueing Theory, Stochastic Processing Networks, Reinforcement Learning and Simulation Optimization. He is the Recipient of "Best Paper Award" in SPCOM 2018.

Siva Theja Maguluri is Fouts Family Early Career Professor and Assistant Professor in the School of Industrial and Systems Engineering at Georgia Tech. He obtained his Ph.D. and MS in ECE as well as MS in Applied Math from UIUC, and B.Tech in Electrical Engineering from IIT Madras. His research interests span the areas of Networks, Control, Optimization, Algorithms, Applied Probability and Reinforcement Learning. He is a recipient of the biennial "Best Publication in Applied Probability" award in 2017, "CTL/BP Junior Faculty Teaching Excellence Award" in 2020 and "Student Recognition of Excellence in Teaching: Class of 1934 CIOS Award" in 2020.